CSE 572 DATA MINING

Project Report: Activity Recognition

By

Hari Meyyappan (1215106204)
Varun Krishna (1214849428)
Shubham Bakre (1215188000)
Venkata Akhil Madaraju (1213038305)

Table of Contents

Introduction 2

Phase 1: Data Cleaning and Organization 3

Phase 2: Feature Extraction 3

Phase 3: PCA 23

Codes in File 29

Introduction

Motivation

Data collection has permeated our lives so thoroughly, that we often forget that it is happening - 2.5 quintillion bytes of data is created each day! The pace is only increasing with the advent of IoT. Data being generated from our financial transactions, entertainment preferences and even health-related activities like sleeping, exercising and eating. This is accomplished using sensors that collect the data, and often these datasets can be huge. They also have missing data, outliers, etc. We have been provided such a dataset that has to be cleaned and processed, before extracting information extracted from it.

Data

We have been provided with real world wristband data for a given activity - specifically eating action mixed with other unknown activities. Our aim is to identify the eating activities amongst the noise. We have two sources of sensor data - the IMU and EMG data.

- IMU file consists of 11 columns: UNIX timestamp, Orientation X, Orientation Y, Orientation Z, Orientation W, Accelerometer X, Accelerometer Y, Accelerometer Z, Gyroscope X, Gyroscope Y, and Gyroscope Z.

- EMG file consists of 9 columns: UNIX timestamp, EMG 1, EMG 2, EMG 3, EMG 4, EMG 5, EMG 6, EMG 7, and EMG 8.

We're also provided with labelled video frame data for eating and non-eating activities.

Phase 1: Data Cleaning and Organization

The Phase 1 involves cleaning and organizing the sensor data given to us. The data is provided for user from 9 through 41, with some missing rows. We have to clean the columns for both EMG and IMU data. The cleaned data is then added with a column representing eating (0) and non eating data (1). Then we synchronize the data between the two datasets by setting the last frame time to the last UNIX timestamp both the IMU or EMG file. The last column that are added with zeros are then changed to 1's based on the validation of the eating tasks. The non-eating tasks are retained as 0 and the files are split according to the columns value (1 for eating and 0 for non-eating). With the help of this data all the graphs are plotted.

Phase 2: Feature Extraction

Our next step is to select features The features we selected are:

- Mean
- Standard Deviation
- Minimum
- Maximum
- Root Mean Square
- Fourier Transform

a) write an explanation on how the feature is extracted.

For each user (9 through 41) in both the EMG and IMU data, we iterate over the columns of the data of the corresponding individuals and create a seperate dataframe for each of them. Then we apply functions that return the result we want. We have used the Pandas, Numpy & Scikit Learn libraries to extract the features:

- **Mean**

```
DataFrame.mean(axis=None, skipna=None, level=None,
numeric_only=None,**kwargs)
```

Return the mean of the values for the requested axis.

- **Standard Deviation**

```
DataFrame.std(axis=None, skipna=None, level=None, numeric_only=None,
```

```
**kwargs)
```

Return the standard deviation of the values for the requested axis.

- **Minimum/Maximum**

```
DataFrame.min(axis=None, skipna=None, level=None,
numeric_only=None, **kwargs)
DataFrame.max(axis=None, skipna=None, level=None,
numeric_only=None, **kwargs)
```

Return the minimum/maximum of the values for the requested axis.

- **Root Mean Square**
  We calculate the RMS by squaring all the terms and taking the mean of the sum.
  `np.sqrt(np.mean(df[col]**2))`

- **Fourier Transform**
  `fft(x[, n, axis, overwrite_x])`
  Return discrete Fourier transform of real or complex sequence.

b) Write an intuition on why you use such a feature.

Mean :
Eating activities have a small range of motion and hence the mean for the values like gyroscope would be concentrated across the eating dataset. It is very unlikely that the mean of non-eating activities would also fall in this range, and hence eating and non-eating activities can be differentiated.

Standard Deviation :
The standard deviation is used to quantify the amount of variation or dispersion of a set of data values. A low standard deviation indicates that the data points tends to be close to the mean. While a high standard deviation indicated that the data points are spread out over a wider range of values. The standard deviation indicates the variance of the data points, and hence can be used to differentiate between the eating and non-eating activities.

Root Mean Square :
The root mean square is modelled as the amplitude modulated Gaussian random process and is related to the standard deviation. The RMS on time-series data can give a single value that could lead to it being an interesting feature depending on the data.

Minimum :

This feature extraction is used to understand the minimum value of the data points. This allows us to differentiate the data from the remaining data. Minimum on a time-series data gives us important data points that can lead to important feature depending on the data. The eating activities might be capped but the non-eating activities might have much lesser values.

Maximum:
Maximum is used to understand the maximum values in the data. This can give us important results based on the data. The eating activities might be capped but the non-eating activities might have much larger values.

Fourier Transform:
The Fourier transform (FT) decomposes a signal into the frequencies that make it up. This is basically transforming our datasets of eating and non-eating activities, and the transformations could result in increasing the variation between the classes, and help us classify them correctly.

c) Write a code to extract that feature from each time series created in task 1.

Below is the code for EMG Data. We have used similar code for IMU dataset, and have attached both the codes, and corresponding ipynb files along with the report.

1. **MEAN**

```
df = pd.read_csv("emgeatingfile.csv")
a = df['Number'].unique()
a.sort()

li = {}
for i in a:
  li[i] = df.loc[df['Number'] == i]

x1, y1 = {}, {}
# Mean of EMG1 for all Eating
means_eating = {}
for j in range(1, 9):
  for i in a: # i is no of persons
      df = li[i] #Returns df for person i
      col = "EMG" + str(j) # Column EMG1, EMG2...
      means_eating[i] = df[col].mean() # means_eating[1] is mean of EMG1
      #print("Done for person ", i)
```

```
  lists = sorted(means_eating.items()) # sorted by key, return a list of
tuples
  u, v = zip(*lists) # unpack a list of pairs into two tuples
  x1[j], y1[j] = u, v
  print("Done for col EMG ", j)

for i in range(1,9):
  plt.plot(x1[i], y1[i])
plt.legend(['EMG1', 'EMG2', 'EMG3', 'EMG4', 'EMG5', 'EMG6', 'EMG7'],
loc='upper left')
plt.title("Feature #1: Mean (Eating)")
plt.show()
```

2. STANDARD DEVIATION

```
df = pd.read_csv("emgeatingfile.csv")
a = df['Number'].unique()
a.sort()

li = {}
for i in a:
  li[i] = df.loc[df['Number'] == i]

x1, y1 = {}, {}
# Std of EMG1 for all Eating
means_eating = {}
for j in range(1, 9):
  for i in a: # i is no of persons
      df = li[i] #Returns df for person i
      col = "EMG" + str(j) # Column EMG1, EMG2...
      means_eating[i] = df[col].std() # means_eating[1] is mean of EMG1
      #print("Done for person ", i)
  lists = sorted(means_eating.items()) # sorted by key, return a list of
tuples
  u, v = zip(*lists) # unpack a list of pairs into two tuples
  x1[j], y1[j] = u, v
  #print("Done for col EMG ", j)

for i in range(1,9):
  plt.plot(x1[i], y1[i])
plt.legend(['EMG1', 'EMG2', 'EMG3', 'EMG4', 'EMG5', 'EMG6', 'EMG7',
```

```
'EMG8'], loc='upper right')
plt.title("Feature #2: Standard Deviation (Eating)")
plt.show()
```

3a. MINIMUM

```
df = pd.read_csv("emgeatingfile.csv")
a = df['Number'].unique()
a.sort()

li = {}
for i in a:
  li[i] = df.loc[df['Number'] == i]

x1, y1 = {}, {}
# Min of EMG1 for all Eating
means_eating = {}
for j in range(1, 9):
  for i in a: # i is no of persons
      df = li[i] #Returns df for person i
      col = "EMG" + str(j) # Column EMG1, EMG2...
      means_eating[i] = df[col].min() # means_eating[1] is mean of EMG1
      #print("Done for person ", i)
  lists = sorted(means_eating.items()) # sorted by key, return a list of
tuples
  u, v = zip(*lists) # unpack a list of pairs into two tuples
  x1[j], y1[j] = u, v
  #print("Done for col EMG ", j)

for i in range(1,9):
  plt.plot(x1[i], y1[i])
plt.legend(['EMG1', 'EMG2', 'EMG3', 'EMG4', 'EMG5', 'EMG6', 'EMG7','EMG8'],
loc='best')
plt.title("Feature #3a: Minimum (Eating)")
plt.show()
```

3b. MAXIMUM

```
df = pd.read_csv("emgeatingfile.csv")
```

```
a = df['Number'].unique()
a.sort()

li = {}
for i in a:
  li[i] = df.loc[df['Number'] == i]

x1, y1 = {}, {}
# Max of EMG1 for all Eating
means_eating = {}
for j in range(1, 9):
  for i in a: # i is no of persons
      df = li[i] #Returns df for person i
      col = "EMG" + str(j) # Column EMG1, EMG2...
      means_eating[i] = df[col].max() # means_eating[1] is mean of EMG1
      #print("Done for person ", i)
  lists = sorted(means_eating.items()) # sorted by key, return a list of
tuples
  u, v = zip(*lists) # unpack a list of pairs into two tuples
  x1[j], y1[j] = u, v
  #print("Done for col EMG ", j)

for i in range(1,9):
  plt.plot(x1[i], y1[i])
plt.legend(['EMG1', 'EMG2', 'EMG3', 'EMG4', 'EMG5', 'EMG6', 'EMG7',
'EMG8'], loc='best')
plt.title("Feature #3b: Maximum (Eating)")
plt.show()
```

4. Root Mean Square

```
df = pd.read_csv("emgeatingfile.csv")
a = df['Number'].unique()
a.sort()

li = {}
for i in a:
  li[i] = df.loc[df['Number'] == i]

x1, y1 = {}, {}
# RMS of EMG1 for all Eating
```

```
means_eating = {}
for j in range(1, 9):
  for i in a: # i is no of persons
      df = li[i] #Returns df for person i
      col = "EMG" + str(j) # Column EMG1, EMG2...
      means_eating[i] = np.sqrt(np.mean(df[col]**2)) # Root Mean Square
      #print("Done for person ", i)
  lists = sorted(means_eating.items()) # sorted by key, return a list of
tuples
  u, v = zip(*lists) # unpack a list of pairs into two tuples
  x1[j], y1[j] = u, v
  #print("Done for col EMG ", j)

for i in range(1,9):
  plt.plot(x1[i], y1[i])
plt.legend(['EMG1', 'EMG2', 'EMG3', 'EMG4', 'EMG5', 'EMG6', 'EMG7',
'EMG8'], loc='best')
plt.title("Feature #4: Root Mean Square (Eating)")
plt.show()
```

5. Fourier Transform

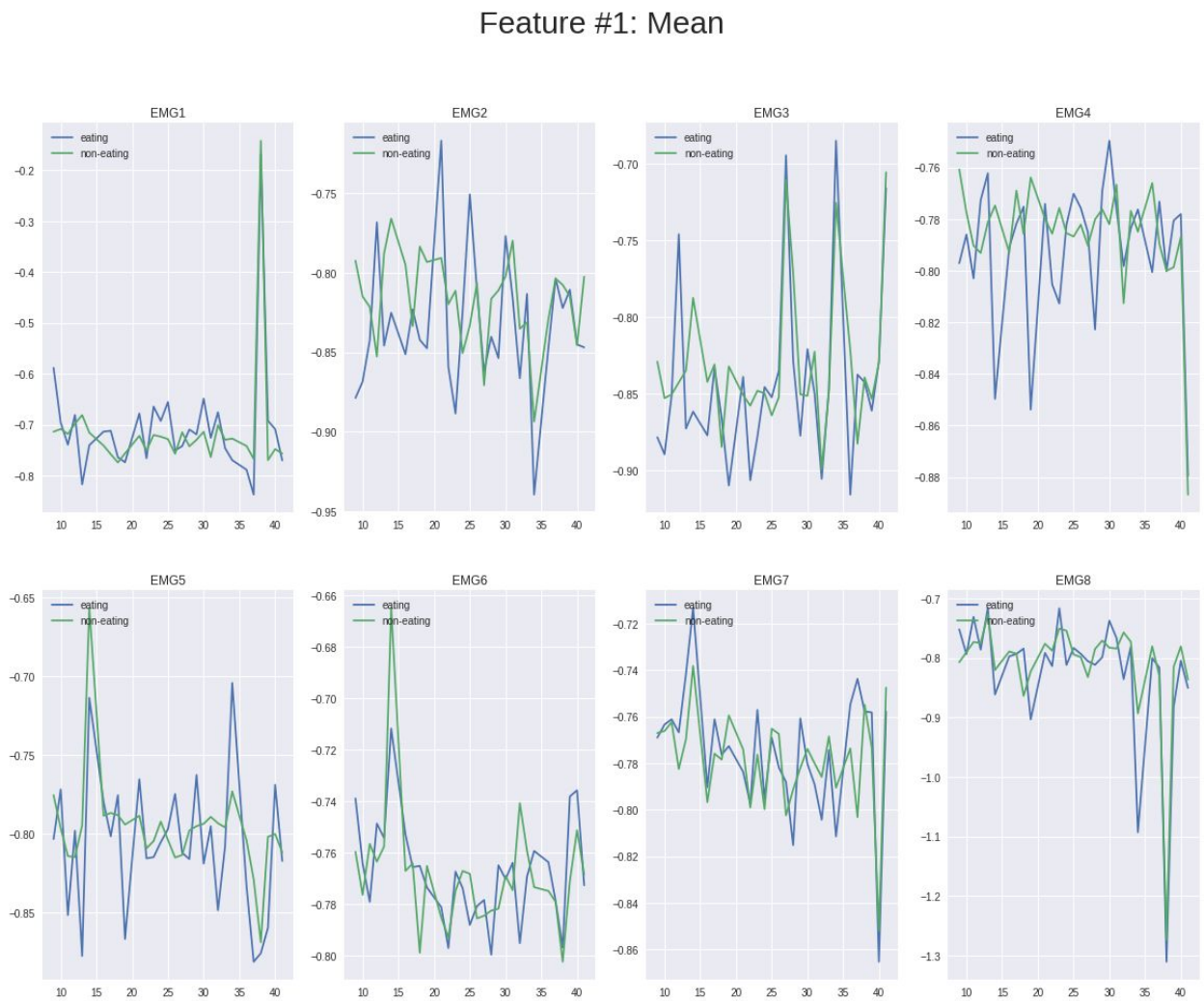
```
import scipy.fftpack

df = pd.read_csv("emgeatingfile.csv")
a = df['Number'].unique()
a.sort()

li = {}
for i in a:
  li[i] = df.loc[df['Number'] == i]
x1, y1 = {}, {}
# FT of EMG1 for all Eating
means_eating = {}
for j in range(1, 9):
  for i in a: # i is no of persons
      df = li[i] #Returns df for person i
      col = "EMG" + str(j) # Column EMG1, EMG2...

      yf = scipy.fft(df[col].values)
```

```
      x = scipy.fftpack.fftfreq(yf.size, 1 / 50) # inverse frequency
  x1[j], y1[j] = x, yf
  plt.legend(['EMG1', 'EMG2', 'EMG3', 'EMG4', 'EMG5', 'EMG6', 'EMG7',
'EMG8'], loc='best')
  plt.title("Feature #5: Fourier Transform (Eating)")
  plt.plot(x[:x.size//2], abs(yf)[:yf.size//2])
```

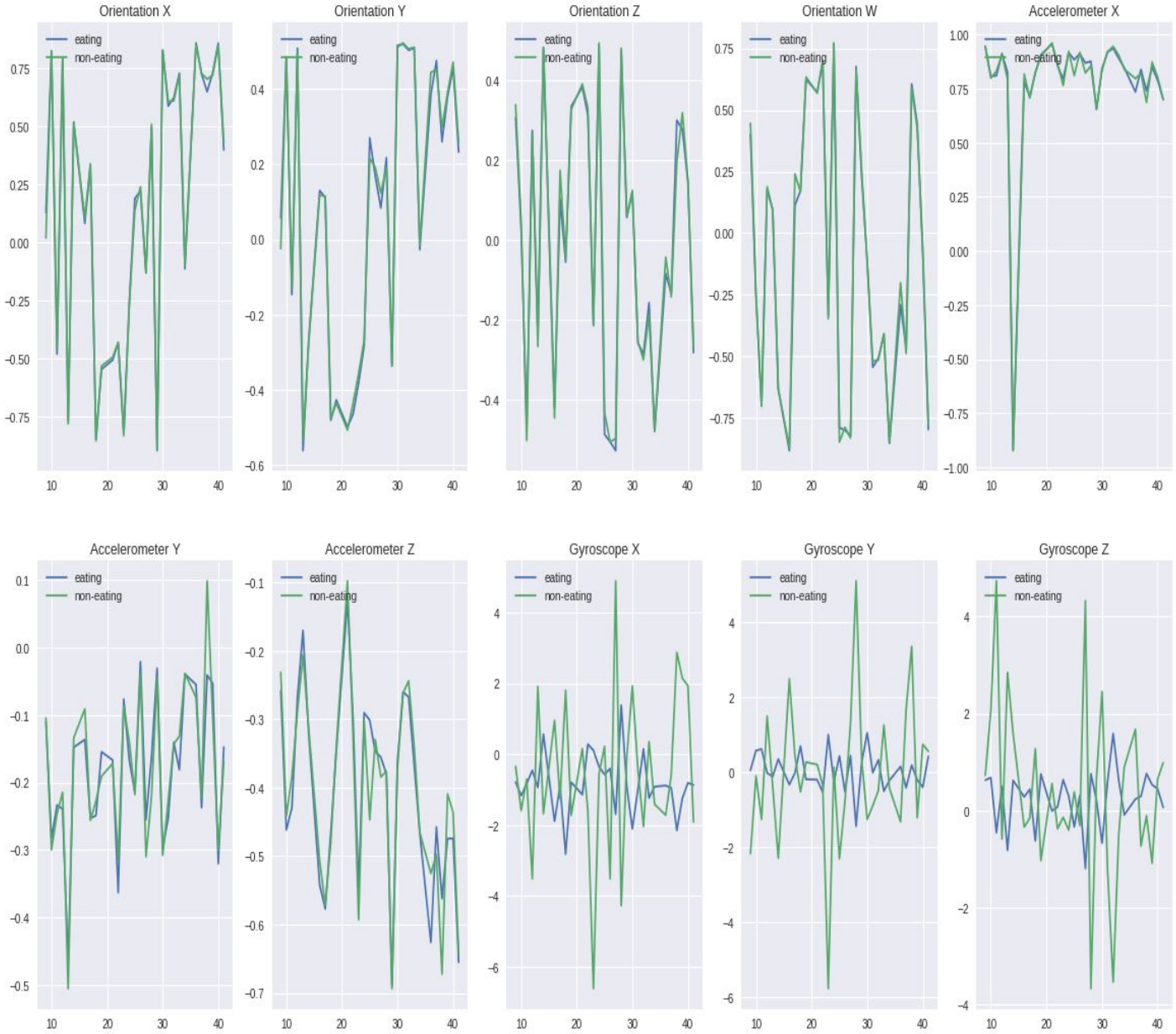
d) Generate plots each corresponding to eating and non-eating activities . This will give you a better idea of potential patterns in the features.

We have generated plots for the 5 features comparing the eating and non-eating data across all the columns in the sensor data.

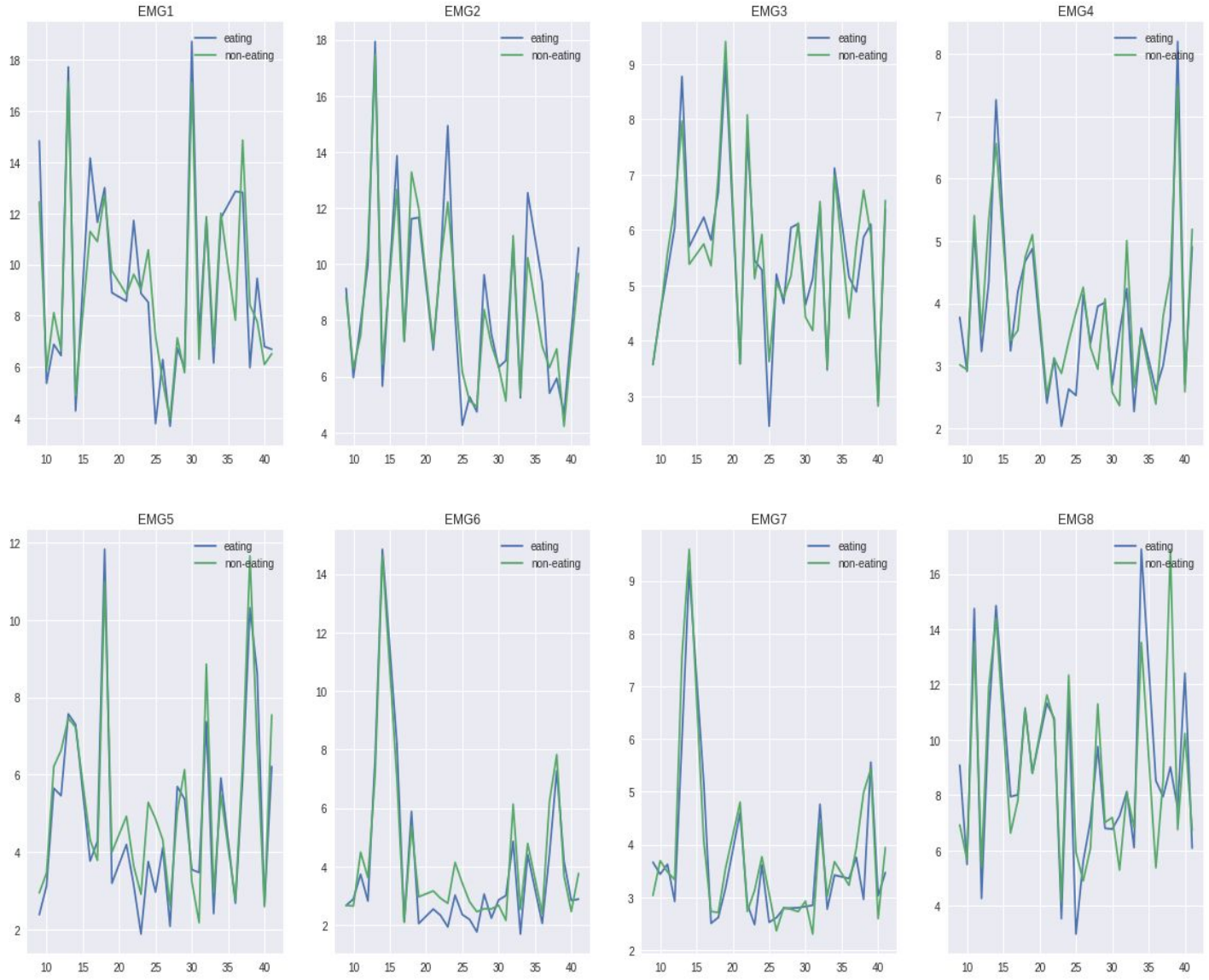
.



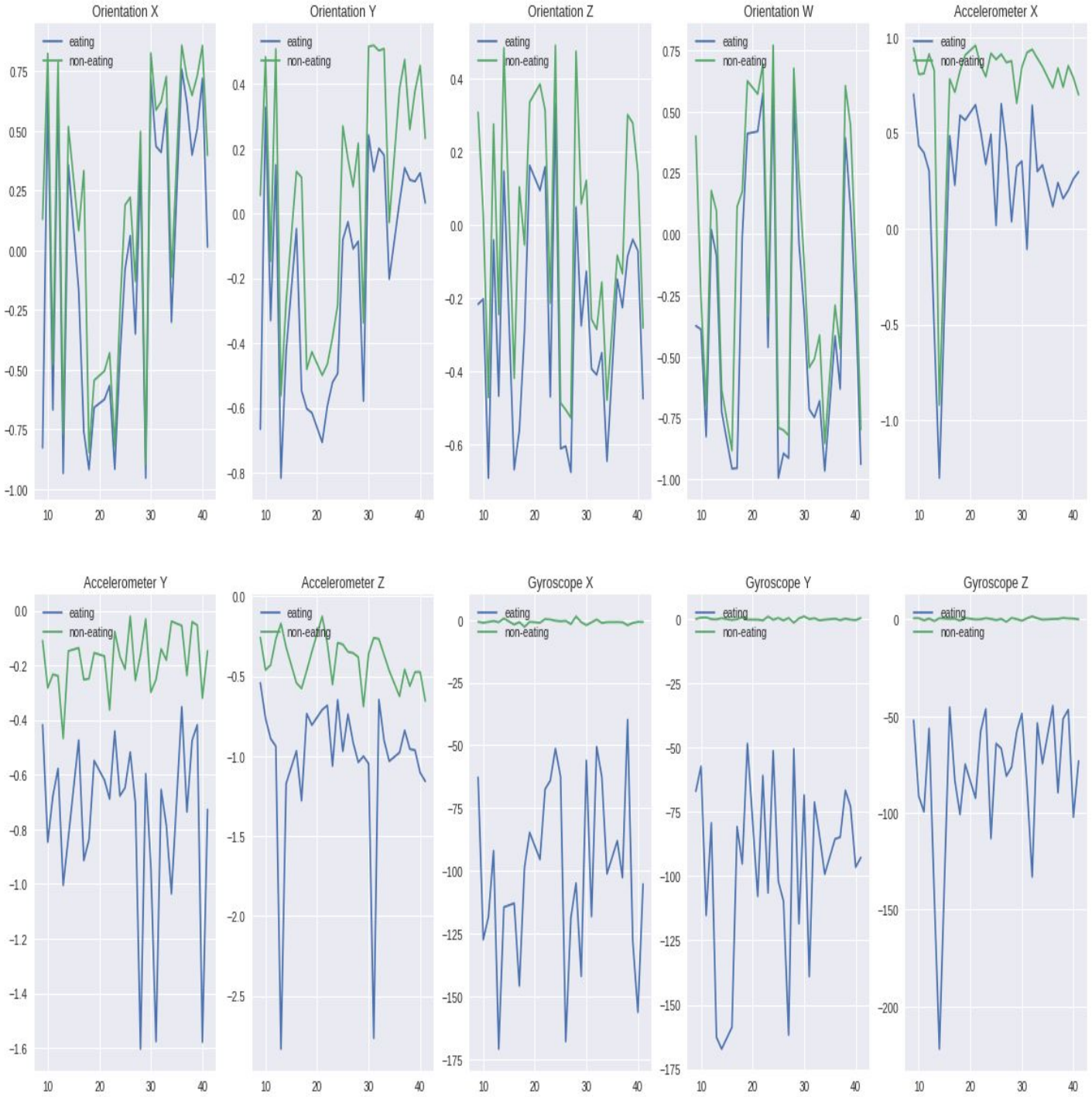
Feature #1: Mean



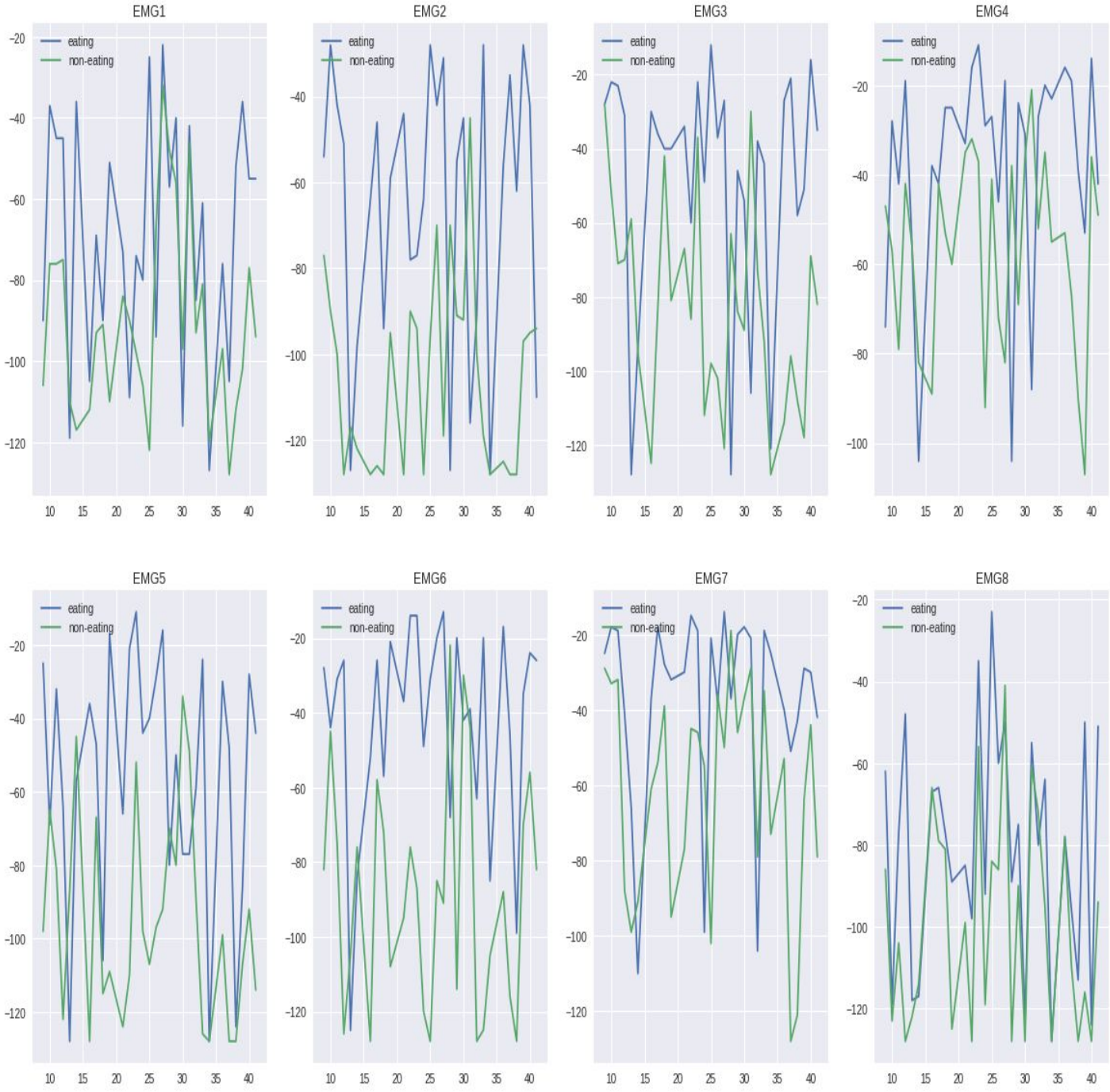
Feature #2: Standard Deviation



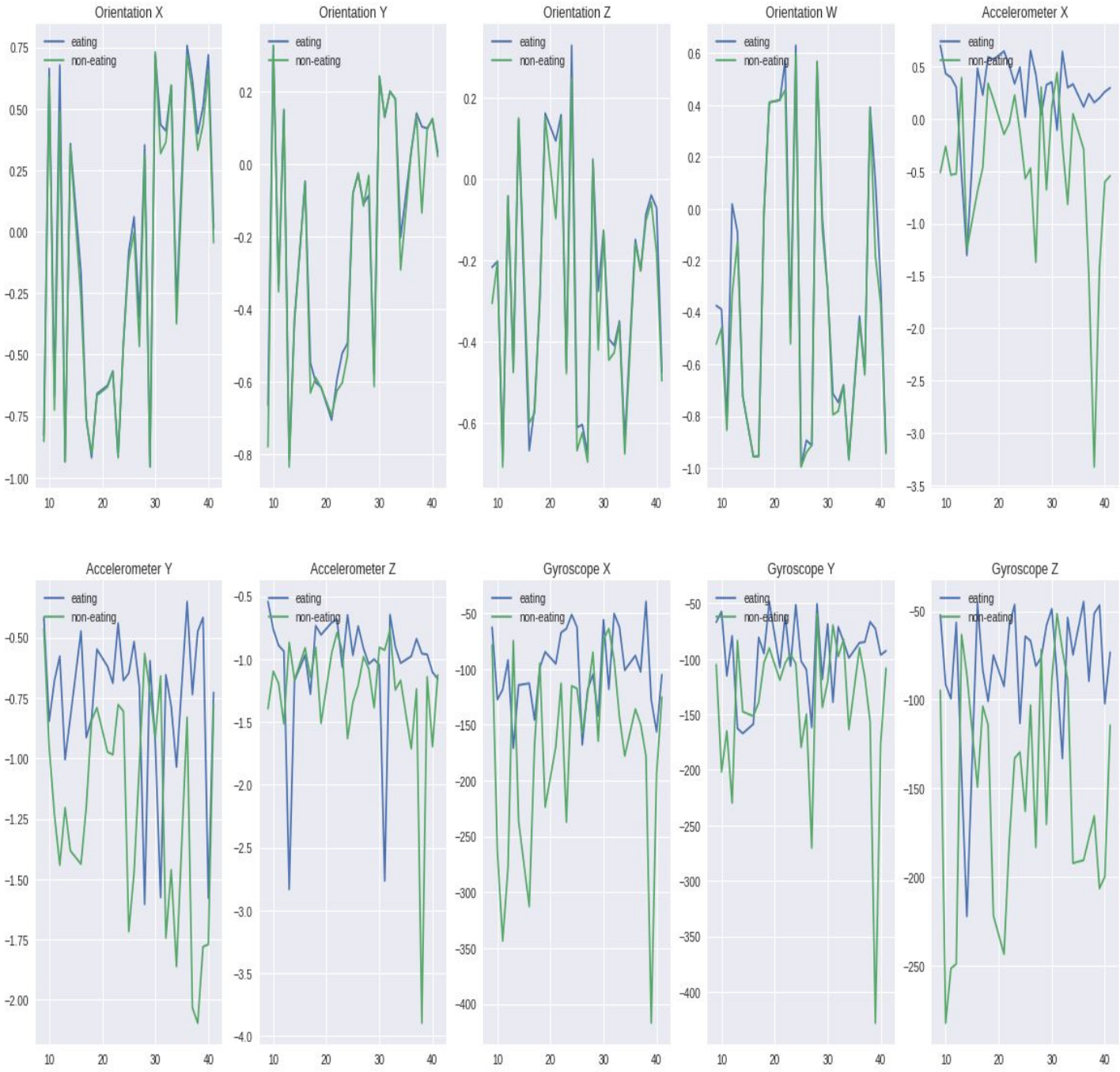
Feature #2: Standard Deviation



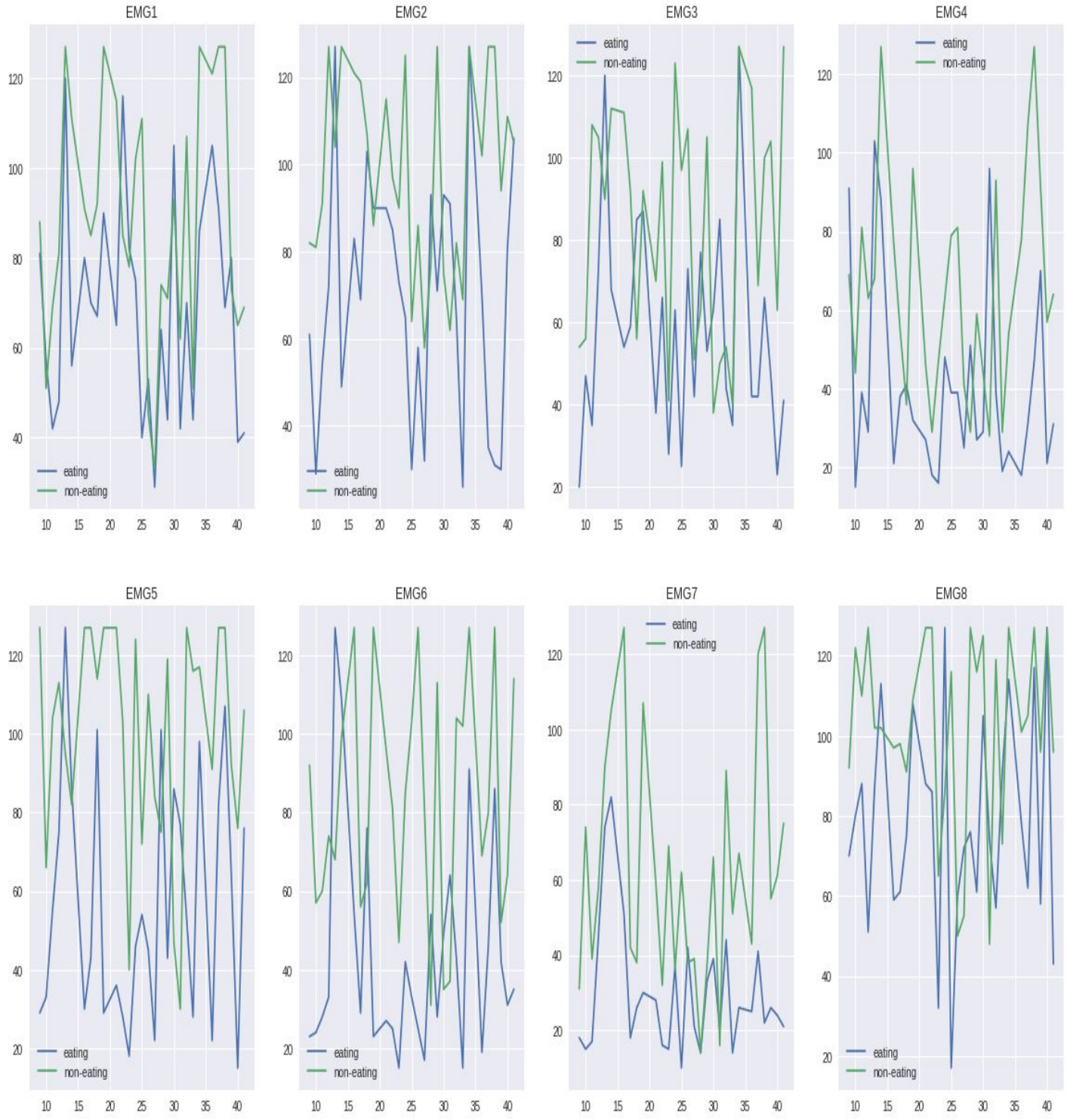
Feature #3a: Minimum



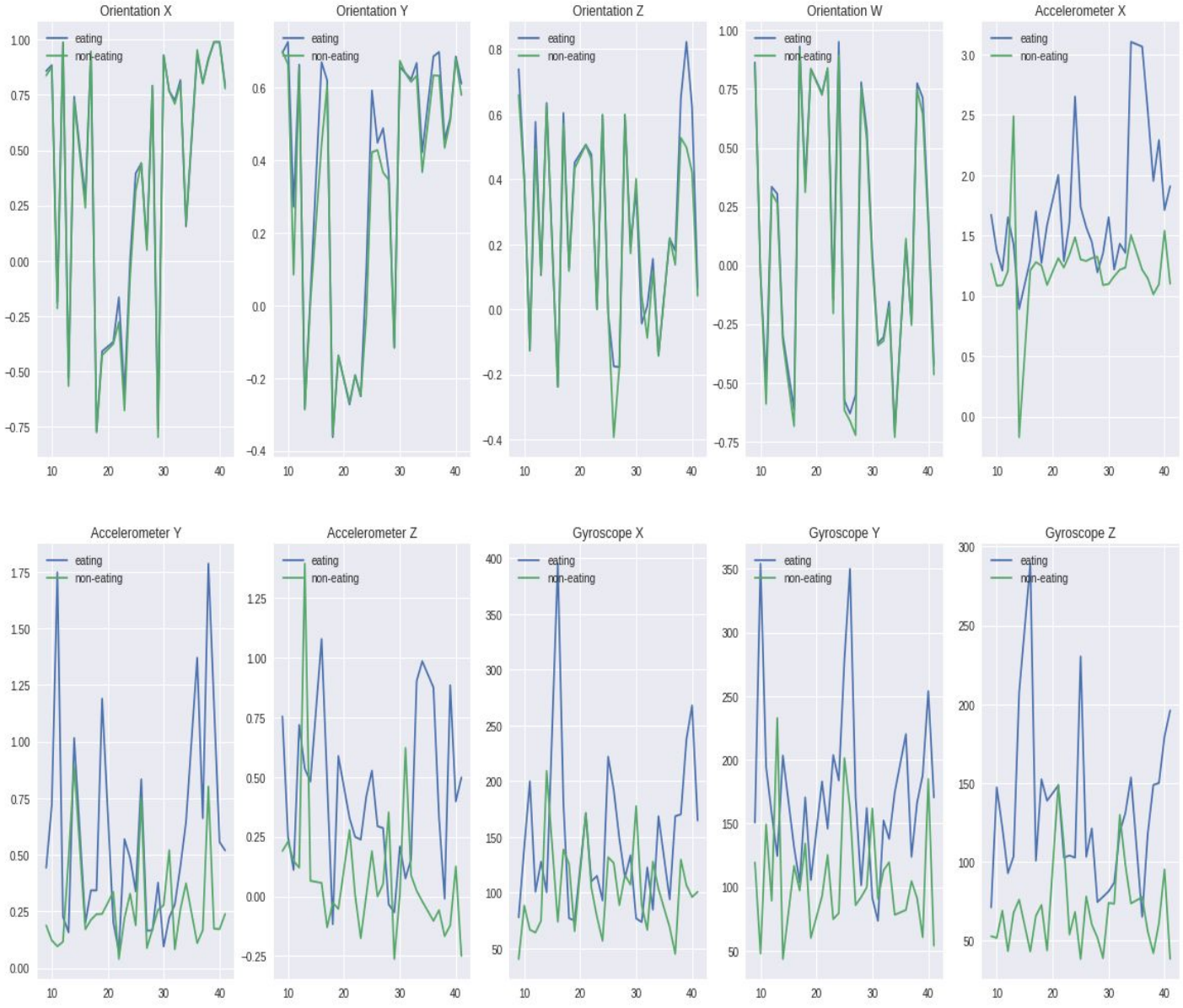
Feature #3a: Minimum

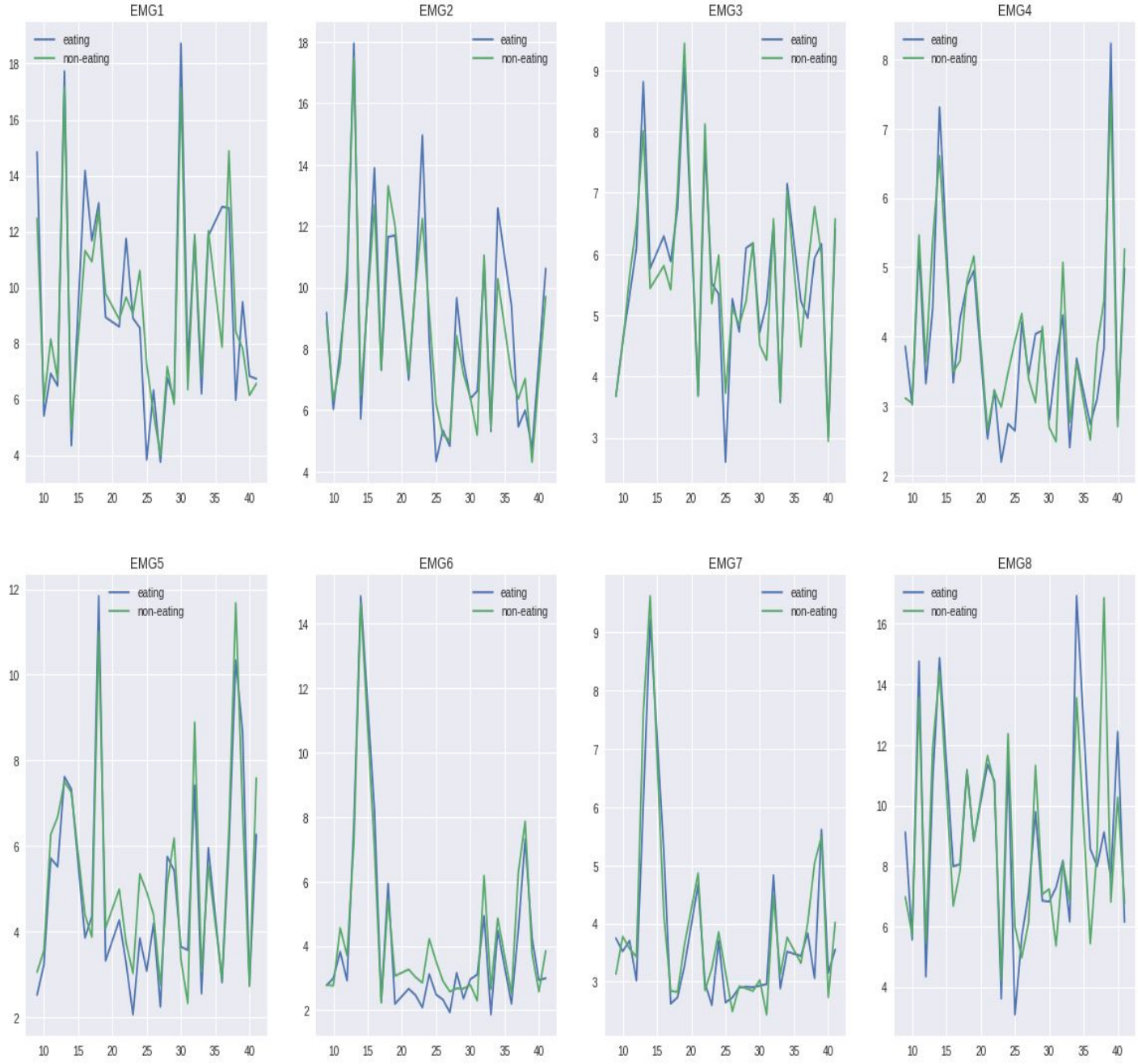


Feature #3b: Maximum

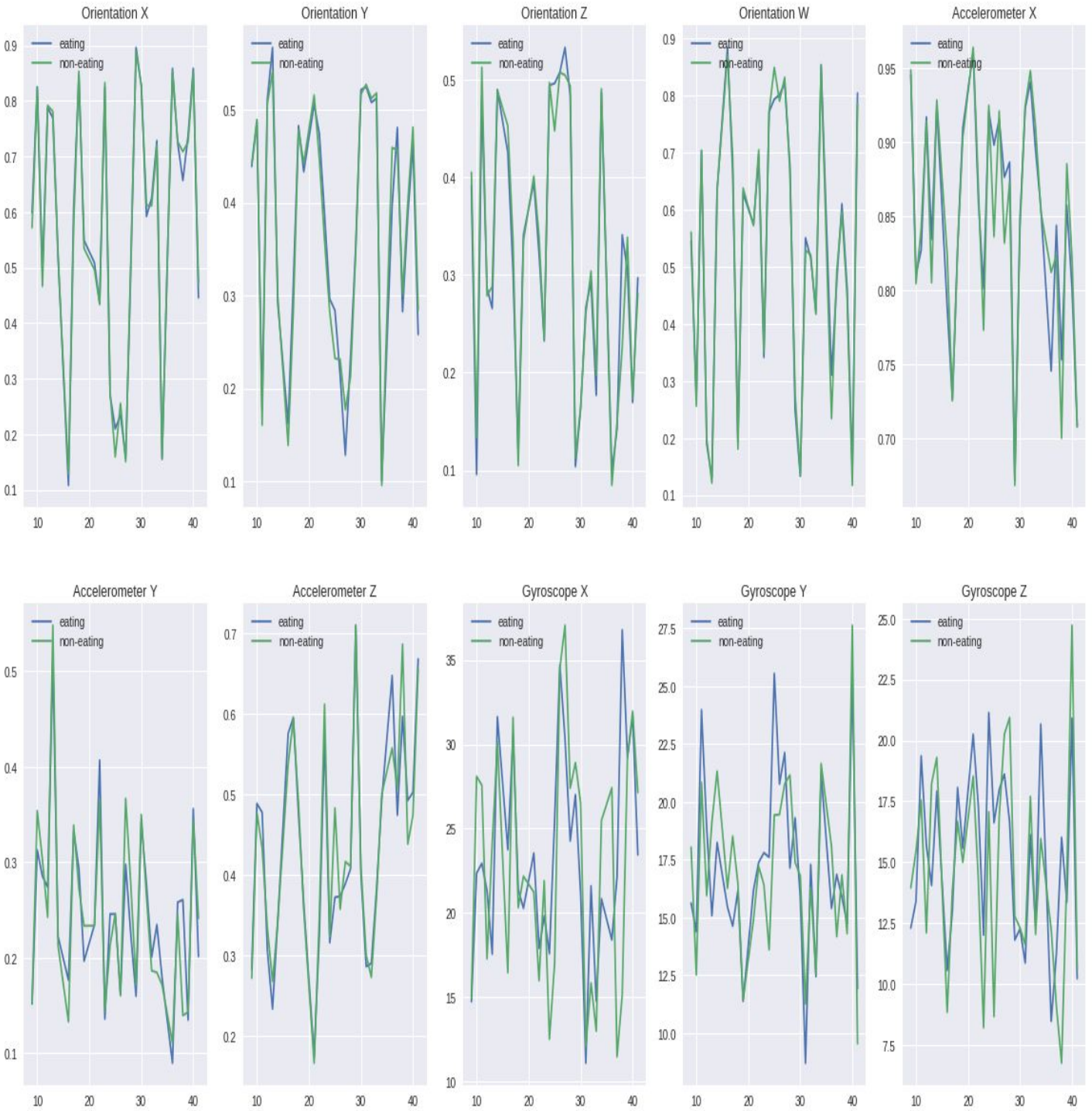


Feature #3b: Maximum

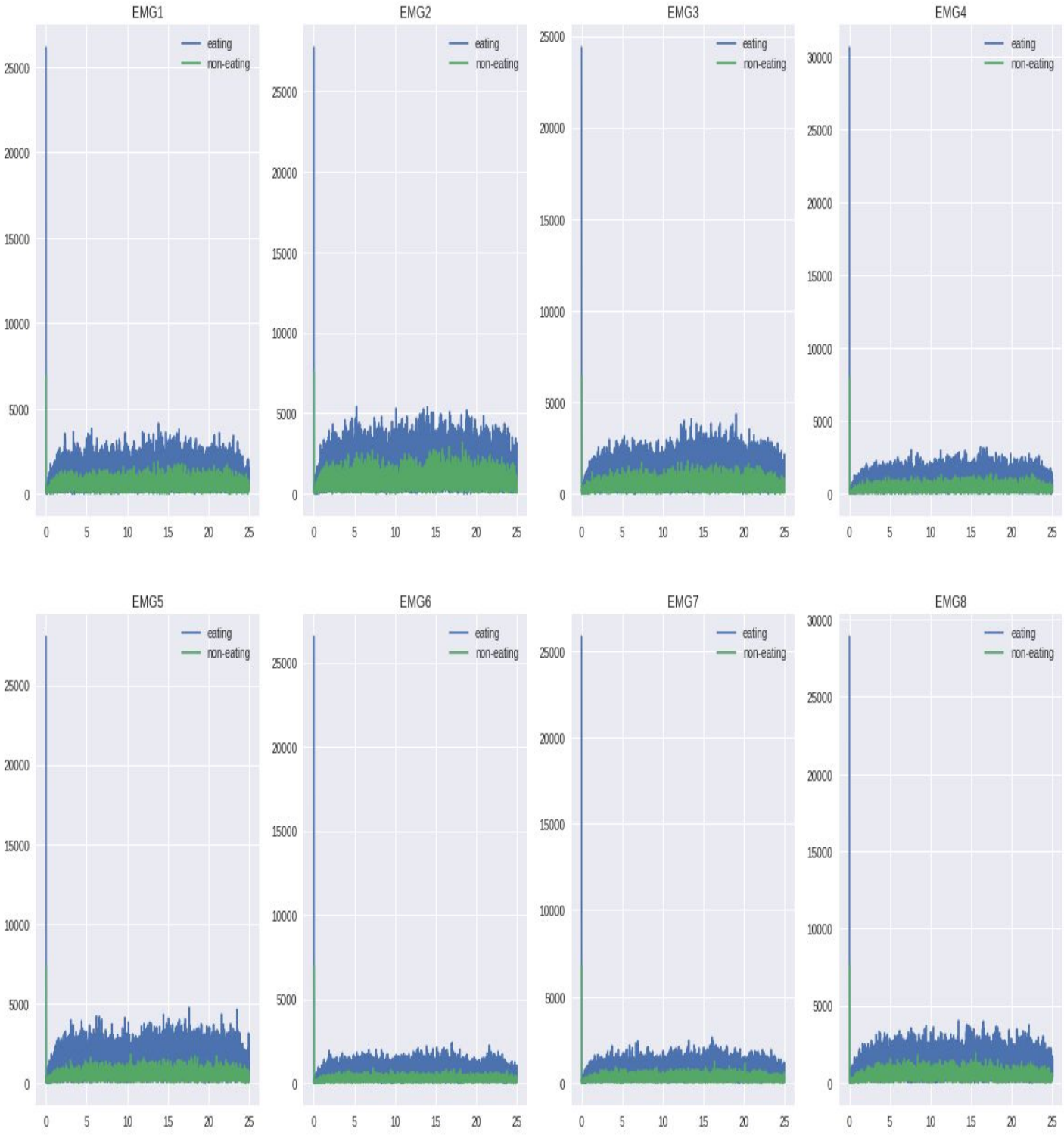



Feature #4: Root Mean Square
EMG1
EMG2
EMG3
EMG4
EMG5
EMG6
EMG7
EMG8
eating
non-eating

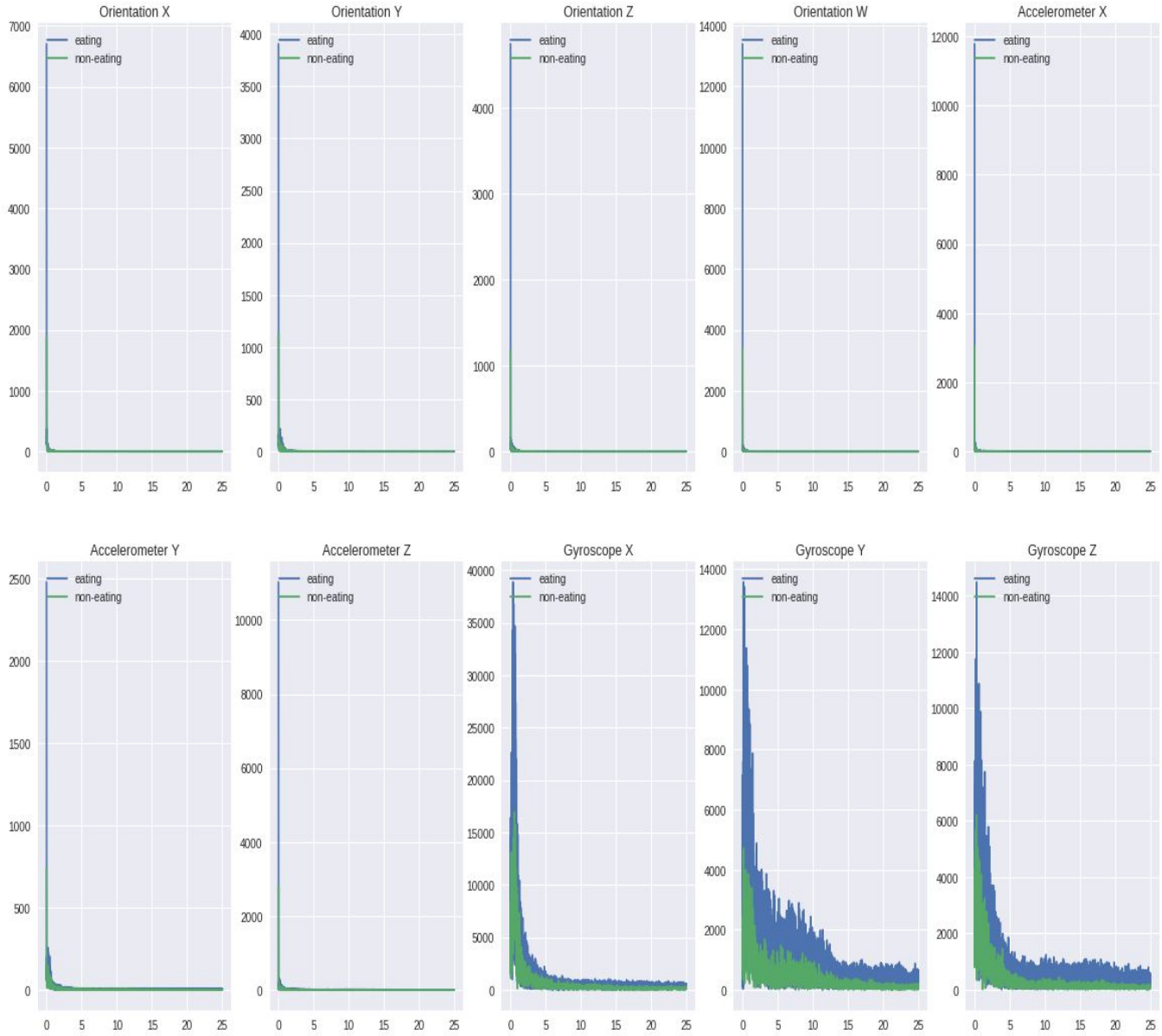
Feature #4: Root Mean Square



Feature #5: Fourier Transform



Feature #5: Fourier Transform



e) Discuss whether your initial intuition about the features that you selected holds true or not.

Mean:
We can see clear difference in the means for Gyroscope across X,Y and Z axes, though not as much when compared to other features we chose. Hence our intuition holds true to an extent.

Standard Deviation:
Standard deviation, among all the graphs shows the strongest difference in values. For Accelerometer, Orientation and Gyroscope we can see obvious separation in values for eating and non-eating activities. This feature will be very useful in classification.

Minimum/Maximum:
These features also show clear difference in values across all the sensors. It seems to be a strong feature to use. Our intuition holds true that non-eating activities have higher maximums and lower minimums than eating activities.

RMS:
Though it's easy to see differences across Gyroscope, other sensors were not as much as we expected. Hence our intuition did not hold true in this case.

FFT:
Across all the features we can see clear differences in values of sensors for eating and non-eating activities. The eating activities consistently have higher values than non-eating activities in this case, and can make a good feature. Thus our intuition holds true.

Phase 3: PCA

The step involves reduction of the feature space and keeping only those features which show maximum distance between the two classes. We will use Principal Component Analysis technique discussed in class for this purpose.

Q. Arranging the feature matrix

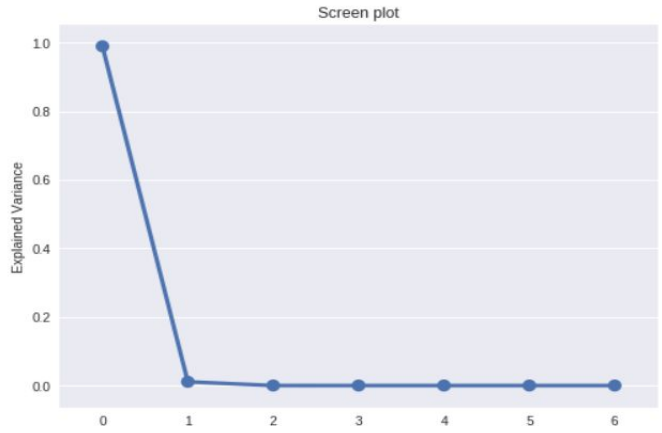
We extracted a total of 6 features across eating and non-eating activities for all the features. We've used these features as the columns of the feature matrix and have chosen the axes where there is a clear separation in values. We get 2 feature matrices - for eating and non-eating activities. From the PCA, we get the new feature matrix which contains the principle components that capture the direction of highest variance.

Q. Execution of PCA

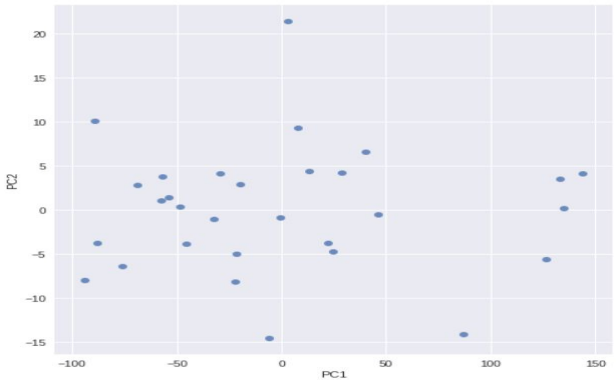
Use the PCA function to run PCA on your feature matrix. Show all the Eigenvectors in a plot.

For Eating Activities:

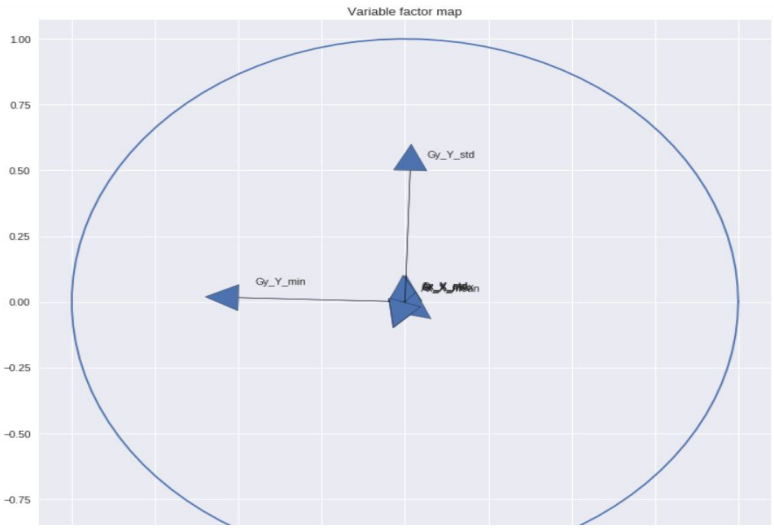
We run PCA with 6 components.
Our Cumulative Explained Variance:
1st Principal Component: 98.903384%
2nd Principal Component: 99.984315%
3rd Principal Component: 99.992898%
4th Principal Component: 99.997321%
5th Principal Component: 99.999506%
6th Principal Component: 99.999924%
7th Principal Component: 100%



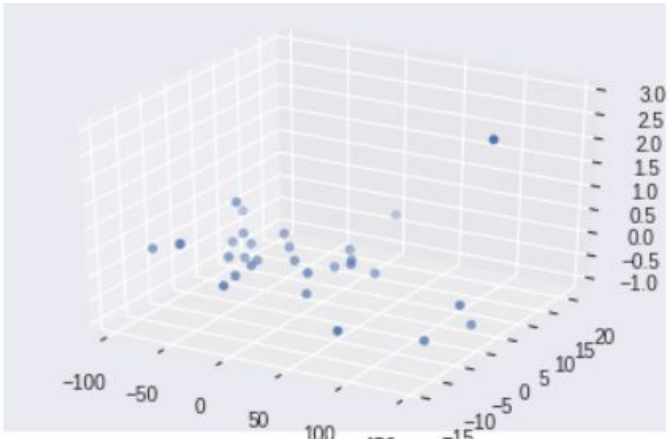
Visualizing our top 2 components in 2-D:



Our variable factor map:

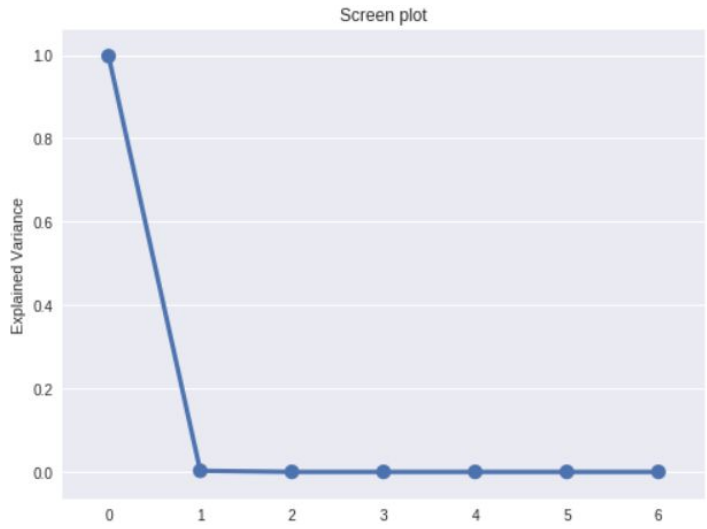


Visualizing top 3 components in 3-D:

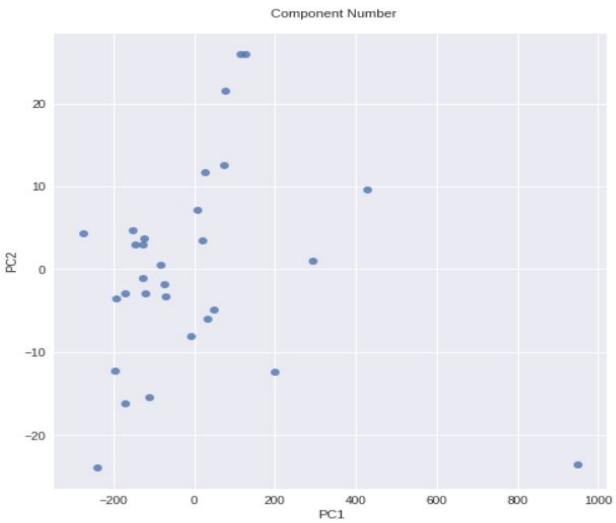


For 'Non-Eating' Activity:

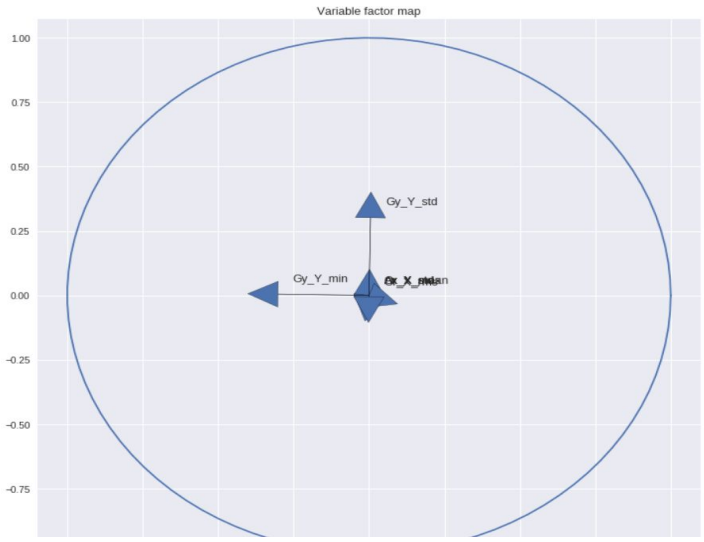
1st Principal Component: 99.726%
2nd Principal Component: 99.992796%
3rd Principal Component: 99.997018%
4th Principal Component: 99.997321%
5th Principal Component: 99.999885%
6th Principal Component: 99.999992%
7th Principal Component: 100%



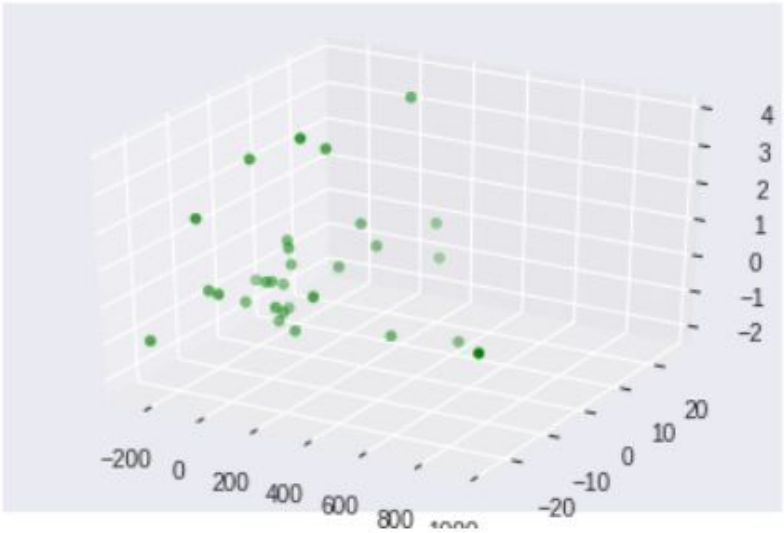
Principal Components in 2 dimensions:



Variable factor map:



Visualizing top 3 components in 3-D:



Q. Results of PCA

Performing the PCA can help us reduce the feature matrix to n*3 dimensions. This new feature matrix captures 99.997% of variance.

New feature matrix for ‘eating’ activity:

PC1	PC2	PC3
-56.931021	3.820634	-0.179274
-76.387198	-6.345230	0.290514
39.833343	6.546899	-0.505611
-32.198075	-1.015903	-0.147131
134.560234	0.201507	-0.348987

New Feature matrix for ‘non-eating’ activity:

PC1	PC2	PC3
-121.112883	-2.959611	-0.418650
198.831079	-12.372468	0.197434
77.722251	21.487771	3.532655
291.927715	1.085142	-1.591288
-193.263617	-3.559945	-1.048253

Q. Argue whether doing PCA was helpful or not.

PCA was primarily useful for reducing the dimensions of our feature matrix. The Gyroscope values contributes to most to the separation, as was clear from the graphs in phase 2, but other features also had a significant contribution. Thus they could not be dismissed, and performing the PCA preserved their variance (almost 100% the top 3 components) when generating new features.

Codes in File:

Phase 1:

- Dataprocessing_final.m

Phase 2:

- IMU_extraction.ipynb
- IMU_extraction.pdf
- EMG_extraction.ipynb
- EMG_extraction.pdf

Phase 3:

- PCA_Eating.ipynb
- PCA_Eating.pdf
- PCA_Noneating.ipynb
- PCA_Noneating.pdf