Walter Piper

Springboard Capstone 3

Muse EEG : Consumer-Driven Discovery

Problem identification / Introduction

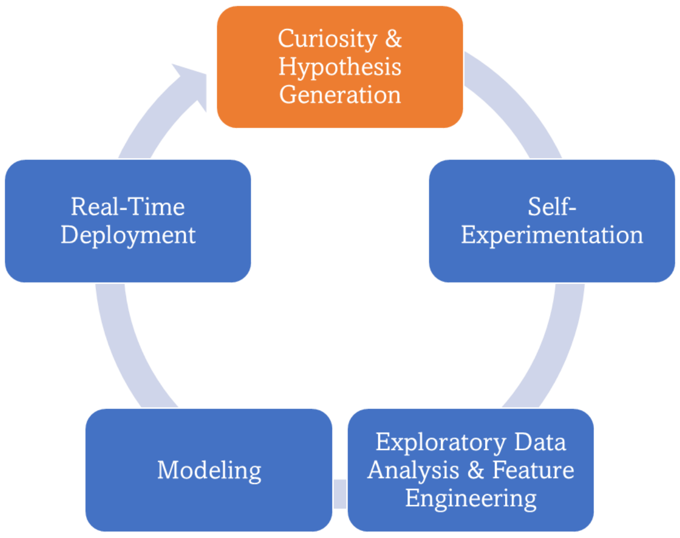
The field of brain-computer interfacing is currently in the midst of a major transition. This once esoteric and academic subject has begun to move out of the lab and into the lives of real people. Perhaps the quintessential idea of a brain-computer interface is a brain implant that allow a paralyzed person to move robotic limbs. While this has been achieved in recent years, it is a very different goal from the non-medical, non-invasive interfaces that would be accessible to the average person.

A number of non-invasive interface start-ups are working hard to create their first products, but the applications for these technologies are still limited in scope. For example, the company Interaxon has a product line of Muse meditation headbands that use electroencephalography (EEG) to non-invasively measure the brain waves of meditators. The EEG data is passed to a mobile app that estimates the calmness of the user’s mind and adjusts an auditory soundscape to facilitate a deeper meditative state. While this approach works brilliantly when a user has eyes closed and is motionless, the reliability quickly diminishes when the user has eyes open or is moving around.

Companies in this non-invasive interface space must market their products for very precise use cases, as Interaxon did for their Muse headbands. Noisy signals must be tamed, which often requires machine learning algorithms to fill in the gaps between moments of clear signal. The muscles of the jaw and neck contribute their own form of noise, as do eye movements. Supplementary features like data from a gyroscope and an accelerometer provide additional resources. If the latent variable of interest is brain calmness, then movement signals can help differentiate electrical signals of muscles from electrical signals of the brain.

The complications inherent in non-invasive brain interfacing are numerous. The marketplace and consumer base for these products are small, but rapidly growing. As the consumer base grows, more ideas are bound to be contributed by consumers themselves. The research framework presented in this report are meant to facilitate machine learning research by consumers themselves.

The figure below shows a research and deployment cycle that, if automated, would require little additional effort from consumers. If a consumer thinks of a mental state that might be recognizable by the EEG device, a semi-automated experiment program could allow the user to collect data from themselves, train and evaluate models, and potentially deploy the models to real-time use.


Curiosity & Hypothesis Generation
Self-Experimentation
Exploratory Data Analysis & Feature Engineering
Modeling
Real-Time Deployment

For example, if I am interested in extending the Muse functionality to mental control of a computer, I must find voluntary mental actions that can be recognized by a classifier that receives EEG data as features. To begin the investigation, I must define a baseline mental state, which I define as eyes open with no particular goal or action in mind. Other mental states could then be compared to this baseline, and a classifier should discriminate between these using EEG features that reflect brain activity rather than movement artifacts.

For the experiments presented here, the first non-baseline condition is mental action-thinking (repeatedly willing an action to occur). Another condition to test is shifting attention to bodily sensations, which in theory could activate additional brain networks.

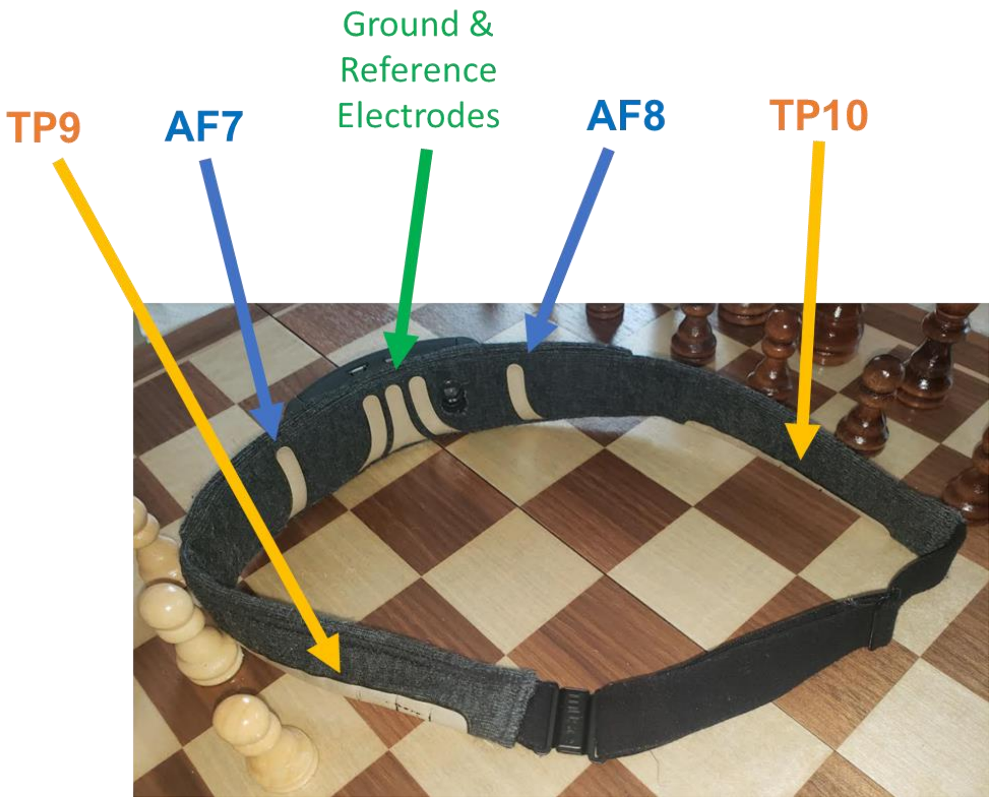
There will be two additional control conditions. The first of these is closed eyes as a positive control. A positive control is a condition that is known to be different from the baseline, as this demonstrates that the current methods are capable of separating conditions in feature space in a situation where theory suggests they should be separable. This closed-eyes condition should be distinguishable from the open eyes baseline, mainly along the dimension of alpha frequencies (approximately 10 Hz).

The final control condition will be physical button pressing, which is a negative control for action thinking. If mental action would be separable from baseline, we would want to see that this is also separable from a simple physical action.

<u>EEG Device & Data Engineering</u>

Muse headband features

The Muse S headband was used to collect data. This headband has 5 electrodes and 4 signal channels providing raw data at 256Hz. The center electrode serves as a ground and reference for the other signal electrodes. The signal channels bear the names of traditional EEG electrode locations. The frontal lobe electrodes (near the temples on the forehead) are AF7 (left) and AF8 (right). The side electrodes, ideally receiving signals from the temporal lobe, are TP9 (left) and TP10 (right). These electrodes sit on the ears and sometimes require the user to move hair out of the way. The Muse S is displayed in this image:



The frontal electrodes (AF7 & AF8) usually have low noise, although eye movements artifacts are a concern. The eyeball has an electromagnetic dipole, so eye movements will appear on frontal electrodes.

The side/temporal electrodes (TP9 & TP10) have high noise. They pick up muscle activity from jaw and neck muscles. Additionally, they tend to have a poor fit to their intended target, where the ear meets the side of the head. Thus, they are often overwhelmed by muscular artifacts and heartbeat artifacts.

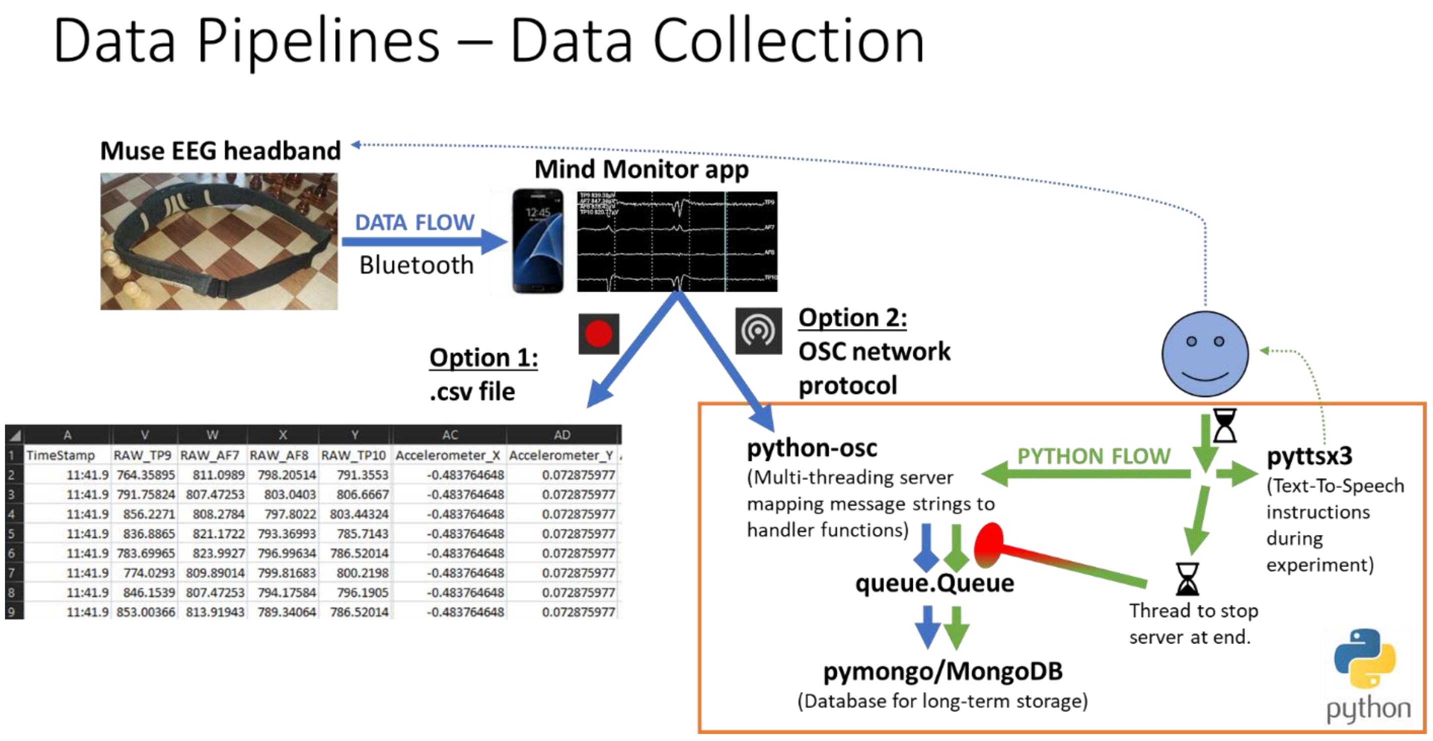
In addition, the Muse S contains an accelerometer and a gyroscope, which each contribute 3 feature columns corresponding to their X, Y, and Z directions. These provide data at 52Hz.

The Muse S has internal computations extracting certain frequency bands and certain noise artifacts, all of which are computed at 10Hz. Powers at the most well-known EEG frequency bands are auto-extracted by the headband's firmware. These frequency bands are delta (1-4Hz), theta (4-8Hz), alpha (8-13Hz), beta (13-30Hz), and gamma (30-44Hz). The noise artifacts identified by the firmware are eye blink and jaw clench. Additionally, there are quality control channels, which include headband contact with the forehead and a coded estimate of brain signal quality for each electrode (1=good signal, 2=poor signal, 4=terrible/no-contact).

Data Collection

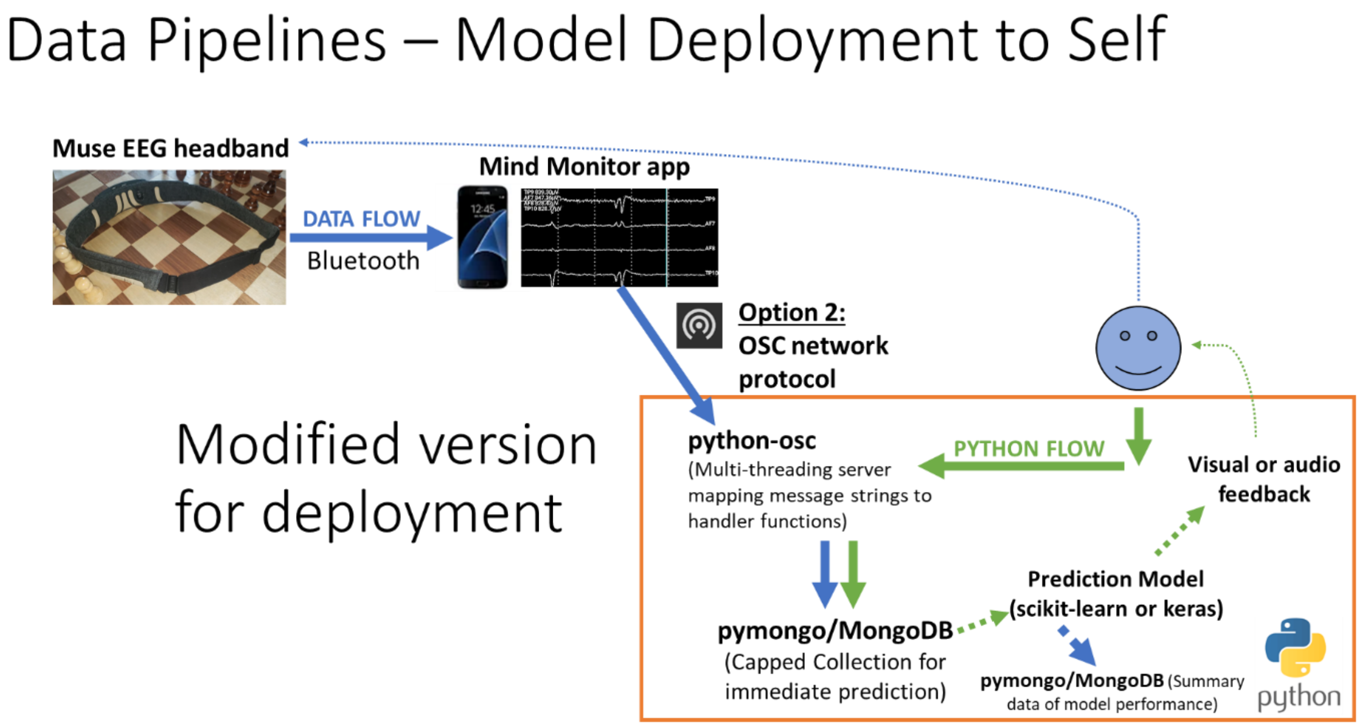
The data intake relied on a 3rd-party mobile app called Mind Monitor. Mind Monitor receives Muse data transmitted by Bluetooth, then offers two options for data output. The first option is saving to a .csv file, which is straightforward. In this case, time markers manually entered in the Mind Monitor app provide time boundaries for later analysis. However, the app's buttons for time markers must be pressed manually during data collection, which in the context of behavioral experiments creates excess mental burdens and movement artifacts. Most data presented in this paper were collected in this manner.

The second option for output from Mind Monitor involves streaming data via OSC network protocol. This second option allows a Python script on a computer to ingest data with precisely defined temporal boundaries, which is ideal for behavioral experiments. The data is ultimately saved in a MongoDB database. The Python script also includes a text-to-speech thread to notify the user/research subject of changes in experimental conditions, so that the user can change mental activity appropriately. This allows experiments without motion artifacts. This figure illustrates these two options:


Data Pipelines – Data Collection
Muse EEG headband
Mind Monitor app
DATA FLOW
Bluetooth
Option 1:
.csv file
Option 2:
OSC network protocol
python-osc
(Multi-threading server mapping message strings to handler functions)
PYTHON FLOW
pyttsx3
(Text-To-Speech instructions during experiment)
queue.Queue
Thread to stop server at end.
pymongo/MongoDB
(Database for long-term storage)
python

	A	V	W	X	Y	AC	AD
1	TimeStamp	RAW_TP9	RAW_AF7	RAW_AF8	RAW_TP10	Accelerometer_X	Accelerometer_Y
2	11:41.9	764.35895	811.0989	798.20514	791.3553	-0.483764648	0.072875977
3	11:41.9	791.75824	807.47253	803.0403	806.6667	-0.483764648	0.072875977
4	11:41.9	856.2271	808.2784	797.8022	803.44324	-0.483764648	0.072875977
5	11:41.9	836.8865	821.1722	793.36993	785.7143	-0.483764648	0.072875977
6	11:41.9	783.69965	823.9927	796.99634	786.52014	-0.483764648	0.072875977
7	11:41.9	774.0293	809.89014	799.81683	800.2198	-0.483764648	0.072875977
8	11:41.9	846.1539	807.47253	794.17584	796.1905	-0.483764648	0.072875977
9	11:41.9	853.00366	813.91943	789.34064	786.52014	-0.483764648	0.072875977

While the ultimate goal of real-time implementation of models has not been completed yet, it would rely on immediate model prediction in real time, which is a scheme illustrated in the following figure:

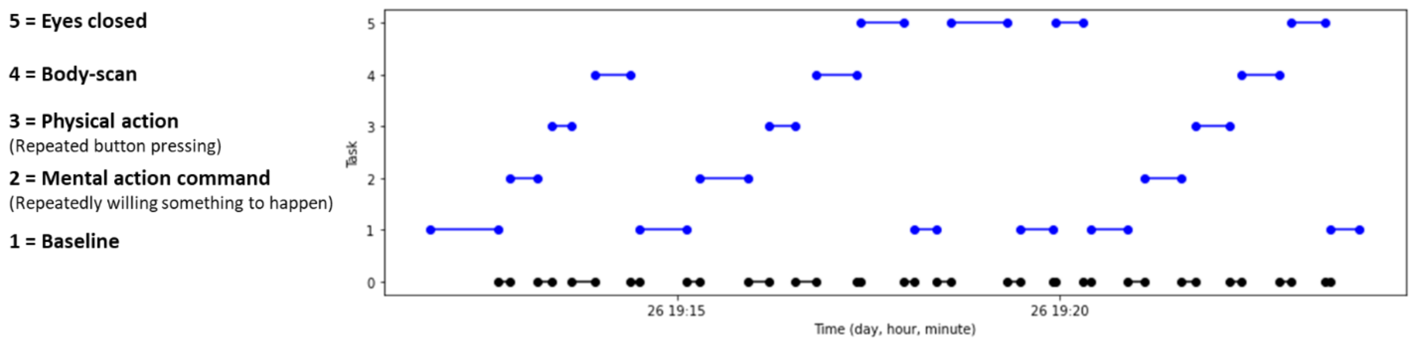


Data Pipelines – Model Deployment to Self
Muse EEG headband
Mind Monitor app
DATA FLOW
Bluetooth
Option 2:
OSC network protocol
Modified version for deployment
python-osc
(Multi-threading server mapping message strings to handler functions)
PYTHON FLOW
Visual or audio feedback
Prediction Model (scikit-learn or keras)
pymongo/MongoDB
(Capped Collection for immediate prediction)
pymongo/MongoDB (Summary data of model performance)
python

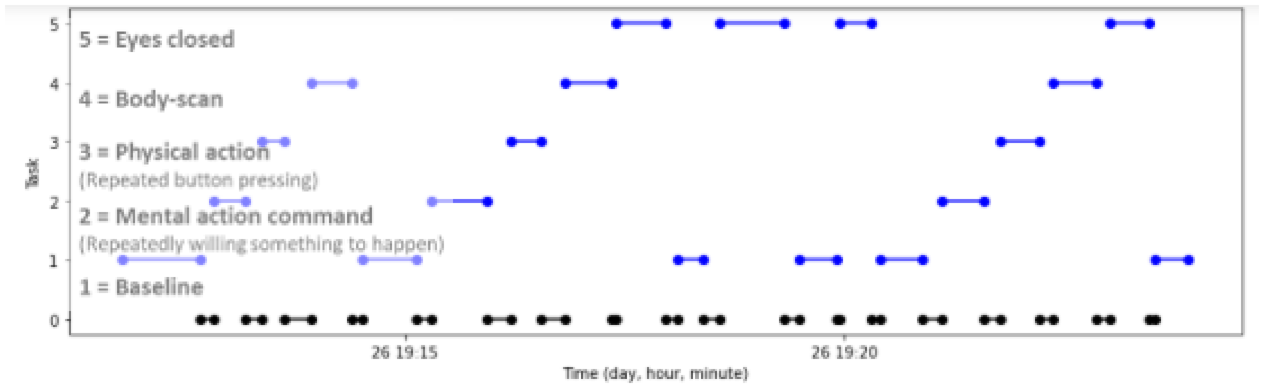
Data Wrangling

Manually collected .csv File Data

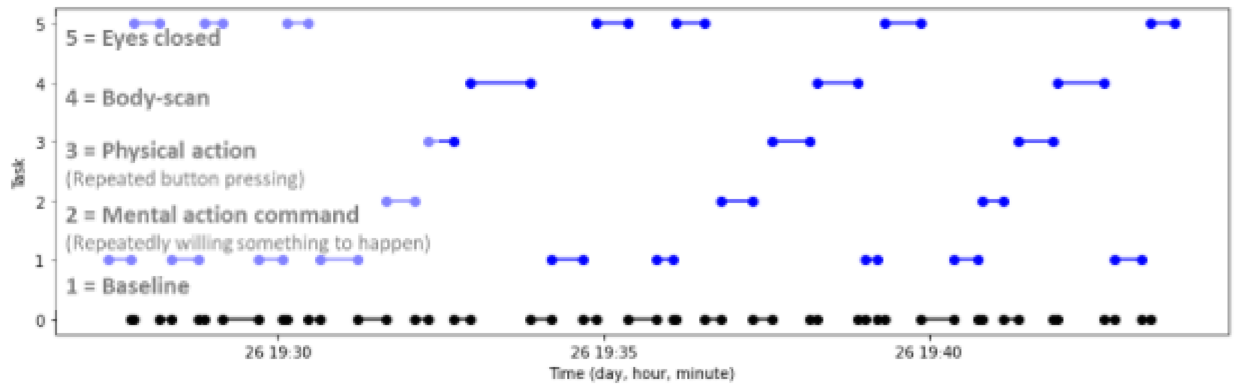
The data collected via the .csv file option (with manually-placed time markers) from 3 experimental recordings were loaded and time markers were utilized to generate a segmented time series with separated experimental tasks, as illustrated by this figure:



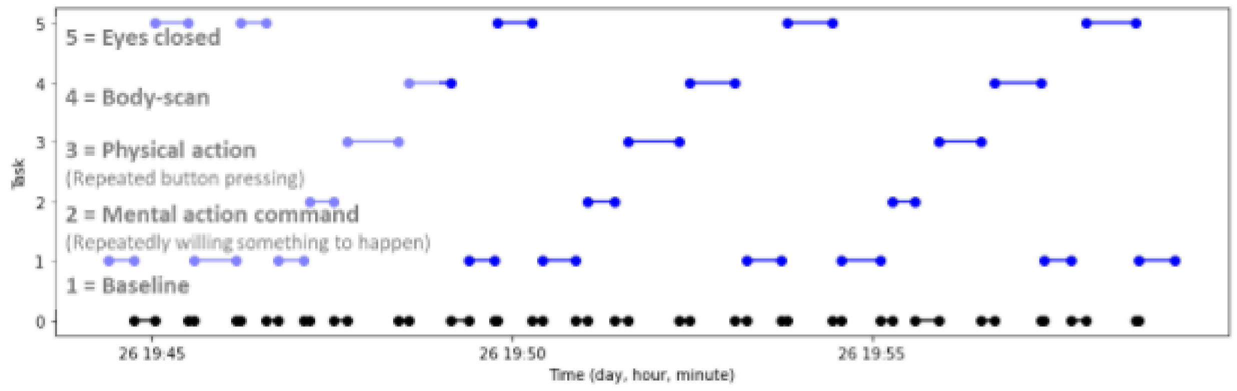
From here, the data was split into 2-second epochs with 1.5 seconds of overlap. This generated a total of 3,603 epochs from experimental conditions, although the effective total was much less due to the need for balanced classes in modeling. The following page shows all 3 recordings used in this approach:



```
/Marker/1 contains 347 epochs
/Marker/2 contains 168 epochs
/Marker/3 contains 116 epochs
/Marker/4 contains 169 epochs
/Marker/5 contains 237 epochs
Between_Tasks contains 291 epochs
```



```
/Marker/1 contains 383 epochs
/Marker/2 contains 136 epochs
/Marker/3 contains 170 epochs
/Marker/4 contains 265 epochs
/Marker/5 contains 316 epochs
Between_Tasks contains 523 epochs
```



```
/Marker/1 contains 448 epochs
/Marker/2 contains 112 epochs
/Marker/3 contains 230 epochs
/Marker/4 contains 212 epochs
/Marker/5 contains 294 epochs
Between_Tasks contains 321 epochs
```

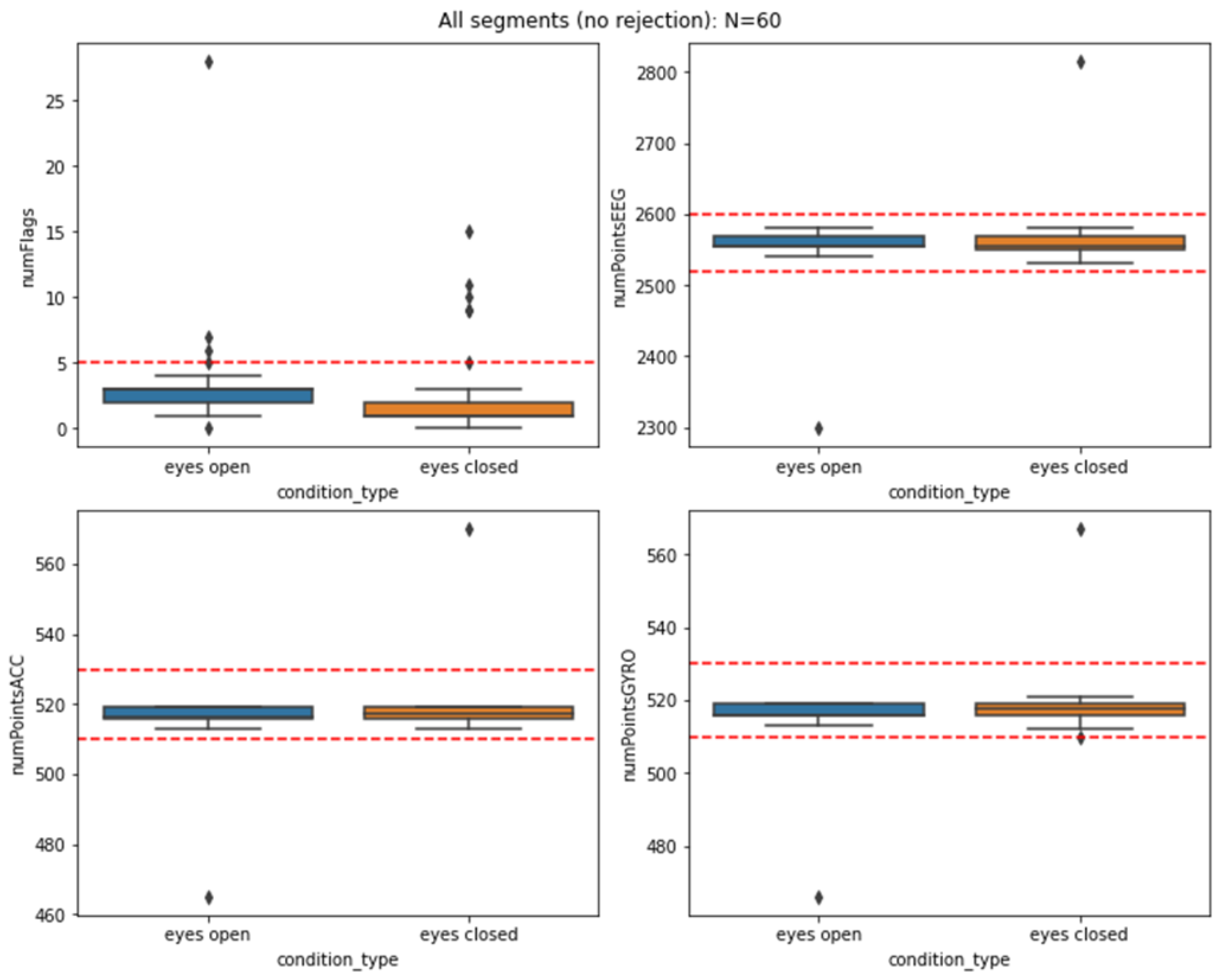
From here, aggregate metrics (mean and variance) could be calculated over each epoch before ingestion into model pipelines.

Automatically collected Streaming Data

Epoch separation was accomplished by the data collection script. Each epoch was represented by a distinct MongoDB document in a RawData collection, with each document containing information about the behavioral condition and containing an experiment identifier. Each experiment had its own parameters saved into a separate Experiments collection. For the streaming data, 10-second epochs were used.

After collection, epochs were screened for quality control issues. Flags were created at collection time in the cases of eye blinks, jaw clenches, or compromised signal connection. An epoch was rejected if 5 or more flags were present. After rejection, only eye blink flags were still present. In addition, a temporally controlled streaming experiment using a chain of devices creates the possibility of connection issues between devices, and this was controlled for by rejecting epochs with a highly irregular number of timepoints. For raw EEG, this was more than 1.5% different from the mean of approximately 2560 timepoints per 10 seconds. For gyroscope and accelerometer data, the mean of approximately 520 timepoints per 10 seconds. After epoch rejection, 48 of 60 epochs remained.

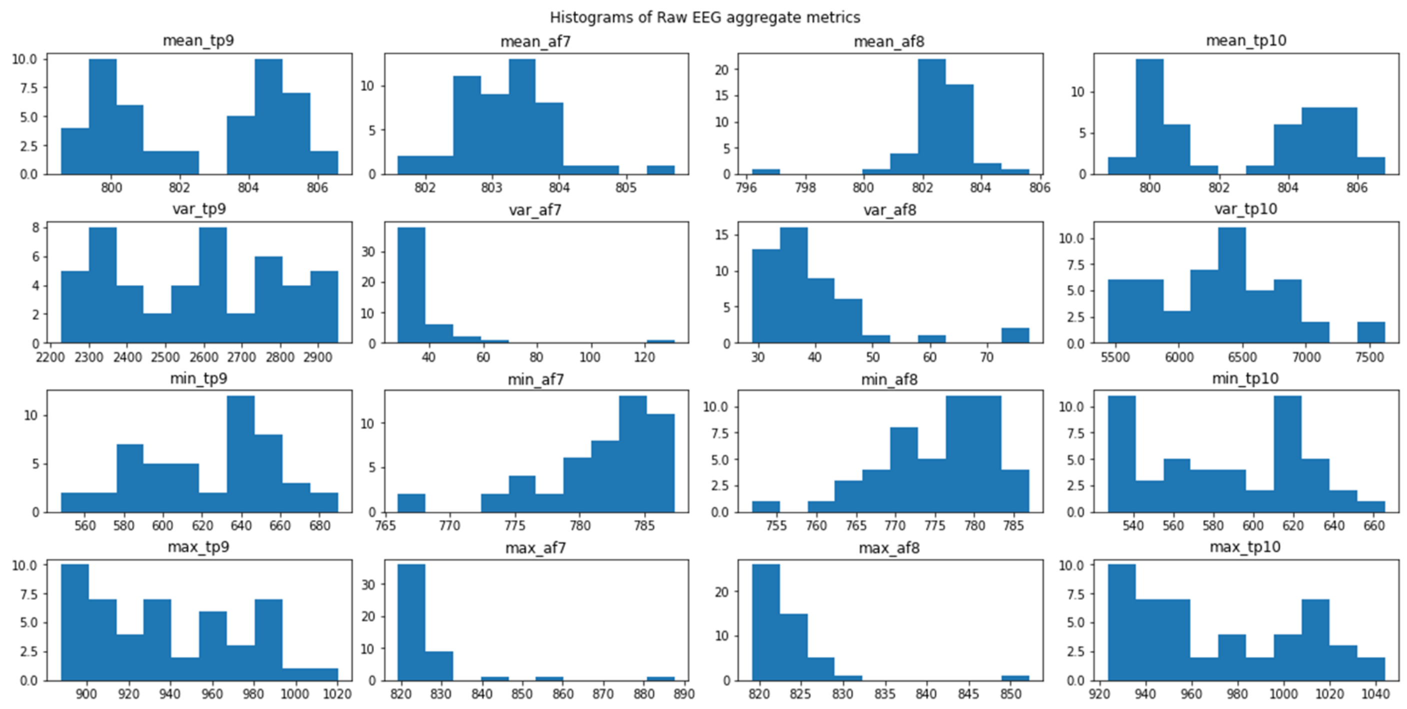
The following figure shows the pre-rejection number of flags and timepoints per 10-second epoch. The dashed red lines show the approximate thresholds outside of which epochs were rejected.



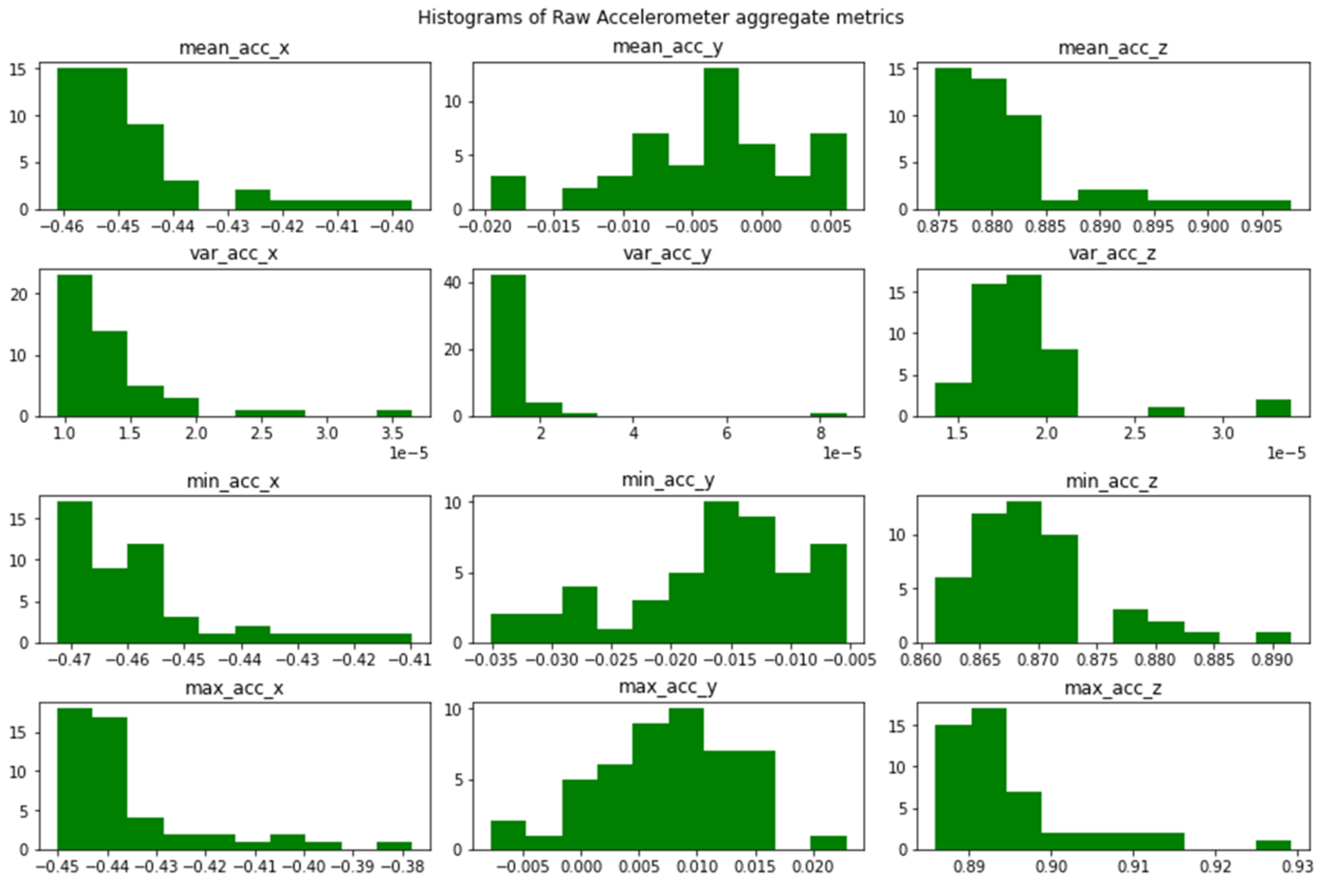
Exploratory Data Analysis

Like most types of neural data, EEG is highly non-linear. Many potential features are non-normal in distribution. Furthermore, there is no single transformation that would convert all features to a normally distributed space. The input to most models will be aggregated over many time points, and the distributions of mean, variance, min, and max for raw EEG, accelerometer, and gyroscope are shown in the below figures.

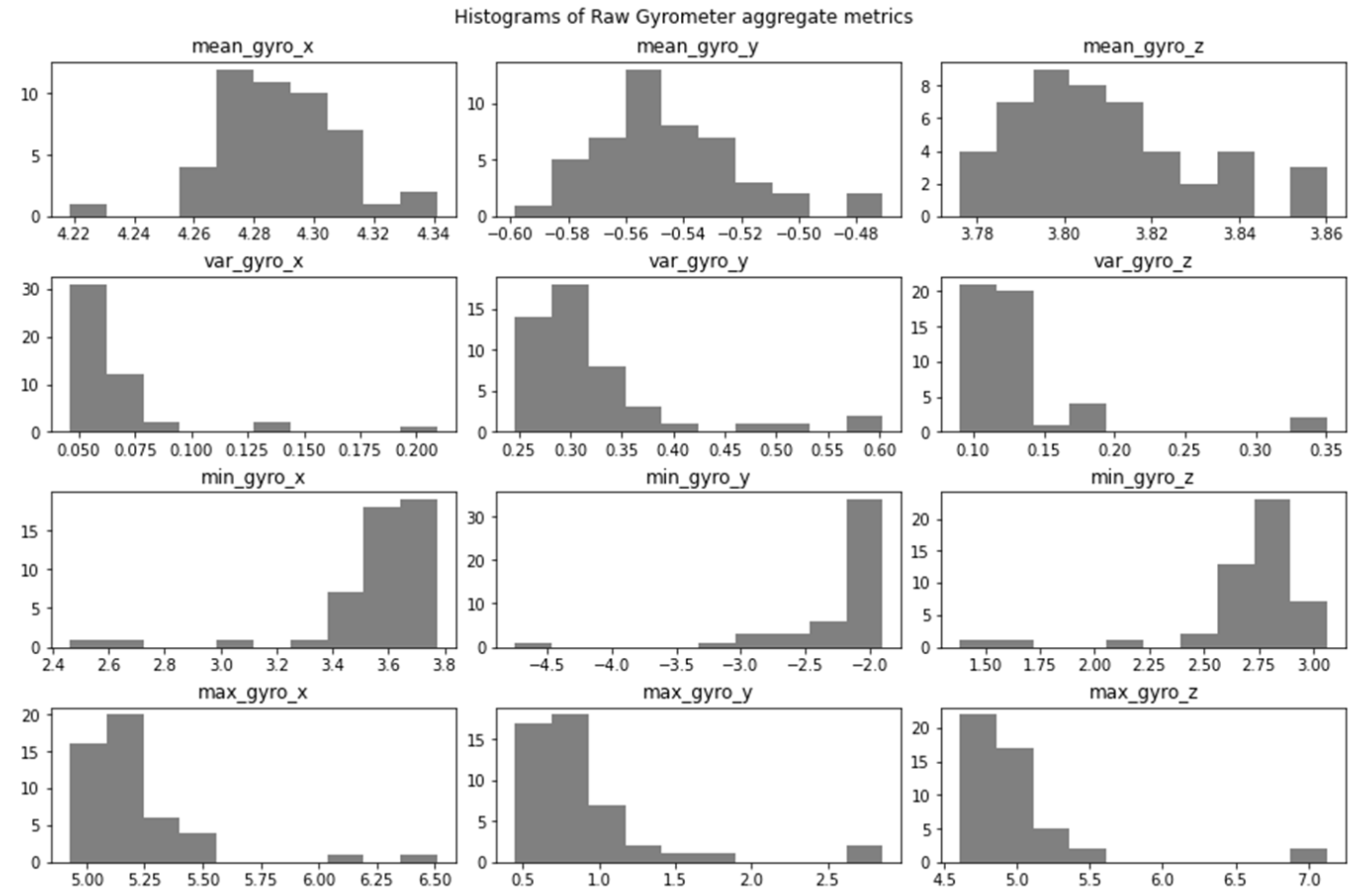
For EEG, note that there are 4 columns representing the electrodes TP9, AF7, AF8, and TP10. The rows show distributions of mean, variance, min, and max for entire 10-second epochs:



For the accelerometer, the columns represent each of the 3 cartesian axes (x, y, z):



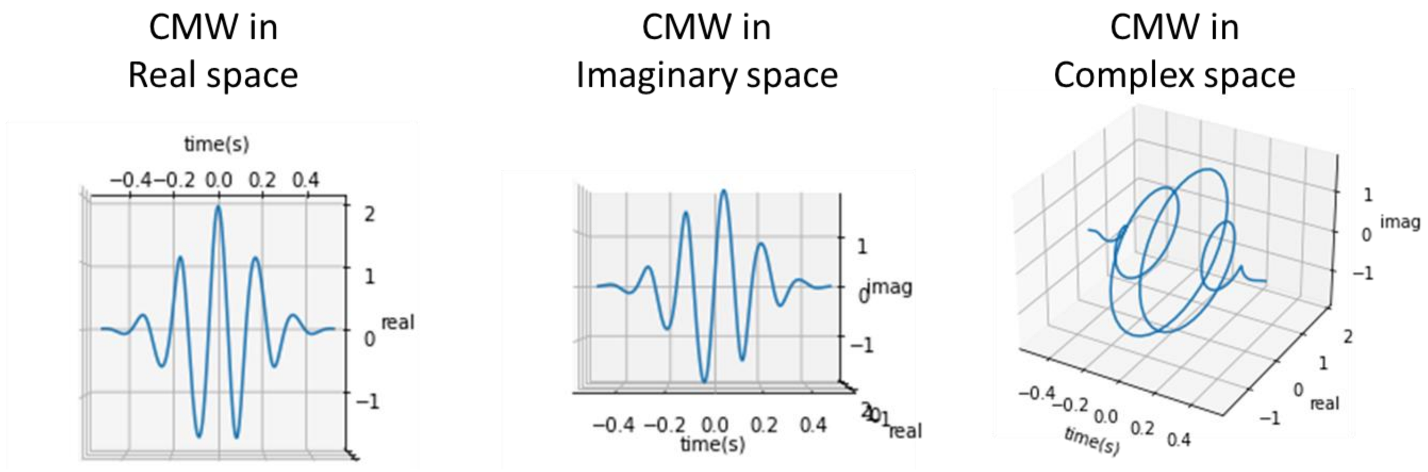
The gyroscope has the same cartesian axes (x, y, z):



Wavelet Transformations

EEG data relies on frequency-based analysis. There are a number of tools to convert time domain EEG signals to frequency domain. Although the most well-known might be the short-time Fourier transform (STFT), I opted for a different approach using Complex Morlet Wavelets (CMW). Wavelet approaches allow power and phase to be extracted from very precise frequency bands by convolving the time series with a kernel of complex numbers.

A Complex Morlet Wavelet (CMW) is constructed by multiplying a sine wave by a gaussian, while using Euler's formula to expand the resulting waveform into imaginary space. When viewed in imaginary space, the wavelet is a spiral.



The 90° phase shift in imaginary space is why the phase at any given frequency can be extracted. Although phase in itself is hardly a meaningful feature, extracting phase information at a given electrode then allows comparison to other electrodes, which theoretically would allow detection of changes in brain activity between brain regions. To put it in neuroscience terms, this means extracting *functional connectivity* between brain regions.

These two boxes give Python code and further explanation for constructing and using a CMW.

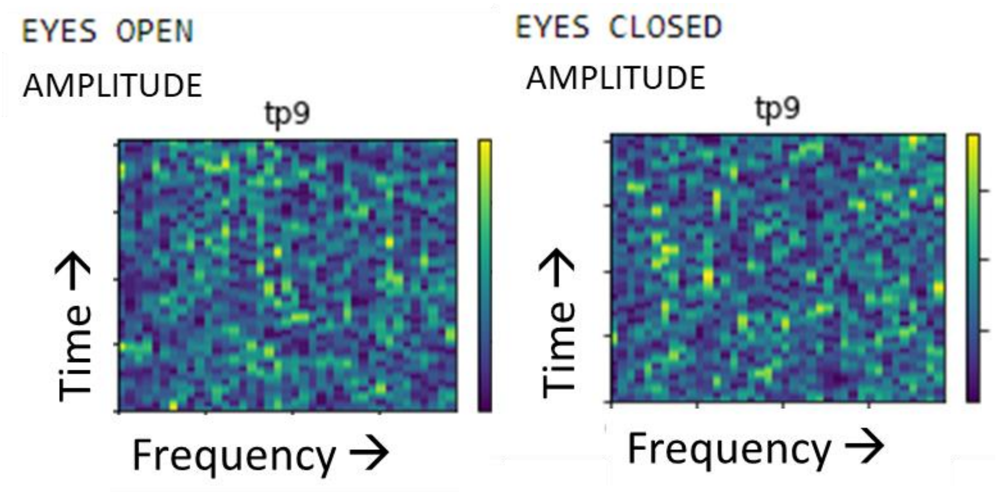
Python code for constructing a Complex Morlet Wavelet (cmw):

```
# Generate a numpy array containing time points along a time axis. This will become the CMW kernel.
x = np.linspace(-windowLengthSeconds/2, windowLengthSeconds/2, int(windowLengthSeconds*fs)+1)
# where fs is sampling rate, and windowLengthSeconds is width (in seconds) of the desired wavelet kernel

std_for_gaussian = lambda n, f: n/(2*np.pi*f)
# where n is number of wavelet cycles, and f is peak frequency of the wavelet

cmw = lambda t, f, s: (1/np.sqrt(s*np.sqrt(np.pi))) * np.exp(-t**2/(2*s**2)) * np.exp(1j*2*np.pi*f*t)
# where t is time-axis points as a numpy array, f is peak frequency of the wavelet, and s is a curve width term
generated by the function "std_for_gaussian"


complex_morlet_wavelet = cmw(x, freq, std_for_gaussian(n=n_wavelet_cycles, f=freq))
# where x is the time points array defined above, freq is the peak frequency of the desired wavelet, and n is the number
of wavelet periods/cycles, which relates to curve width

# Notes:
# wavelets must be flat before reaching sides of kernel (see figure above)
# n_wavelets_cycles can be scaled with peak frequency to give more stable comparisons between frequencies
# This project scaled n_wavelets cycles so that n_wavelets_cycles = peak frequency
```

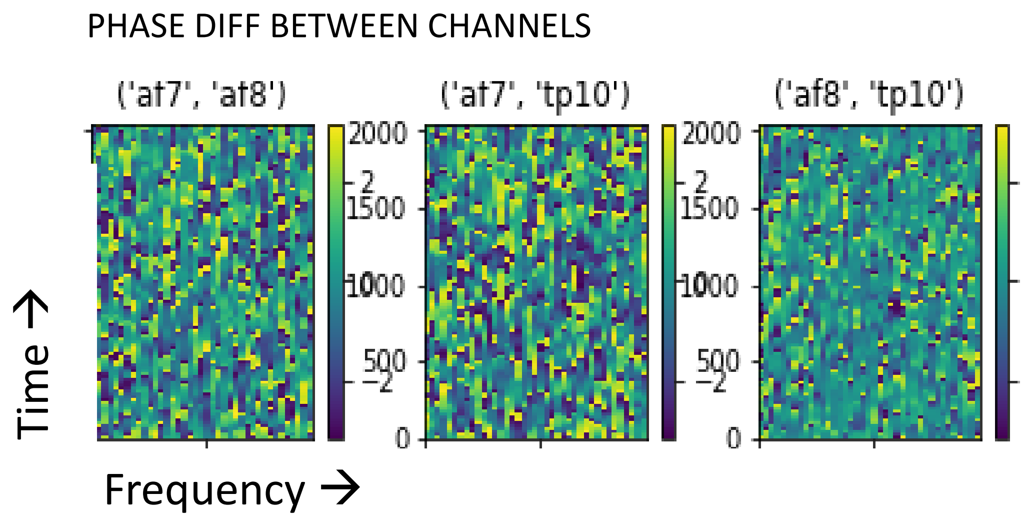
Python code for using a Complex Morlet Wavelet (cmw) for feature extraction:

```
# Generate wavelet
complex_morlet_wavelet = cmw(t, freq, std_for_gaussian(n=n_cycles, f=freq))

# Convolve wavelet with 1-dimensional data array
complex_convolved = np.convolve(complex_morlet_wavelet, data[:,i], mode=mode)

# Convert complex number to real number metric, with any or all of these 4 options:
BANDPASS_FILTERED_SIGNAL = np.real(complex_convolved)
AMPLITUDE = np.abs(complex_convolved)
POWER = np.abs(complex_convolved**2)
PHASE = np.angle(complex_convolved)
```

The following two figures show examples of CMW applied during exploratory data analysis to generate interesting insights. Forty CMW kernels with peak frequencies at 2Hz, 4Hz, ..., 78Hz, and 80Hz were used to generate amplitude, power, and phase differences. The first pair of heatmaps show amplitude in the color-axis, with time on the y-axis and frequency on the x-axis. A 10-second epoch with eyes open (left) is compared to a 10-second epoch with eyes closed (right). In the eyes closed epoch, near the left side of the right plot, we see activity at 10Hz and 12Hz (alpha band) that is not apparent in the preceding epoch with eyes open.



This second example below shows phase differences (color-axis) over time (y-axis) and frequency (x-axis) between electrode channels. The interesting observation is that comparisons between two electrodes from a given hemisphere (such as on the right-most plot between AF8 and TP10) show much more stability than comparisons between electrodes on opposite hemispheres (such as the left-most and center plots).



Altogether, the chaotic non-linear dynamics of the brain make modeling difficult. One approach would be to aggregate each epoch into summary statistics, which allows input to most machine learning model types. Another approach not taken here would be to input sequences of timepoints into a recurrent neural network or multivariate autoregressive model.

Further exploratory analysis can be seen at https://github.com/WalterPiTheScienceGuy/Capstone_3_EEG/blob/main/Notebooks/Data_Exploration_Simple-Copy1.ipynb

Preprocessing

For modeling with the manual experimentation data as described earlier, each 2-second epoch was aggregated with mean and variance. This produced a mean value and variance value for every numeric feature, including raw data from gyroscope and accelerometer, and power, amplitude, and phase difference information from CMW transformations on each EEG channel. This also included mean and variance of internally calculated frequency band powers (delta, theta, alpha, beta, and gamma).

Due to the manual and imprecise control of experimental condition durations during data collection, not all possible epochs were used in modeling. Depending on the number of epochs belonging to different classes that were to be compared by a model, some epochs were dropped to ensure balanced classes.

Before modeling, train and test sets were made with a modified train-test split procedure. The modified procedure used the first 80% of epochs of a given class in a given recording as the train set, then used the following 20% of each class in each recording as the test set. This avoids shuffling examples, which is important when examples represent segmented epochs from a continuous recording.

Modeling Procedure

Aggregated data from 2-second epochs were input into 12 different candidate scikit-learn pipelines. Each pipeline began with a z-scoring StandardScaler. Half of the models then used principal component analysis (PCA) as the next step. The final step was one of 6 model types: Support Vector Classifier, Linear Discriminant Analysis, Random Forest Classifier, AdaBoost Classifier, Gradient Boosting Classifier, or a Decision Tree Classifier. Area under the curve of the receiver operating characteristic curve (AUC ROC) was used as the primary evaluation metric. For the decision tree methods, feature importances were extracted and displayed.

This procedure was used in 6 different experiments, each of which compared 2 or more classes from among the 5 classes of behavior conditions during the data collection.

Condition
Eyes Open
Mental Action
Physical Action
Body-focused Attention/Meditation
Eyes Closed

Each experiment report below will include model metrics and top features used by decision tree models.

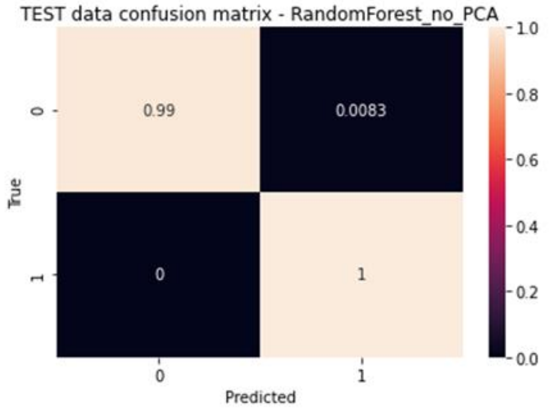
Experiment 1: Eyes Open vs Eyes Closed

Highly accurate classification was easily achieved by most model types. The best model types were the Random Forest Classifier without PCA and the Support Vector Classifiers with PCA and without PCA. The most important features used by decision tree model types were either high gamma frequencies (40Hz-80Hz) at the frontal electrodes, or alpha/10Hz frequency at the right temporal lobe (electrode TP10). The alpha frequency effect is known in neuroscience literature.

Experiment 1: Eyes Open vs Eyes Closed		
pipeline_name	**test_ROC_score**	**test_confusion_matrix**
RandomForest_no_PCA	100.0%	[[1. 0.] [0. 1.]]
SVC_no_PCA	99.6%	[[0.99166667 0.00833333] [0. 1.]]
SVC_with_PCA	99.6%	[[0.99166667 0.00833333] [0. 1.]]
GradientBoosting_with_PCA	97.5%	[[0.96666667 0.03333333] [0.01666667 0.98333333]]
AdaBoost_no_PCA	96.3%	[[0.925 0.075] [0. 1.]]
AdaBoost_with_PCA	96.3%	[[0.95833333 0.04166667] [0.03333333 0.96666667]]
DecisionTree_no_PCA	93.3%	[[0.88333333 0.11666667] [0.01666667 0.98333333]]
LDA_no_PCA	92.9%	[[0.89166667 0.10833333] [0.03333333 0.96666667]]
GradientBoosting_no_PCA	92.9%	[[0.875 0.125] [0.01666667 0.98333333]]
RandomForest_with_PCA	80.0%	[[0.68333333 0.31666667] [0.08333333 0.91666667]]
DecisionTree_with_PCA	60.8%	[[0.76666667 0.23333333] [0.55 0.45]]
LDA_with_PCA	53.8%	[[0.575 0.425] [0.5 0.5]]

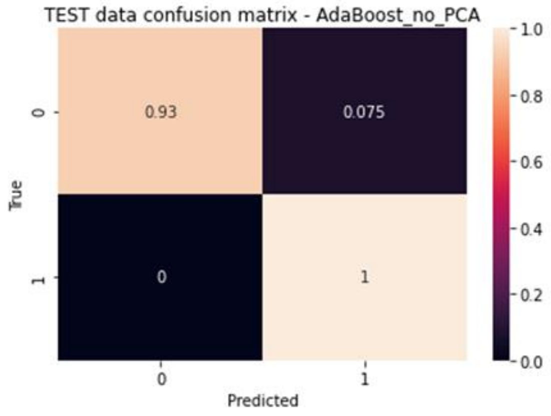
Experiment 1: Open Eyes vs Closed Eyes

```
RandomForest_no_PCA
for TESTING DATA
ROC_AUC Score: 0.9958333333333333
Accuracy Score: 0.9958333333333333
```



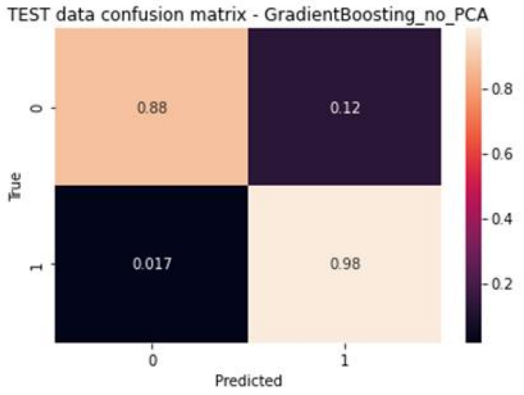
```
mean_power76.0HzCh1                                  0.037261
mean_amplitude58.0HzCh1                              0.036103
var_amplitude78.0HzCh1                               0.034273
mean_power10.0HzCh3                                  0.028574
var_phaseDifference_phase42.0HzCh1_phase42.0HzCh3    0.027517
```

```
AdaBoost_no_PCA
for TESTING DATA
ROC_AUC Score: 0.9625
Accuracy Score: 0.9625
```



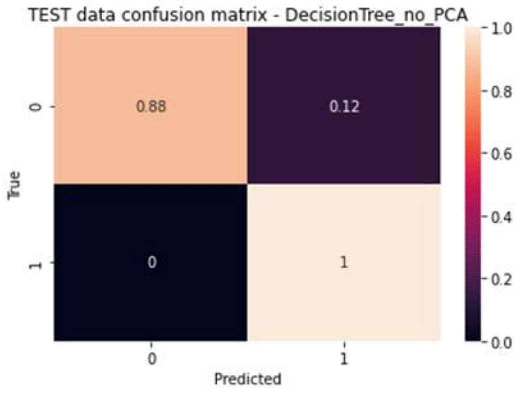
```
mean_Alpha_TP10                    0.10
mean_amplitude72.0HzCh1            0.04
var_power10.0HzCh3                 0.04
mean_Delta_AF8                     0.04
mean_power70.0HzCh1                0.04
```

```
GradientBoosting_no_PCA
for TESTING DATA
ROC_AUC Score: 0.9291666666666666
Accuracy Score: 0.9291666666666667
```



```
mean_power70.0HzCh1         9.146086e-01
mean_Alpha_TP10             4.527634e-02
mean_power74.0HzCh1         1.319184e-02
var_amplitude74.0HzCh1      5.784964e-03
mean_amplitude74.0HzCh1     5.783374e-03
```

```
DecisionTree_no_PCA
for TESTING DATA
ROC_AUC Score: 0.9375
Accuracy Score: 0.9375
```



```
mean_power70.0HzCh1                                  0.914609
mean_Alpha_TP10                                      0.045276
mean_amplitude74.0HzCh1                              0.028735
var_power44.0HzCh2                                   0.007222
var_phaseDifference_phase80.0HzCh1_phase80.0HzCh2    0.004158
```

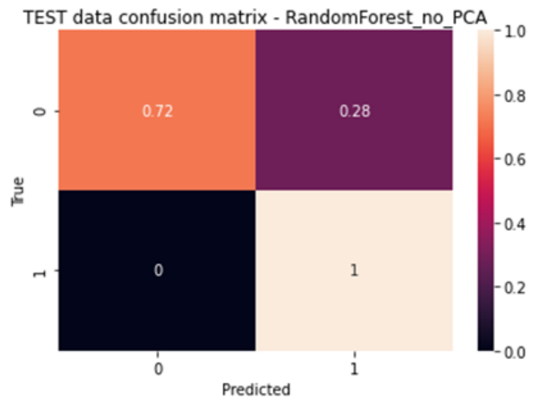
Experiment 2: Eyes Open vs Mental Action

This comparison was much more difficult for the models, and the feature importances suggest the models were overfitting to noise artifacts or to session-level variables, as the dataset was derived from 3 separate recordings. The best model types were the Random Forest Classifier, AdaBoost Classifier, and Gradient Boosting Classifier, all without PCA. The most important features used by decision tree model types were either accelerometer features or aspects of the 60Hz frequency, which is probably electrical noise. This is almost certainly a case of overfitting, and these models should not be considered valid.

Experiment 2: Eyes Open vs Mental Action		
pipeline_name	**test_ROC_score**	**test_confusion_matrix**
RandomForest_no_PCA	87.5%	[[0.75 0.25] [0. 1.]]
AdaBoost_no_PCA	84.2%	[[0.68333333 0.31666667] [0. 1.]]
GradientBoosting_no_PCA	83.3%	[[0.66666667 0.33333333] [0. 1.]]
SVC_with_PCA	74.2%	[[0.58333333 0.41666667] [0.1 0.9]]
SVC_no_PCA	70.0%	[[0.6 0.4] [0.2 0.8]]
DecisionTree_no_PCA	70.0%	[[0.4 0.6] [0. 1.]]
GradientBoosting_with_PCA	68.3%	[[0.6 0.4] [0.23333333 0.76666667]]
AdaBoost_with_PCA	62.5%	[[0.31666667 0.68333333] [0.06666667 0.93333333]]
LDA_no_PCA	61.7%	[[0.51666667 0.48333333] [0.28333333 0.71666667]]
DecisionTree_with_PCA	56.7%	[[0.83333333 0.16666667] [0.7 0.3]]
RandomForest_with_PCA	55.8%	[[0.45 0.55] [0.33333333 0.66666667]]
LDA_with_PCA	39.2%	[[0.5 0.5] [0.71666667 0.28333333]]

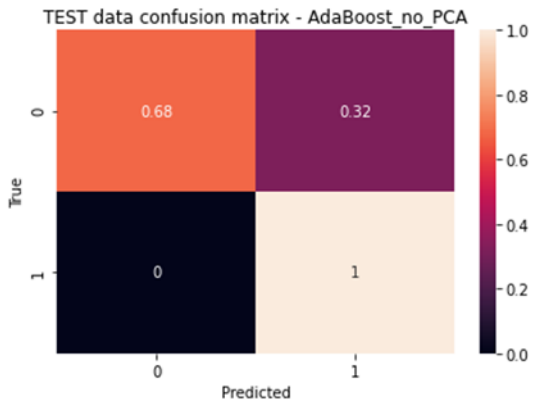
Experiment 2: Open Eyes vs Mental Action

```
RandomForest_no_PCA
for TESTING DATA
ROC_AUC Score: 0.8583333333333334
Accuracy Score: 0.8583333333333334
```



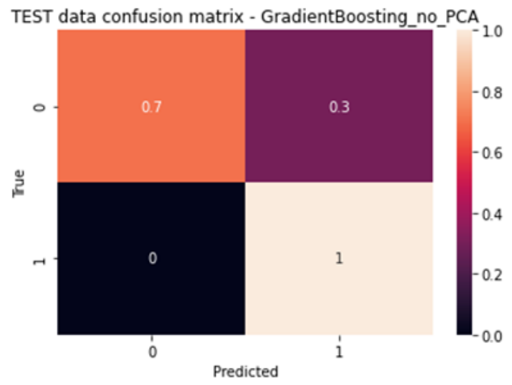
```
mean_Accelerometer_Z                                 0.023615
mean_Accelerometer_X                                 0.023422
var_power60.0HzCh0                                   0.021041
mean_amplitude60.0HzCh0                              0.020900
var_phaseDifference_phase60.0HzCh0_phase60.0HzCh3    0.019460
```

```
AdaBoost_no_PCA
for TESTING DATA
ROC_AUC Score: 0.8416666666666667
Accuracy Score: 0.8416666666666667
```



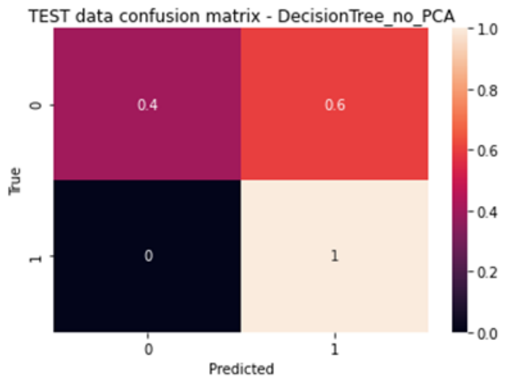
```
mean_Accelerometer_X                                   0.08
var_phaseDifference_phase60.0HzCh0_phase60.0HzCh3      0.06
var_phaseDifference_phase62.0HzCh0_phase62.0HzCh3      0.06
mean_phaseDifference_phase58.0HzCh0_phase58.0HzCh3     0.04
var_amplitude62.0HzCh0                                 0.04
```

```
GradientBoosting_no_PCA
for TESTING DATA
ROC_AUC Score: 0.85
Accuracy Score: 0.85
```



```
mean_Accelerometer_X                                 2.526275e-01
var_phaseDifference_phase60.0HzCh0_phase60.0HzCh3    1.894172e-01
mean_amplitude58.0HzCh0                              1.230162e-01
var_phase58.0HzCh0                                   1.109842e-01
var_power62.0HzCh0                                   8.815802e-02
```

```
DecisionTree_no_PCA
for TESTING DATA
ROC_AUC Score: 0.7
Accuracy Score: 0.7
```



```
mean_amplitude58.0HzCh0                              0.362678
var_phaseDifference_phase60.0HzCh0_phase60.0HzCh3    0.293952
var_power62.0HzCh0                                   0.184247
var_phaseDifference_phase34.0HzCh2_phase34.0HzCh3    0.054264
mean_Accelerometer_X                                 0.032231
```

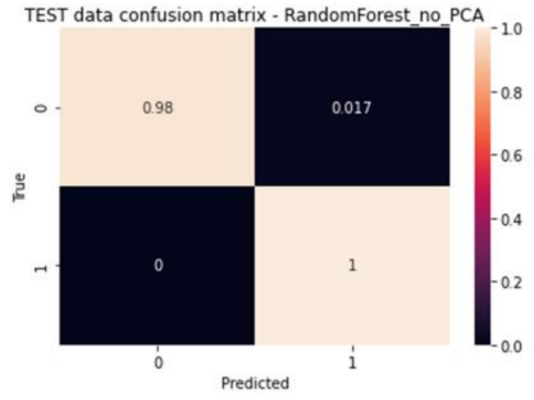
Experiment 3: Eyes Open vs Physical Action (button pressing)

This classification appears to have been driven by movement artifacts, as indicated by the top feature placement of gyroscope and accelerometer variables. The Random Forest Classifier again had the best performance.

Experiment 3: Eyes Open vs Physical Action (button pressing)		
pipeline_name	**test_ROC_score**	**test_confusion_matrix**
RandomForest_no_PCA	95.8%	[[0.91666667 0.08333333] [0. 1.]]
GradientBoosting_no_PCA	86.7%	[[0.73333333 0.26666667] [0. 1.]]
AdaBoost_no_PCA	83.3%	[[0.66666667 0.33333333] [0. 1.]]
LDA_no_PCA	81.7%	[[0.65 0.35] [0.01666667 0.98333333]]
DecisionTree_no_PCA	77.5%	[[0.88333333 0.11666667] [0.33333333 0.66666667]]
SVC_with_PCA	75.0%	[[0.76666667 0.23333333] [0.26666667 0.73333333]]
SVC_no_PCA	73.3%	[[0.78333333 0.21666667] [0.31666667 0.68333333]]
LDA_with_PCA	70.0%	[[0.66666667 0.33333333] [0.26666667 0.73333333]]
DecisionTree_with_PCA	70.0%	[[0.83333333 0.16666667] [0.43333333 0.56666667]]
AdaBoost_with_PCA	54.2%	[[1. 0.] [0.91666667 0.08333333]]
RandomForest_with_PCA	51.7%	[[1. 0.] [0.96666667 0.03333333]]
GradientBoosting_with_PCA	50.0%	[[1. 0.] [1. 0.]]

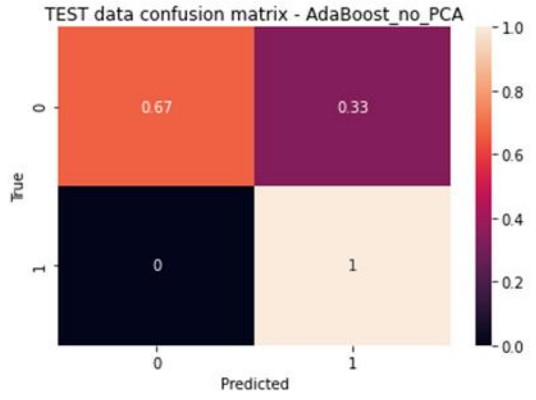
Experiment 3: Open Eyes vs Physical Action (button pressing)

```
RandomForest_no_PCA
for TESTING DATA
ROC_AUC Score: 0.9916666666666666
Accuracy Score: 0.9916666666666667
```



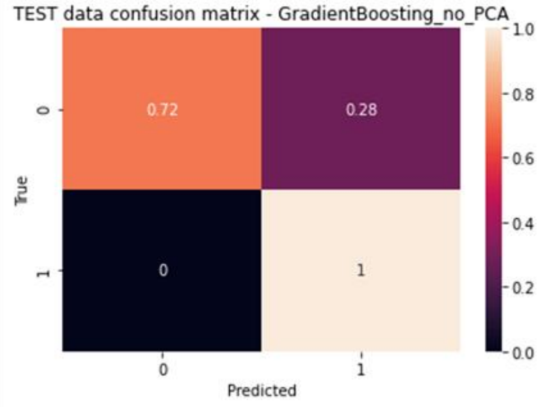
```
var_Accelerometer_Y                    0.048230
mean_Accelerometer_X                   0.046693
var_Gyro_Z                             0.037279
mean_Accelerometer_Z                   0.032886
var_phase60.0HzCh3                     0.021521
```

```
AdaBoost_no_PCA
for TESTING DATA
ROC_AUC Score: 0.8333333333333334
Accuracy Score: 0.8333333333333333
```



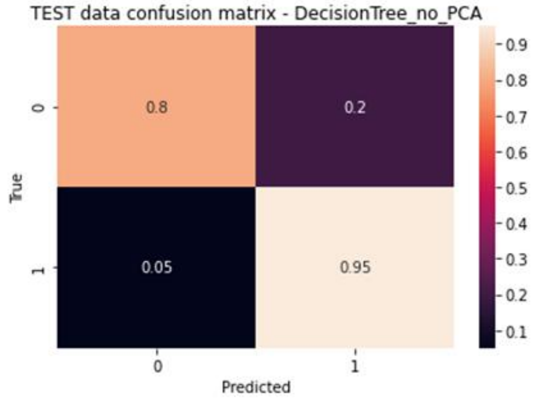
```
mean_Accelerometer_X                          0.72
mean_Accelerometer_Z                          0.26
var_Gyro_Z                                    0.02
mean_Delta_TP9                                0.00
var_amplitude44.0HzCh1                        0.00
```

```
GradientBoosting_no_PCA
for TESTING DATA
ROC_AUC Score: 0.8583333333333334
Accuracy Score: 0.8583333333333334
```



```
var_Gyro_Z                                           0.727947
mean_Accelerometer_X                                 0.069183
mean_Accelerometer_Z                                 0.055364
mean_phaseDifference_phase62.0HzCh0_phase62.0HzCh3   0.038017
var_Accelerometer_Z                                  0.029711
```

```
DecisionTree_no_PCA
for TESTING DATA
ROC_AUC Score: 0.875
Accuracy Score: 0.875
```



```
var_Gyro_Z                                           0.730222
mean_Accelerometer_X                                 0.127368
mean_phaseDifference_phase62.0HzCh0_phase62.0HzCh3   0.061424
var_phaseDifference_phase6.0HzCh0_phase6.0HzCh3      0.032754
mean_amplitude58.0HzCh0                              0.024217
```

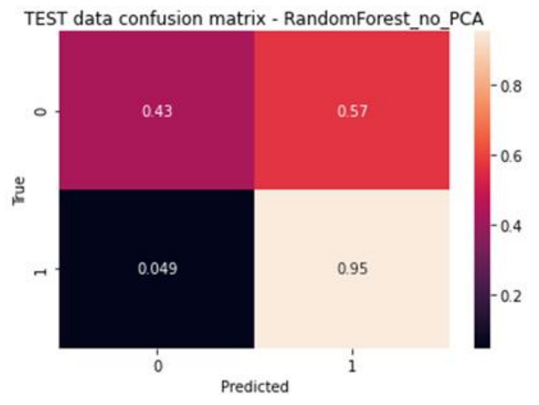
Experiment 4: Eyes Open vs Body-focused Attention

This classification was not reliable. The model metrics show poor performance with identifying the eyes-open baseline. The feature importances show the decision tree models were fit to accelerometer data.

Experiment 4: Eyes Open vs Body-focused Attention		
pipeline_name	**test_ROC_score**	**test_confusion_matrix**
SVC_with_PCA	74.5%	[[0.54901961 0.45098039] [0.05882353 0.94117647]]
SVC_no_PCA	74.0%	[[0.53921569 0.46078431] [0.05882353 0.94117647]]
AdaBoost_with_PCA	71.1%	[[0.54901961 0.45098039] [0.12745098 0.87254902]]
DecisionTree_no_PCA	68.1%	[[0.44117647 0.55882353] [0.07843137 0.92156863]]
AdaBoost_no_PCA	67.6%	[[0.37254902 0.62745098] [0.01960784 0.98039216]]
GradientBoosting_with_PCA	67.6%	[[0.41176471 0.58823529] [0.05882353 0.94117647]]
RandomForest_no_PCA	65.2%	[[0.38235294 0.61764706] [0.07843137 0.92156863]]
DecisionTree_with_PCA	64.7%	[[0.5 0.5] [0.20588235 0.79411765]]
GradientBoosting_no_PCA	62.7%	[[0.25490196 0.74509804] [0. 1.]]
LDA_no_PCA	56.4%	[[0.53921569 0.46078431] [0.41176471 0.58823529]]
LDA_with_PCA	55.9%	[[0.59803922 0.40196078] [0.48039216 0.51960784]]
RandomForest_with_PCA	46.1%	[[0.35294118 0.64705882] [0.43137255 0.56862745]]

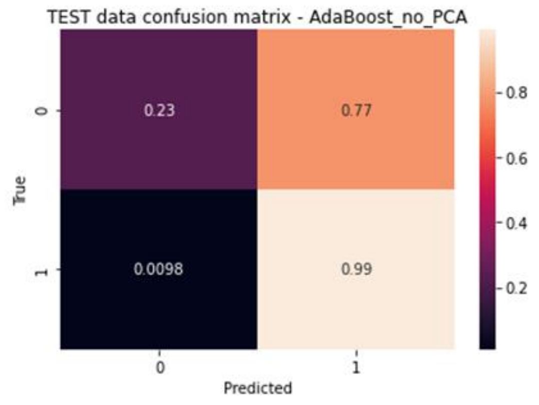
Experiment 4: Eyes Open vs Body-focused Attention

```
RandomForest_no_PCA
for TESTING DATA
ROC_AUC Score: 0.6911764705882353
Accuracy Score: 0.6911764705882353
```



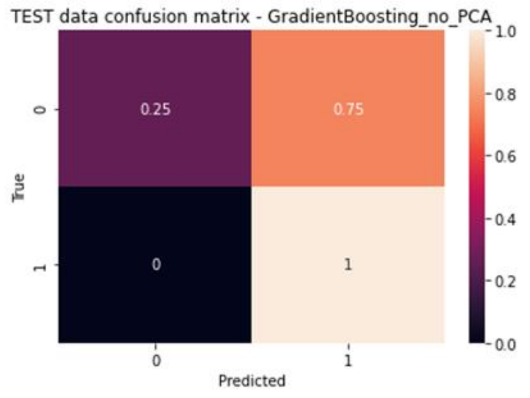
```
var_power60.0HzCh3                 0.027127
mean_Accelerometer_X               0.026263
var_amplitude4.0HzCh1              0.018707
mean_amplitude60.0HzCh3            0.015983
var_amplitude58.0HzCh3             0.015691
```

```
AdaBoost_no_PCA
for TESTING DATA
ROC_AUC Score: 0.607843137254902
Accuracy Score: 0.607843137254902
```



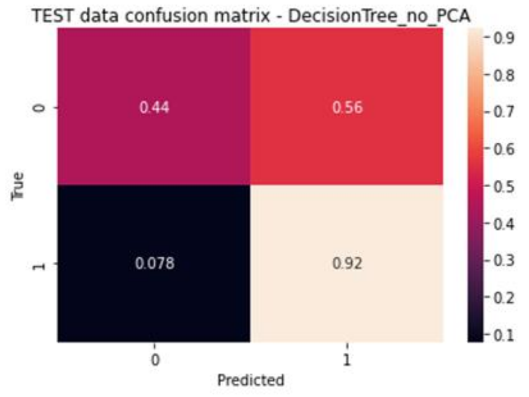
```
mean_Accelerometer_X                                    0.16
mean_Accelerometer_Y                                    0.14
var_phaseDifference_phase58.0HzCh0_phase58.0HzCh3       0.04
mean_Theta_AF7                                          0.04
mean_Mellow                                             0.04
```

```
GradientBoosting_no_PCA
for TESTING DATA
ROC_AUC Score: 0.6274509803921569
Accuracy Score: 0.6274509803921569
```



```
mean_Accelerometer_X                                    0.412758
var_power60.0HzCh3                                      0.273761
var_phaseDifference_phase60.0HzCh0_phase60.0HzCh3       0.047876
mean_Accelerometer_Y                                    0.045291
var_amplitude6.0HzCh1                                   0.016711
```

```
DecisionTree_no_PCA
for TESTING DATA
ROC_AUC Score: 0.6813725490196078
Accuracy Score: 0.6813725490196079
```

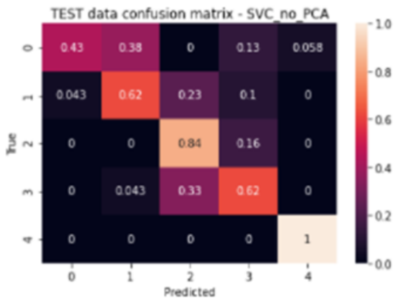


```
mean_Accelerometer_X                                    0.437724
var_power60.0HzCh3                                      0.288235
mean_Accelerometer_Y                                    0.055319
var_phaseDifference_phase60.0HzCh0_phase60.0HzCh3       0.045233
mean_power48.0HzCh0                                     0.039643
```

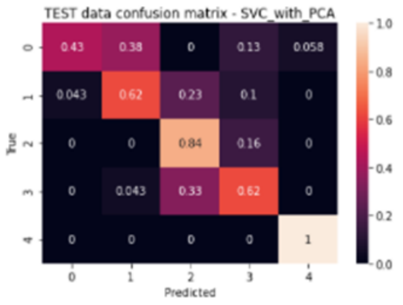
Experiment 5: All Classes Together

The models failed when trying to generalize to all 5 classes simultaneously. However, the eyes closed condition was still separable from the others in almost all cases.

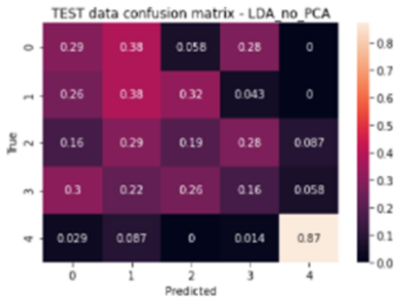
```
SVC_no_PCA
for TESTING DATA
Accuracy Score: 0.7043478260869566
```



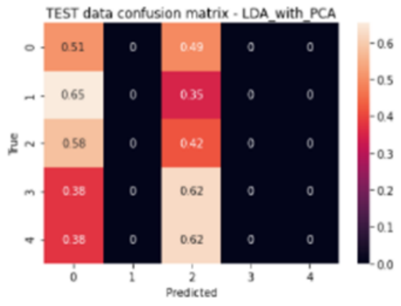
```
SVC_with_PCA
for TESTING DATA
Accuracy Score: 0.7043478260869566
```



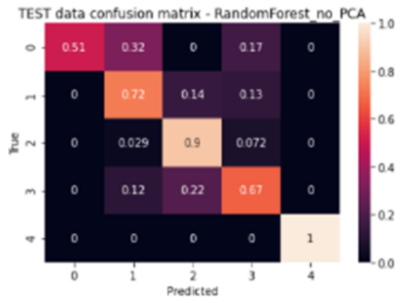
```
LDA_no_PCA
for TESTING DATA
Accuracy Score: 0.37681159420289856
```



```
LDA_with_PCA
for TESTING DATA
Accuracy Score: 0.1855072463768116
```

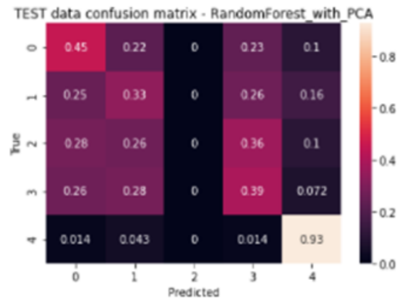


```
RandomForest_no_PCA
for TESTING DATA
Accuracy Score: 0.7594202898550725
```

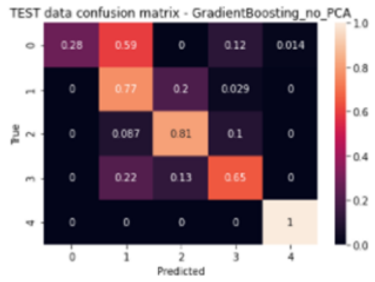


```
mean_power70.0HzCh1        0.013442
mean_Accelerometer_X       0.012673
var_Gyro_Z                 0.011025
mean_Accelerometer_Z       0.010582
var_Accelerometer_Y        0.008667
```

```
RandomForest_with_PCA
for TESTING DATA
Accuracy Score: 0.42028985507246375
```

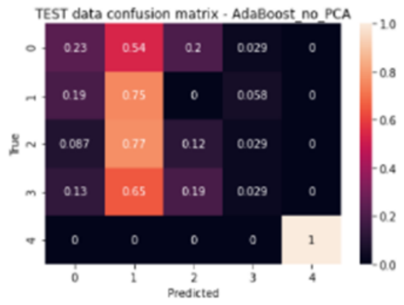


```
GradientBoosting_no_PCA
for TESTING DATA
Accuracy Score: 0.7014492753623188
```



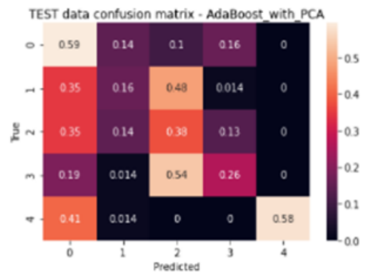
```
mean_amplitude46.0HzCh1                                   2.044194e-01
mean_Accelerometer_X                                      9.896705e-02
var_Gyro_Z                                                7.800960e-02
mean_Accelerometer_Z                                      6.597769e-02
var_amplitude62.0HzCh0                                    5.693320e-02
```

```
AdaBoost_no_PCA
for TESTING DATA
Accuracy Score: 0.42608695652173906
```

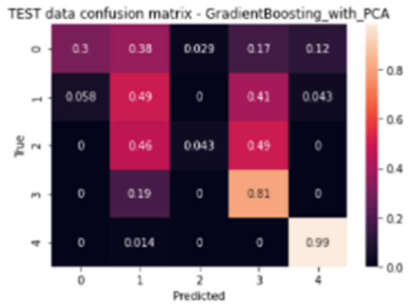


```
mean_amplitude46.0HzCh1                                   0.20
mean_power10.0HzCh3                                       0.04
var_amplitude10.0HzCh1                                    0.04
mean_Alpha_TP10                                           0.04
mean_amplitude10.0HzCh3                                   0.04
```

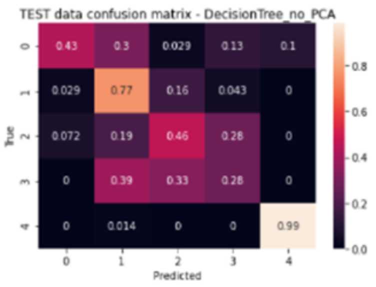
```
AdaBoost_with_PCA
for TESTING DATA
Accuracy Score: 0.3942028985507247
```



```
GradientBoosting_with_PCA
for TESTING DATA
Accuracy Score: 0.527536231884058
```

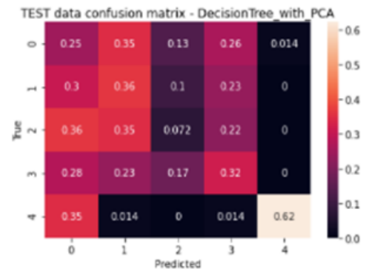


```
DecisionTree_no_PCA
for TESTING DATA
Accuracy Score: 0.5855072463768116
```



```
mean_amplitude46.0HzCh1                          0.219128
mean_Accelerometer_X                             0.132253
var_Gyro_Z                                       0.093260
mean_Accelerometer_Z                             0.092604
mean_Accelerometer_Y                             0.059958
```

```
DecisionTree_with_PCA
for TESTING DATA
Accuracy Score: 0.3246376811594203
```

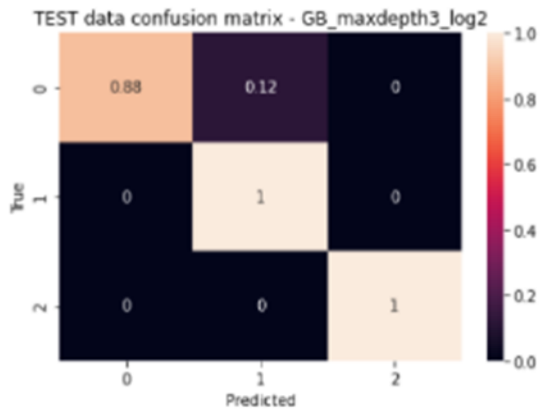


Experiment 6: Narrow and Optimize the Multi-Class Modeling

Here, the models were optimized to distinguish at least one active mental condition from the open eyes baseline and the closed eyes comparison. Some versatility was achieved, as shown below. The top features for the optimized models were gamma frequencies at the frontal electrodes.

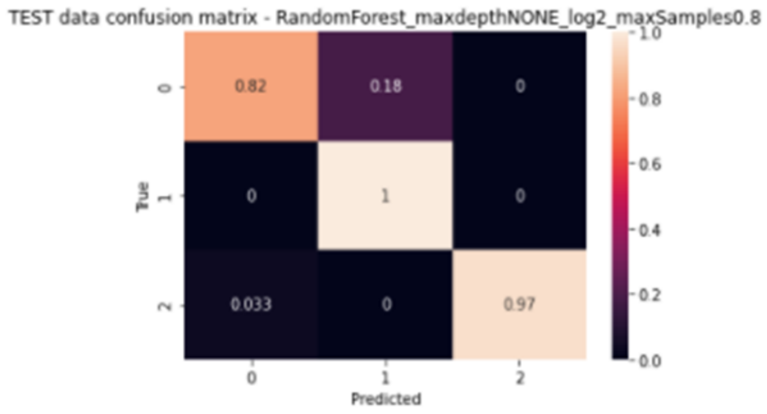
Experiment 6: Open Eyes vs Body Attention vs Closed Eyes			
pipeline_name	**named_steps**	**test_acc_score**	**test_confusion_matrix**
GradientBoosting_maxdepth3_log2	{'scaler': StandardScaler(), 'est': GradientBoostingClassifier(max_features='log2', n_estimators=300)}	96.1%	[[0.88333333 0.11666667 0.] [0. 1. 0.] [0. 0. 1.]]
RandomForest_maxdepthNONE_log2_maxSamples0.8	{'scaler': StandardScaler(), 'est': RandomForestClassifier(max_features='log2', max_samples=0.8, n_estimators=2000)}	92.8%	[[0.81666667 0.18333333 0.] [0. 1. 0.] [0.03333333 0. 0.96666667]]

```
GB_maxdepth3_log2
for TESTING DATA
Accuracy Score: 0.9611111111111111
```



```
mean_amplitude46.0HzCh1                                  0.066416
var_power36.0HzCh1                                       0.036009
var_power66.0HzCh1                                       0.030429
mean_power44.0HzCh2                                      0.024595
mean_power40.0HzCh1                                      0.024051
```

```
RandomForest_maxdepthNONE_log2_maxSamples0.8
for TESTING DATA
Accuracy Score: 0.9277777777777777
```



```
mean_amplitude44.0HzCh1    0.005346
mean_power70.0HzCh1        0.005292
mean_power60.0HzCh3        0.005239
var_power60.0HzCh3         0.005211
mean_amplitude60.0HzCh3    0.005178
```

Findings & Recommendations

Models were dominated by features that were either neural in origin, were movement artifacts, or were overfit to other types of noise. The most convincing models were separating open-eyes epochs from closed-eyes epochs, and this was done based on alpha frequencies at temporal lobe electrodes and gamma frequencies at frontal lobe electrodes. Gyroscope and accelerometer features are important for identifying movement artifacts and overfitting to posture.

Wavelet-extracted features were treated similarly to Muse's extracted frequency bands (delta, theta, alpha, beta, gamma). Among the wavelet-extracted features, the power and amplitude features were recruited by many models. Variance in phase difference between channels might have utility.

Exploratory data analysis showed that phase difference is more stable within a brain hemisphere than across hemispheres. Phase shifts are known to reflect functional connectivity between brain regions. Other research groups have employed Granger causality or similar strategies to find this functional connectivity. Further research may be able to use phase difference information from wavelet transformations to detect large-scale neural network changes.

Regarding model types, decision tree ensemble methods can be very effective in an unknown feature space. Support vector classifiers may be best for well-understood feature spaces, and they performed well in eyes-open vs eyes-closed discrimination.

Seeing how classification failed when testing for 5 classes, parallel pipelines for binary classification may be more effective than a single multiclass predictor.

Overall, this research shows that consumer EEG interfaces like Muse may have undiscovered potential. There is reason to believe that a highly automated self-experimentation framework could be used by consumers to identify mental activity patterns that can be detected by the EEG device.