Activity Recognition

CSE 572: Data Mining Assignment 1

TEAM 3

Pratik Bartakke - 1213175715

Vihar Bhatt - 1213139146

Venkata Sai Sandeep Nadella - 1215185621

Darsh Parikh - 1213185088

Abstract

The project aims at developing a computing system to recognize and distinguish eating activities from noise. Several video sequences have been acquired with various users performing eating actions with the fork, spoon wearing Myo gesture control armband. The Myo wristband data includes data from accelerometer, gyroscope, orientation and EMG sensors. The dataset is provided along with the eating action labels throughout the video session. The assignment aims at 1) Cleaning and Organization of sensor data 2) Feature extraction and 3) Feature selection. In the data cleaning phase, the sensor data along with the ground truth data is synced to a common axis and merged. The next phase of feature extraction involved computing six features namely mean, maximum, root mean square (RMS), standard deviation, entropy, and power across the sensor data. The extracted features are then analyzed and compared by checking for the cosine distance. Based on the similarity measure, individual features are handpicked and Principal Component Analysis (PCA) is performed on the matrix to reduce the dimensionality of the data. The report walks through the details on the process followed at each of these individual phases backing up with logical reasoning and plots.

Keywords: Myo Armband, Data Mining, Activity Recognition, feature extraction, feature selection, PCA, eigen vectors.

Introduction

In this assignment, we built a system that could identify human activities, specifically eating and non-eating activities. The dataset provided was created based on 2 sources

1. Wristband data - In this, the user was wearing the wrist band and performed eating actions, and further non-eating actions were also recorded. The wristband provided data which included

a. Accelerometer data

b. Gyroscope data

c. Orientations hand movements

d. EMG sensor data

2. Video Data – A video of the user doing eating and non-eating habits was also recorded, for classifying the actions as eating/non-eating actions.

The assignment consists of 3 phases

1. Data cleaning, noise reduction and organizing the data for further machine learning stages.

2. Feature extraction – Based on the organized data got from the first phase, we applied 6 different feature extraction methods and compared the results, to see which method would provide a way the new features can be used to distinguish between eating/non-eating actions most clearly.

3. Feature selection – In this phase, we utilized Principal Component Analysis, to reduce the number of features extracted from phase 2 and keep only those features which would provide the maximum information which would help to distinguish between eating and non-eating actions.

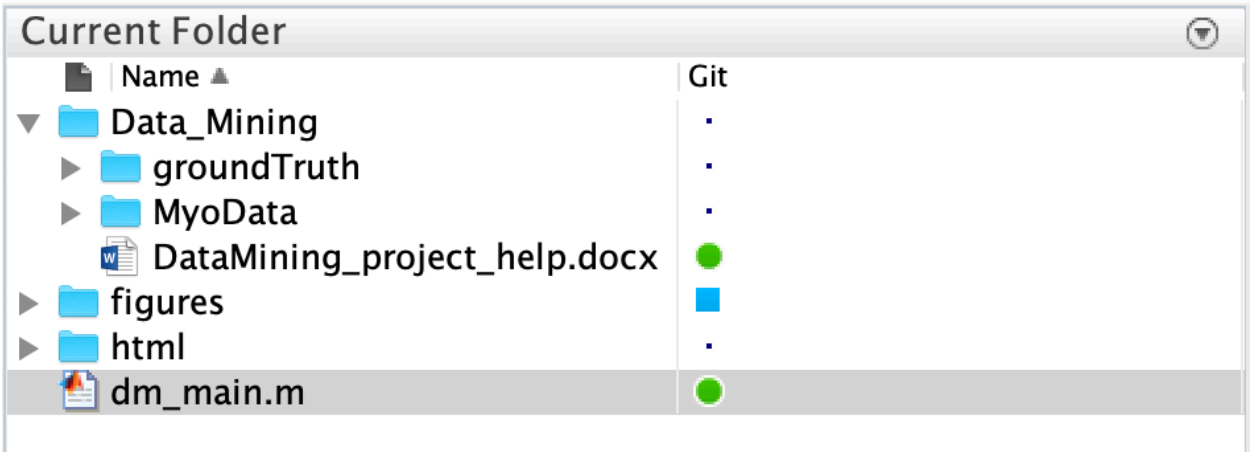
Phase 1 - Data organization

In this stage the preprocessed data provided is read and stored. As stated in the dataset help document file 'DataMining_project_help.docx', the sensor and ground truth datasets are synchronized by setting the last frame time to the last UNIX timestamp in IMU, EMG files. The following report analyzes the eating, non-eating actions of the user using 'fork'. The similar can be done on actions performed using spoon as well.

Note: The code expects the user names to be the same across the sensor and ground truth datasets. Hence, the 'user9' folder under 'groundTruth' folder has been renamed to 'user09' to maintain consistency with 'user09' folder under 'MyoData'.

The dataset provided had the following

1. The ground truth label files, which has the starting and ending frames of all the eating actions a user had performed during the video. It was for each user while using fork, spoon and is located under 'groundTruth' folder.
2. IMU sensor data files for fork and spoon actions for each user. Each file has 11 columns with UNIX timestamp for Orientation X, Orientation Y, Orientation Z, Orientation W, Accelerometer X, Accelerometer Y, Accelerometer Z, Gyroscope X, Gyroscope Y, Gyroscope Z.
3. EMG sensor data files for fork and spoon actions for each user. Each file has 8 columns corresponding to UNIX timestamp (at which these sensor data readings were taken), EMG 1, EMG 2, EMG 3, EMG 4, EMG 5, EMG 6, EMG 7, EMG 8.




```
addpath(genpath(pwd));  % Adding all the folders and subfolders of the current working directory to the work path

% Reading the sample dataset
fork_list = dir('Data_Mining/MyoData/*/fork/*');
fork_list = fork_list(~ismember({fork_list.name},{'.','..'}));
spoon_list = dir('Data_Mining/MyoData/*/spoon/*');
spoon_list = spoon_list(~ismember({spoon_list.name},{'.','..'}));
ground_truth_fork_list  = dir('Data_Mining/groundTruth/*/fork/*.txt');
ground_truth_spoon_list  = dir('Data_Mining/groundTruth/*/spoon/*.txt');
samples_matrix = [];    % Used for storing the entire sample data set after data cleaning


if size(fork_list,1)~=(size(spoon_list,1))
   throw(MException('DM:DatasetMismatch', 'The number of fork and spoon eating action entries are not the same'));
end

if size(fork_list,1)/3~=size(ground_truth_fork_list,1)
   throw(MException('DM:DatasetMismatch', 'The number of eating actions and ground truth label entries are not the same'));
end

if size(ground_truth_fork_list,1)~=size(ground_truth_spoon_list,1)
   throw(MException('DM:DatasetMismatch', 'The number of ground truth labels for spoon and fork entries are not the same'));
end
num_users  = size(ground_truth_fork_list,1);
```

The following steps were followed for organizing the data to merge into a common table.

- The ground truth file is picked for a particular user.
- The corresponding IMU and EMG data in the fork folder is read.
- The first timestamp of sensor data for the initial sample and the last timestamp of the sensor data for the final sample are taken as reference for the video recording.
- The ground truth axis extracted from the dataset is in the frame axis format. Hence, this is converted to ground truth time axis format by synchronizing the end frames and calculating the timestamps as stated in the dataset help document.

- The time axis extracted from IMU and EMG files is not in sync with the common video time axis we need. Hence, we need to interpolate the IMU, EMG and the ground truth time axis into the common time axis of the video recording. In this process only labels with highest confidence are retained.

```
% The OOTB code only runs on fork dataset. Can be tweaked based on need to
% run on spoon dataset as well.

for i = 1:num_users

   path_split  = strsplit(ground_truth_fork_list(i).folder, '/');
   user  = path_split{end-1};  % contains the current user for which the data is being processed
   ground_truth_fork_data = csvread(ground_truth_fork_list(i).name);

   imu_fork_data = [];
   emg_fork_data = [];
   video_info_fork_data = [];

   %    Populate the dataset for current user
   for j=1:size(fork_list,1)
      if contains(fork_list(j).folder, user) && contains(fork_list(j).name,"IMU")
         imu_fork_data = csvread(fork_list(j).name);
      elseif contains(fork_list(j).folder, user) && contains(fork_list(j).name,"EMG")
         emg_fork_data = csvread(fork_list(j).name);
      elseif contains(fork_list(j).folder, user) && contains(fork_list(j).name,"video_info")
         video_info_fork_data = csvread(fork_list(j).name);
      end
      if isempty(imu_fork_data) || isempty(emg_fork_data) || isempty(video_info_fork_data)
         continue;
      else
         break;
      end
   end

   % Get the time axis of the video recording
   video_time_axis = linspace(max(imu_fork_data(1,1), emg_fork_data(1,1)),min(imu_fork_data(end,1),
emg_fork_data(end,1)),(ground_truth_fork_data(end,2)-ground_truth_fork_data(1,1)+1));
   % Get the ground truth axis
   gt_time_axis = linspace(min(imu_fork_data(end,1), emg_fork_data(end,1))-
(ground_truth_fork_data(end,2)*(1000/30)),min(imu_fork_data(end,1),
emg_fork_data(end,1)),ground_truth_fork_data(end,2));

   gt_label = zeros(1,ground_truth_fork_data(end,2)); % the indices indicate ground truth frame axis
   for s=1:size(ground_truth_fork_data,1)
      for h = 1:length(gt_label)
         if h>=ground_truth_fork_data(s,1) && h<=ground_truth_fork_data(s,2)
            gt_label(1,h)=1; % Indicate it as eating frame
         end
      end
   end

   imu_fork_time = imu_fork_data(:,1);
   emg_fork_time = emg_fork_data(:,1);

   %    interpolate the IMU fork data to a common video time axis to combine
   %    IMU, EMG and GroundTruth data
   interpolated_fork_data = [];
```

```
    interpolated_imu_data = [];
    interpolated_emg_data = [];
    interpolated_gt_data = [];
    interpolated_fork_data_cleaned = [];

    for k = 2:11    % there are 18 sensors in the given dataset
        interpolated_imu_data = vertcat(interpolated_imu_data,interp1(imu_fork_time, imu_fork_data(:,k), video_time_axis,
'linear'));
        if k~=10 && k~=11
            interpolated_emg_data = vertcat(interpolated_emg_data,interp1(emg_fork_time, emg_fork_data(:,k), video_time_axis,
'linear'));
        end
    end
    interpolated_gt_data = interp1(gt_time_axis, gt_label, video_time_axis, 'linear');
    interpolated_fork_data = vertcat(interpolated_gt_data, interpolated_imu_data, interpolated_emg_data);
    interpolated_fork_data = vertcat(repmat(str2num(strrep(user,'user','')),1,length(video_time_axis)), video_time_axis,
interpolated_fork_data);

    for g = 1:size(interpolated_fork_data, 2)
        if (interpolated_fork_data(3,g)==1||interpolated_fork_data(3,g)==0)
            interpolated_fork_data_cleaned = [interpolated_fork_data_cleaned interpolated_fork_data(:,g)];
        end
    end
    % concatenate and store the data corresponding to each user along
    % with timestamps across all users in the samples_matrix
    samples_matrix = [samples_matrix interpolated_fork_data_cleaned];
end

clearvars -except samples_matrix
```

- A source dataset matrix - “sample_matrix”, with sensor sample timestamps, sensor data is created for all the user actions across all video recording sessions.
- The class label associated with timestamps depicting eating actions is 1 and the class label associated with timestamps depicting non-eating actions is 0.

The final “sample matrix”, has data recordings from all the 18 sensors of the Myo band.

samples_table

21x2000 table

	1 Var1	2 Var2	3 Var3	4 Var4	5 Var5	6 Var6	7 Var7
1 user	9	9	9	9	9	9	9
2 time_stamp	1.5035e...	1.5035e...	1.5035e...	1.5035e...	1.5035e...	1.5035e...	1.5035e...
3 ground_truth	0	0	0	0	0	0	0
4 OrientationX	-0.7170	-0.7168	-0.7160	-0.7150	-0.7180	-0.7231	-0.7230
5 OrientationY	-0.6400	-0.6400	-0.6400	-0.6393	-0.6364	-0.6321	-0.6361
6 OrientationZ	0.1579	0.1554	0.1593	0.1668	0.1682	0.1636	0.1578
7 OrientationW	0.2290	0.2300	0.2300	0.2293	0.2276	0.2246	0.2178
8 AccelerometerX	0.8966	0.9178	0.8664	0.8611	0.8214	0.7954	0.9454
9 AccelerometerY	0.0016	-0.0178	0.0549	0.1159	0.1061	0.0968	0.0398
10 AccelerometerZ	-0.1231	-0.0789	0.0185	0.1211	0.1425	0.0763	0.0272
11 GyroscopeX	13.5486	1.9839	-12.4371	-12.5754	6.6846	20.2676	33.9369
12 GyroscopeY	-3.9494	-0.2579	1.8333	-8.1907	-34.4398	-25.9380	5.3663
13 GyroscopeZ	24.8709	5.3641	-3.3069	-4.1444	-0.8447	5.7630	4.4050
14 EMG1	-3.2537	7.2441	2.9283	1.4321	-1.9421	-3.9719	3.9799
15 EMG2	18.5208	5.5272	21.3570	5.8617	-14.7434	1.9478	-11.9916
16 EMG3	8.2255	-1.3780	1.5716	3.8370	-10	-2.7309	-2.9833
17 EMG4	1.8309	-2.8661	-0.8572	-0.8642	3.6197	-6.0281	-3.9967
18 EMG5	28.2389	-20.3780	0.1419	-16.5901	18.7684	-24.7349	4.9749
19 EMG6	12.4362	5.2915	-5.5719	-8.5654	4.6776	-12.2731	-0.0084
20 EMG7	0.8027	-0.6220	-0.4287	-1.3778	0	-2.1084	-1.9950
21 EMG8	-4.9436	-1.3780	-0.4287	-0.9185	-2.4711	0.1084	4

Phase 2 - Features extraction

Explanation on how each feature is extracted

In this phase, we used 6 feature extraction methods to get useful features from the samples_matrix generated in the previous phase. The feature computation is performed by selecting each action sequence from the ground truth dataset and computing the mean, maximum, standard deviation, RMS, entropy and power across this sequence. This is done on both eating and non-eating action sequences across all the users. This helps in the dimensionality reduction across the observation samples and extracts the feature space that is robust to noise. The reduced feature matrix is then passed to PCA in phase 3 for further dimensionality reduction across the variable/feature length.

1. **Mean**
   a. **Explanation on how the feature is computed**

   For a random variable vector A, made up of N scalar observations, the mean is defined as

$$\mu = \frac{1}{N}\sum_{i=1}^{N} A_i \qquad [1]$$

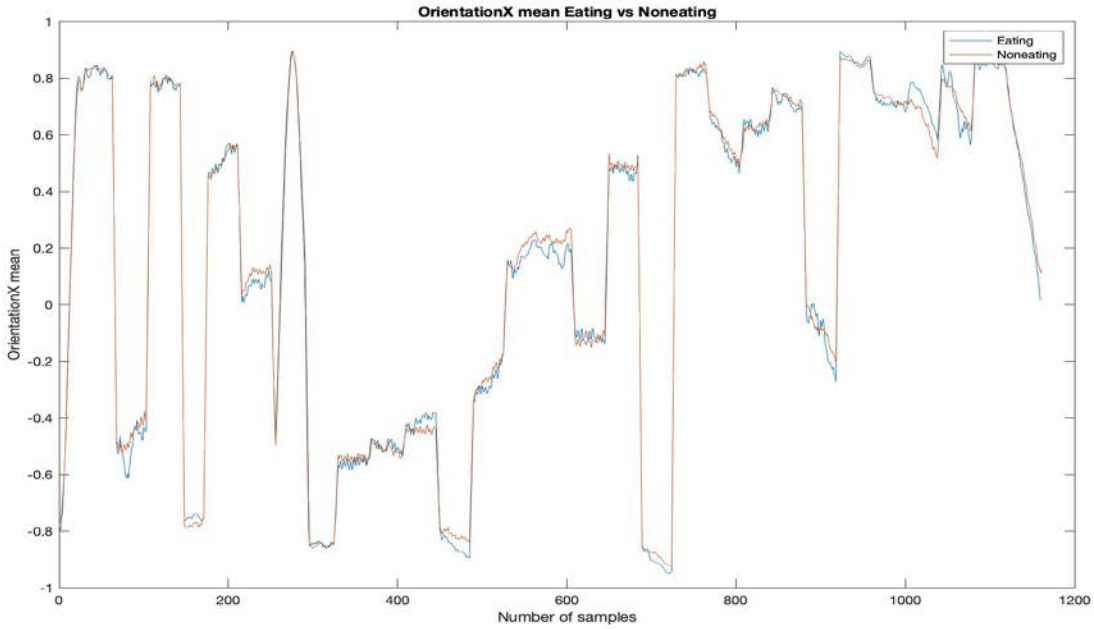
b. <u>The intuition behind using Mean</u>

Mean is a measure of central tendency, which gives us an idea about the mid-value or the central value in data. Using mean on all attributes, would part from reducing data sample size, would also provide us with a value that is most representative of the whole data, which is easier to analyze. It also helps in differentiating between the eating and non-eating labels we had to predict. Also, using mean helps to normalize the data points and reduces noise and outliers [2].

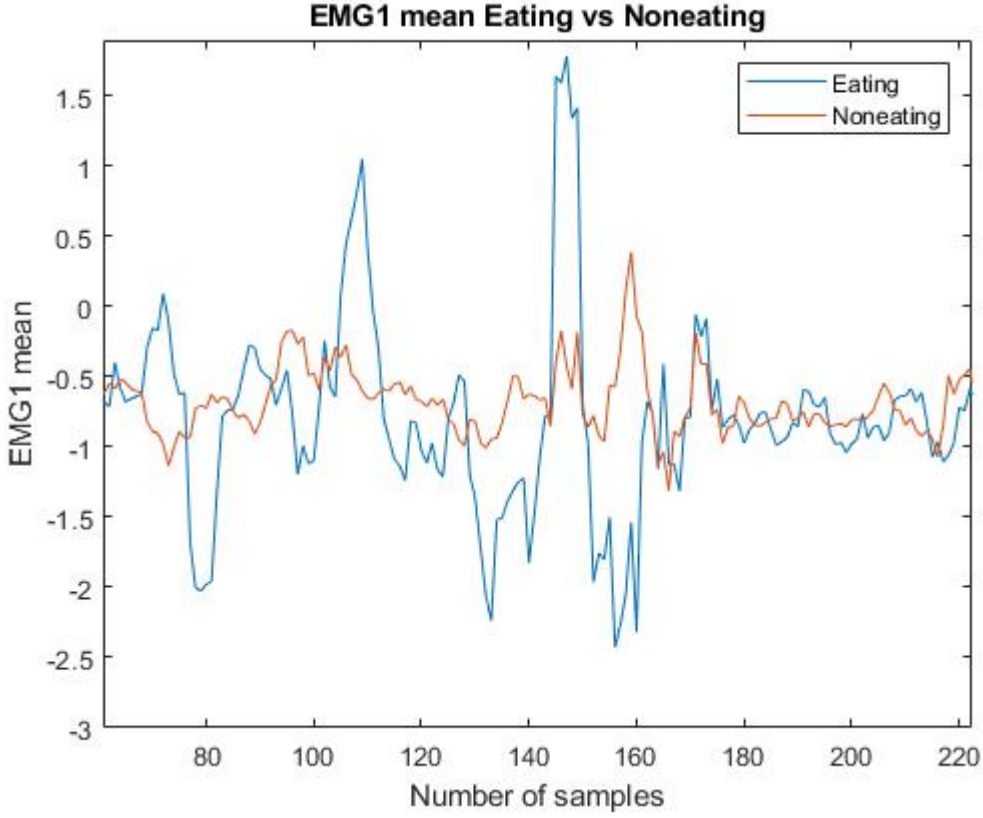
c. <u>The plots for the eating and non-eating actions, using mean as a feature</u>

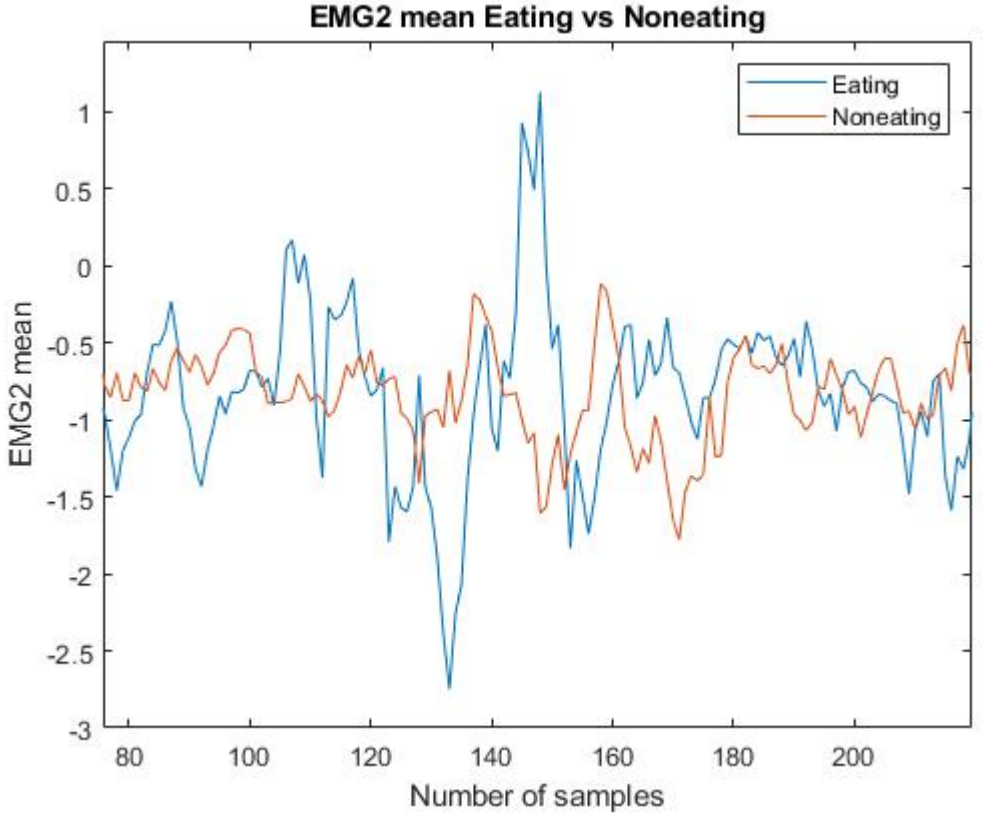
The mean was calculated across all eating and non-eating windows for each sensor and the ones with low cosine similarity were picked.

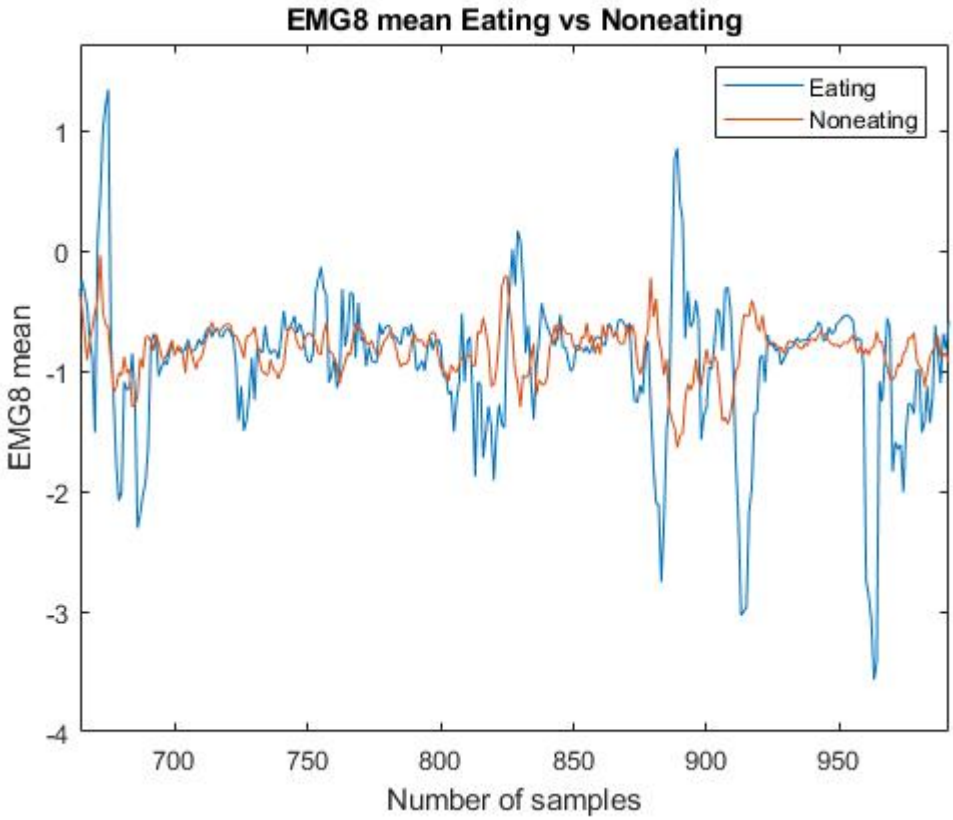
Consider the below graph of OrientationX which doesn't show clear distinction between eating and noneating. In line with our intuition, the cosine similarity turned out to be higher and hence this feature wasn't picked for passing to PCA in phase 3.



On the other hand, the following features showed clear distinction across eating and non-eating and backing our intuition these features were picked for PCA by our cosine similarity method.


EMG1 mean Eating vs Noneating
Eating
Noneating
EMG1 mean
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
80
100
120
140
160
180
200
220
Number of samples


EMG2 mean Eating vs Noneating
Eating
Noneating
EMG2 mean
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
80
100
120
140
160
180
200
Number of samples



2. **RMS (Root Mean Square)**

   a. **Explanation on how the feature is computed**

   The RMS (Root mean square) can be calculated by taking the mean of the square of all values and then taking the square root of that mean.

$$x_{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}|x_n|^2} \quad [3]$$

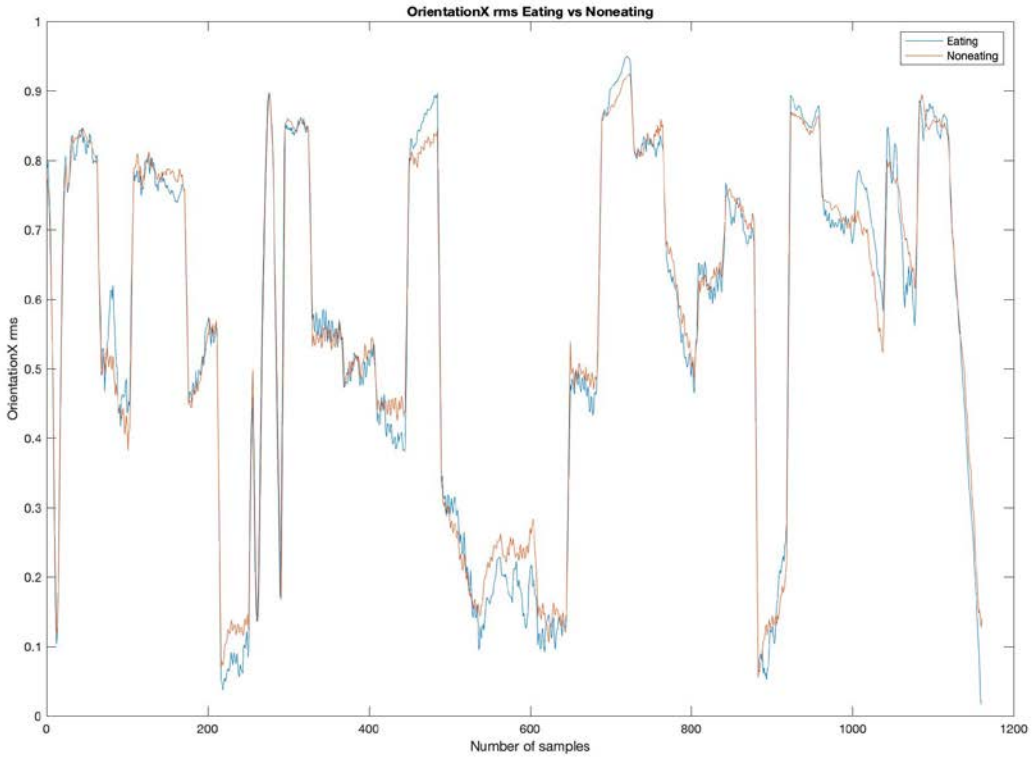
   b. **The intuition behind using RMS is as follows.**

   Root mean square can be used as a method for feature extraction since the dataset contains both positive and negative values and using mean may not give an accurate representation of the entire dataset. RMS can also be understood as deviation from the dataset mean [4].

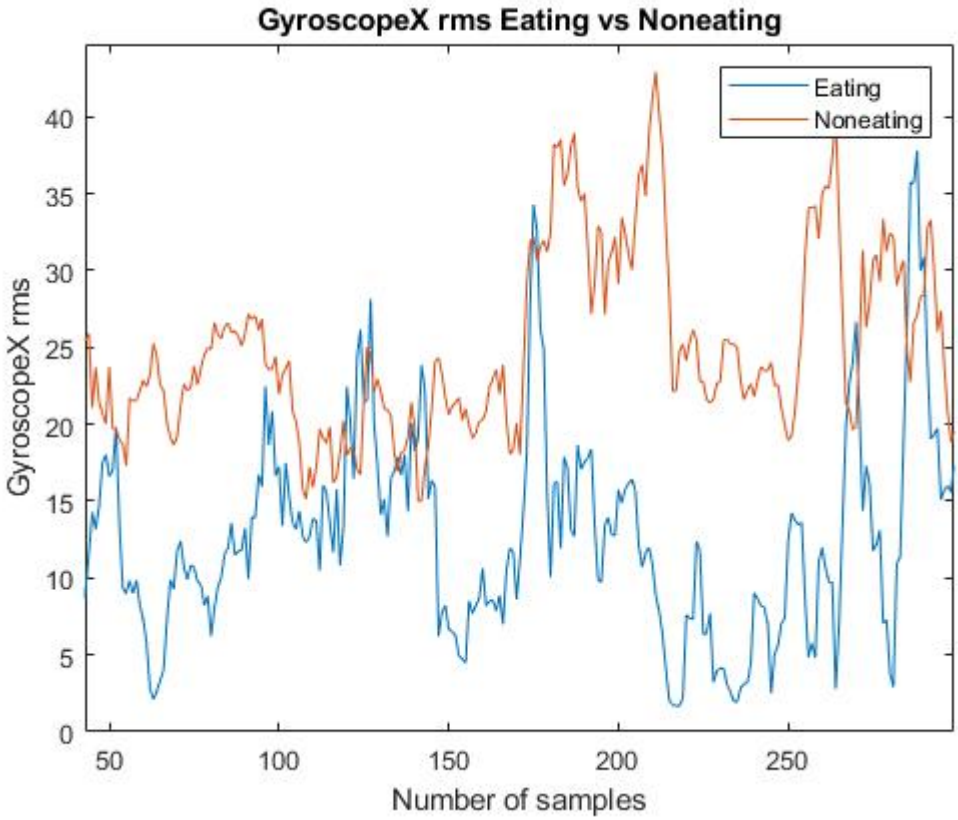
   c. **The plots for the eating and non-eating actions, using RMS as a feature**

   The RMS was calculated across all eating and non-eating windows for each sensor and the ones with low cosine similarity were picked similar to the way we picked mean above.

   Consider the below graph of OrientationX which doesn't show clear distinction between eating and noneating. In line with our intuition, the cosine similarity turned out to be higher and hence this feature wasn't picked for passing to PCA in phase 3.



On the other hand, the following features showed clear distinction across eating and non-eating and backing our intuition these features were picked for PCA by our cosine similarity method.



3. **Standard Deviation**

   a. **Explanation on how the feature is computed**

   For a random variable vector A made up of N scalar observations, the standard deviation is defined as

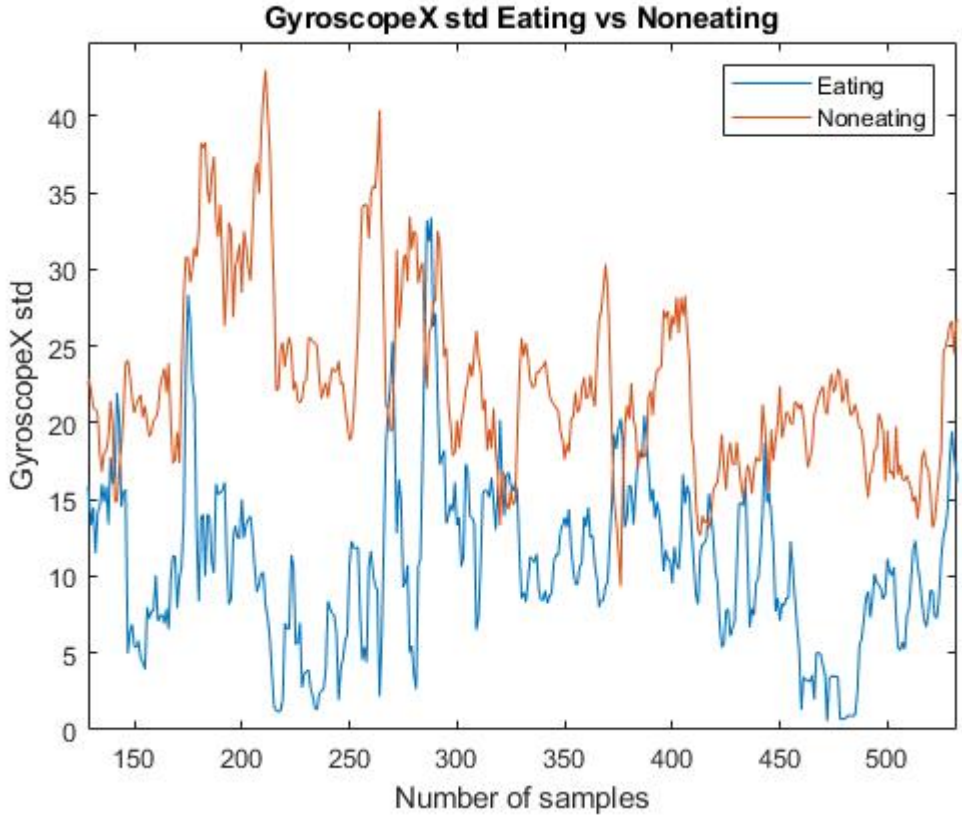
$$S = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}|A_i - \mu|^2} \qquad [5]$$

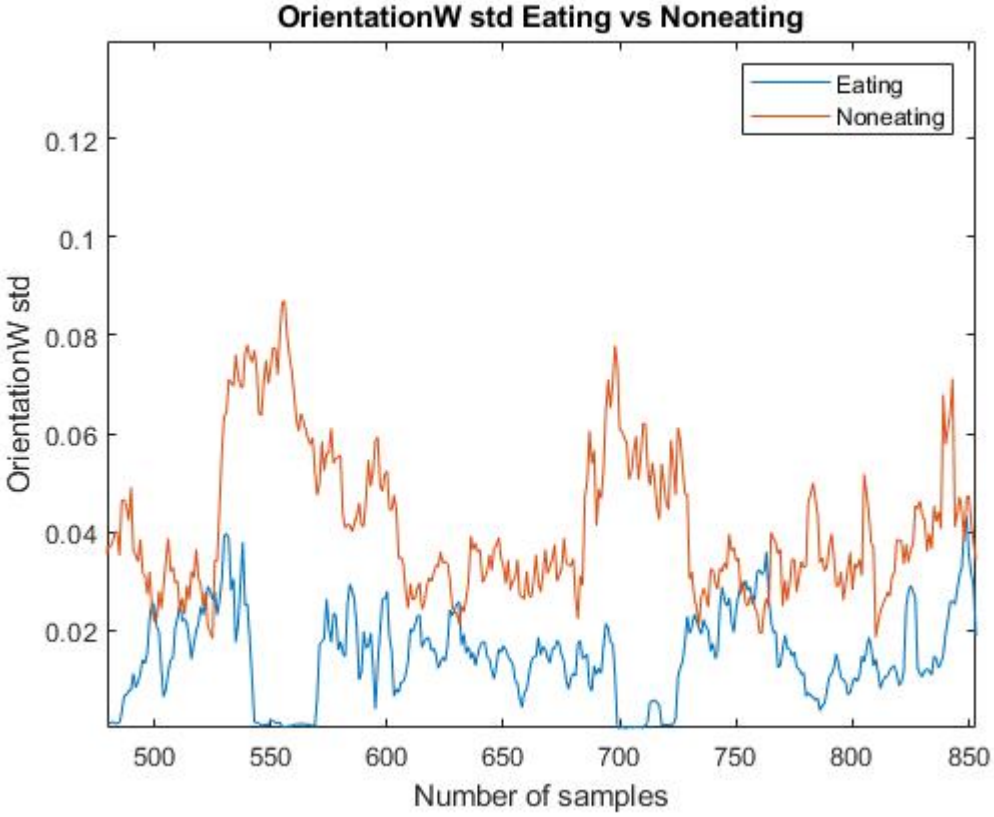
   where μ is the mean of A.

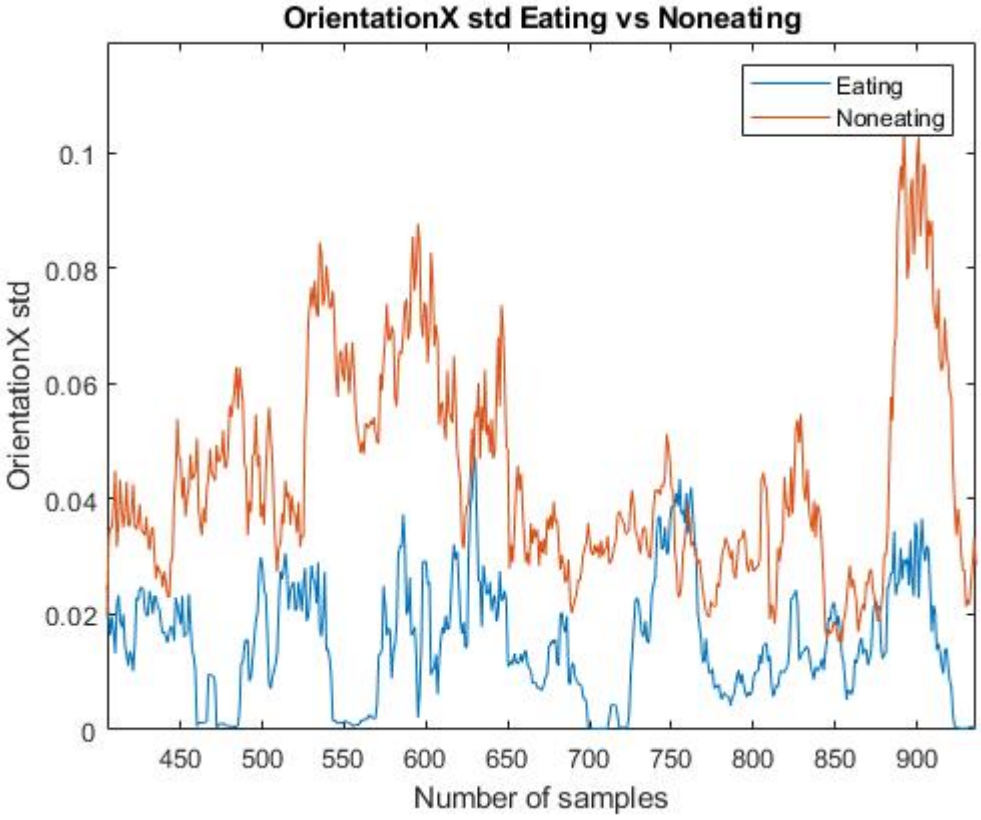
   b. **The intuition behind using standard deviation**

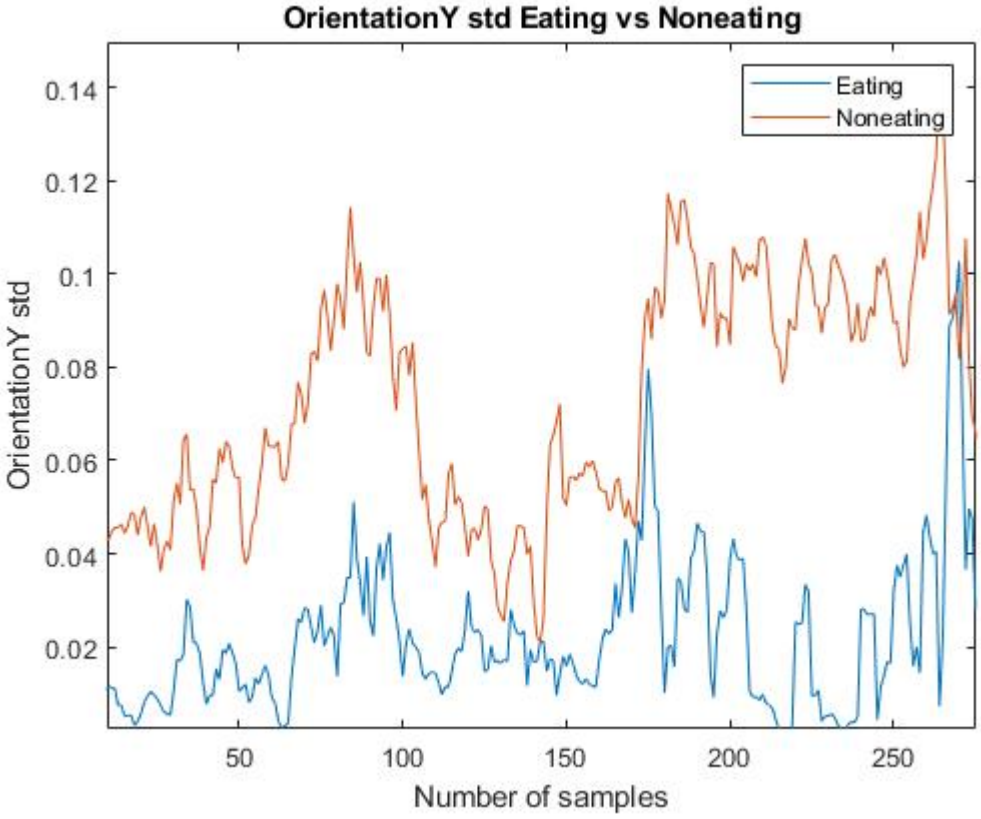
   Standard deviation shows how much the values of the dataset vary from the average value of the dataset. A small deviation value means that the values in the dataset are pretty close to the average value of the dataset, whereas, high standard deviation value means that the values in the dataset are at large distance from the average value of the dataset. As this method helps determine the features with large deviation, it can be a useful method for feature extraction [6].

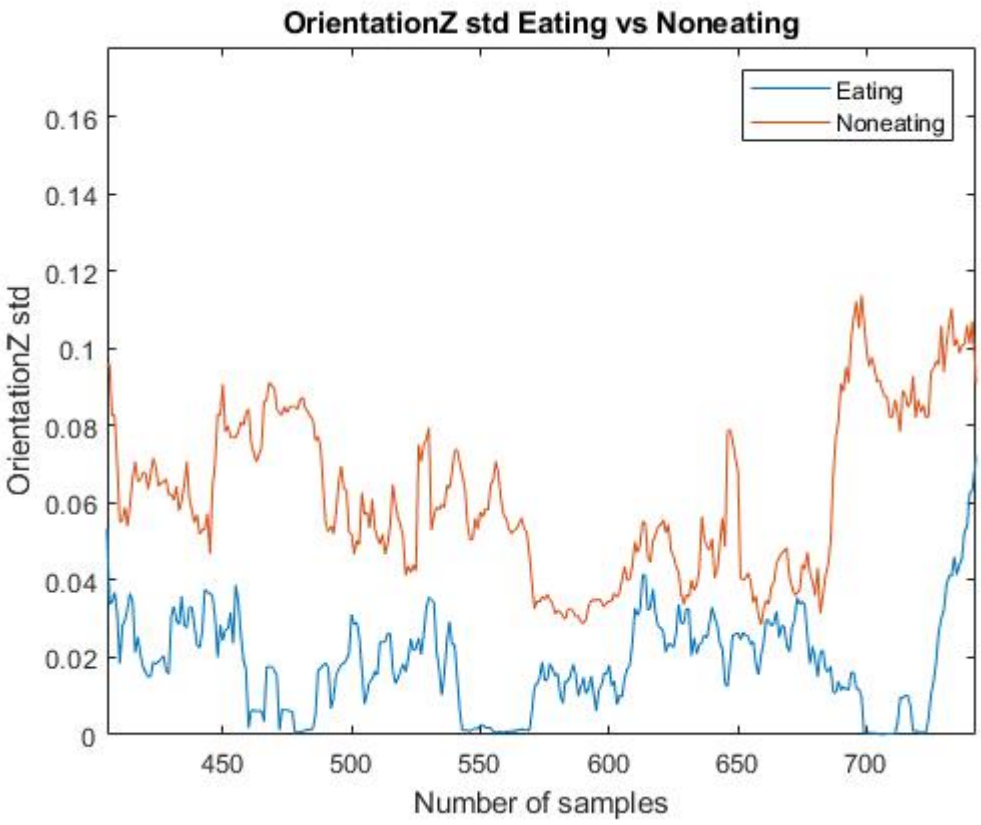
   c. **The plots for the eating and non-eating actions, using standard deviation as a feature**


GyroscopeX std Eating vs Noneating
Eating
Noneating
GyroscopeX std
Number of samples
0
5
10
15
20
25
30
35
40
150
200
250
300
350
400
450
500


OrientationW std Eating vs Noneating
Eating
Noneating
OrientationW std
Number of samples
0.02
0.04
0.06
0.08
0.1
0.12
500
550
600
650
700
750
800
850


OrientationX std Eating vs Noneating
Eating
Noneating
OrientationX std
0.1
0.08
0.06
0.04
0.02
0
450
500
550
600
650
700
750
800
850
900
Number of samples


OrientationY std Eating vs Noneating
Eating
Noneating
OrientationY std
0.14
0.12
0.1
0.08
0.06
0.04
0.02
50
100
150
200
250
Number of samples



4. **Maximum**

   a. **Explanation on how the feature is computed**

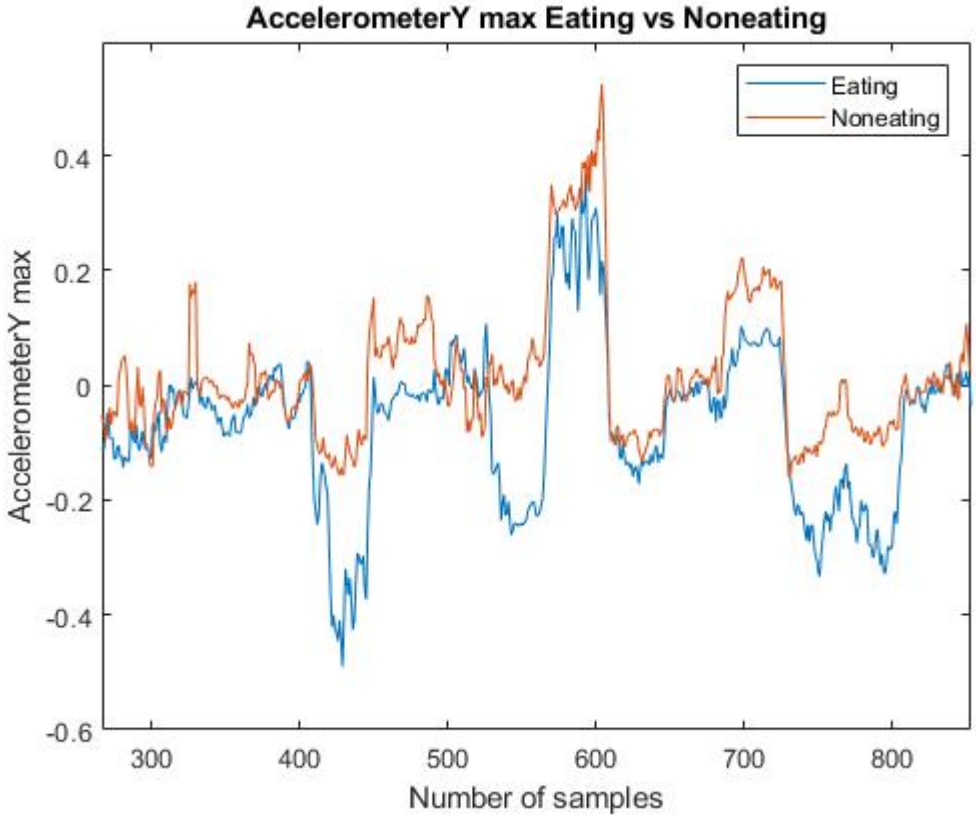
   The Maximum of a feature vector sequence of n samples is defined as

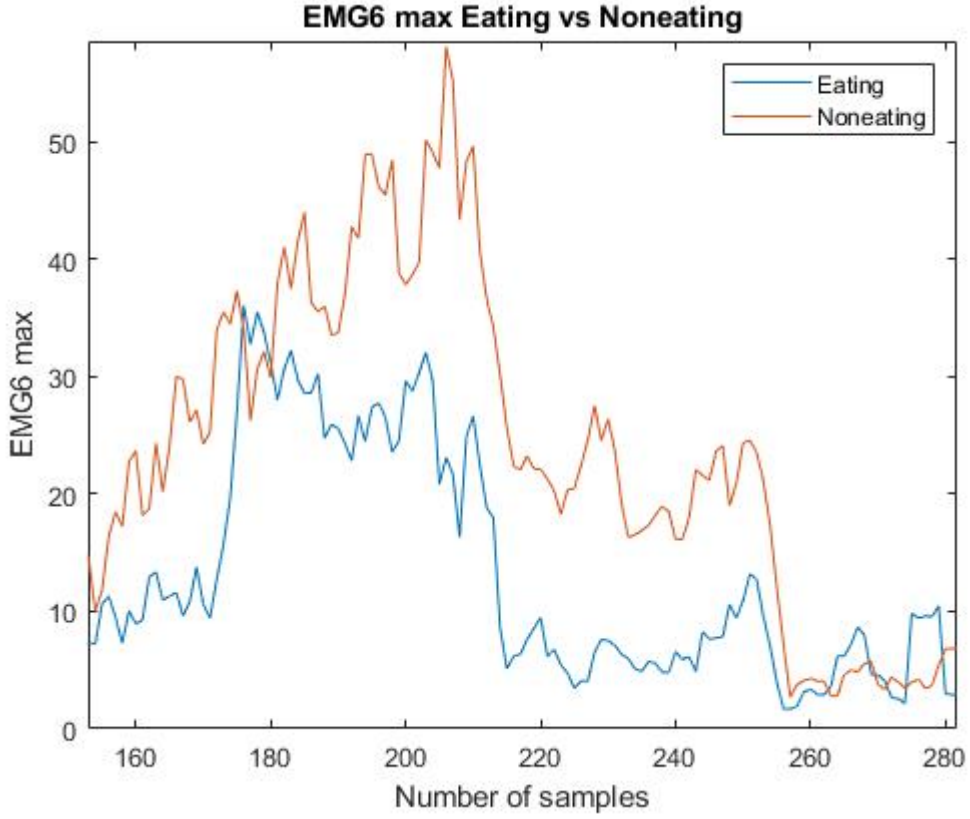
   $$x_{max} = max(x_1, x_2, x_3, \dots, x_n) \quad [7]$$

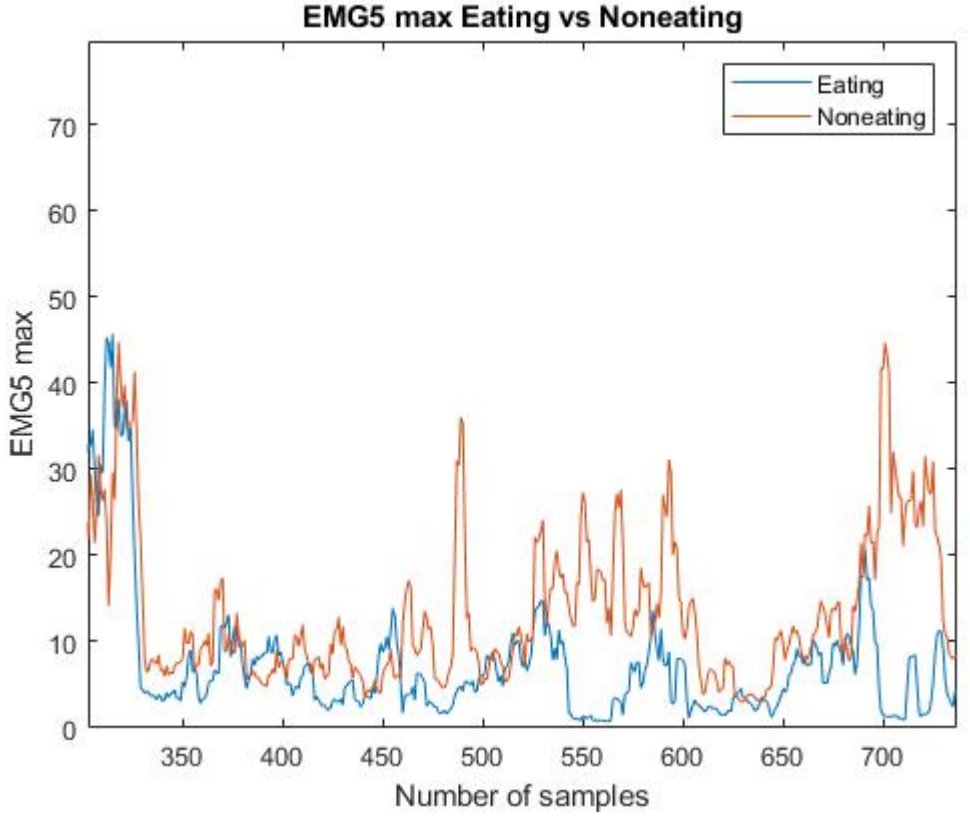
   b. **The intuition behind using Maximum**

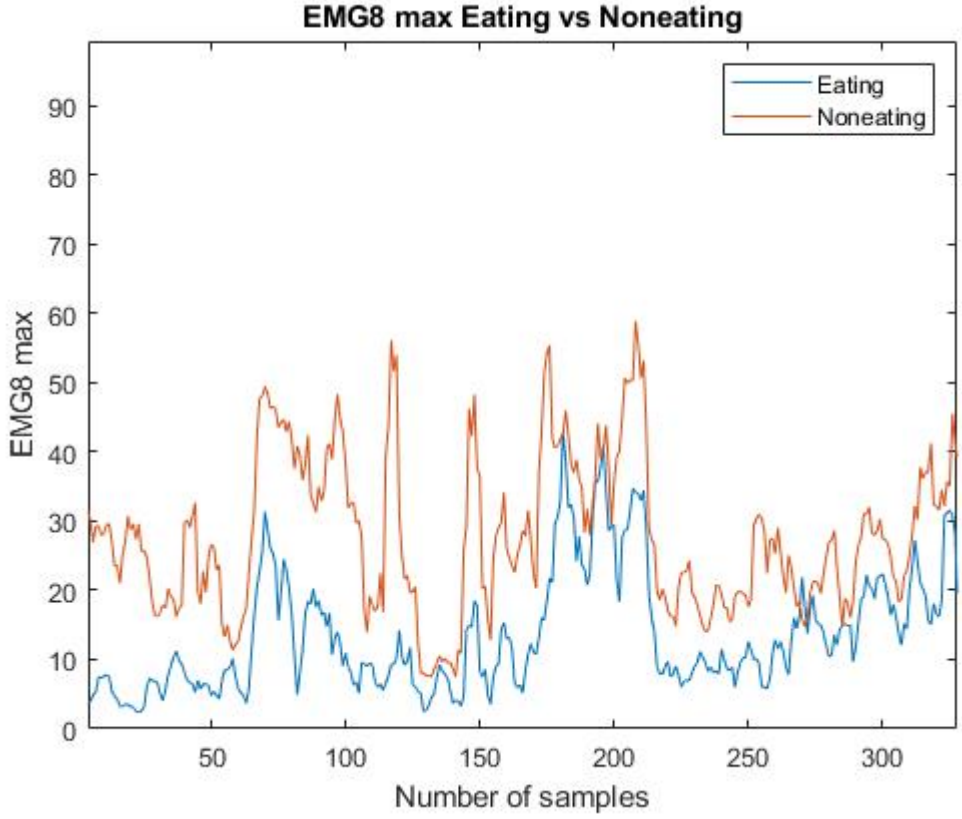
   The reason for using maximum as a method of feature extraction is that it helps us in finding the data points in the samples that have the highest variance.

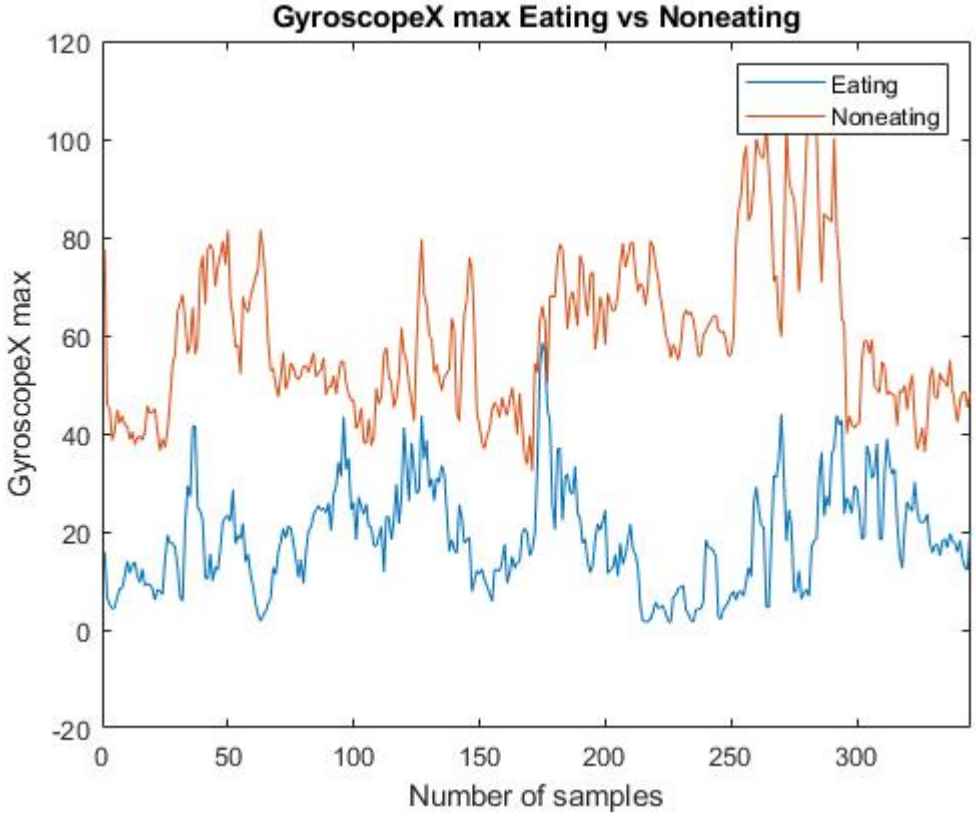
   c. **The plots for the eating and non-eating actions, using maximum as a feature**

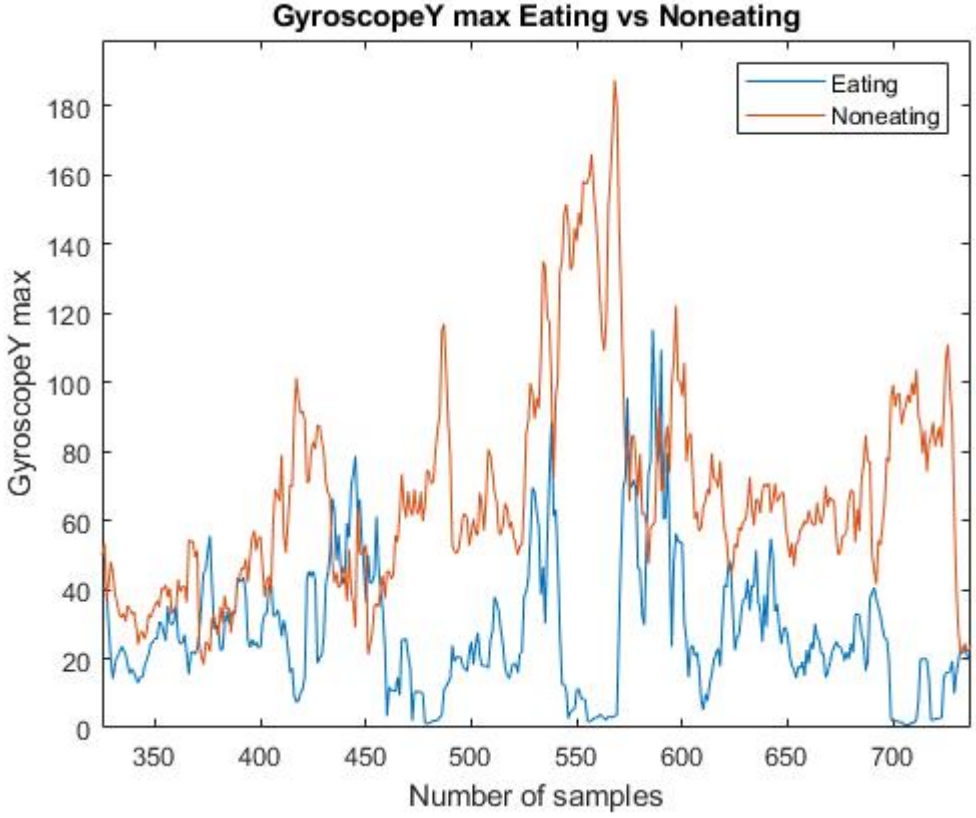

AccelerometerY max Eating vs Noneating
Eating
Noneating
AccelerometerY max
0.4
0.2
0
-0.2
-0.4
-0.6
300
400
500
600
700
800
Number of samples

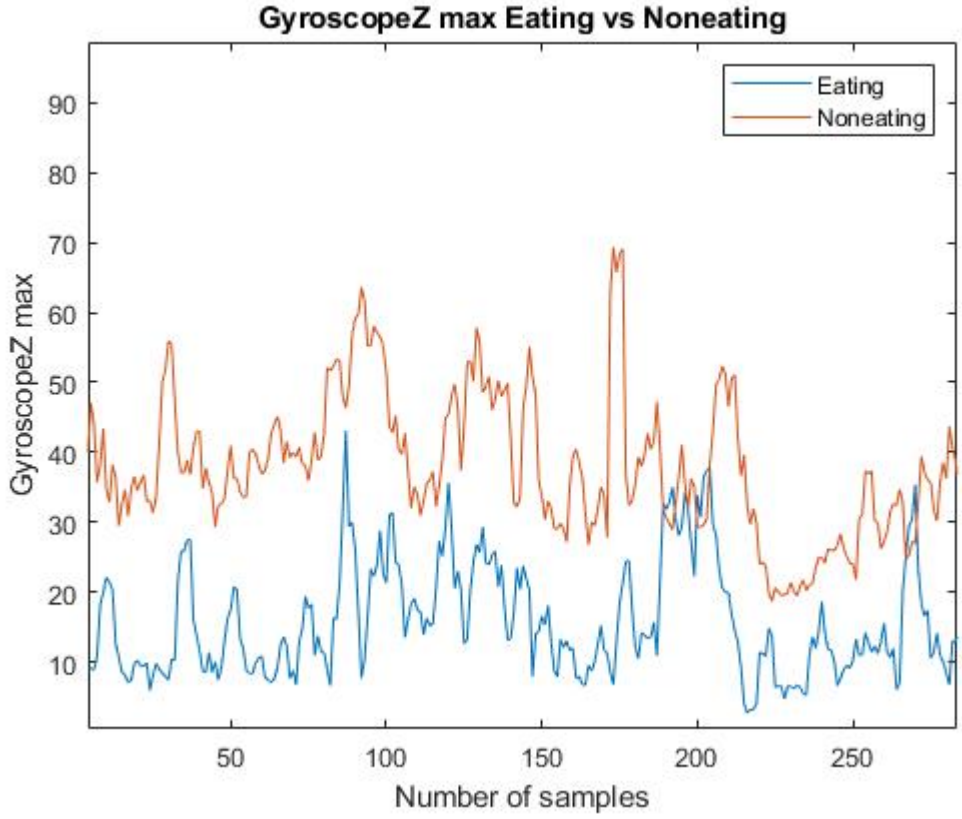

EMG6 max Eating vs Noneating
Eating
Noneating
EMG6 max
0
10
20
30
40
50
160
180
200
220
240
260
280
Number of samples


EMG5 max Eating vs Noneating
Eating
Noneating
EMG5 max
0
10
20
30
40
50
60
70
350
400
450
500
550
600
650
700
Number of samples


EMG8 max Eating vs Noneating
Eating
Noneating
EMG8 max
Number of samples
0
10
20
30
40
50
60
70
80
90
50
100
150
200
250
300


GyroscopeX max Eating vs Noneating
Eating
Noneating
GyroscopeX max
Number of samples
-20
0
20
40
60
80
100
120
0
50
100
150
200
250
300


GyroscopeY max Eating vs Noneating
Eating
Noneating
GyroscopeY max
Number of samples
0
20
40
60
80
100
120
140
160
180
350
400
450
500
550
600
650
700


GyroscopeZ max Eating vs Noneating
Eating
Noneating
GyroscopeZ max
Number of samples
10
20
30
40
50
60
70
80
90
50
100
150
200
250

5. **Entropy**

   a. **Explanation on how the feature is computed**

   The equation for entropy arises from equations of power of a data sequence.

$$H = -\sum_{m=1}^{N} P(m) \log_2 P(m) \quad [8]$$

   Normalizing

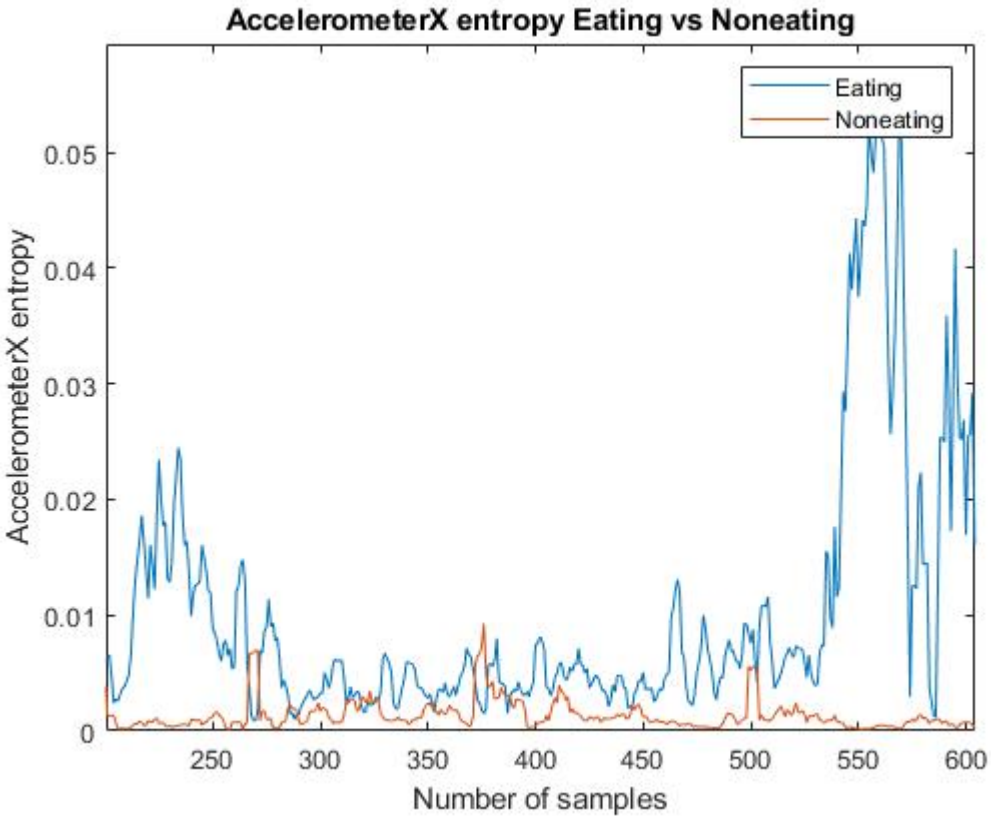
$$H_n = -\frac{\sum_{m=1}^{N} P(m) \log_2 P(m)}{\log_2 N}$$

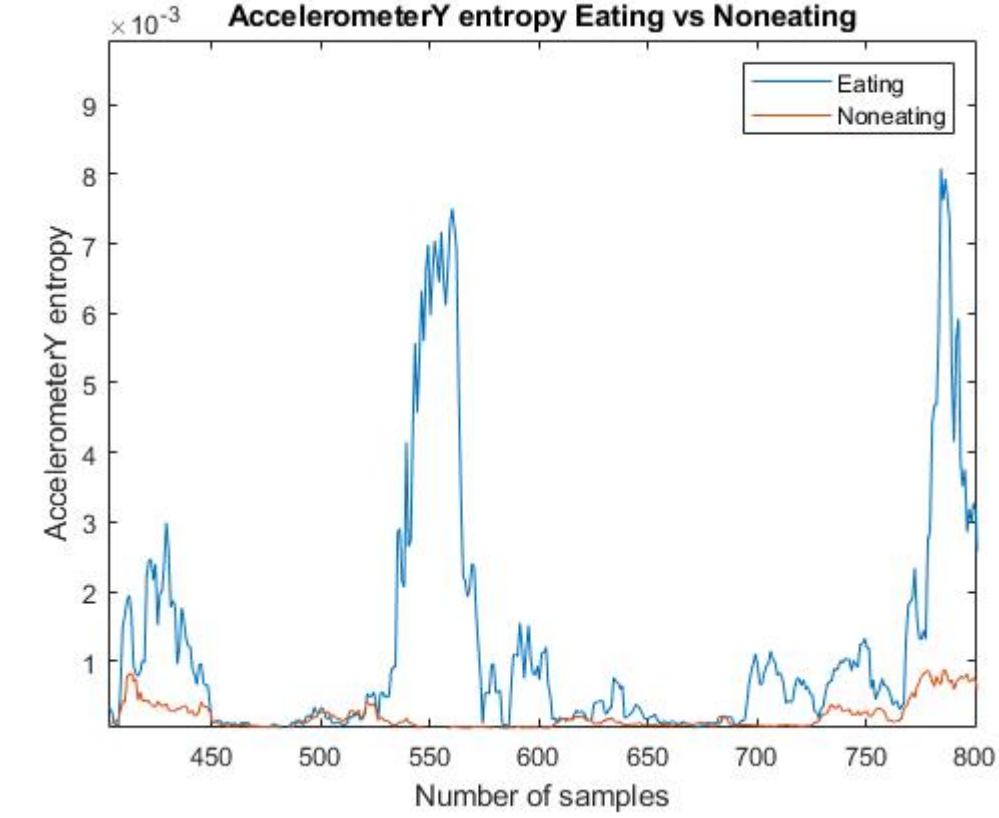
   where *N* is the total frequency points.

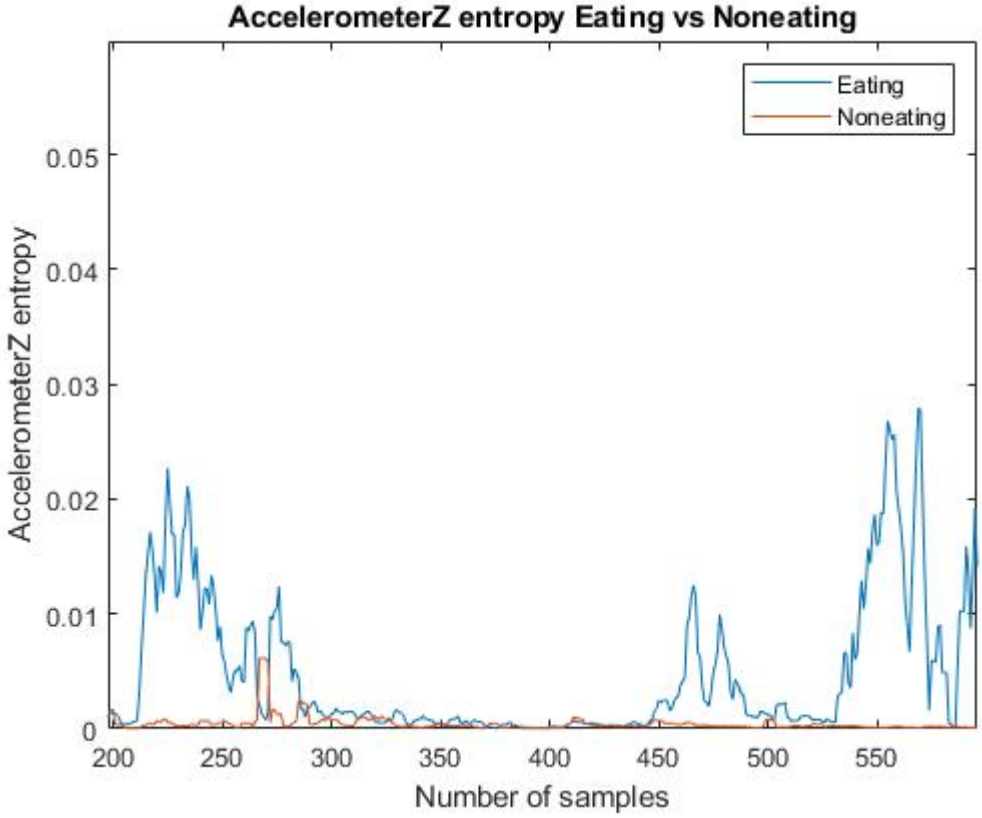
   b. **The intuition behind using Entropy**

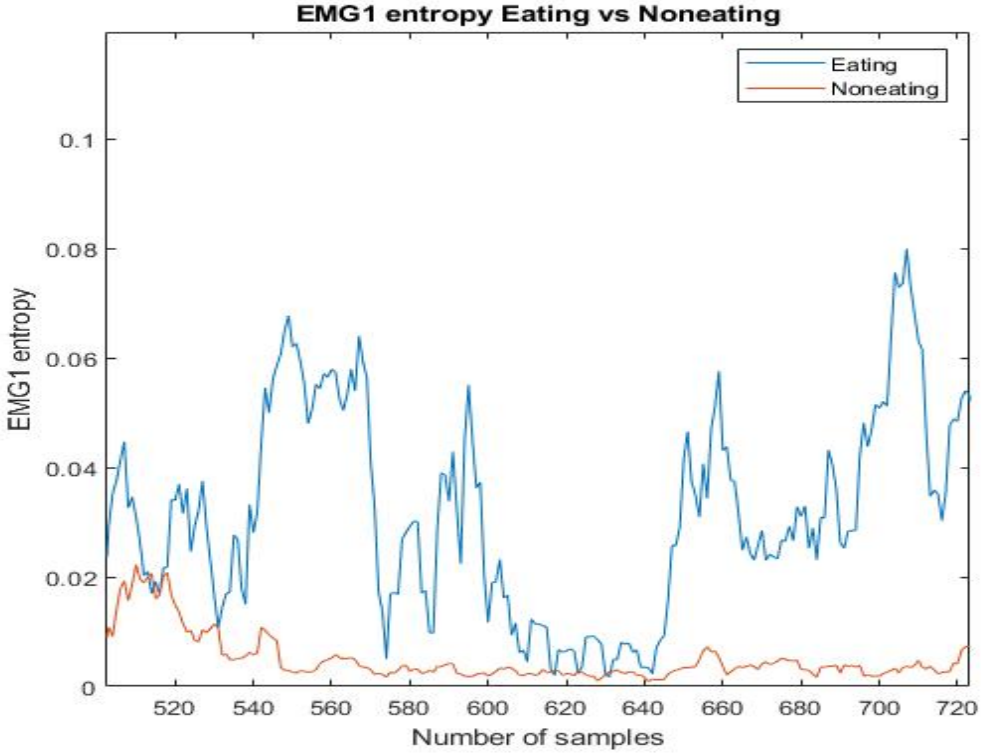
   Entropy is a statistical measure of randomness that can be used to distinguish between the eating and the non-eating activity in the dataset.

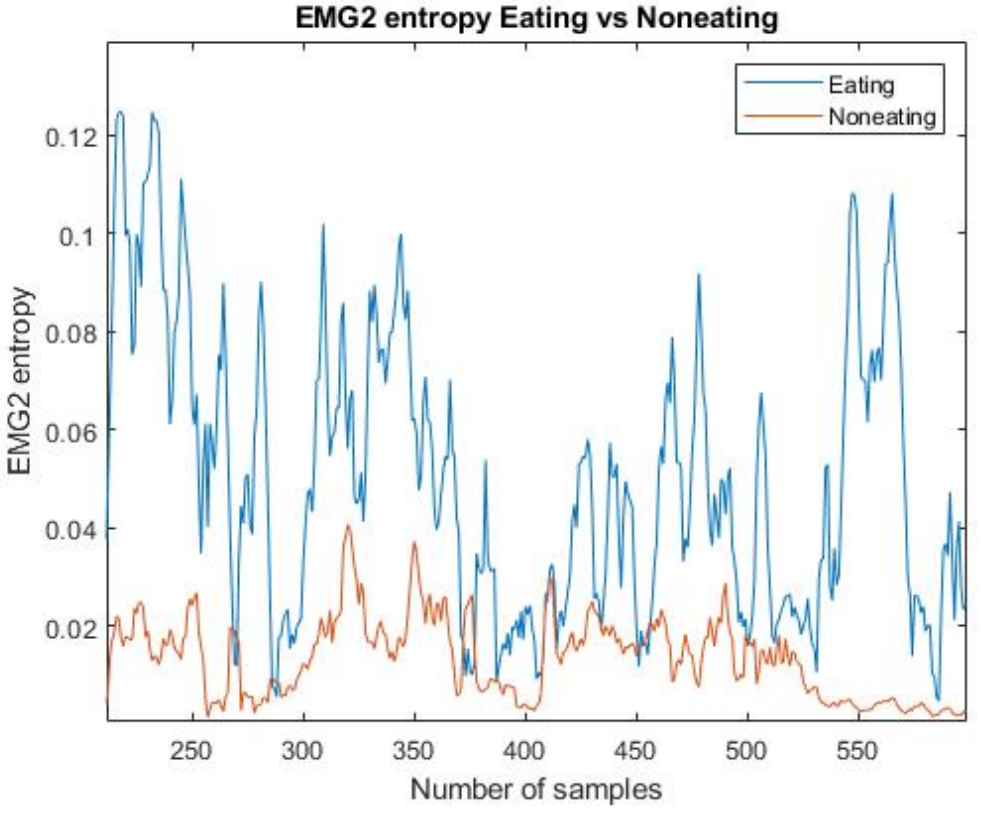
   c. **The plots for the eating and non-eating actions, using entropy as a feature**

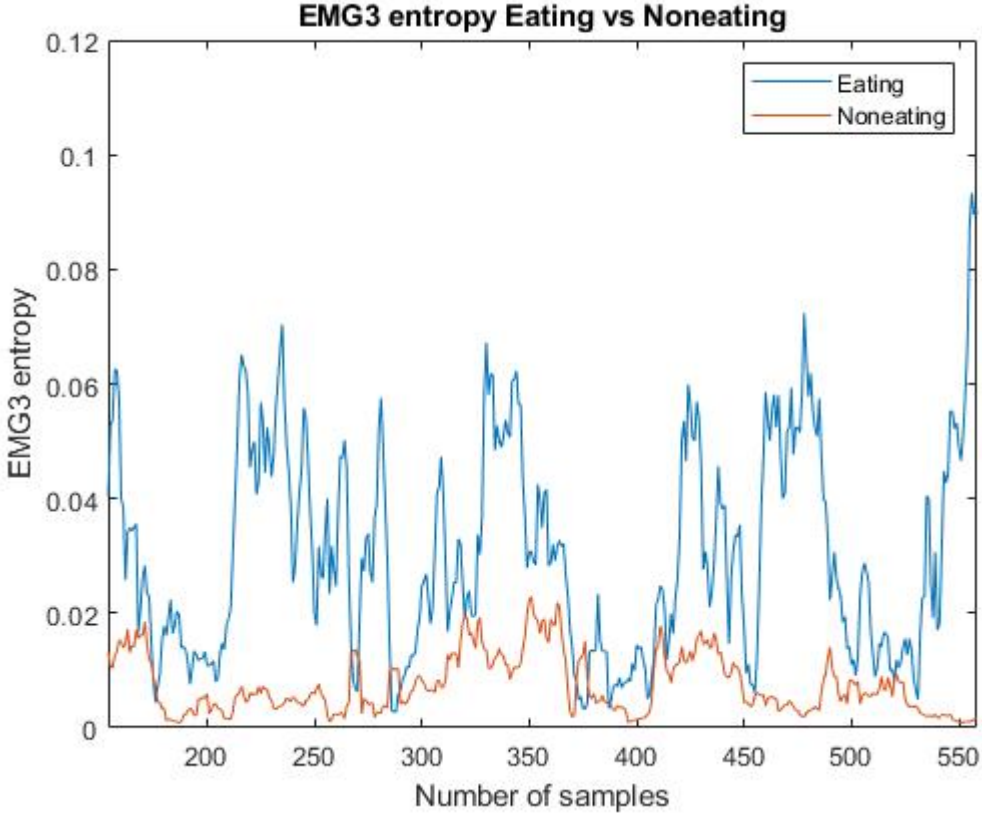


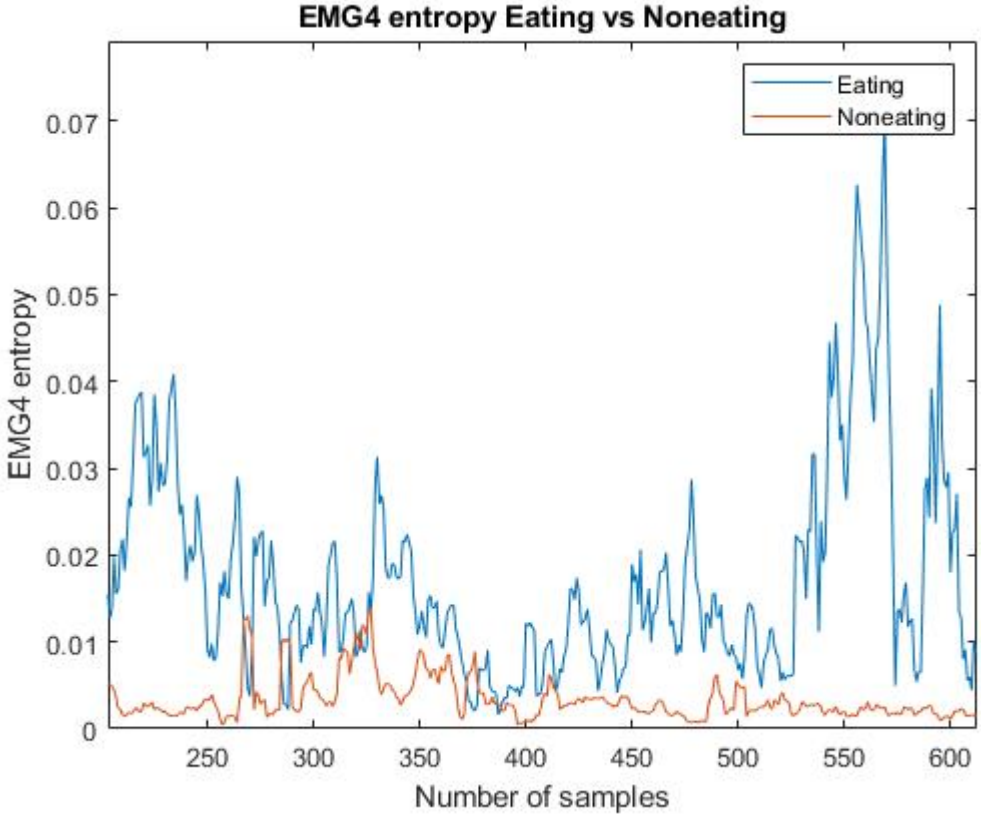

AccelerometerY entropy Eating vs Noneating
×10^-3
Eating
Noneating
AccelerometerY entropy
Number of samples

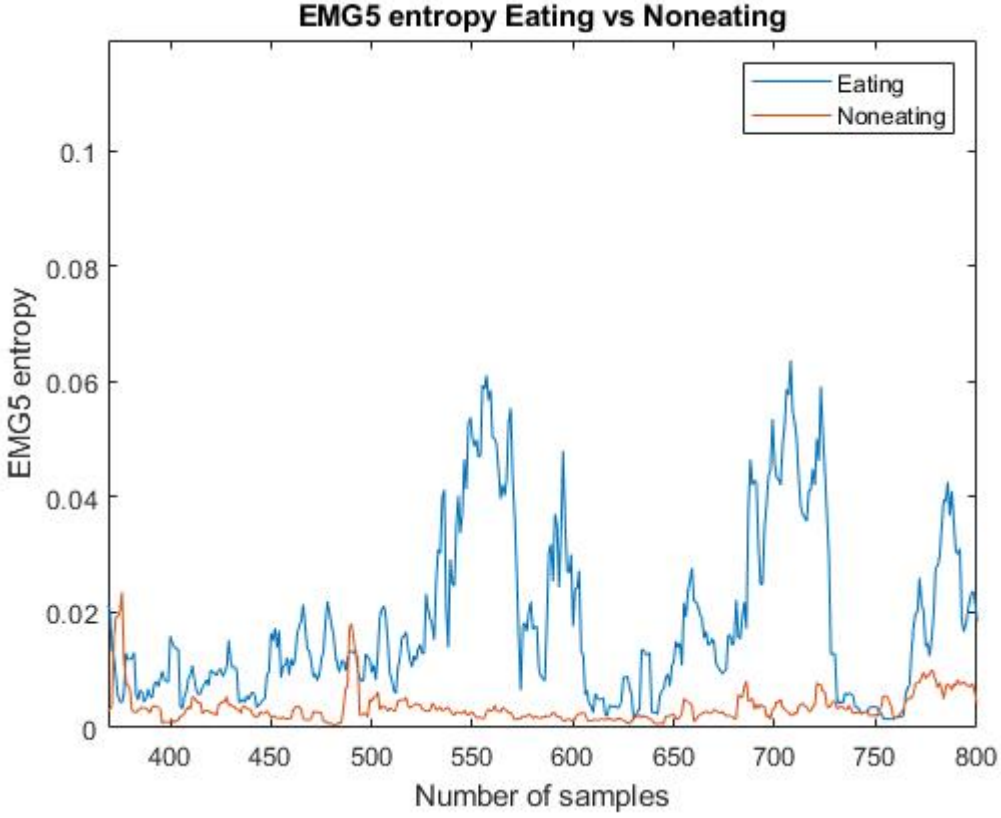

AccelerometerZ entropy Eating vs Noneating
Eating
Noneating
AccelerometerZ entropy
Number of samples

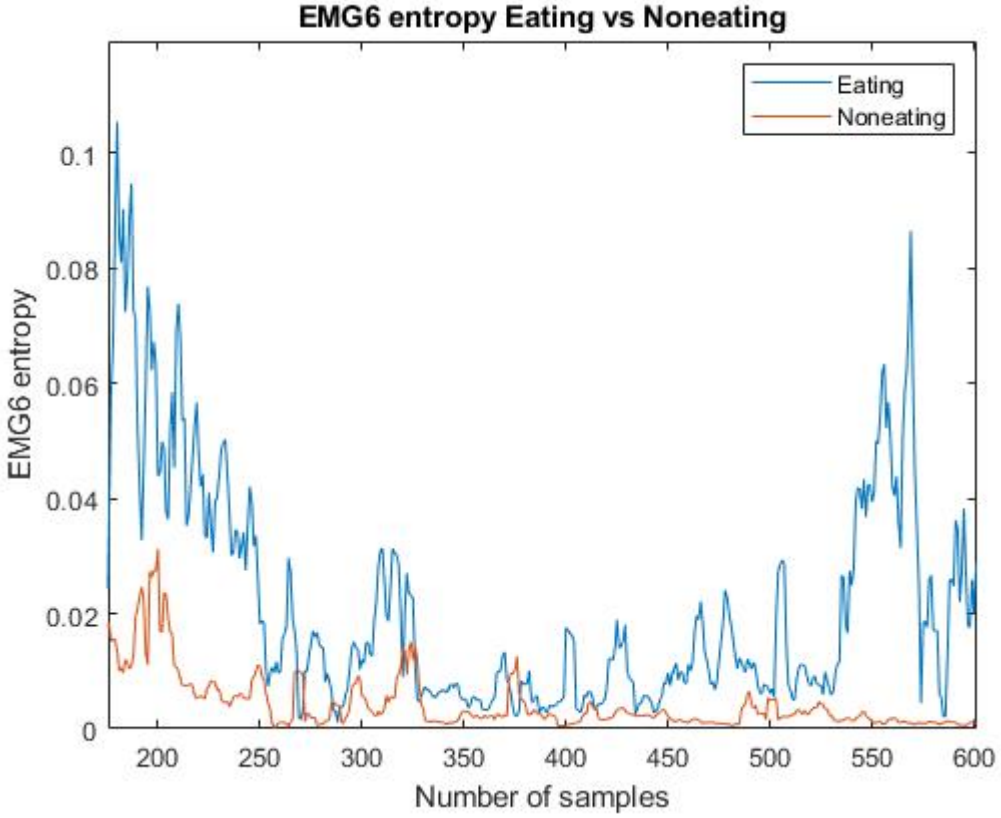

EMG1 entropy Eating vs Noneating
Eating
Noneating
EMG1 entropy
0.1
0.08
0.06
0.04
0.02
0
520
540
560
580
600
620
640
660
680
700
720
Number of samples

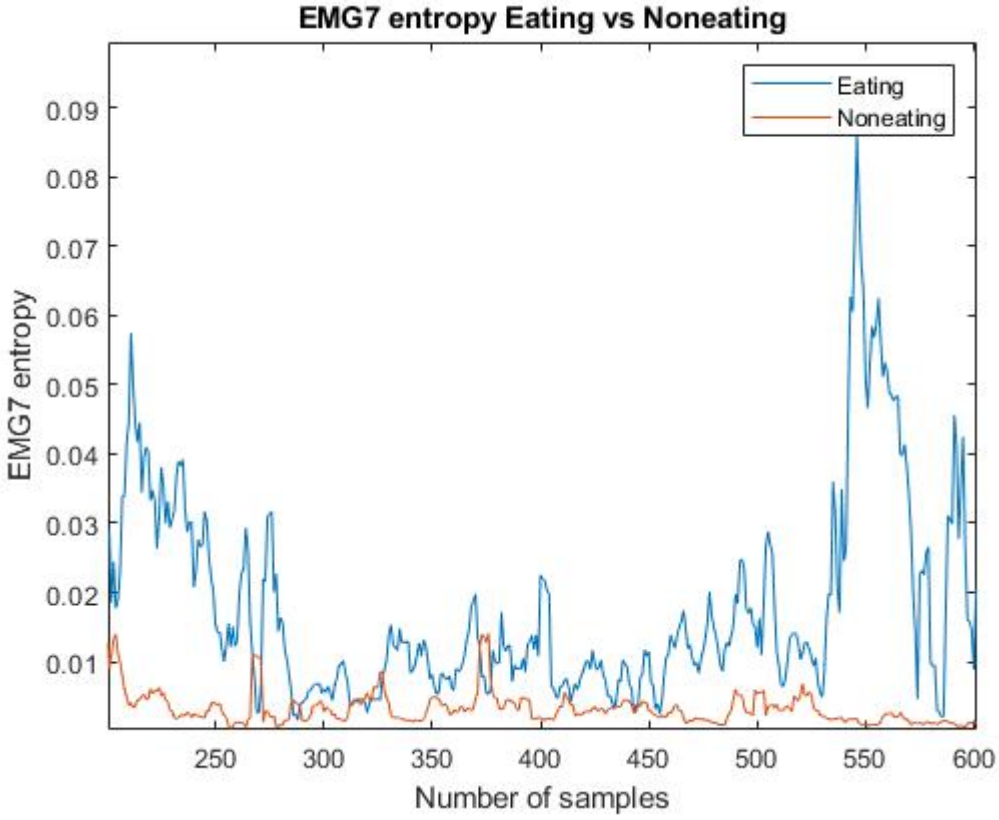

EMG2 entropy Eating vs Noneating
Eating
Noneating
EMG2 entropy
0.12
0.1
0.08
0.06
0.04
0.02
250
300
350
400
450
500
550
Number of samples

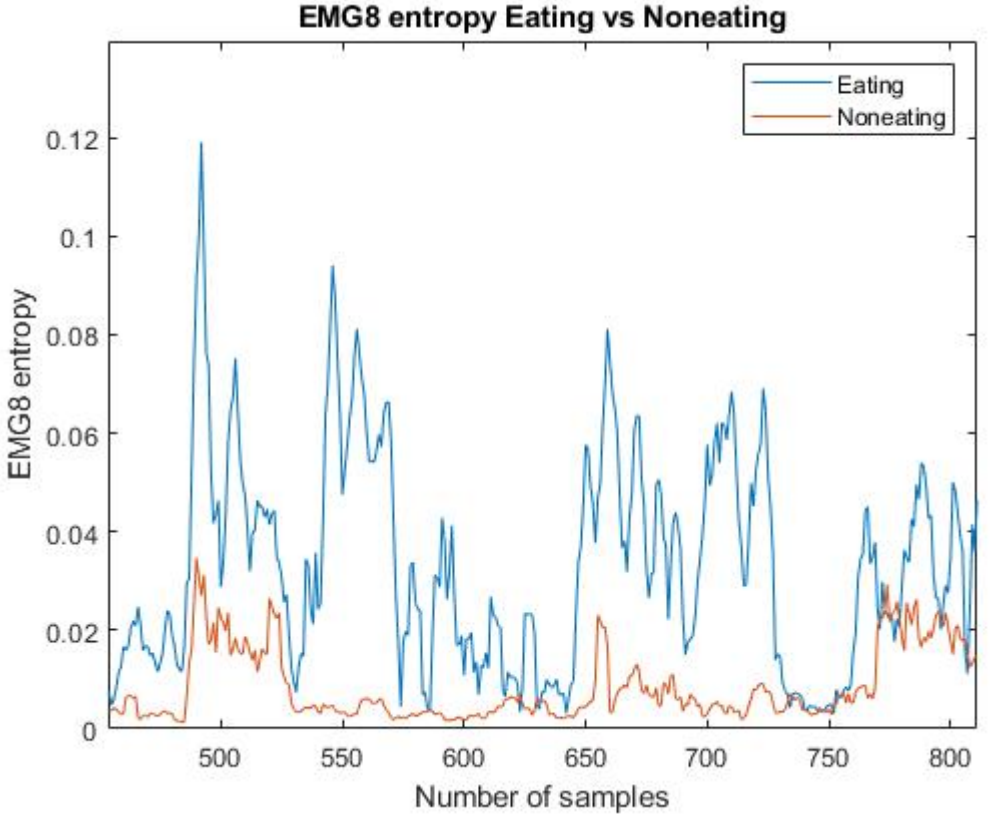

EMG3 entropy Eating vs Noneating
Eating
Noneating
EMG3 entropy
0.12
0.1
0.08
0.06
0.04
0.02
0
200
250
300
350
400
450
500
550
Number of samples

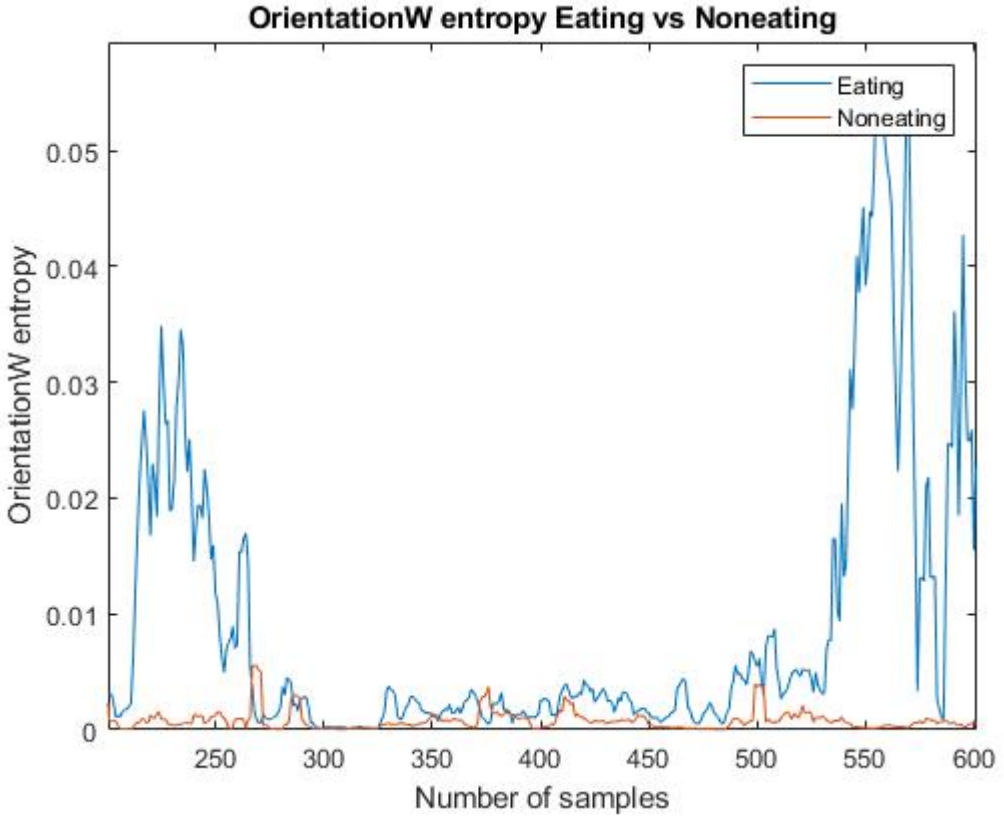

EMG4 entropy Eating vs Noneating
Eating
Noneating
EMG4 entropy
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
250
300
350
400
450
500
550
600
Number of samples

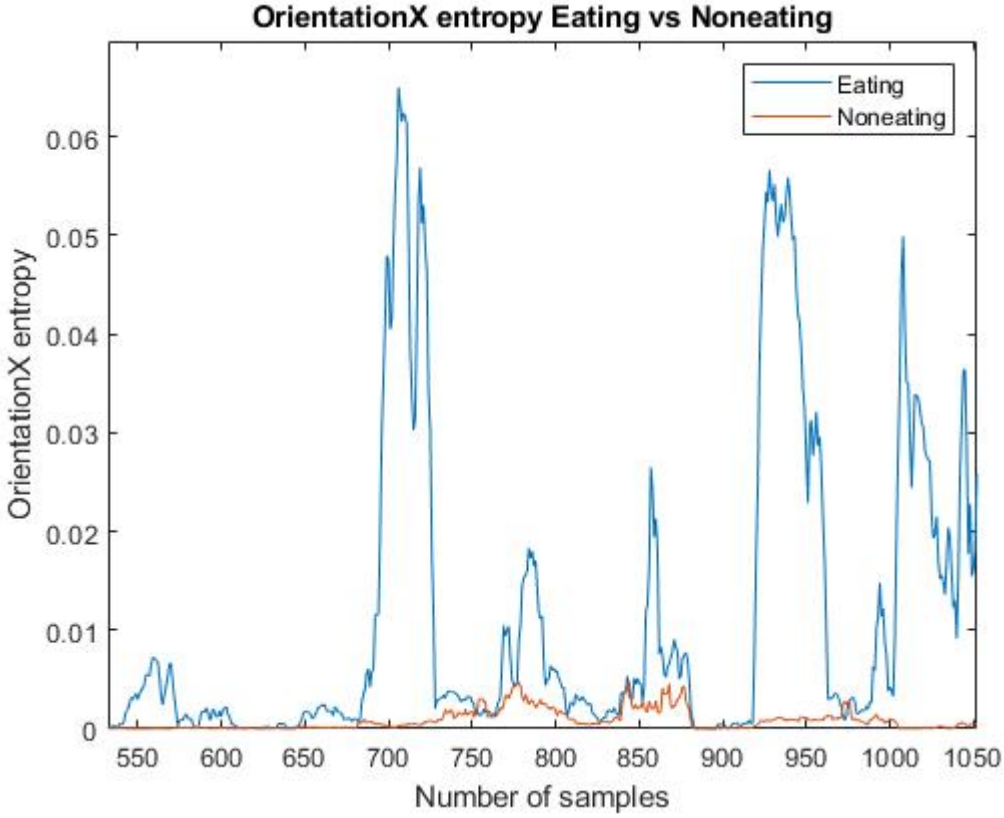

EMG5 entropy Eating vs Noneating
Eating
Noneating
EMG5 entropy
0
0.02
0.04
0.06
0.08
0.1
400
450
500
550
600
650
700
750
800
Number of samples

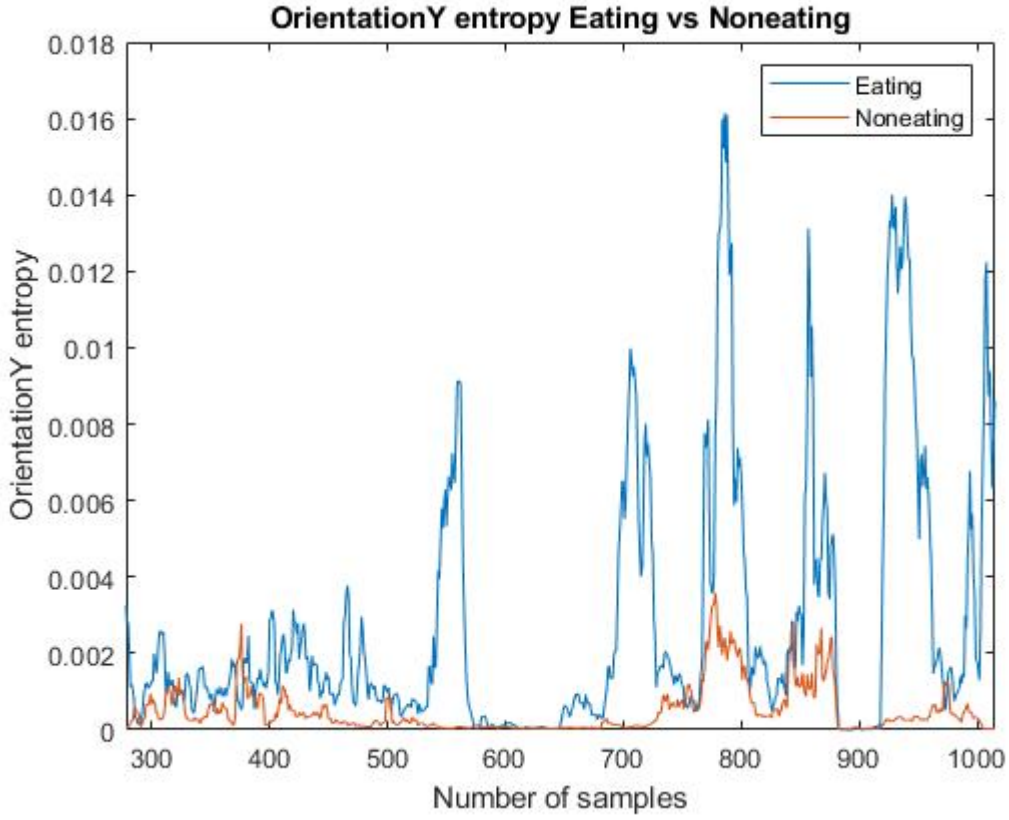

EMG6 entropy Eating vs Noneating
Eating
Noneating
EMG6 entropy
0
0.02
0.04
0.06
0.08
0.1
200
250
300
350
400
450
500
550
600
Number of samples

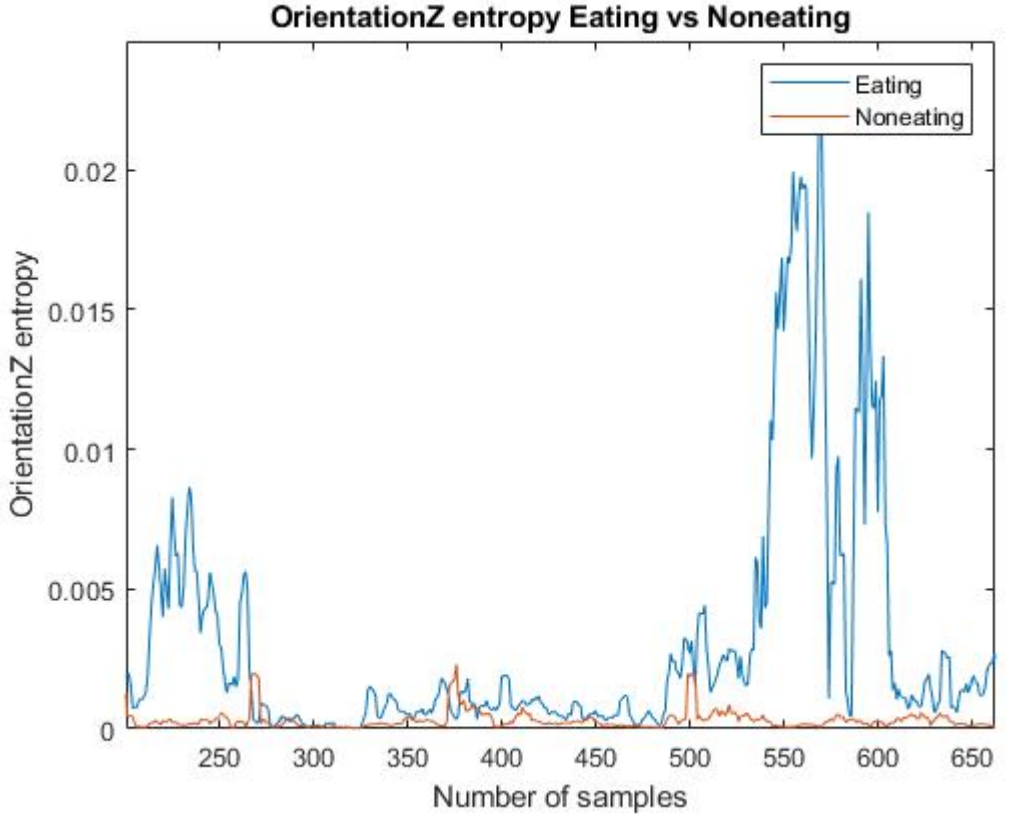

EMG7 entropy Eating vs Noneating
Eating
Noneating
EMG7 entropy
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
250
300
350
400
450
500
550
600
Number of samples


EMG8 entropy Eating vs Noneating
Eating
Noneating
EMG8 entropy
0.12
0.1
0.08
0.06
0.04
0.02
0
500
550
600
650
700
750
800
Number of samples


OrientationW entropy Eating vs Noneating
Eating
Noneating
OrientationW entropy
0
0.01
0.02
0.03
0.04
0.05
250
300
350
400
450
500
550
600
Number of samples


OrientationX entropy Eating vs Noneating
Eating
Noneating
OrientationX entropy
0
0.01
0.02
0.03
0.04
0.05
0.06
550
600
650
700
750
800
850
900
950
1000
1050
Number of samples


OrientationY entropy Eating vs Noneating
Eating
Noneating
OrientationY entropy
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
300
400
500
600
700
800
900
1000
Number of samples


OrientationZ entropy Eating vs Noneating
Eating
Noneating
OrientationZ entropy
0.02
0.015
0.01
0.005
0
250
300
350
400
450
500
550
600
650
Number of samples

6. **Power**

   a. **Explanation on how the feature is computed**

   The power of a sequence is computed using the equation

$$P(m) = \frac{S(m)}{\sum_i S(i)} \quad [8]$$

   where the power spectrum is given by

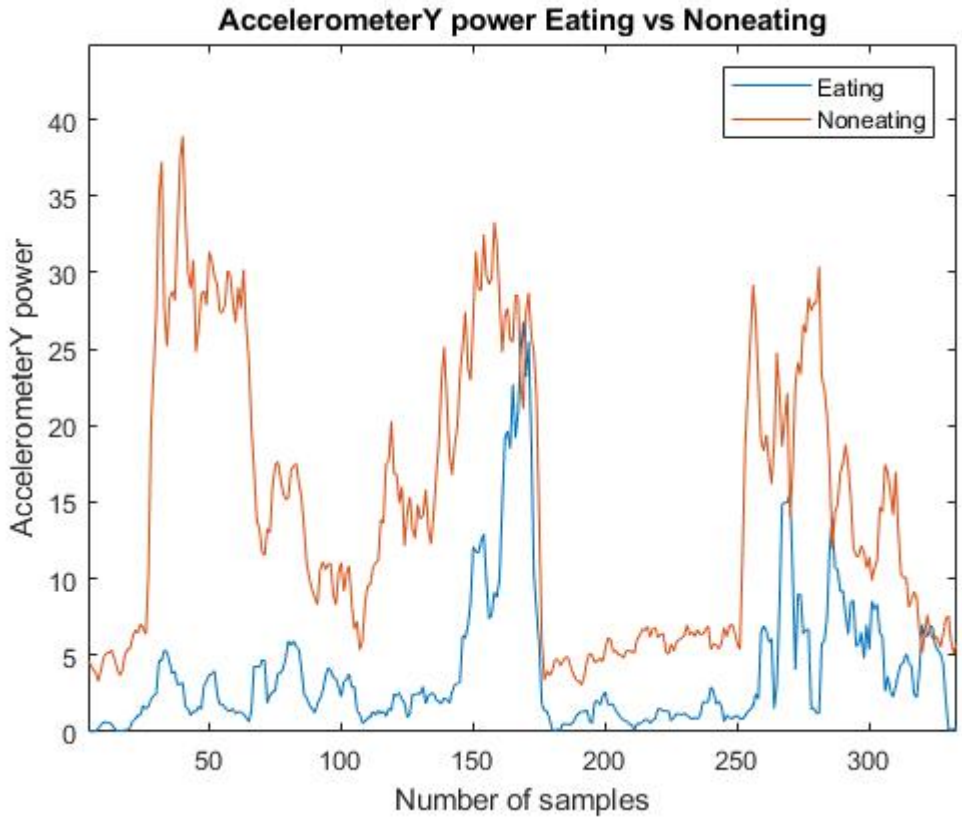
$$S(m) = |X(m)|^2$$

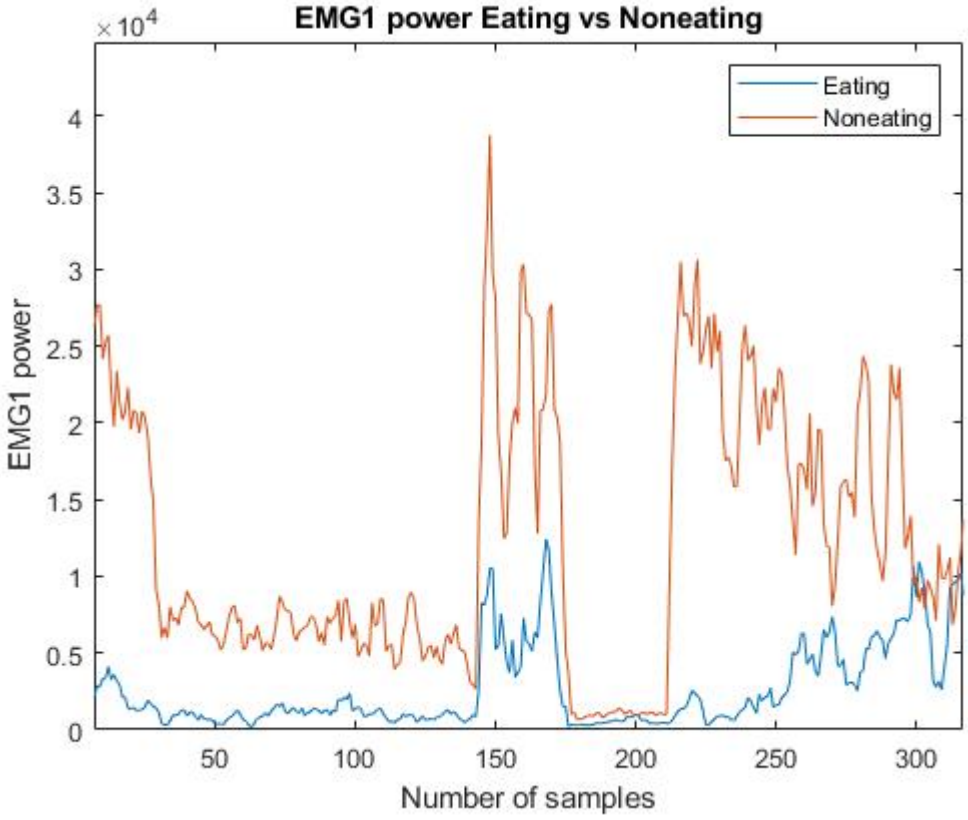
   where *X(m)* is the discrete Fourier transform of *x(n)*.

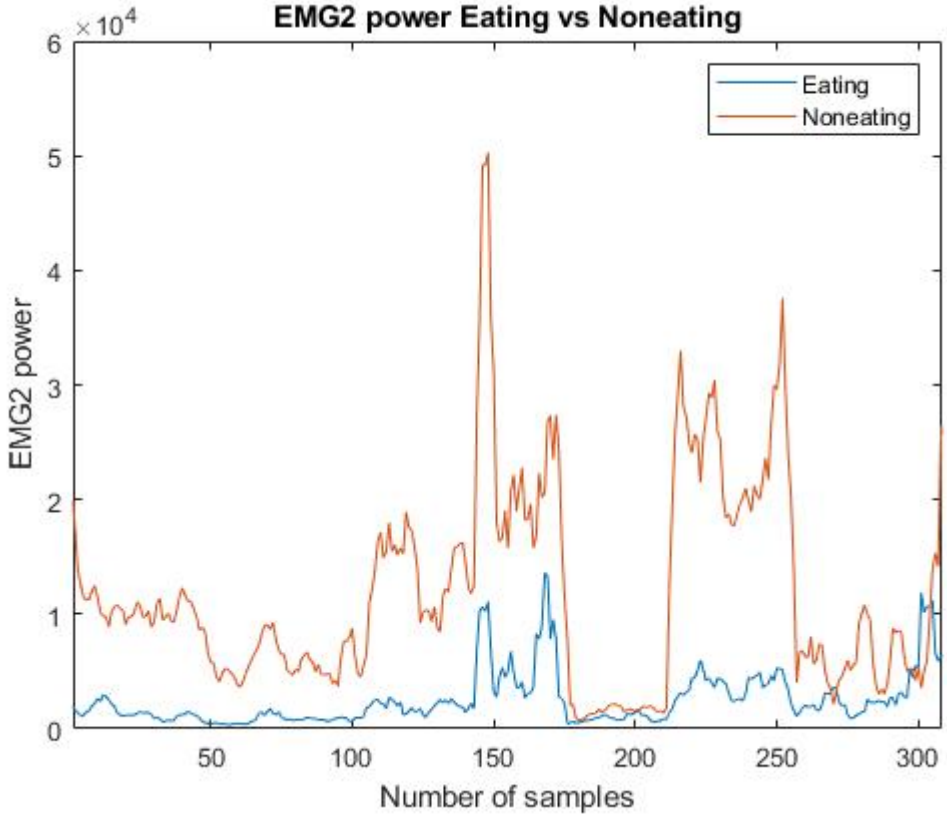
   b. **The intuition behind using Power**

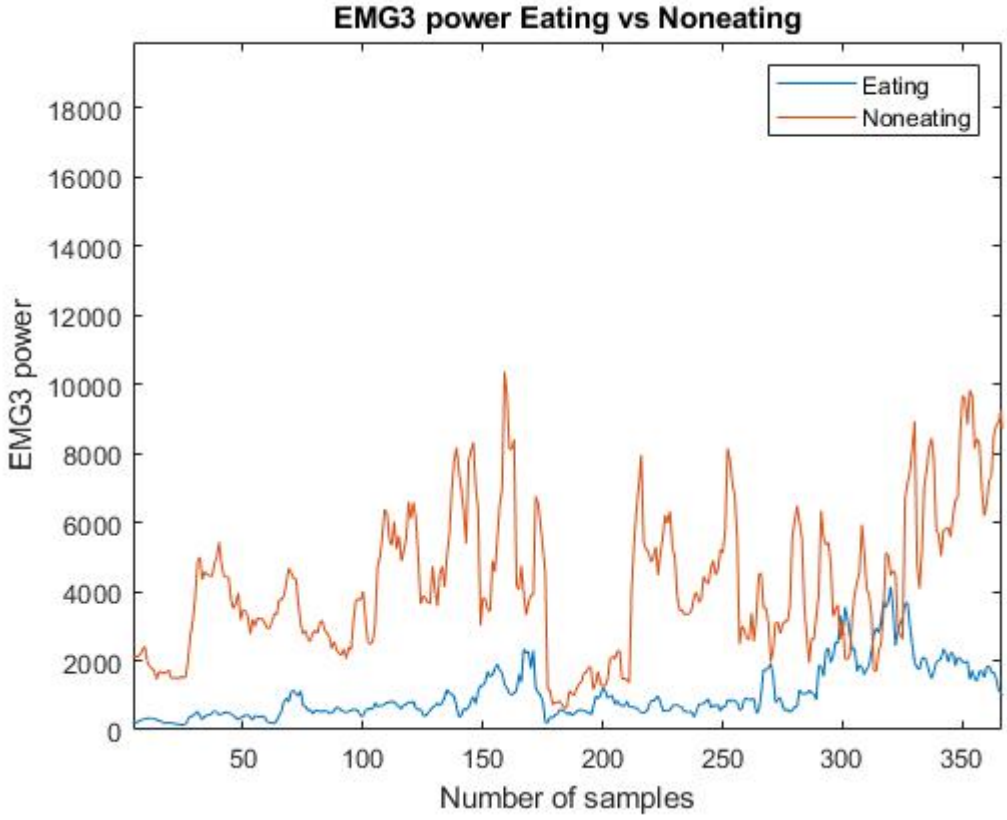
   The reason for using power, is that the peak values obtained from the sum of the squares of coefficients of the FFT help us distinguish in the eating and non-eating activities by capturing the peak effort needed for carrying out a particular activity (whether eating or non-eating) for a particular time instance.

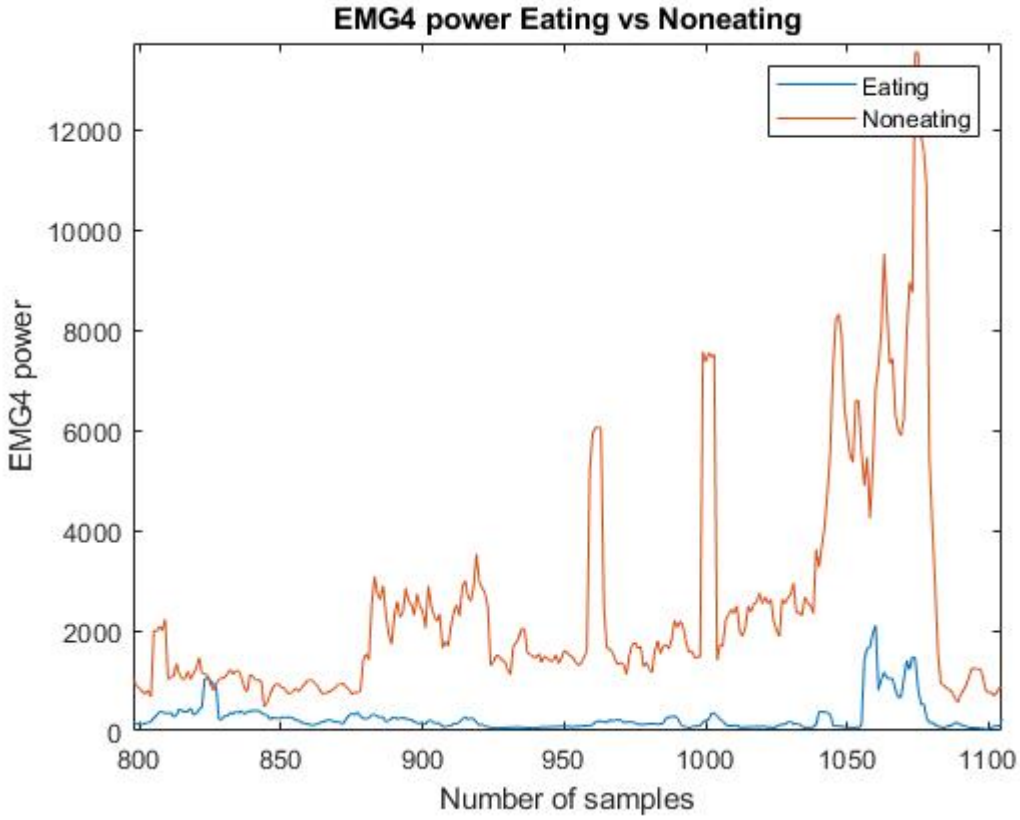
   c. **The plots for the eating and non-eating actions, using power as a feature**

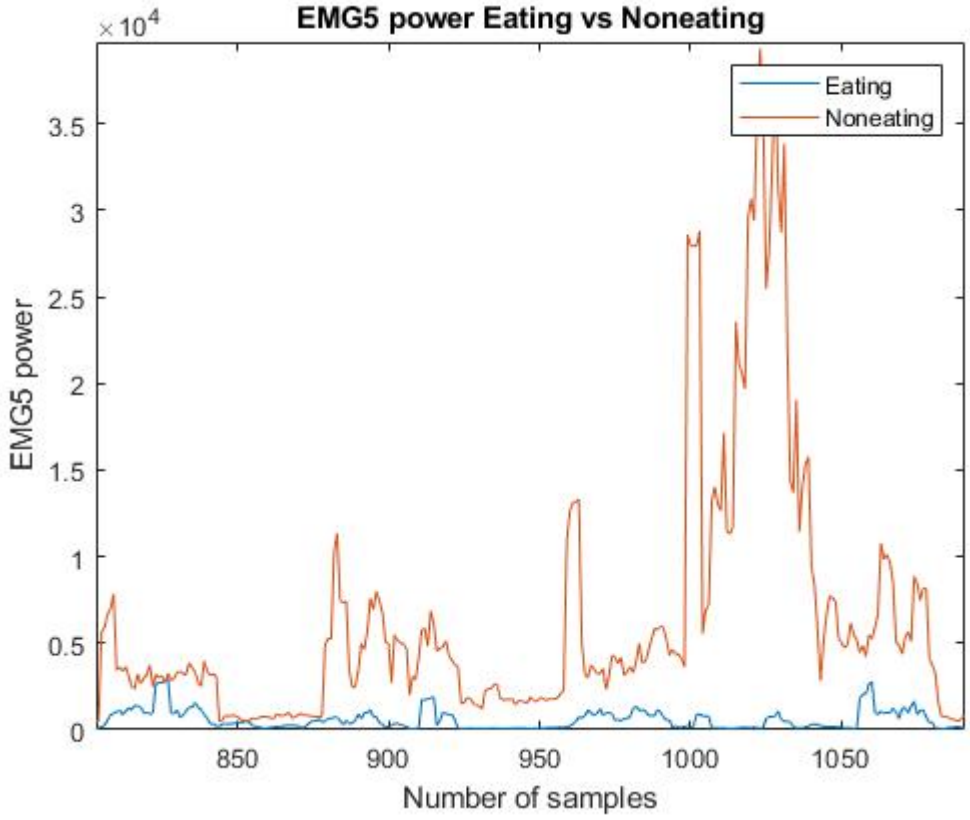


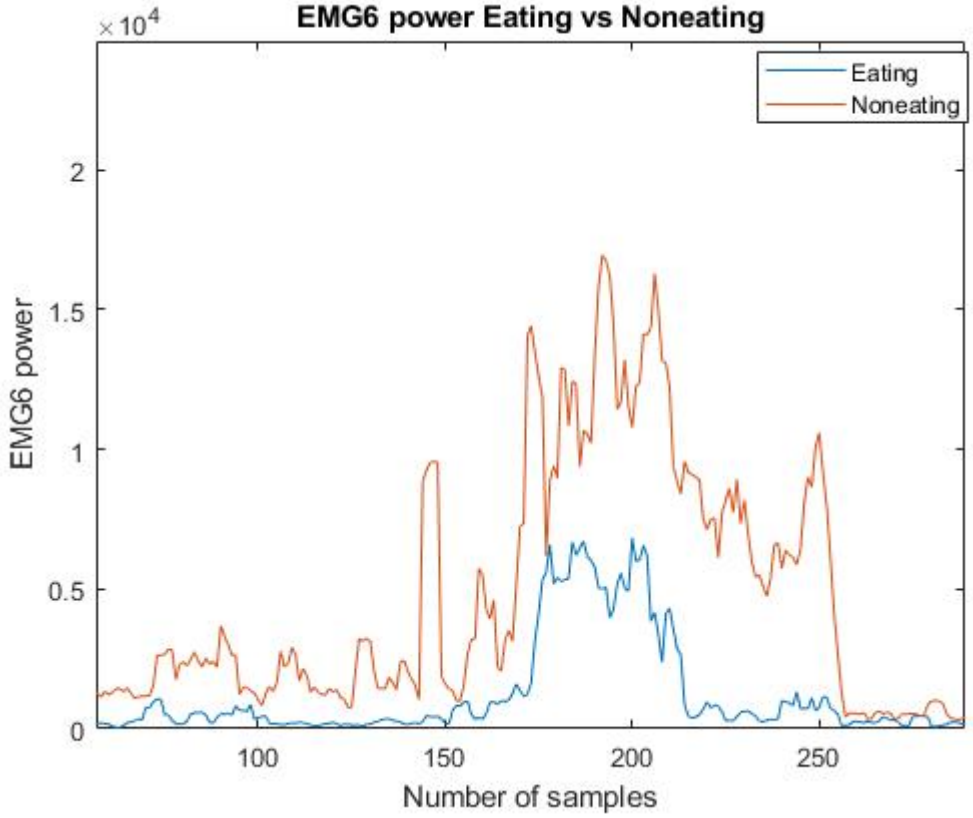

EMG1 power Eating vs Noneating
×10^4
Eating
Noneating
EMG1 power
Number of samples
0
0.5
1
1.5
2
2.5
3
3.5
4
50
100
150
200
250
300

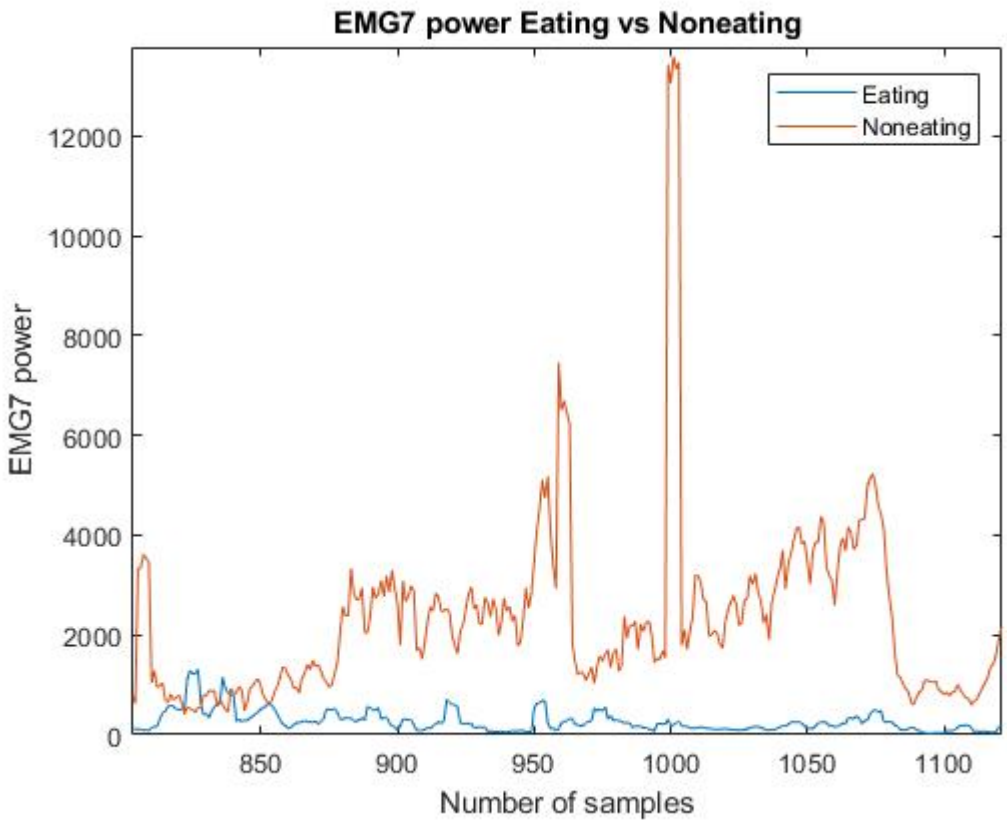

EMG2 power Eating vs Noneating
×10^4
Eating
Noneating
EMG2 power
Number of samples
0
1
2
3
4
5
6
50
100
150
200
250
300

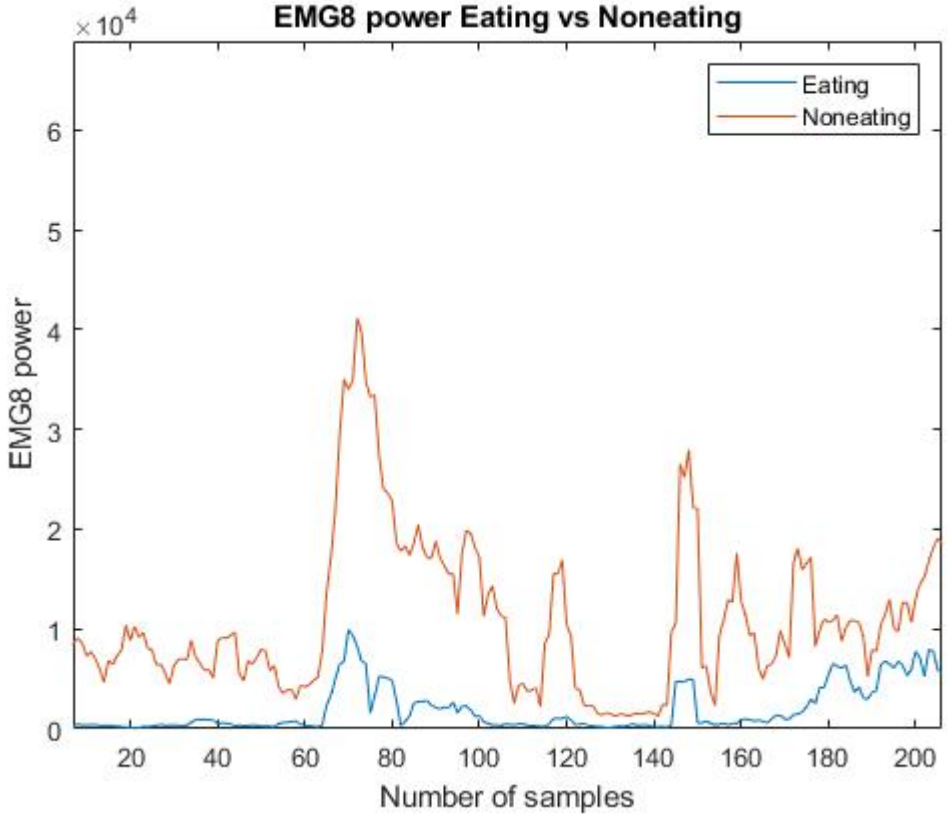

EMG3 power Eating vs Noneating
Eating
Noneating
EMG3 power
Number of samples
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
50
100
150
200
250
300
350

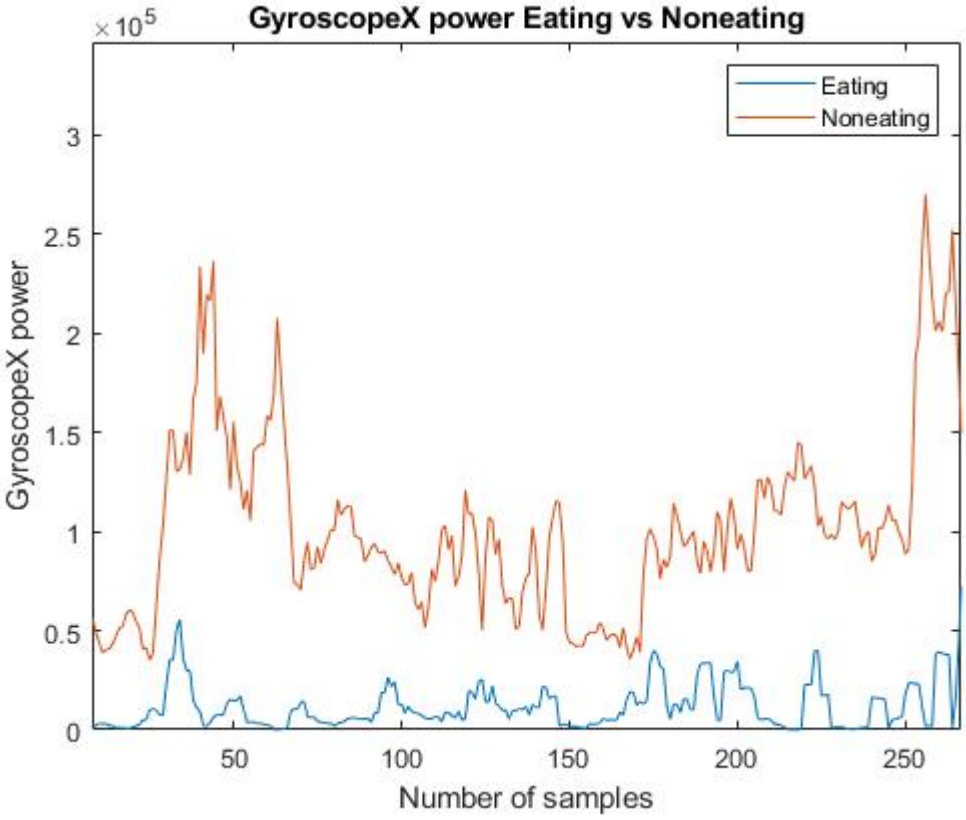

EMG4 power Eating vs Noneating
Eating
Noneating
EMG4 power
Number of samples
0
2000
4000
6000
8000
10000
12000
800
850
900
950
1000
1050
1100

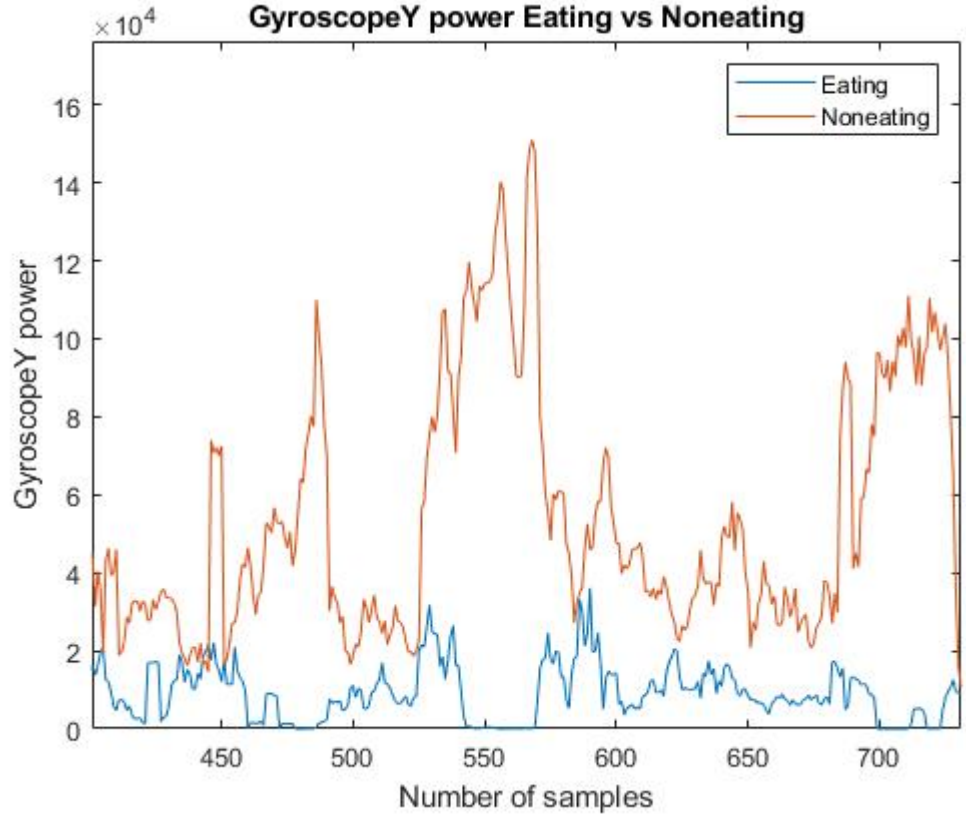

EMG5 power Eating vs Noneating
×10^4
Eating
Noneating
EMG5 power
Number of samples
0
0.5
1
1.5
2
2.5
3
3.5
850
900
950
1000
1050

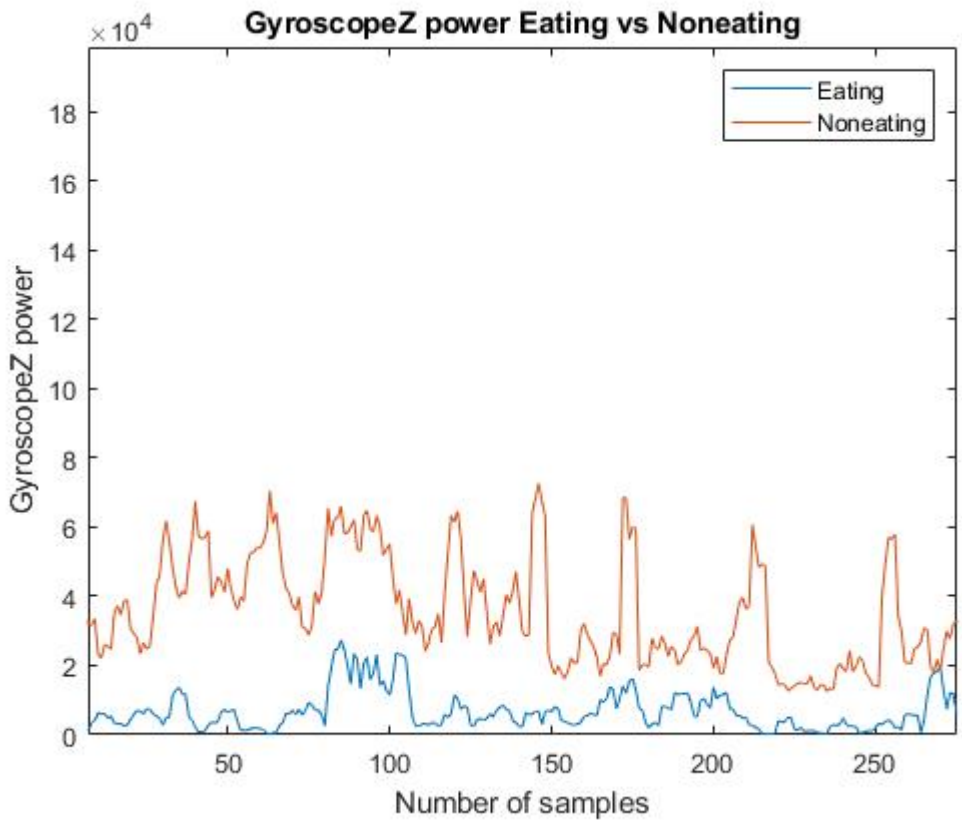

EMG6 power Eating vs Noneating
×10^4
Eating
Noneating
EMG6 power
Number of samples
0
0.5
1
1.5
2
100
150
200
250


EMG7 power Eating vs Noneating
Eating
Noneating
EMG7 power
0
2000
4000
6000
8000
10000
12000
850
900
950
1000
1050
1100
Number of samples


EMG8 power Eating vs Noneating
×10^4
Eating
Noneating
EMG8 power
0
1
2
3
4
5
6
20
40
60
80
100
120
140
160
180
200
Number of samples


GyroscopeX power Eating vs Noneating
×10^5
Eating
Noneating
GyroscopeX power
Number of samples
0
0.5
1
1.5
2
2.5
3
50
100
150
200
250


GyroscopeY power Eating vs Noneating
×10^4
Eating
Noneating
GyroscopeY power
Number of samples
0
2
4
6
8
10
12
14
16
450
500
550
600
650
700




```
feature_eat_matrix = [];
feature_noneat_matrix = [];

prev = 0;
curr=0;
t_mat = [];

% Compute the feature matrix. 6 features were considered for this
% assignment. 'mean', 'max', 'rms', 'std', 'entropy', 'power'
for i = 1:size(samples_matrix,2)
   curr = samples_matrix(3,i);
   if (curr == 0 && prev == 0) || (curr==1 && prev==1)
      t_mat = [t_mat samples_matrix(4:end,i)];
   elseif curr==0 && prev==1
      fft_feature = abs(fft(t_mat,[],2));
      power = fft_feature.*conj(fft_feature)/size(t_mat,2);
      total_power = sum(power,2);

      p=power;
      p=p/sum(p+ 1e-12);
      logd = log2(p + 1e-12);
      entropy = -sum(p.*logd,2)/log2(length(p));

      t_mat = vertcat(mean(t_mat,2), max(t_mat,[],2), rms(t_mat,2), std(t_mat,[],2), entropy, total_power);
      feature_eat_matrix = [feature_eat_matrix t_mat];
      t_mat=samples_matrix(4:end,i);
   else
```

```
        fft_feature = abs(fft(t_mat,[],2));
        power = fft_feature.*conj(fft_feature)/size(t_mat,2);
        total_power = sum(power,2);

        p=power;
        p=p/sum(p+ 1e-12);
        logd = log2(p + 1e-12);
        entropy = -sum(p.*logd,2)/log2(length(p));

        t_mat = vertcat(mean(t_mat,2), max(t_mat,[],2), rms(t_mat,2), std(t_mat,[],2), entropy, total_power);
        feature_noneat_matrix = [feature_noneat_matrix t_mat];
        t_mat=samples_matrix(4:end,i);
    end
    prev=curr;
end
row_labels = {'OrientationX mean','OrientationY mean','OrientationZ mean','OrientationW mean','AccelerometerX
mean','AccelerometerY mean','AccelerometerZ mean','GyroscopeX mean','GyroscopeY mean','GyroscopeZ mean','EMG1
mean','EMG2 mean','EMG3 mean','EMG4 mean','EMG5 mean','EMG6 mean','EMG7 mean','EMG8 mean','OrientationX
max','OrientationY max','OrientationZ max','OrientationW max','AccelerometerX max','AccelerometerY max','AccelerometerZ
max','GyroscopeX max','GyroscopeY max','GyroscopeZ max','EMG1 max','EMG2 max','EMG3 max','EMG4 max','EMG5
max','EMG6 max','EMG7 max','EMG8 max','OrientationX rms','OrientationY rms','OrientationZ rms','OrientationW
rms','AccelerometerX rms','AccelerometerY rms','AccelerometerZ rms','GyroscopeX rms','GyroscopeY rms','GyroscopeZ
rms','EMG1 rms','EMG2 rms','EMG3 rms','EMG4 rms','EMG5 rms','EMG6 rms','EMG7 rms','EMG8 rms','OrientationX
std','OrientationY std','OrientationZ std','OrientationW std','AccelerometerX std','AccelerometerY std','AccelerometerZ
std','GyroscopeX std','GyroscopeY std','GyroscopeZ std','EMG1 std','EMG2 std','EMG3 std','EMG4 std','EMG5 std','EMG6
std','EMG7 std','EMG8 std','OrientationX entropy','OrientationY entropy','OrientationZ entropy','OrientationW
entropy','AccelerometerX entropy','AccelerometerY entropy','AccelerometerZ entropy','GyroscopeX entropy','GyroscopeY
entropy','GyroscopeZ entropy','EMG1 entropy','EMG2 entropy','EMG3 entropy','EMG4 entropy','EMG5 entropy','EMG6
entropy','EMG7 entropy','EMG8 entropy','OrientationX power','OrientationY power','OrientationZ power','OrientationW
power','AccelerometerX power','AccelerometerY power','AccelerometerZ power','GyroscopeX power','GyroscopeY
power','GyroscopeZ power','EMG1 power','EMG2 power','EMG3 power','EMG4 power','EMG5 power','EMG6 power','EMG7
power','EMG8 power'};
samples_labels = ['user', 'time_stamp', 'ground_truth',
"OrientationX","OrientationY","OrientationZ","OrientationW","AccelerometerX","AccelerometerY","AccelerometerZ","Gyrosco
peX","GyroscopeY","GyroscopeZ","EMG1","EMG2","EMG3","EMG4","EMG5","EMG6","EMG7","EMG8"];

% Refer to the below tables for getting a visual sense of the samples, feature
% matrices. Warning: Doesn't contain all observations. Just for visualization
% purpose.
feature_eat_matrix_table  = array2table(feature_eat_matrix,'RowNames', row_labels);
feature_noneat_matrix_table  = array2table(feature_noneat_matrix,'RowNames', row_labels);
samples_table = array2table(samples_matrix(:,1:2000),'RowNames',samples_labels);
```

Variables – feature_eat_matrix_table

feature_eat_matrix_table

108x1160 table

	1 feature_eat_matrix1	2 feature_eat_matrix2	3 feature_eat_matrix3	4 feature_eat_matrix4
1 OrientationX mean	-0.7900	-0.8150	-0.7895	-0.72(
2 OrientationY mean	-0.5971	-0.5132	-0.4861	-0.47:
3 OrientationZ mean	-0.0100	0.1008	0.2152	0.26
4 OrientationW mean	0.1374	0.2495	0.3067	0.42:
5 AccelerometerX mean	0.9539	0.8996	0.9094	0.92:
6 AccelerometerY mean	-0.1747	-0.0868	0.0490	-0.00
7 AccelerometerZ mean	-0.2349	-0.4059	-0.3892	-0.36:
8 GyroscopeX mean	3.3770	-0.6653	-3.3975	-2.07:
9 GyroscopeY mean	2.6043	0.8235	-1.6713	0.65
10 GyroscopeZ mean	2.0608	1.1204	1.8279	0.27:
11 EMG1 mean	-1.0430	-1.1914	0.4979	-1.14
12 EMG2 mean	0.0094	-1.2121	-1.1765	-0.58
13 EMG3 mean	-0.5456	-0.6294	-0.5400	-1.35
14 EMG4 mean	-0.7726	-0.8111	-0.7396	-1.25
15 EMG5 mean	-0.7495	-0.7252	-0.7283	-0.53
16 EMG6 mean	-0.8070	-0.5809	-0.5604	-0.69
17 EMG7 mean	-0.5107	-0.7078	-0.3692	-0.76:
18 EMG8 mean	-0.8340	-0.5744	-0.4210	-1.22(
19 OrientationX max	-0.7810	-0.8110	-0.7870	-0.72
20 OrientationY max	-0.5780	-0.5100	-0.4770	-0.46
21 OrientationZ max	0.0170	0.1058	0.2190	0.27:
22 OrientationW max	0.1457	0.2550	0.3246	0.42(
23 AccelerometerX max	1.0050	0.9118	0.9476	0.94(
24 AccelerometerY max	-0.1162	-0.0695	0.0802	0.01(
25 AccelerometerZ max	-0.1857	-0.3956	-0.3639	-0.34
26 GyroscopeX max	15.8157	2.3243	1.8702	3.32
27 GyroscopeY max	33.2784	6.2276	6.4559	7.74(
28 GyroscopeZ max	21.7005	5.3189	11.3903	9.30:
29 EMG1 max	21.8234	11.4161	16.7262	14.51
30 EMG2 max	16.5935	12.8711	21.0008	19.21
31 EMG3 max	6.3293	7.0246	5.7730	3.40
32 EMG4 max	5.9958	1.0571	0.7299	0.62
33 EMG5 max	1.8548	1.0571	0.7848	
34 EMG6 max	1.8484	1.7290	0.7848	

Intuition behind selecting the features

We applied 6 feature extraction methods, across 18 features, giving us 108 observable features. In order to further identify the set of features that gave us a clear distinction between eating/non-eating actions, we used **the cosine similarity measure**. The process we followed is as follows

- Calculation of the cosine similarity for each of the 108 features for eating vs non-eating activities.
- Calculation of the mean value of all the 108 cosine similarity values.
- Selection of features showing clear distinction between eating and non-eating activities by identifying features having cosine similarity value between [0, mean_cosine_sim).

After applying the above steps, we got a final set of 43 features. Out of these features, we found out that the **power and entropy feature selection methods** can give us the clearest distinction between eating and non-eating activities since their similarity measure is low.

Thus, the **final 43 features** that we get showing the lowest similarity measure between eating and non-eating activities are as follows:

Using mean for feature extraction – EMG1 Mean, EMG2 Mean, EMG8 Mean

Using RMS for feature extraction – GyroscopeX RMS

Using STD for feature extraction – GyroscopeX STD, OrientationX STD, OrientationY STD, OrientationZ STD, OrientationW STD

Using max for feature extraction – AccelerometerY Max, EMG5 Max, EMG6 Max, EMG8 Max, GyroscopeX Max, GyroscopeY Max, GyroscopeZ Max

Using entropy for feature extraction - AccelerometerX Entropy, AccelerometerY Entropy, AccelerometerZ Entropy, EMG1 Entropy, EMG2 Entropy, EMG3 Entropy, EMG4 Entropy, EMG5 Entropy, EMG6 Entropy, EMG7 Entropy, EMG8 Entropy, OrientationX Entropy, OrientationY Entropy, OrientationZ Entropy, OrientationW Entropy

Using power for feature extraction - AccelerometerY Power, EMG1 Power, EMG2 Power, EMG3 Power, EMG4 Power, EMG5 Power, EMG6 Power, EMG7 Power, EMG8 Power, OrientationX Power, OrientationY Power, OrientationZ Power

```
% Compute the cosine similarity between the plots to get the
% dissimilar features with more distance
cosine_similarity=[];
feature_noneat_mat=feature_noneat_matrix(:,(1:1160)); % Discard one observation and compare the equal number of eating and non eating observations
num_features = size(feature_eat_matrix_table,1);
for k=1:num_features
   cosine_similarity=[cosine_similarity 1 - pdist([feature_eat_matrix(k,:); feature_noneat_mat(k,:)], 'cosine')];
end
mean_cosine_sim = mean(cosine_similarity,2);

picked_features=[];

% Pick the features with cosine similarity less(or highly dissimilar/distant) than the mean cosine
% similarity across all the observations
for k=1:num_features
   m = 1 - pdist([feature_eat_matrix(k,:); feature_noneat_mat(k,:)], 'cosine');
   if m>=0 && m<mean_cosine_sim
      t = [feature_eat_matrix(k,:) feature_noneat_matrix(k,:)];
      picked_features = vertcat(picked_features, t);
      figure('Name',row_labels{k});
      clf
      plot(1:size(feature_eat_matrix,2),smooth(feature_eat_matrix(k,:)));
      hold on;
      plot(1:size(feature_noneat_matrix,2),smooth(feature_noneat_matrix(k,:)));
      xlabel('Number of samples');
```

```
    ylabel(row_labels{k});
    legend('Eating','Noneating');
  end
end
```

Phase 3 – Feature Selection

In this phase, we had to select the features from the feature matrix generated in the previous phase which would clearly distinguish between eating and non-eating activities. We chose, Principal Component Analysis (PCA), a popular method dimensionality reduction method.

When a set of features are given as an input to PCA, it gives out the same number principal components back, but the values of these generated components are the combination of the features given as an input. After this, we select the components which show the maximum variance, and drop the one's having less variance [9]. So, basically, the variance helps us in deciding whether the feature helps in distinguishing between the eating and non-eating actions.

```
% Apply PCA over the distant features picked. Refer '>> doc pca' for more info
% how PCA works and the outputs returned
[coeff,score,latent,tsquared,explained,mu]=pca(picked_features.');
% Observe that the top 11 variances constitute to 100% variance and hence
% we can safely pick only these components
explained

% Plotting the first 3 Principal Components
figure('Name','Biplot for PC1, PC2, PC3');
biplot(coeff(:,1:3),'scores',score(:,1:3));

legend_eigen={};
figure('Name','Plot for Eigen vectors');
for i=1:11
  plot(coeff(:,i));
  hold on;
  xlabel('Number of samples');
  ylabel('Eigen Vectors for Principal Components');
  legend_eigen{i} = ['Eigen Vector ' num2str(i)];
end
legend(legend_eigen);

new_feature_mat = score;
new_feature_mat = [new_feature_mat vertcat(ones(1160,1), zeros(1161,1))];

for i=1:11
  figure('Name',['Plot of new feature matrix for PC ' num2str(i) ' with variance ' num2str(explained(i,1)) '%'])
  plot(smooth(new_feature_mat([1:1160],i)));
  hold on;
  plot(smooth(new_feature_mat([1161:2321],i)));
  xlabel('Number of samples');
  ylabel('Principle Component Values');
  legend('Eating','Noneating');
end
```

```
figures_folder_name = 'figures'; % Destination folder for storing the plots
mkdir(figures_folder_name);
figures_list = findobj(allchild(0), 'flat', 'Type', 'figure');
for i = 1:length(figures_list)
   figure_handle = figures_list(i);
   figure_name   = get(figure_handle, 'Name');
   savefig(figure_handle, fullfile(figures_folder_name, [figure_name, '.fig']));
end

clearvars -except samples_table feature_eat_matrix_table feature_noneat_matrix_table
```

explained' =

Columns 1 through 17

82.3409 11.2408 3.7703 1.2797 0.6385 0.3237 0.2705 0.0678 0.0411 0.0179 0.0089 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

Columns 18 through 34

0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

Columns 35 through 43

0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000

Sub-Task 1: Arranging the Feature Matrix

Now since PCA requires only one matrix as the input, we will combine the eating and non-eating matrix. The process to do so is as follows:

- Create a matrix of the 43 features identified in the previous phase which is a combination of both eating and non-eating activities.
- Annotate this matrix by adding another column as class label to the matrix having 2 values, 1 – eating and 0 – non-eating.

Sub-Task 2 Execution of PCA

Now we will execute the PCA function on the feature matrix generated in sub-task 1.

The signature for the PCA method is as follows:
[coeff, score, latent, tsquared, explained, mu] = pca(X) [12]

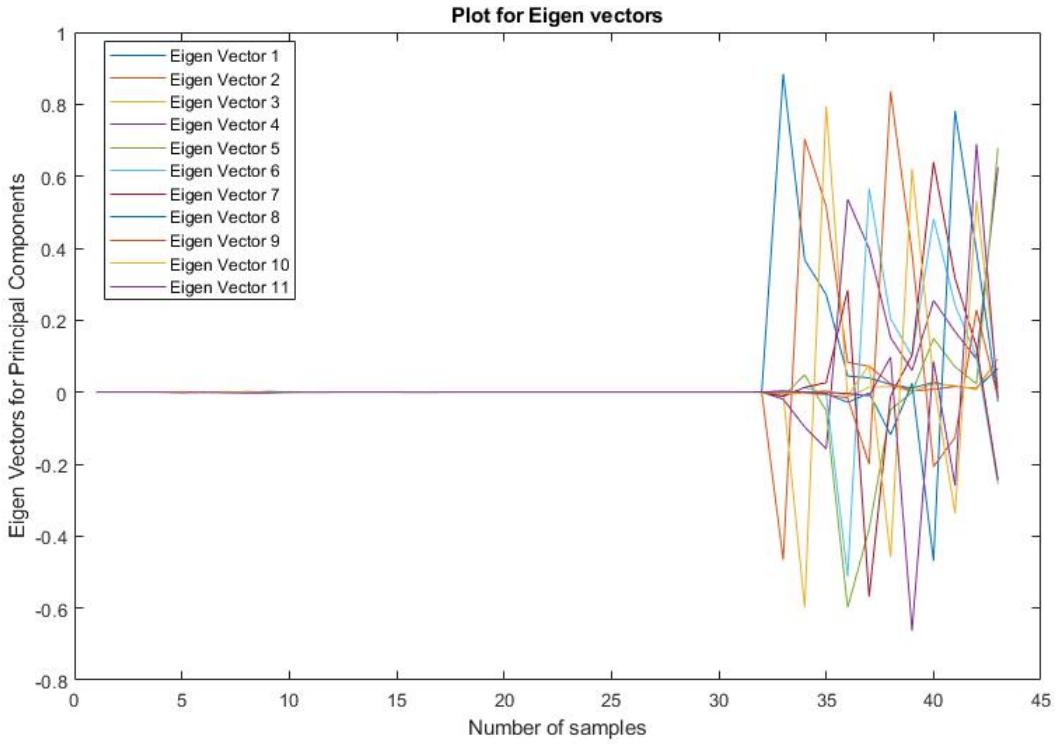
The output generated by this PCA method is as follows:

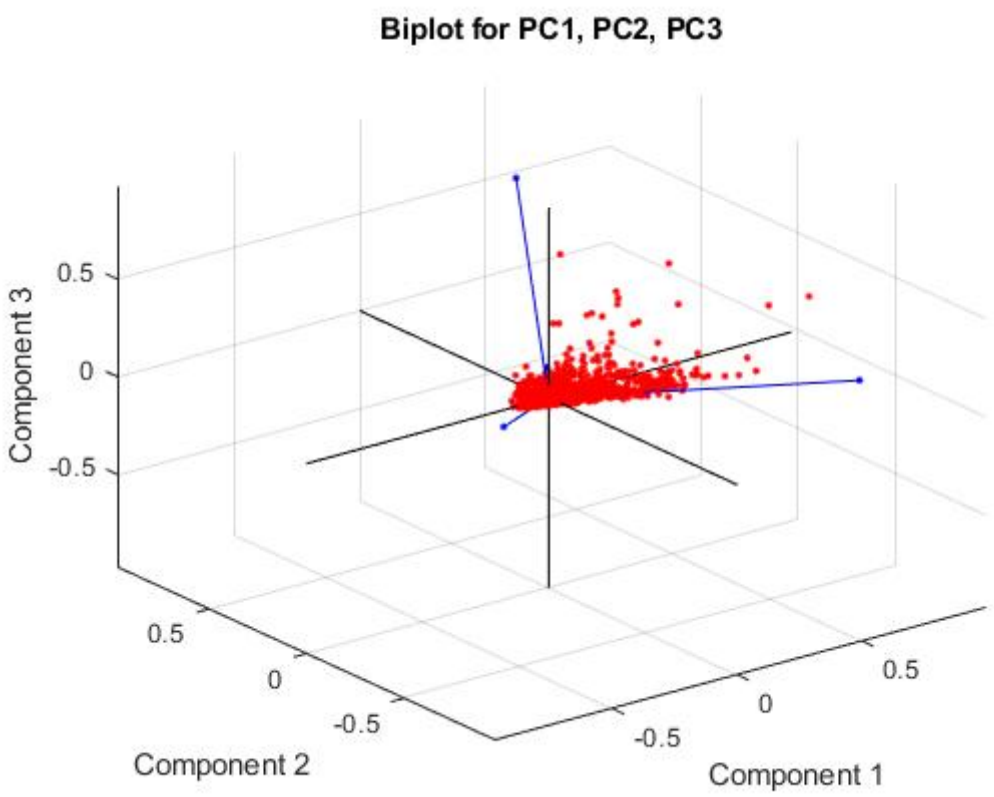
- The **coeff** gives us the output of principal component coefficients, also known as loadings, for our input feature matrix X. In our case X is obtained previously from sub task 1. Its dimensions are 2321 * 43 (2321 – Total eating and non-eating activities in the feature matrix, 43 – features selected for PCA from phase 2). Basically, this **coeff** matrix has dimensions 43 * 43 (for the 43 features now transformed into principal components). The PCA method by default centers the data and uses SVD algorithm. Its columns are arranged in descending order with respect to the component variance. [12]

- The **score** returns the representations of X in the principal component space. Rows of score correspond to observations, and columns correspond to components. Thus, in our case the dimensions for score matrix are 2321 * 43. [12]
- The **explained** returns the percentage of the total variance explained by each principal component. [12]

Sub-Task 3 Make sense of the PCA eigen vectors

In our case upon careful observation of the explained structure we conclude that only the first 11 principal components combine for 99.996 % of variance. Thus, we will use only the first 11 principal components since they retain 99.996% variance. The plots for eigen vectors for these 11 components are as follows:





Sub-Task 4 Results of PCA

The new feature matrix will be obtained from the score matrix. So, to get the new feature matrix, we do new_feature_matrix = score.

Also, in the previous subtask we identified that out of the 43 principal components only 11 were necessary as they contributed to 99.996% variance.

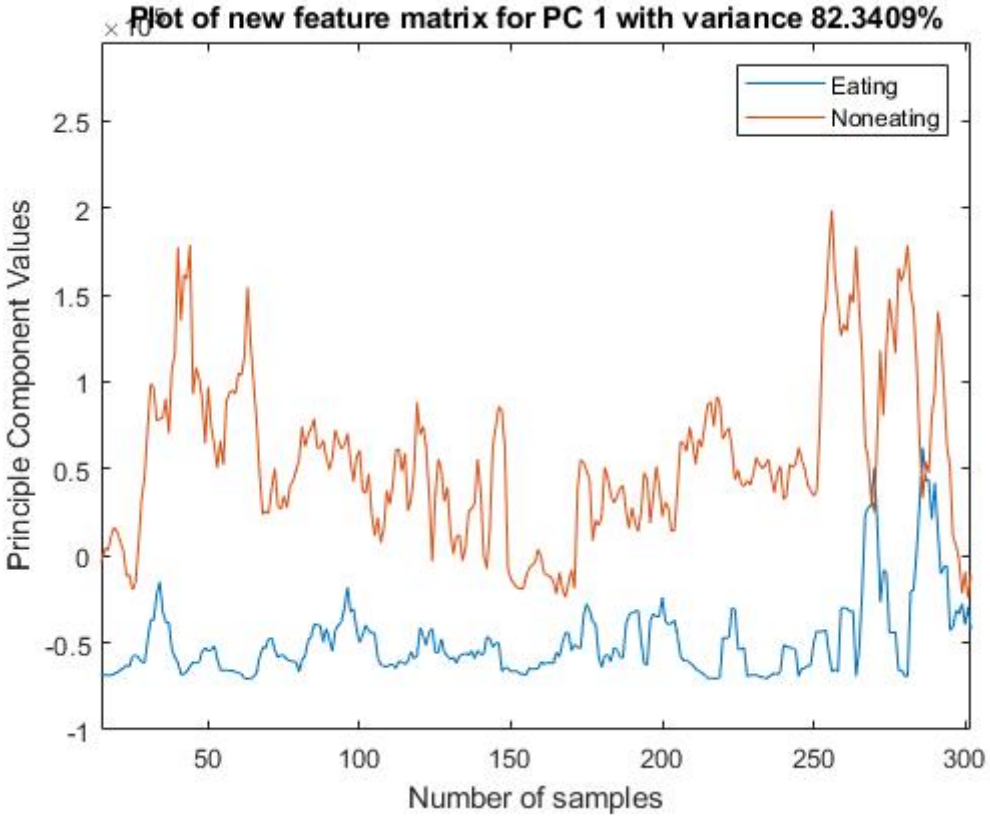
Hence, the new feature matrix will have the same 2321 no of observations, but it will only have 11 columns corresponding to the 11 principal components we identified in the previous subtask.

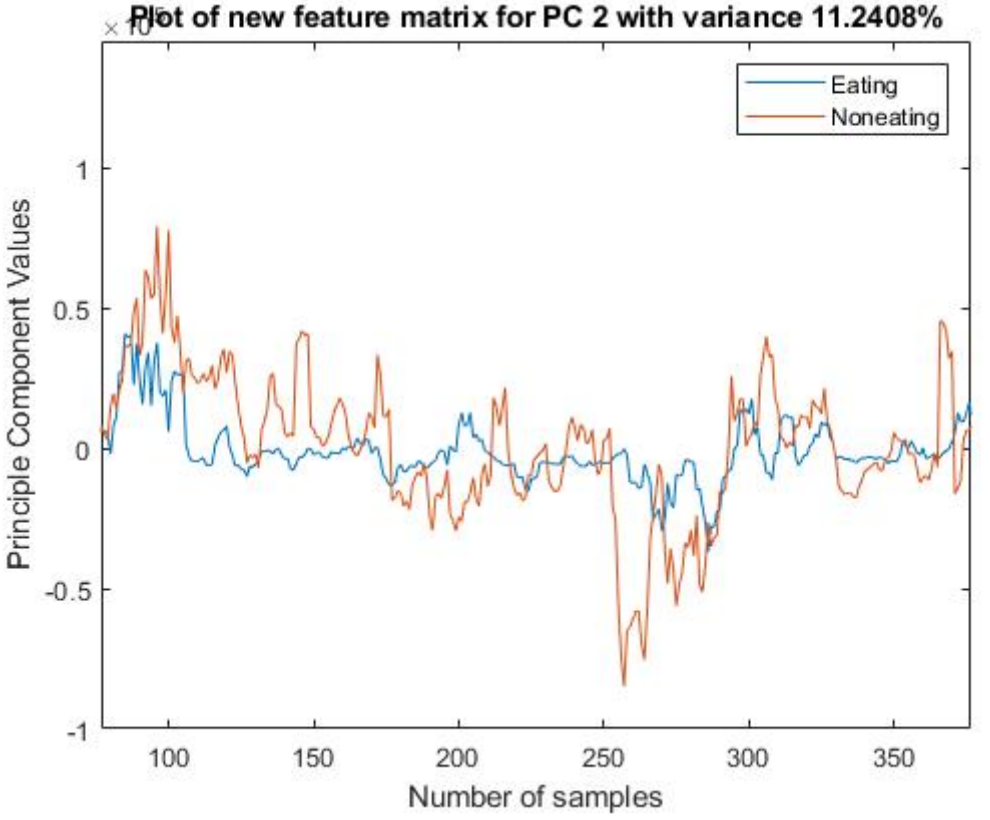
Sub-Task 5 PCA was helpful

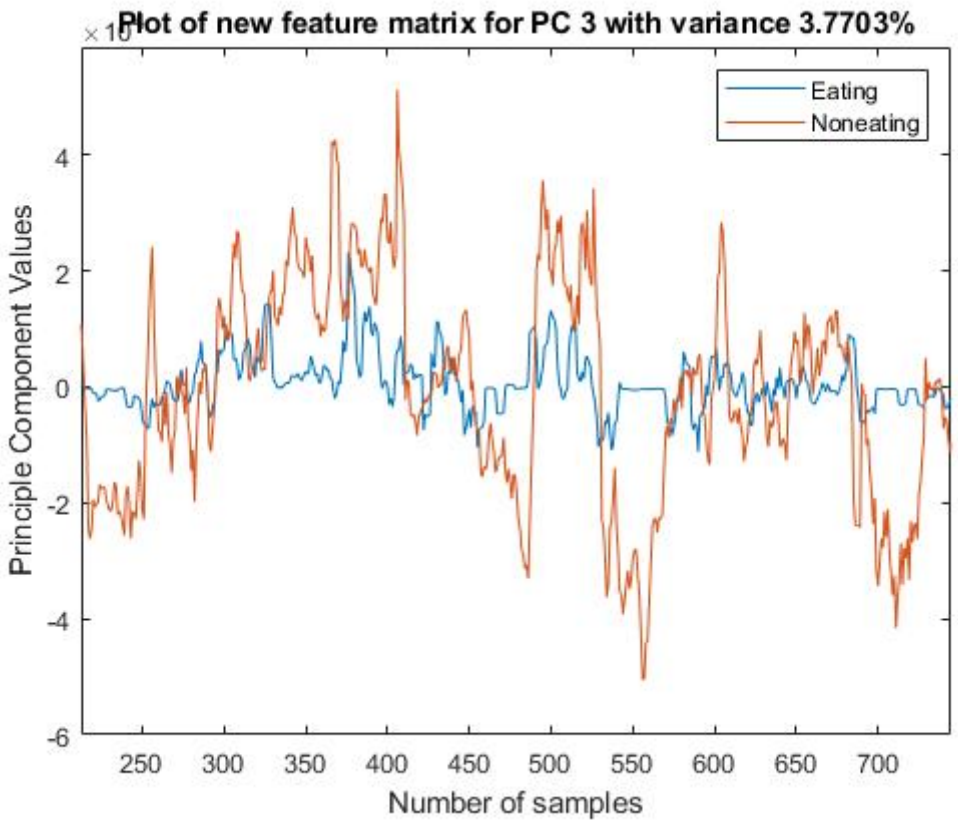
Using PCA was helpful in the following aspects:

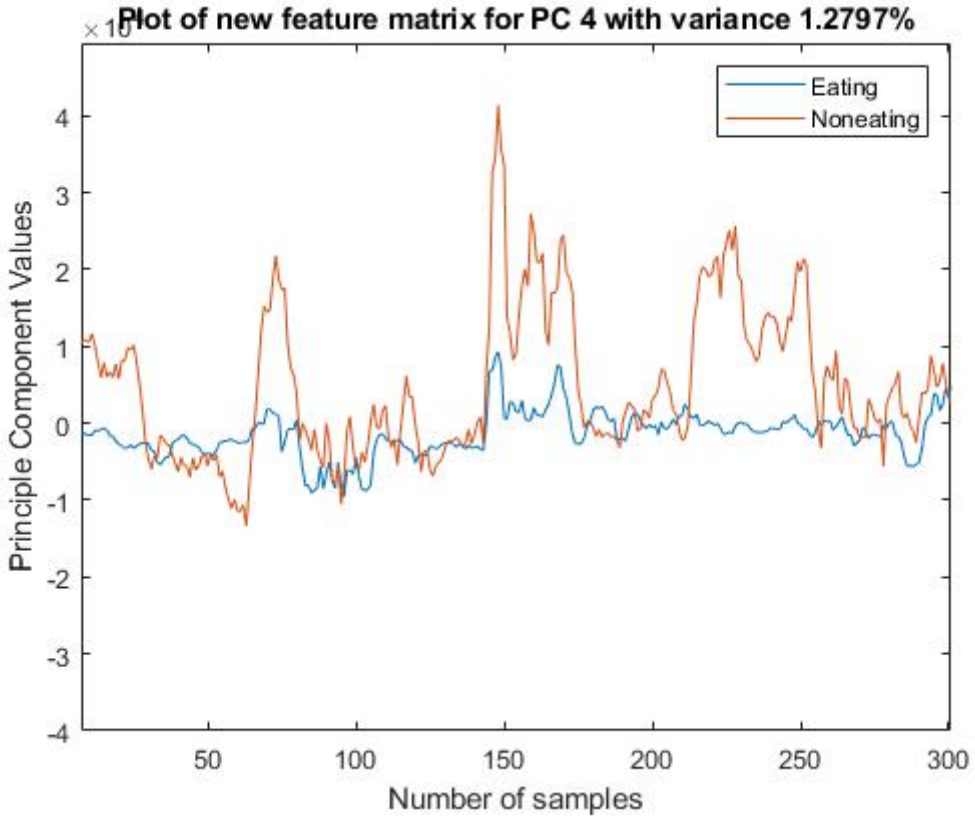
- It helped reduce the feature space from a staggering 108 features to only 11 principal components that contributed most towards identifying whether a given activity was eating or non-eating.
- The plots displayed below for the 11 principal components show a clearer distinction between the eating and non-eating activities as compared to the feature plots we obtained in phase 2.

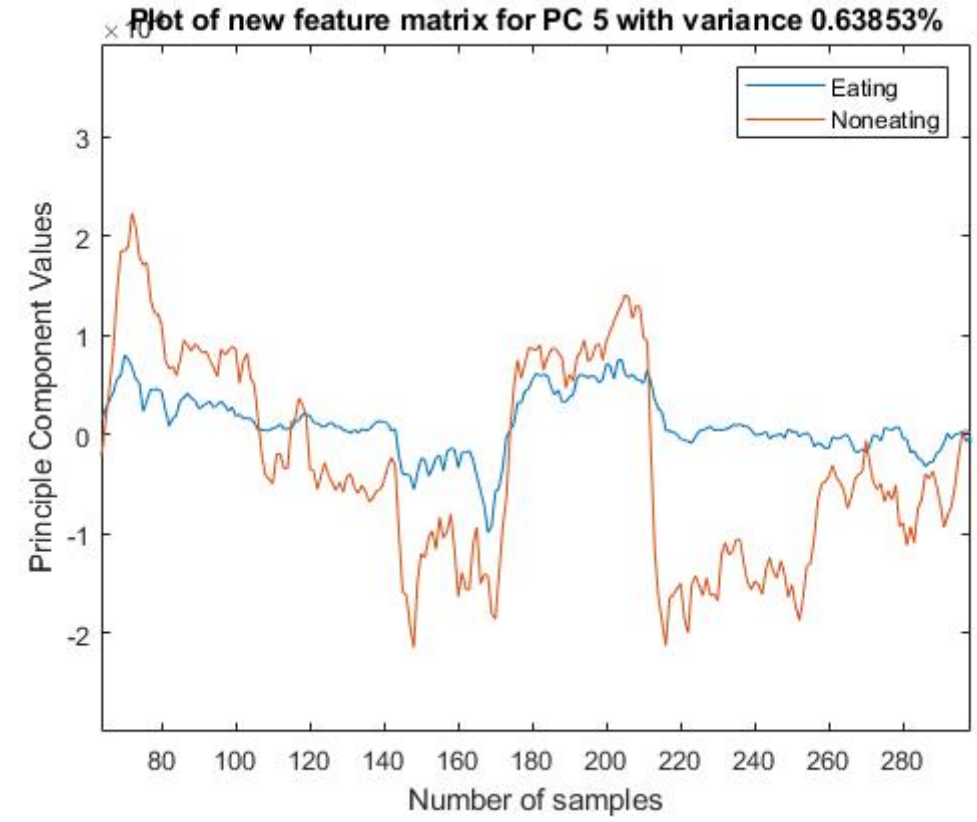
The plots for the 11 principal components with regards to differentiating in eating and non-eating activities are as follows:

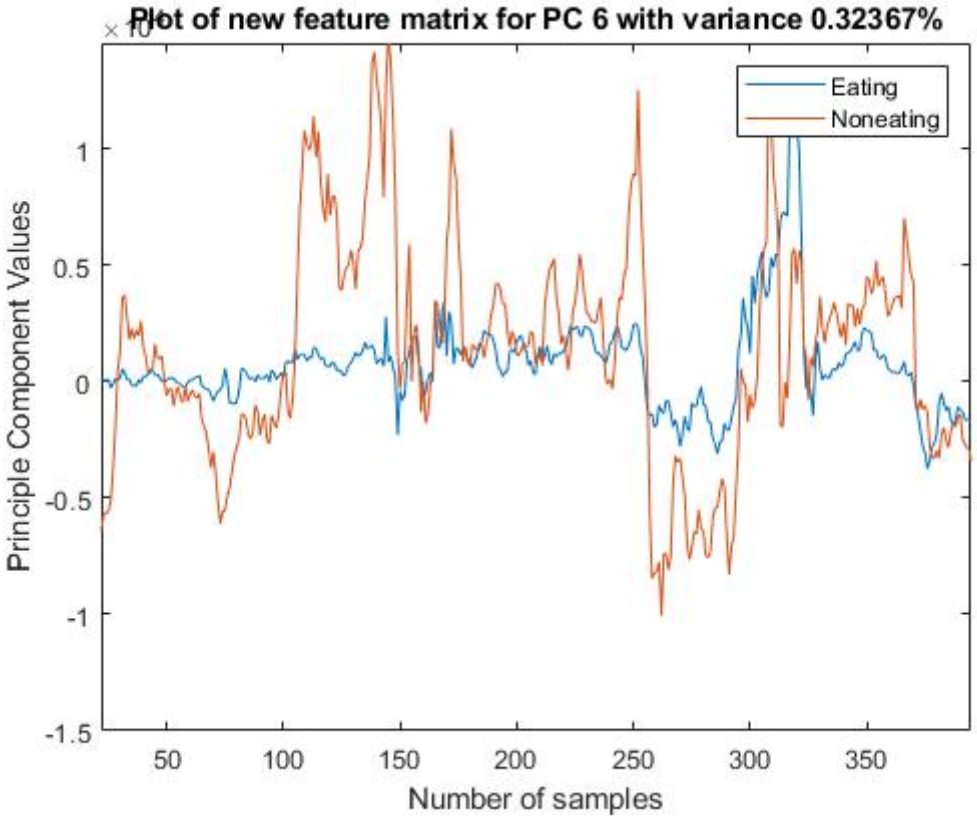

Plot of new feature matrix for PC 1 with variance 82.3409%
Eating
Noneating
Principle Component Values
Number of samples
2.5
2
1.5
1
0.5
0
-0.5
-1
50
100
150
200
250
300

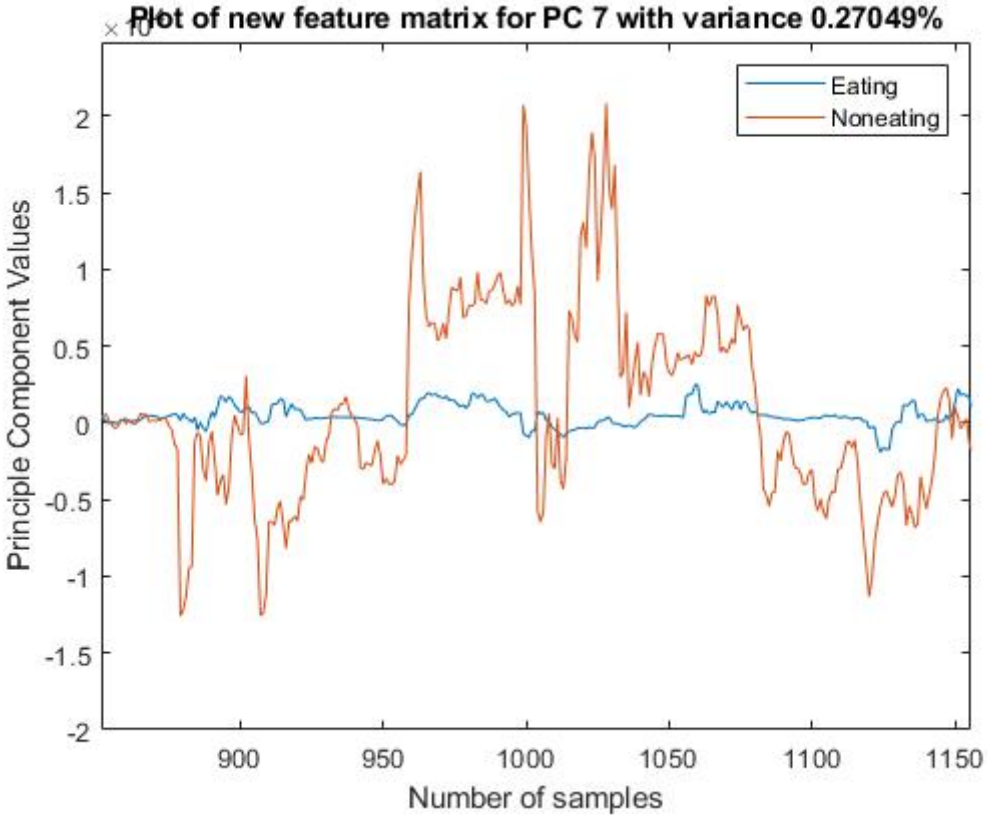

Plot of new feature matrix for PC 2 with variance 11.2408%
Eating
Noneating
Principle Component Values
Number of samples
1
0.5
0
-0.5
-1
100
150
200
250
300
350

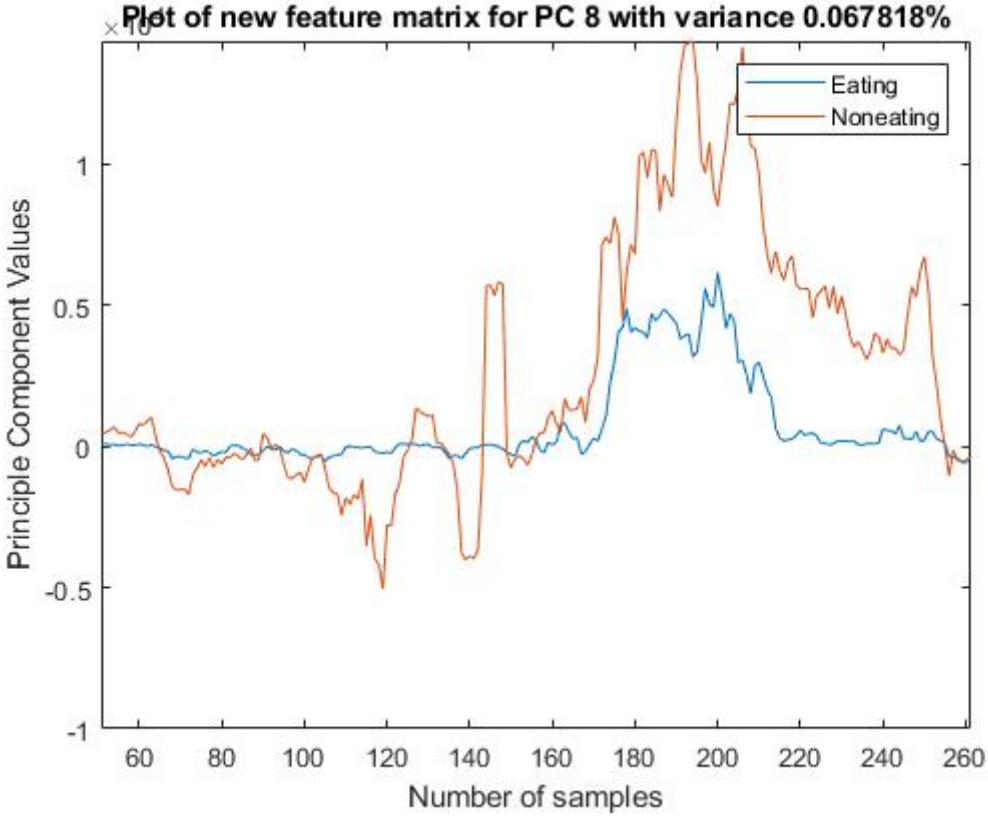

Plot of new feature matrix for PC 3 with variance 3.7703%
Eating
Noneating
Principle Component Values
Number of samples

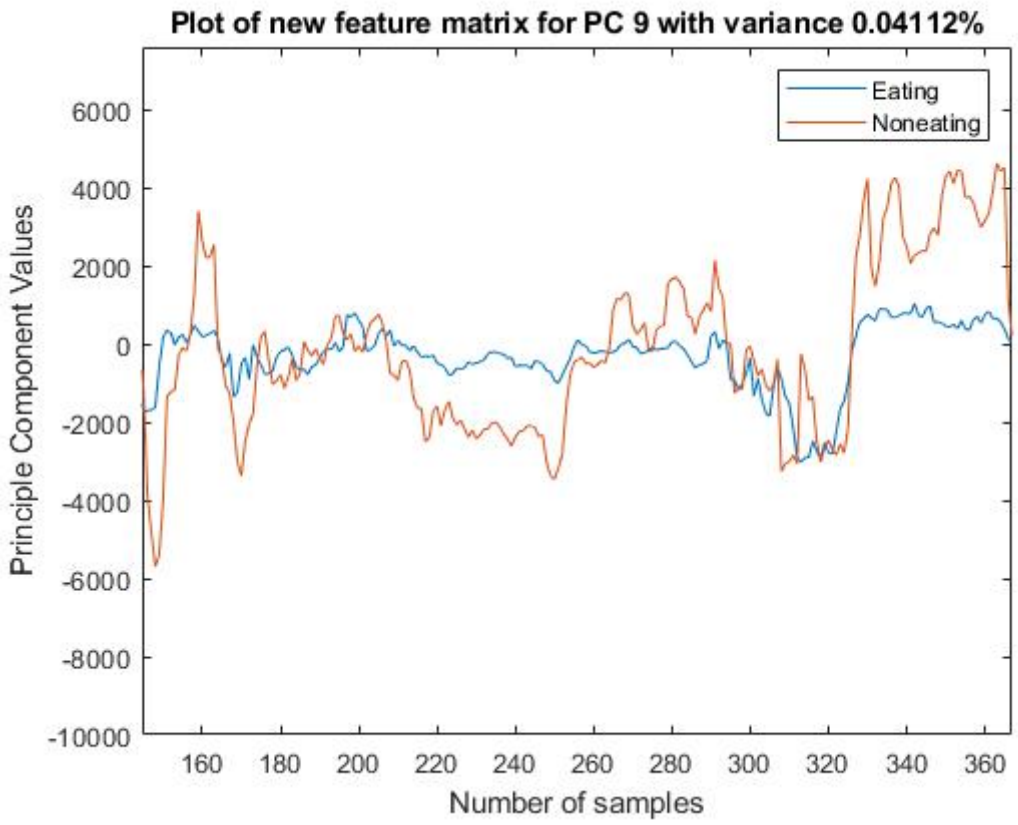

Plot of new feature matrix for PC 4 with variance 1.2797%
Eating
Noneating
Principle Component Values
Number of samples

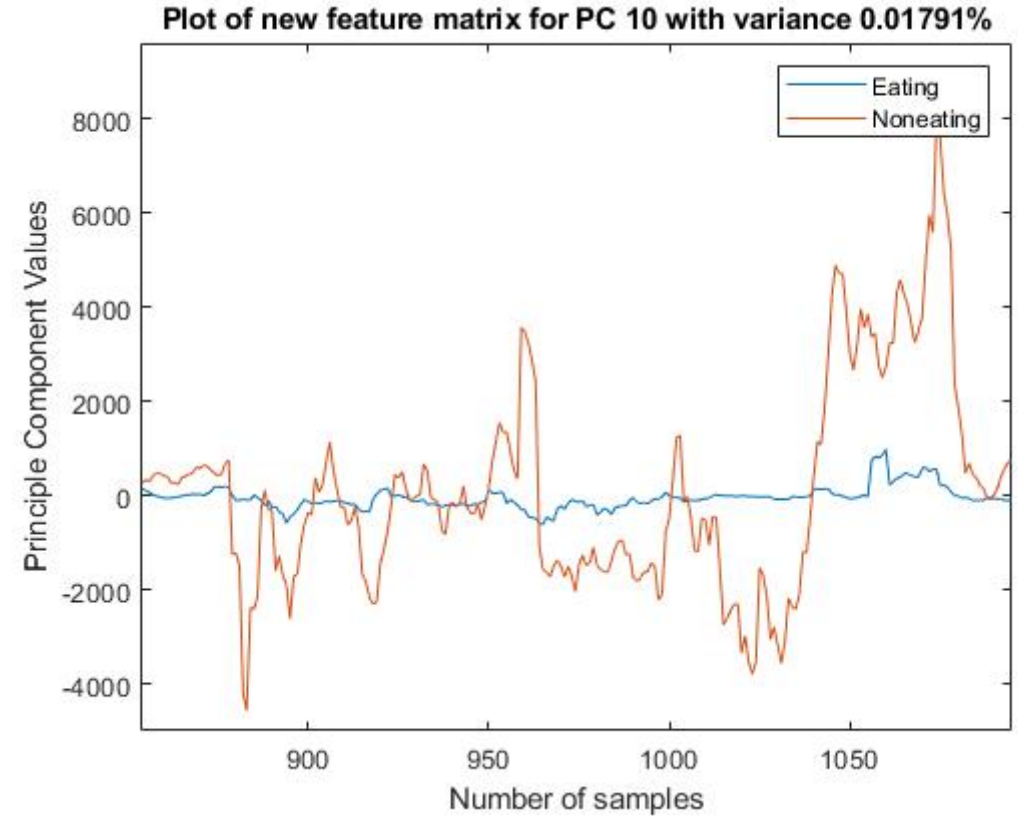

Plot of new feature matrix for PC 5 with variance 0.63853%
Eating
Noneating
Principle Component Values
Number of samples

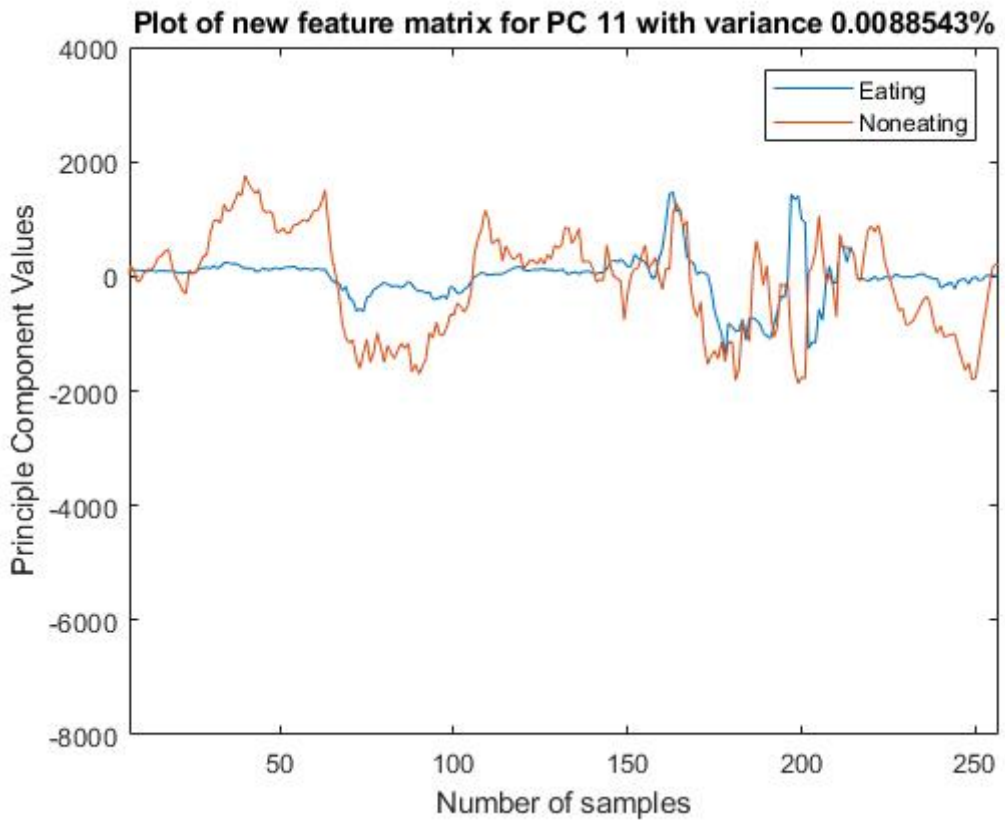

Plot of new feature matrix for PC 6 with variance 0.32367%
Eating
Noneating
Principle Component Values
Number of samples


Plot of new feature matrix for PC 7 with variance 0.27049%
Eating
Noneating
Principle Component Values
Number of samples
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
900
950
1000
1050
1100
1150


Plot of new feature matrix for PC 8 with variance 0.067818%
Eating
Noneating
Principle Component Values
Number of samples
1
0.5
0
-0.5
-1
60
80
100
120
140
160
180
200
220
240
260


Plot of new feature matrix for PC 9 with variance 0.04112%
Eating
Noneating
Principle Component Values
6000
4000
2000
0
-2000
-4000
-6000
-8000
-10000
160
180
200
220
240
260
280
300
320
340
360
Number of samples


Plot of new feature matrix for PC 10 with variance 0.01791%
Eating
Noneating
Principle Component Values
8000
6000
4000
2000
0
-2000
-4000
900
950
1000
1050
Number of samples



Citations

[1] "https://www.mathworks.com/help/matlab/ref/mean.html," [Online]. Available: https://www.mathworks.com/help/matlab/ref/mean.html.

[2] "https://searchdatacenter.techtarget.com/definition/statistical-mean-median-mode-and-range," [Online]. Available: https://searchdatacenter.techtarget.com/definition/statistical-mean-median-mode-and-range.

[3] "https://www.mathworks.com/help/signal/ref/rms.html," [Online]. Available: https://www.mathworks.com/help/signal/ref/rms.html.

[4] "http://www.differencebetween.net/science/difference-between-rms-and-average/," [Online]. Available: http://www.differencebetween.net/science/difference-between-rms-and-average/.

[5] "https://www.mathworks.com/help/matlab/ref/std.html," [Online]. Available: https://www.mathworks.com/help/matlab/ref/std.html.

[6] "https://corporatefinanceinstitute.com/resources/knowledge/standard-deviation/," [Online]. Available: https://corporatefinanceinstitute.com/resources/knowledge/standard-deviation/.

[7] "https://www.mathworks.com/help/matlab/ref/max.html," [Online]. Available: https://www.mathworks.com/help/matlab/ref/max.html.

[8] "https://www.mathworks.com/help/signal/ref/pentropy.html," [Online]. Available: https://www.mathworks.com/help/signal/ref/pentropy.html.

[9] "https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c," [Online]. Available: https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c.

[10] "https://www.statisticshowto.datasciencecentral.com/quadratic-mean/," [Online]. Available: https://www.statisticshowto.datasciencecentral.com/quadratic-mean/.

[11] "https://www.dummies.com/education/math/statistics/how-to-interpret-standard-deviation-in-a-statistical-data-set/," [Online]. Available: https://www.dummies.com/education/math/statistics/how-to-interpret-standard-deviation-in-a-statistical-data-set/.