Phase 2: IMU Data

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

```
column = ['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Acceleromet
```

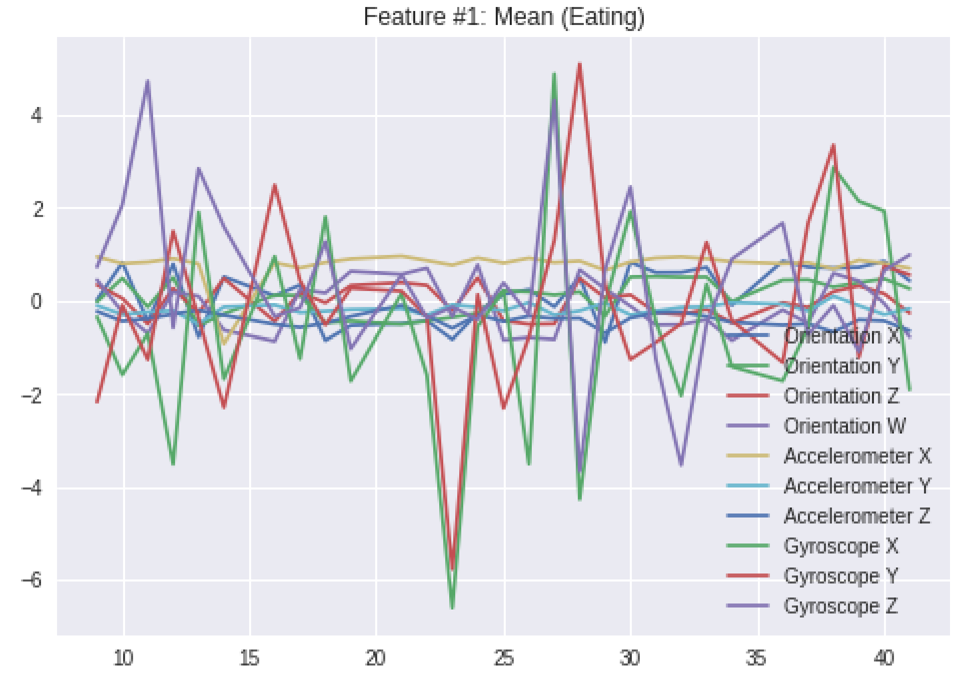
```
df = pd.read_csv("IMUeatingfile.csv")
df.head()
```

	Number	Timestamp	Orientation X	Orientation Y	Orientation Z	Orientation W	Acceler
0	9	1.500000e+12	-0.666	-0.665	0.292	0.169	
1	9	1.500000e+12	-0.667	-0.665	0.292	0.169	
2	9	1.500000e+12	-0.667	-0.664	0.292	0.169	
3	9	1.500000e+12	-0.667	-0.664	0.293	0.169	
4	9	1.500000e+12	-0.667	-0.664	0.293	0.169	

Feature 1: Mean

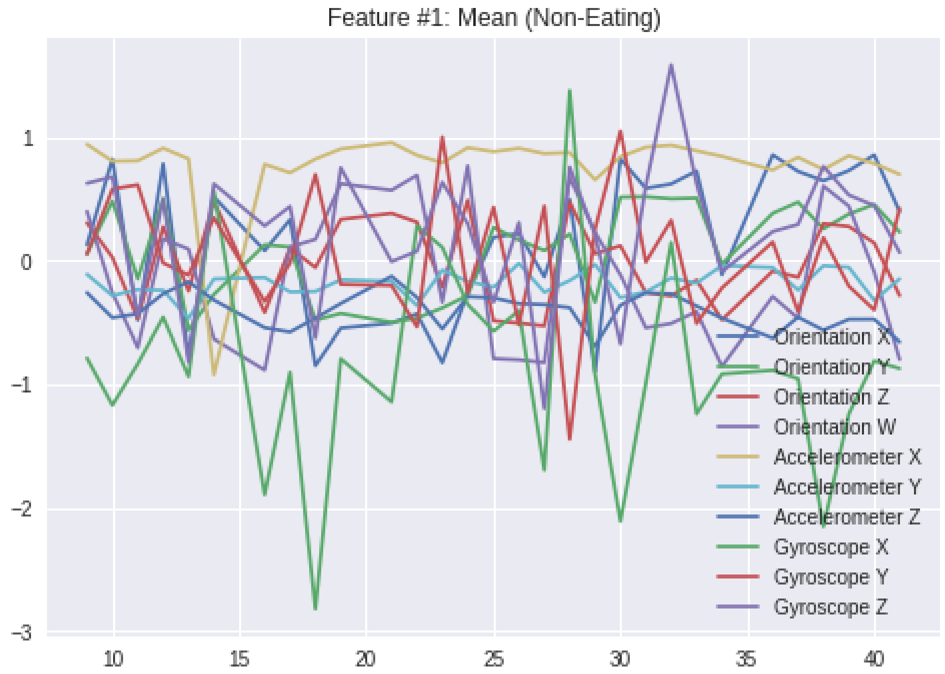
```
df = pd.read_csv("IMUeatingfile.csv")
a = df['Number'].unique()
a.sort()

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 column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i
    col = j # Column IMU1, IMU2...
    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 ",col )


for i in column:
  plt.plot(x1[i], y1[i])
plt.legend(['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Acceleromo
plt.title("Feature #1: Mean (Eating)")
plt.figure(figsize=(40,40))
plt.show()
```

```
Done for col  Orientation X
Done for col  Orientation Y
Done for col  Orientation Z
Done for col  Orientation W
Done for col  Accelerometer X
Done for col  Accelerometer Y
Done for col  Accelerometer Z
Done for col  Gyroscope X
Done for col  Gyroscope Y
Done for col  Gyroscope Z
```



```
<Figure size 2880x2880 with 0 Axes>
```

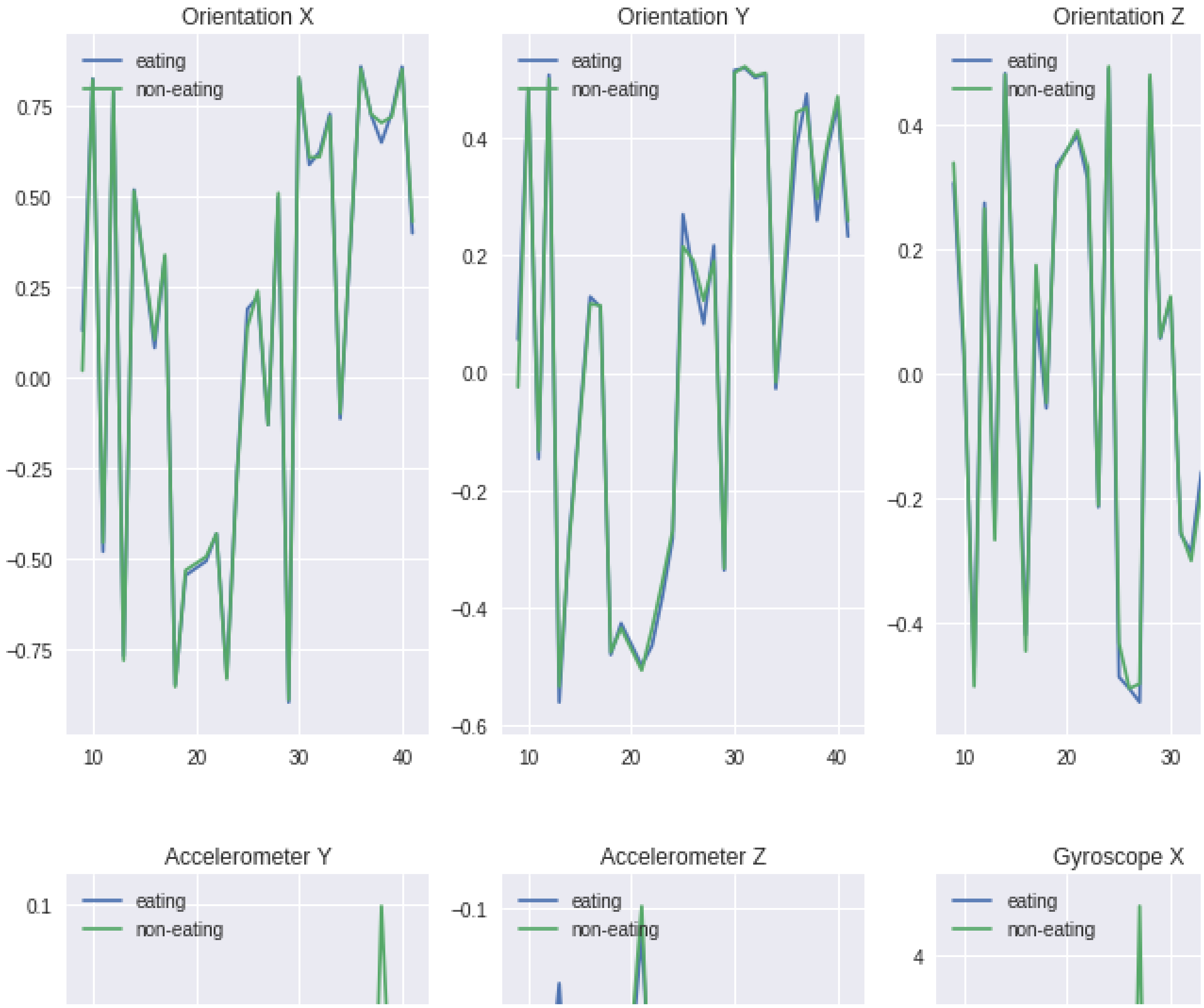
```
df = pd.read_csv("IMUnoneatingfile.csv")
a = df['Number'].unique()
a.sort()

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

x2, y2 = {}, {}
# Mean of EMG1 for all Eating
means_eating = {}
for j in column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i
    col = j # Column IMU1, IMU2...
    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
  x2[j], y2[j] = u, v
  #print("Done for col ",col )


for i in column:
  plt.plot(x2[i], y2[i])

plt.legend(['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Acceleromε
plt.title("Feature #1: Mean (Non-Eating)")
plt.figure(figsize=(40,40))
plt.show()
```



```
<Figure size 2880x2880 with 0 Axes>
```

```
fig = plt.figure(figsize=(20,15))
fig.suptitle('Feature #1: Mean', fontsize=30)
j = 1
for i in column:
  plt.subplot(2, 5, j).set_title('{}'.format(i))
  plt.plot(x2[i], y2[i])
  plt.plot(x1[i], y1[i])
  plt.legend(['eating', 'non-eating'], loc='upper left')
  j = j+1
plt.show()
```

Feature #1: M

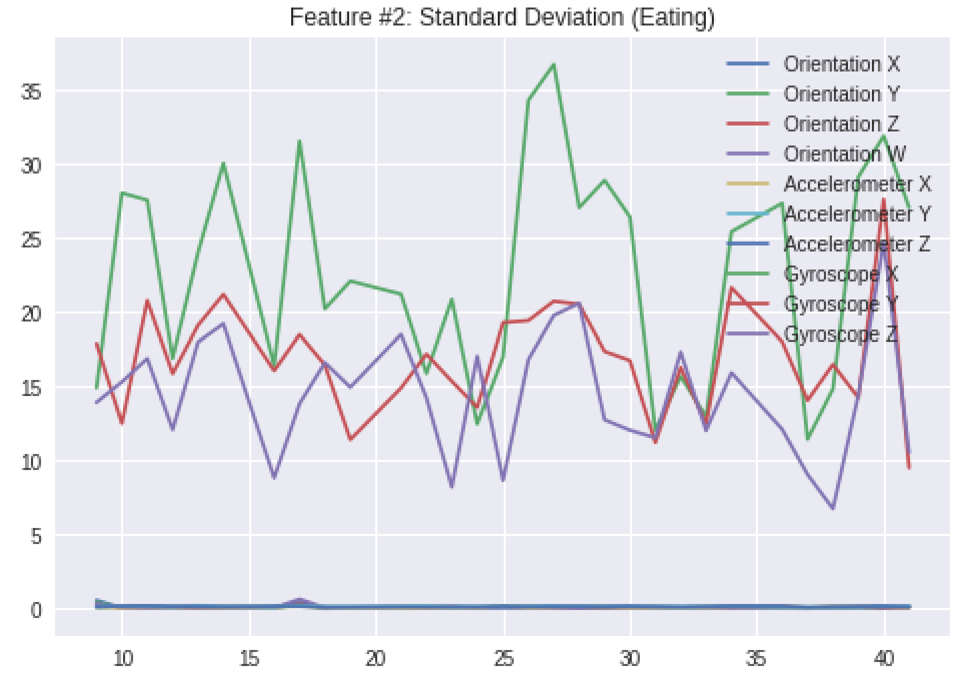


Feature 2: Standard Deviation

```
df = pd.read_csv("IMUeatingfile.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 column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i
    col = j # Column IMU1, IMU2...
    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 ",col )
```

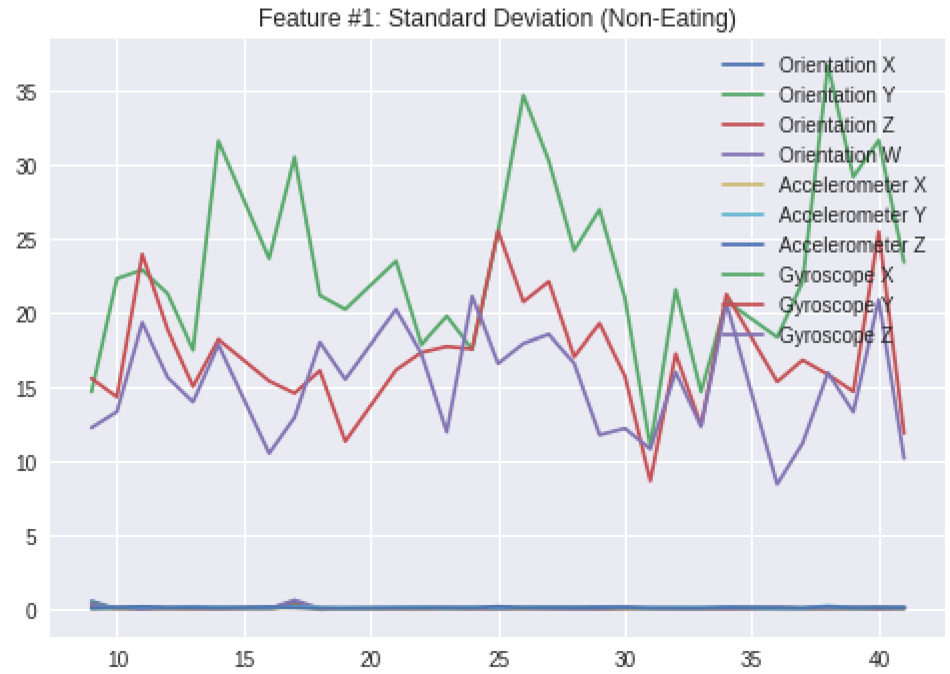
```
for i in column:
  plt.plot(x1[i], y1[i])
plt.legend(['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Accelerom
plt.title("Feature #2: Standard Deviation (Eating)")
plt.figure(figsize=(40,40))
plt.show()
```

```
Done for col  Orientation X
Done for col  Orientation Y
Done for col  Orientation Z
Done for col  Orientation W
Done for col  Accelerometer X
Done for col  Accelerometer Y
Done for col  Accelerometer Z
Done for col  Gyroscope X
Done for col  Gyroscope Y
Done for col  Gyroscope Z
```



```
<Figure size 2880x2880 with 0 Axes>
```

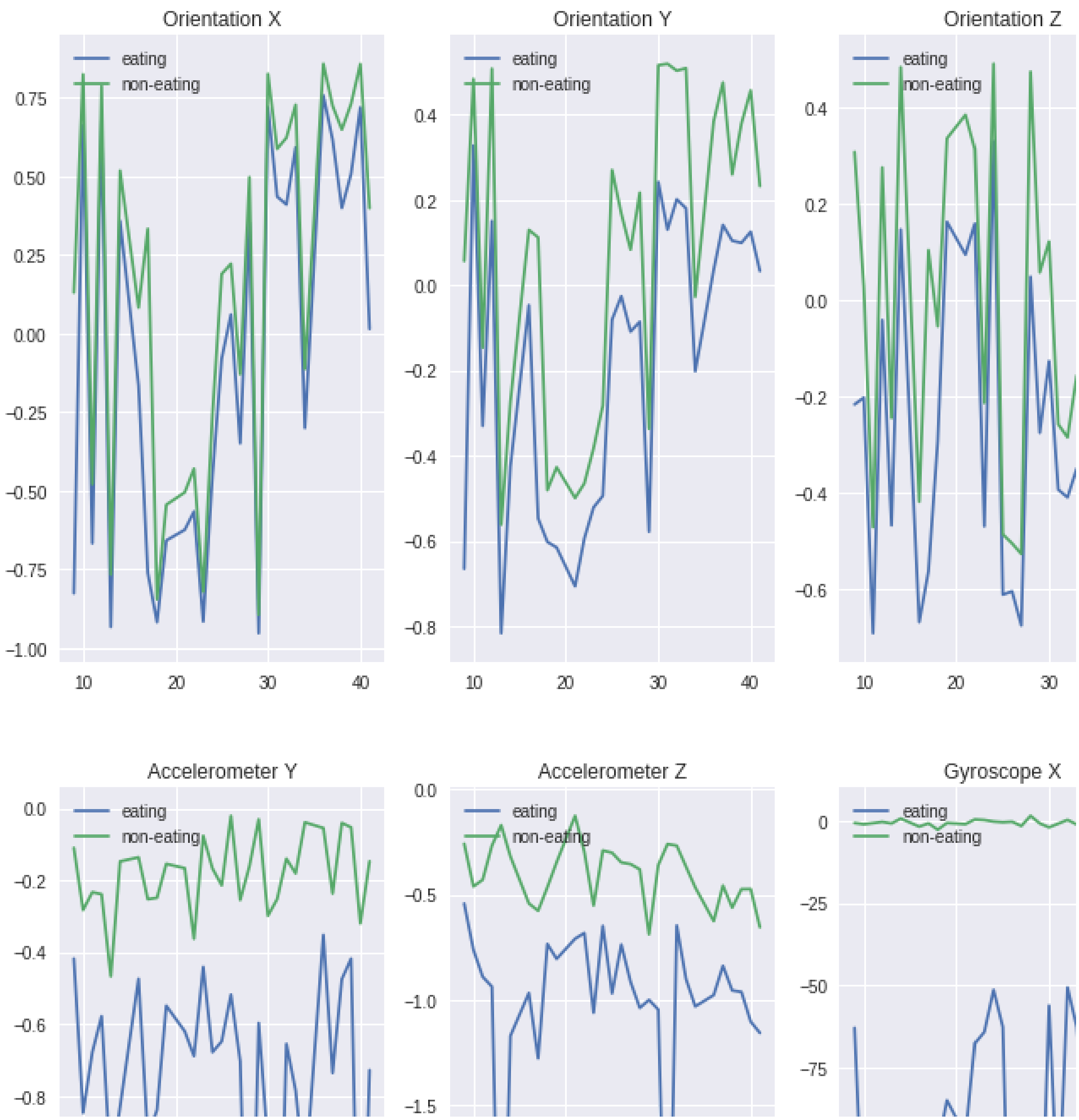
```
df = pd.read_csv("IMUnoneatingfile.csv")
a = df['Number'].unique()
a.sort()

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

x2, y2 = {}, {}
# Mean of EMG1 for all Eating
means_eating = {}
for j in column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i
    col = j # Column IMU1, IMU2...
    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
  x2[j], y2[j] = u, v
  #print("Done for col ",col )


for i in column:
  plt.plot(x2[i], y2[i])
plt.legend(['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Acceleromo
plt.title("Feature #1: Standard Deviation (Non-Eating)")
plt.figure(figsize=(40,40))
plt.show()
```



```
<Figure size 2880x2880 with 0 Axes>
```

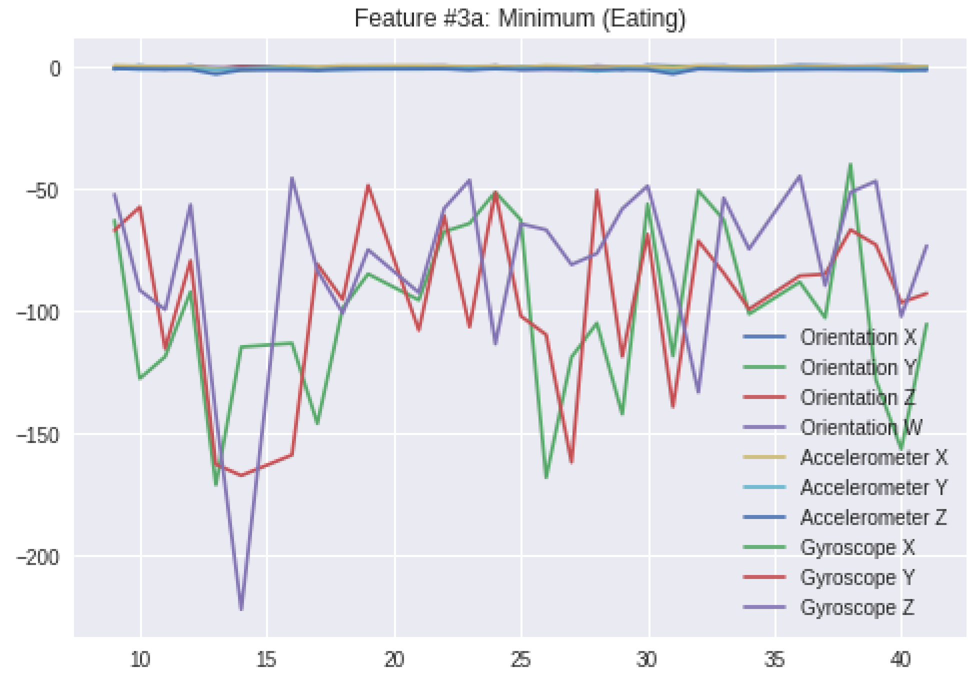
```
fig = plt.figure(figsize=(20,15))
fig.suptitle('Feature #2: Standard Deviation', fontsize=30)
j = 1
for i in column:
  plt.subplot(2, 5, j).set_title('{}'.format(i))
  plt.plot(x1[i], y1[i])
  plt.plot(x2[i], y2[i])
  plt.legend(['eating', 'non-eating'], loc='upper left')
  j = j+1
plt.show()
```

Feature #2: Standar



Feature 3a: Minimum

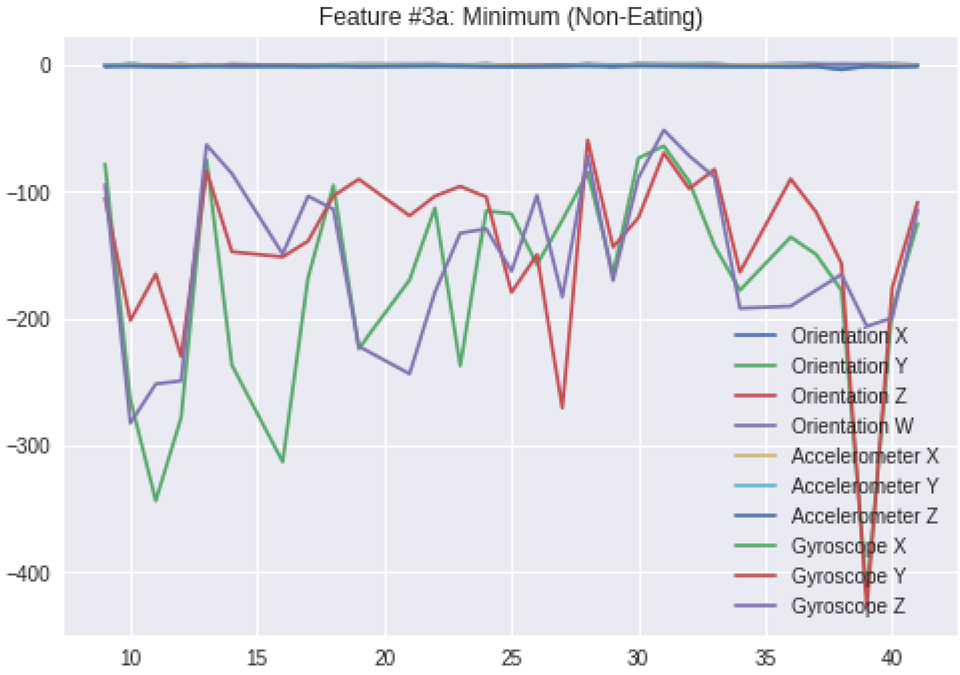
```
df = pd.read_csv("IMUeatingfile.csv")
a = df['Number'].unique()
a.sort()

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 column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i
    col = j # Column IMU1, IMU2...
    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 ",col )


for i in column:
  plt.plot(x1[i], y1[i])
plt.legend(['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Accelerom
plt.title("Feature #3a: Minimum (Eating)")
plt.figure(figsize=(40,40))
plt.show()
```



```
<Figure size 2880x2880 with 0 Axes>
```

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

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

x2, y2 = {}, {}
# Mean of EMG1 for all Eating
means_eating = {}
for j in column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i
    col = j # Column IMU1, IMU2...
    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

  x2[j], y2[j] = u, v
  #print("Done for col ",col )
```

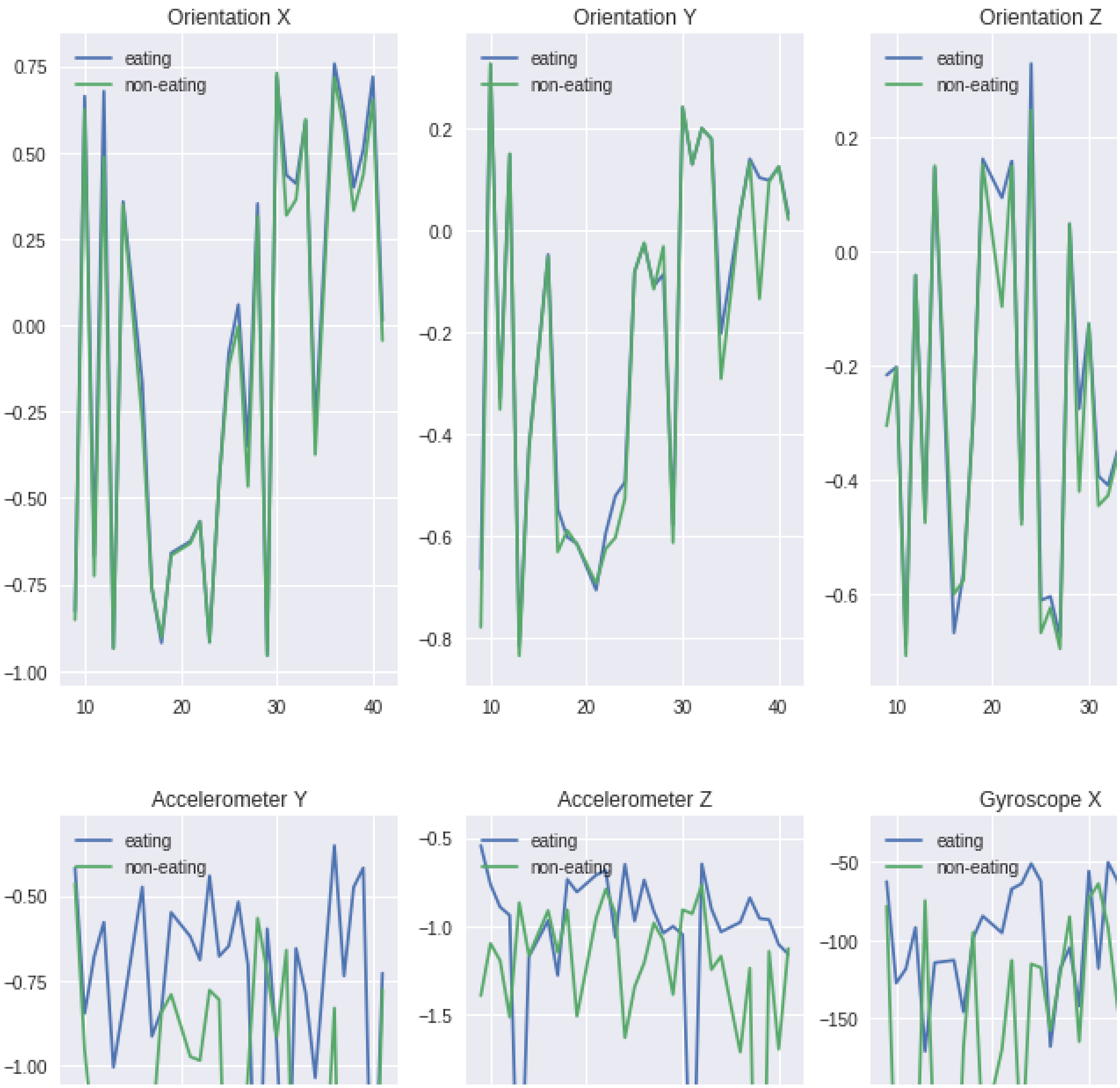
```
for i in column:
  plt.plot(x2[i], y2[i])
plt.legend(['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Accelerom
plt.title("Feature #3a: Minimum (Non-Eating)")
plt.figure(figsize=(40,40))
plt.show()
```



```
<Figure size 2880x2880 with 0 Axes>
```

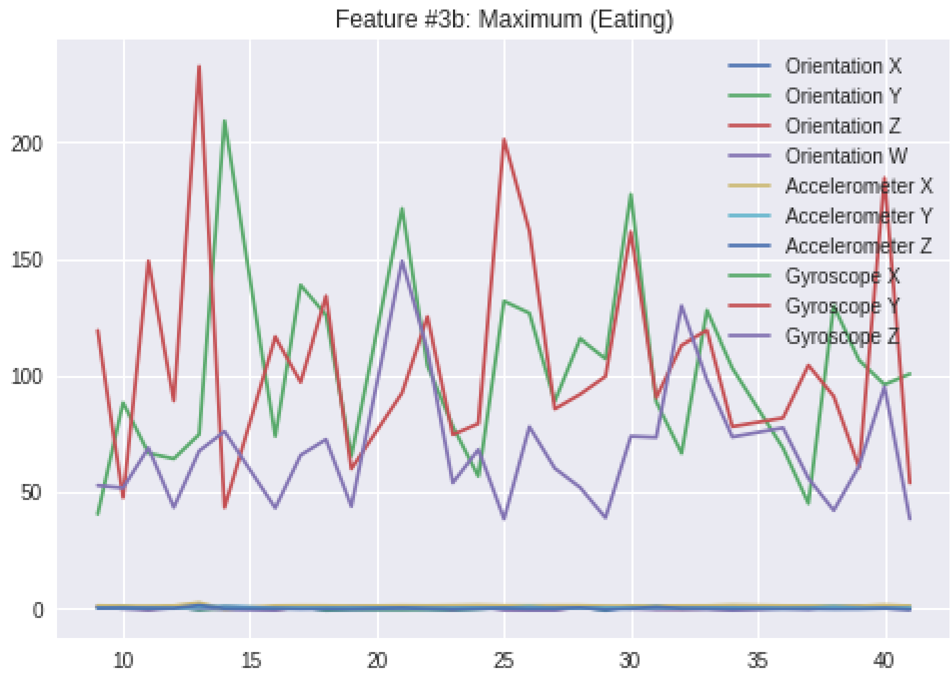
```
fig = plt.figure(figsize=(20,15))
fig.suptitle('Feature #3a: Minimum', fontsize=30)
j = 1
for i in column:
  plt.subplot(2, 5, j).set_title('{}'.format(i))
  plt.plot(x1[i], y1[i])
  plt.plot(x2[i], y2[i])
  plt.legend(['eating', 'non-eating'], loc='upper left')
  j = j+1
plt.show()
```

Feature #3a: Mi



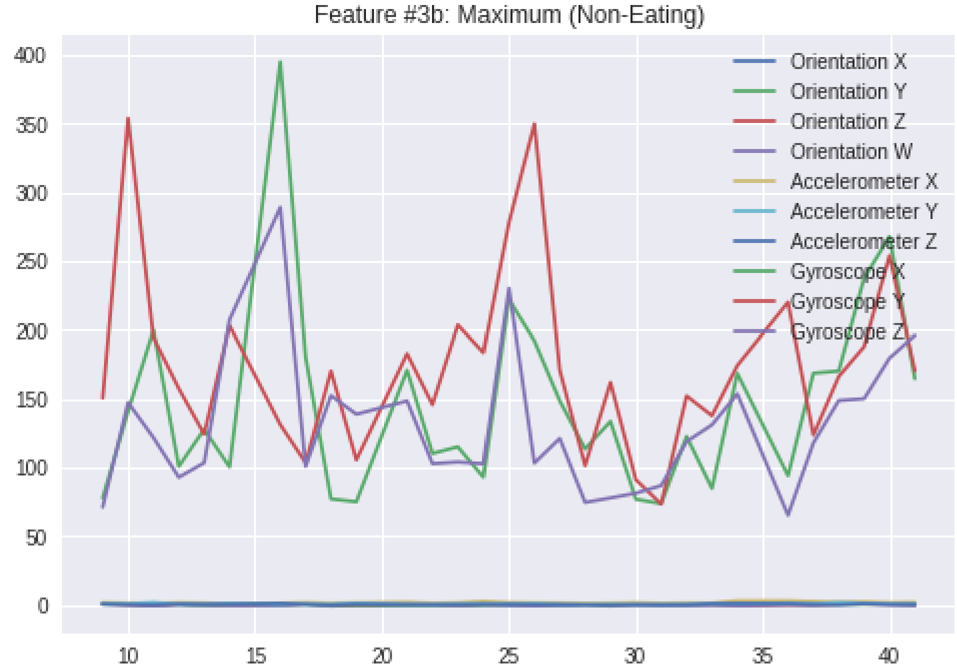
Feature 3b: Maximum

```
df = pd.read_csv("IMUeatingfile.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 column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i
    col = j # Column IMU1, IMU2...
    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 ",col )


for i in column:
  plt.plot(x1[i], y1[i])
plt.legend(['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Accelerom
plt.title("Feature #3b: Maximum (Eating)")
plt.figure(figsize=(40,40))
plt.show()
```



```
<Figure size 2880x2880 with 0 Axes>
```

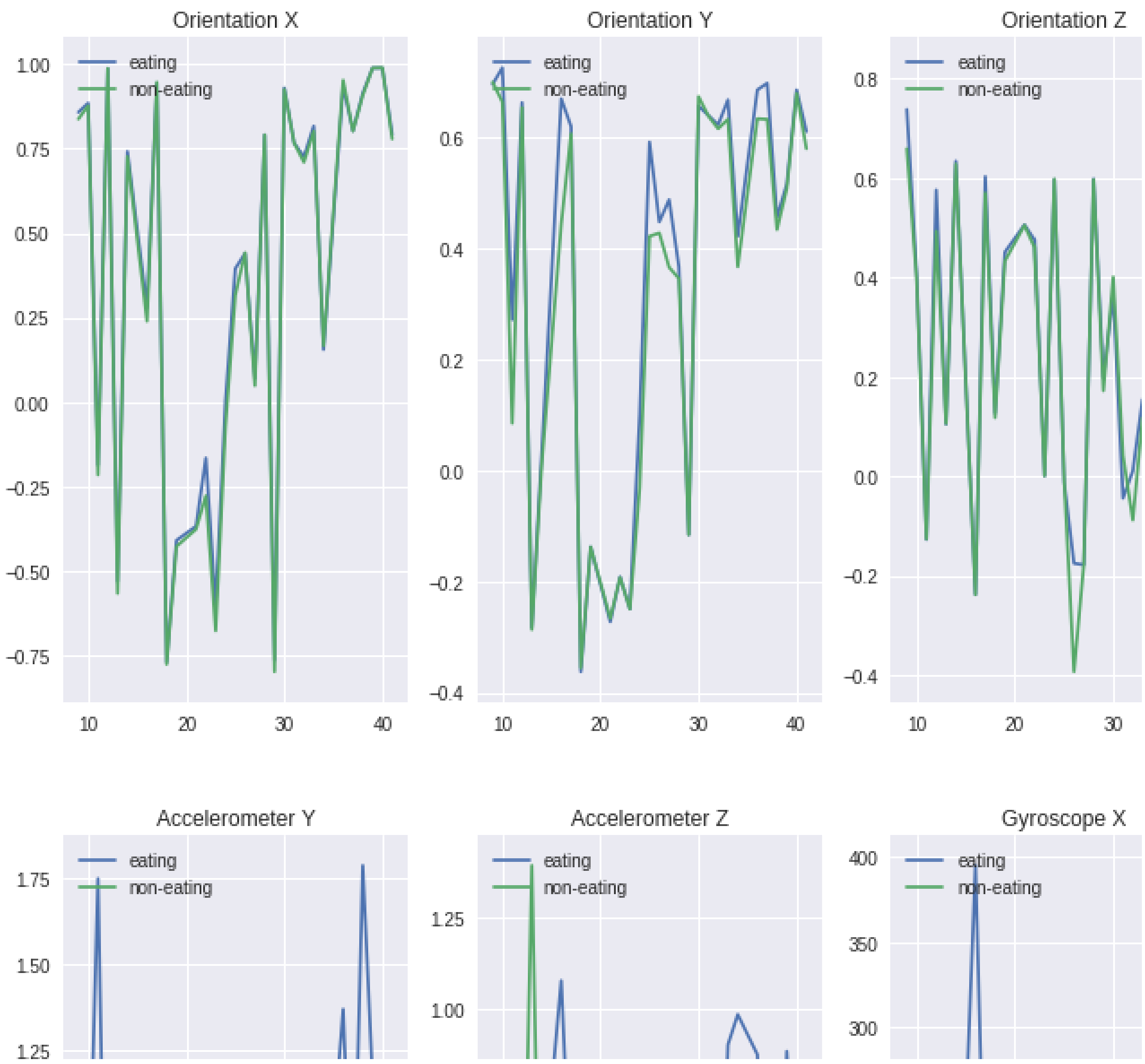
```
df = pd.read_csv("IMUnoneatingfile.csv")
a = df['Number'].unique()
a.sort()

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

x2, y2 = {}, {}
# Mean of EMG1 for all Eating
means_eating = {}
for j in column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i
    col = j # Column IMU1, IMU2...
    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
  x2[j], y2[j] = u, v
  #print("Done for col ",col )



for i in column:
  plt.plot(x2[i], y2[i])
plt.legend(['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Accelerom
plt.title("Feature #3b: Maximum (Non-Eating)")
plt.figure(figsize=(40,40))
plt.show()
```



```
<Figure size 2880x2880 with 0 Axes>
```

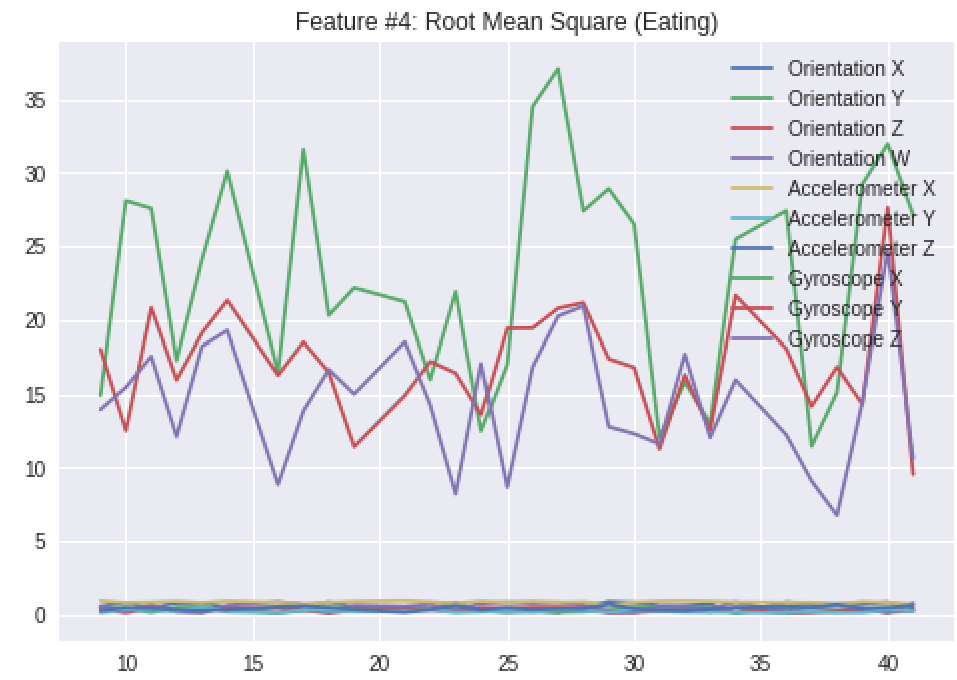
```
fig = plt.figure(figsize=(20,15))
fig.suptitle('Feature #3b: Maximum', fontsize=30)
j = 1
for i in column:
  plt.subplot(2, 5, j).set_title('{}'.format(i))
  plt.plot(x2[i], y2[i])
  plt.plot(x1[i], y1[i])

  plt.legend(['eating', 'non-eating'], loc='upper left')
  j = j+1
plt.show()
```

Feature #3b: Ma



Feature 4: Root Mean Square

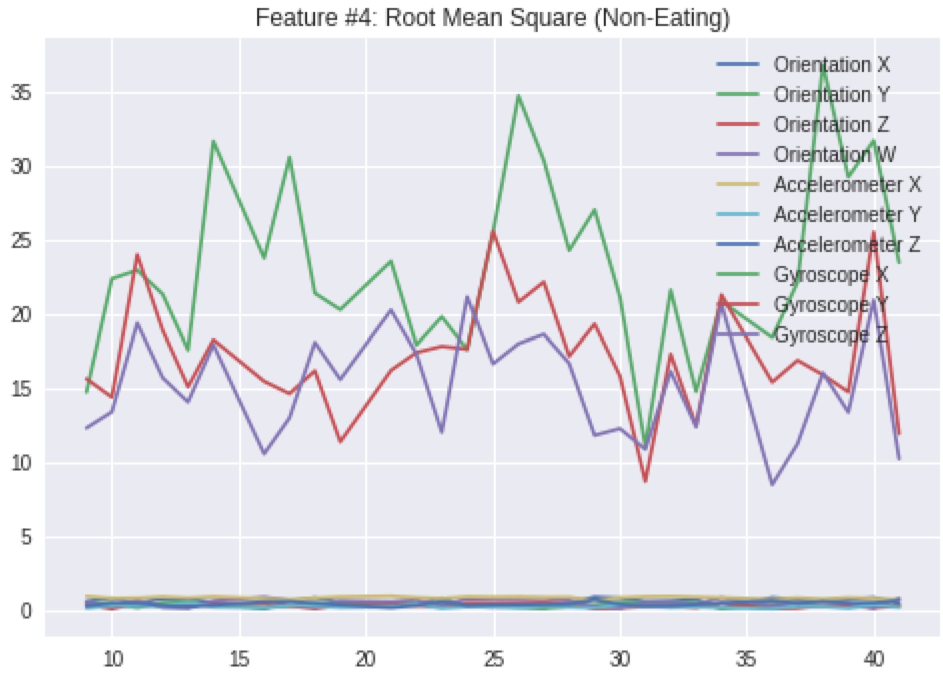
```
df = pd.read_csv("IMUeatingfile.csv")
a = df['Number'].unique()
a.sort()

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 column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i

    col = j # Column IMU1, IMU2...
    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 ",col )


for i in column:
  plt.plot(x1[i], y1[i])
plt.legend(['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Accelerom
plt.title("Feature #4: Root Mean Square (Eating)")
plt.figure(figsize=(40,40))
plt.show()
```



```
<Figure size 2880x2880 with 0 Axes>
```

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

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

x2, y2 = {}, {}
# Mean of EMG1 for all Eating
means_eating = {}
for j in column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i
    col = j # Column IMU1, IMU2...
    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
  x2[j], y2[j] = u, v

  #print("Done for col ",col )
```

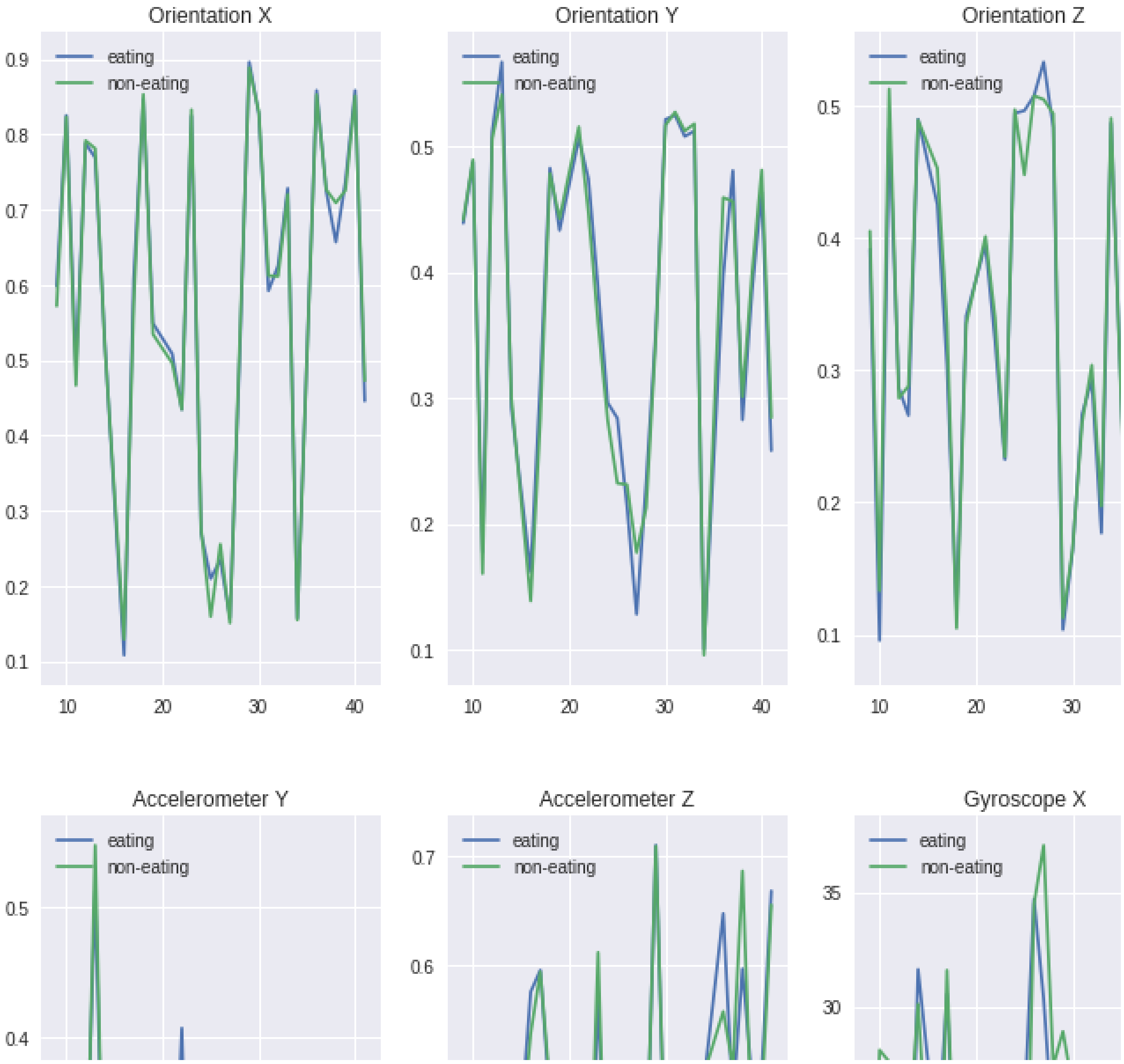
```
for i in column:
  plt.plot(x2[i], y2[i])
plt.legend(['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Accelerom
plt.title("Feature #4: Root Mean Square (Non-Eating)")
plt.figure(figsize=(40,40))
plt.show()
```



```
<Figure size 2880x2880 with 0 Axes>
```

```
fig = plt.figure(figsize=(20,15))
fig.suptitle('Feature #4: Root Mean Square', fontsize=30)
j = 1
for i in column:
  plt.subplot(2, 5, j).set_title('{}'.format(i))
  plt.plot(x2[i], y2[i])
  plt.plot(x1[i], y1[i])
  plt.legend(['eating', 'non-eating'], loc='upper left')
  j = j+1
plt.show()
```

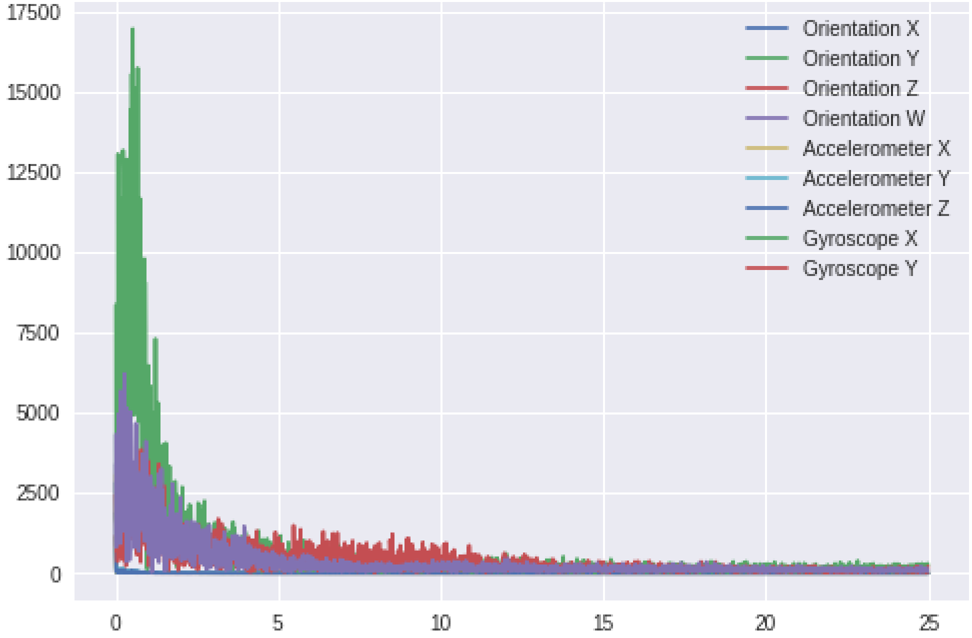
Feature #4: Root Mea



Feature 5: Fast Fourier Transform

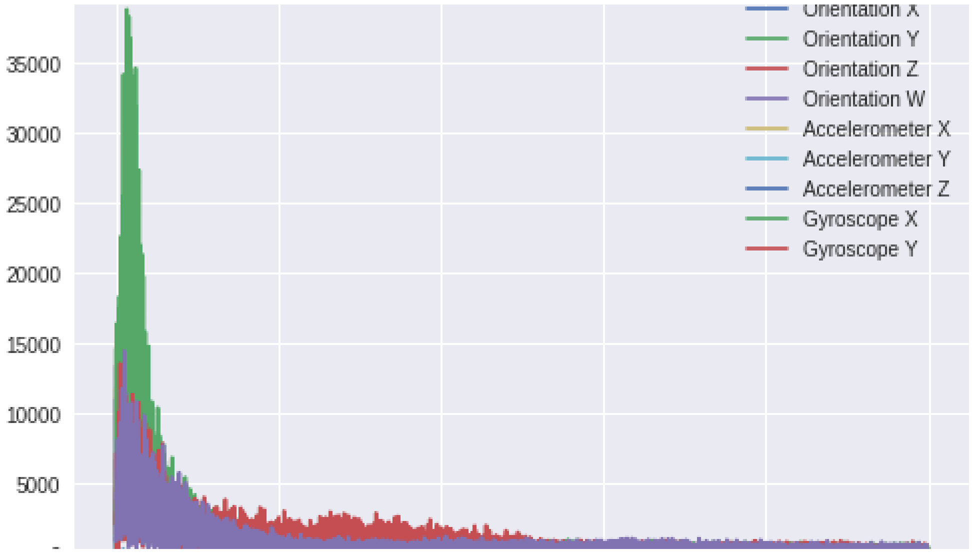
```
import scipy.fftpack

df = pd.read_csv("IMUeatingfile.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 column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i
    col = j # Column IMU1, IMU2...

    yf = scipy.fft(df[col].values)
    x = scipy.fftpack.fftfreq(yf.size, 1 / 50)
  x1[j], y1[j] = x, yf
  plt.legend(['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Acceler
  plt.title("Feature #5: Fourier Transform (Eating)")
  plt.plot(x[:x.size//2], abs(yf)[:yf.size//2])
```

Feature #5: Fourier Transform (Eating)



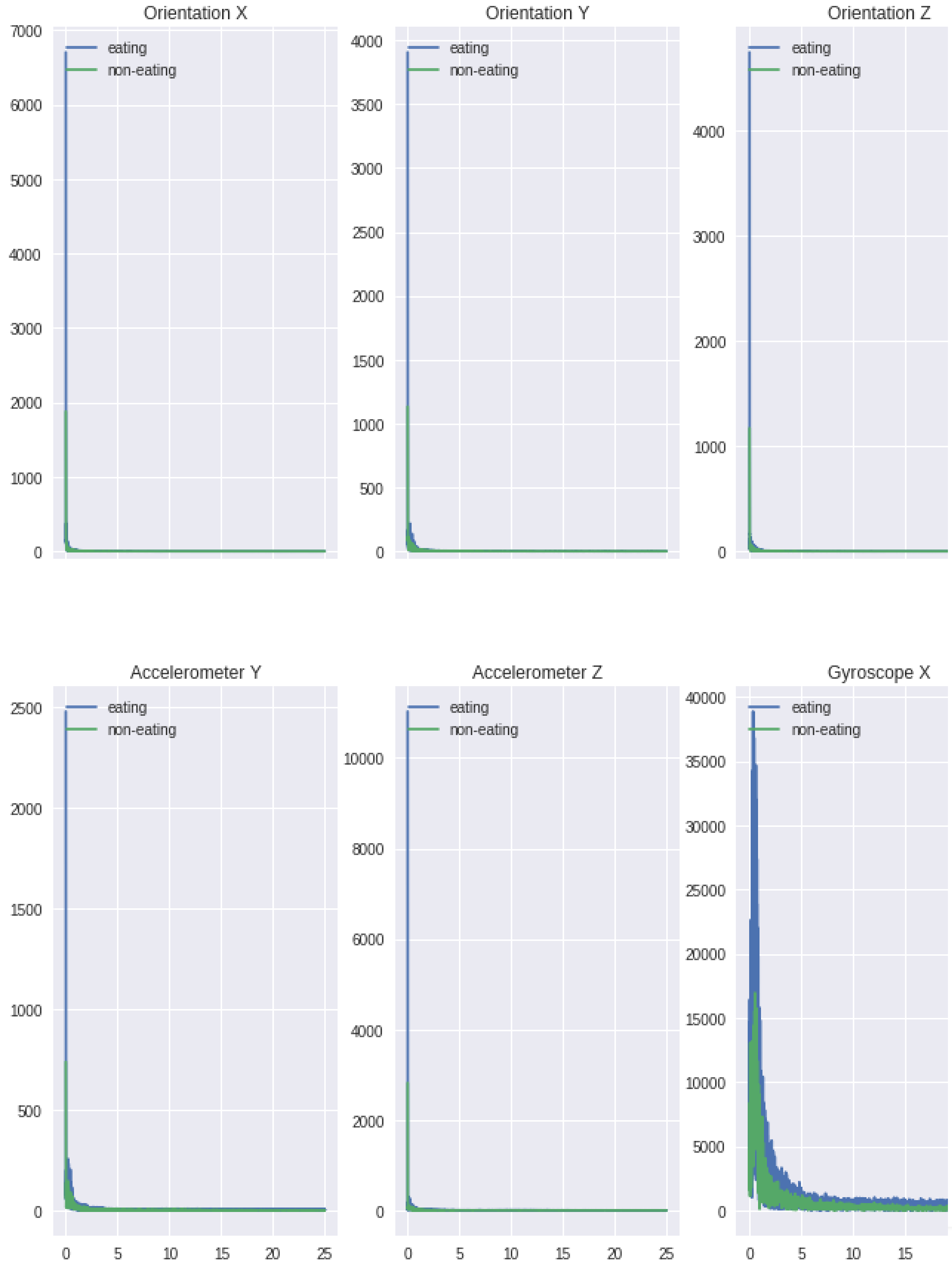
```
import scipy.fftpack

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

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

x2, y2 = {}, {}
# Mean of EMG1 for all Eating
means_eating = {}
for j in column:
  for i in a: # i is no of persons
    df = li[i] #Returns df for person i
    col = j # Column IMU1, IMU2...

    yf = scipy.fft(df[col].values)
    x = scipy.fftpack.fftfreq(yf.size, 1 / 50)
  x2[j], y2[j] = x, yf
  plt.legend(['Orientation X', 'Orientation Y', 'Orientation Z', 'Orientation W', 'Acceler
  plt.title("Feature #5: Fourier Transform (Non-Eating)")
  plt.plot(x[:x.size//2], abs(yf)[:yf.size//2])
```

Feature #5: Fourier Transform (Non-Eating)



```
fig = plt.figure(figsize=(20,15))
fig.suptitle('Feature #5: Fourier Transform', fontsize=30)
j = 1
for i in column:
  plt.subplot(2, 5, j).set_title('{}'.format(i))
  plt.plot(x2[i][:x2[i].size//2], abs(y2[i])[:y2[i].size//2])
  plt.plot(x1[i][:x1[i].size//2], abs(y1[i])[:y1[i].size//2])
  plt.legend(['eating', 'non-eating'], loc='upper left')
  j = j+1
plt.show()
```

Feature #5: Fourier


Orientation X
7000
6000
5000
4000
3000
2000
1000
0
eating
non-eating
Orientation Y
4000
3500
3000
2500
2000
1500
1000
500
0
eating
non-eating
Orientation Z
4000
3000
2000
1000
0
eating
non-eating
Accelerometer Y
2500
2000
1500
1000
500
0
eating
non-eating
0
5
10
15
20
25
Accelerometer Z
10000
8000
6000
4000
2000
0
eating
non-eating
0
5
10
15
20
25
Gyroscope X
40000
35000
30000
25000
20000
15000
10000
5000
0
eating
non-eating
0
5
10
15