BME 511 Midterm Project

Ben McAteer

0029592670

Code

Part A

1. Rref electrodes are subtracted, filter is added, epochs are extracted from the events to 1000ms after, Baseline DC value is subtracted, and noisy trials are rejected leaving 389 good target epochs and 2225 good non target epochs seen in the legend of Figure 1. Filtering must be done at the beginning to reduce the 0Hz noise and 60Hz noise that would effect the 40microvolts cutoff and filtering at the beginning reduces the amount of times that the data needs to be run through a filter. Rather than every epoch, it can just be ran once for all the data.

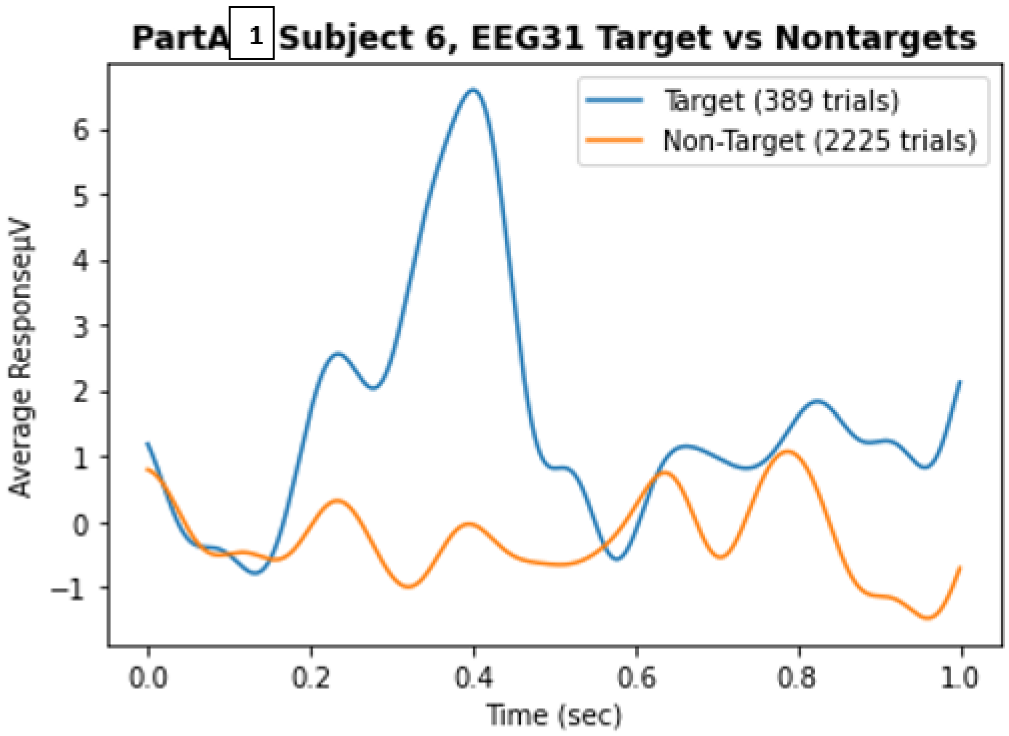

Figure 1: Average Target and Non-target Epochs Across all Trials

2. In part 2 of A, the avg target is subtracted by the average non target for each electrode seen in figure 2. There are 389 targets and 2225 nontargets as seen in Figure 1. There is a very apparent P300 response on almost all the electrodes. As seen in figure 2.2, the electrodes located in the back of the head gave the least significant response. The channels most useful for discriminating target and non-target events based on this data would be the EEG's located near the forehead and in the front of the grid like 1, 30, 2, 29, and 31.

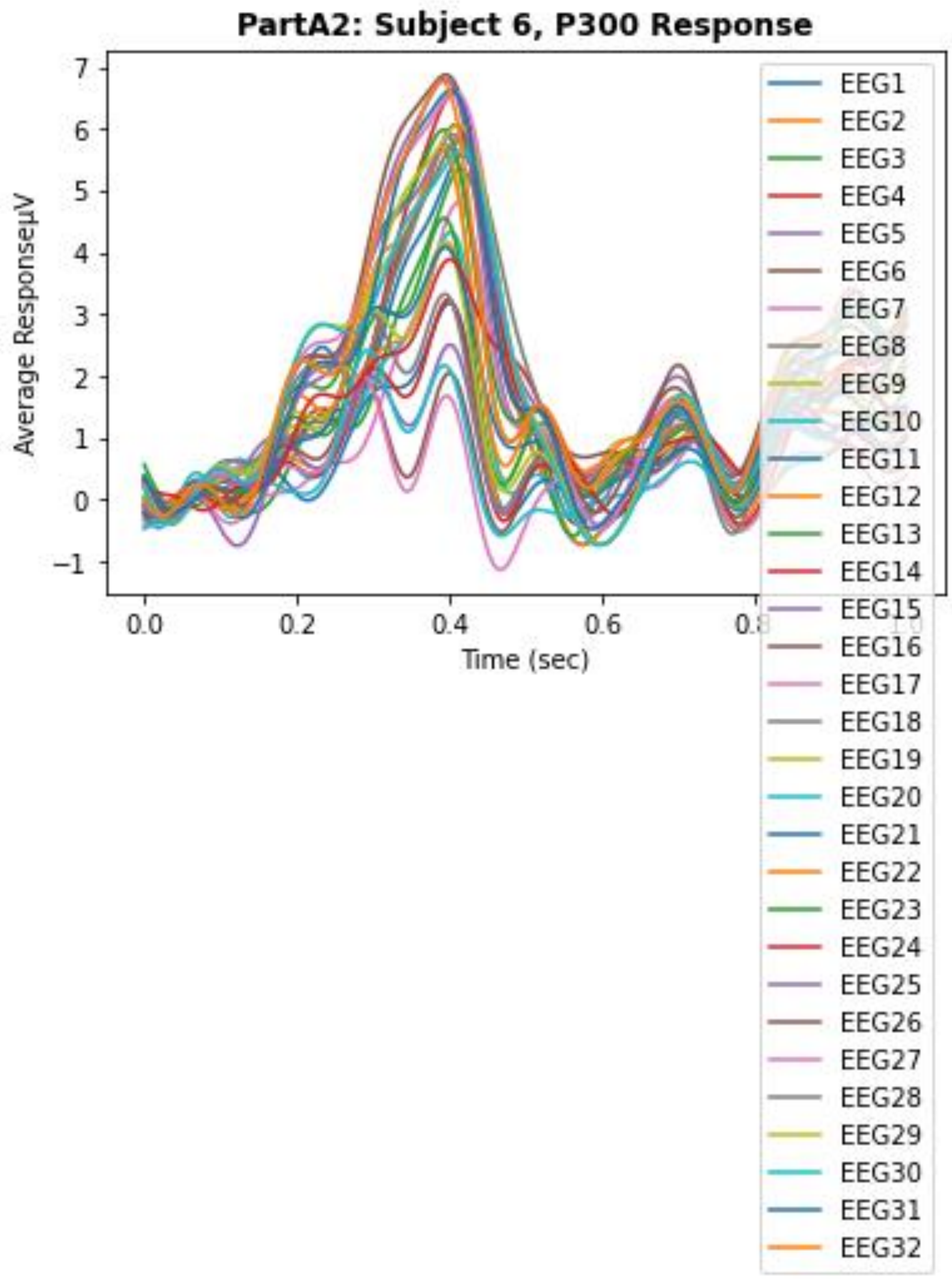

Figure 2.1: P300 Response for Subject 6 All EEG

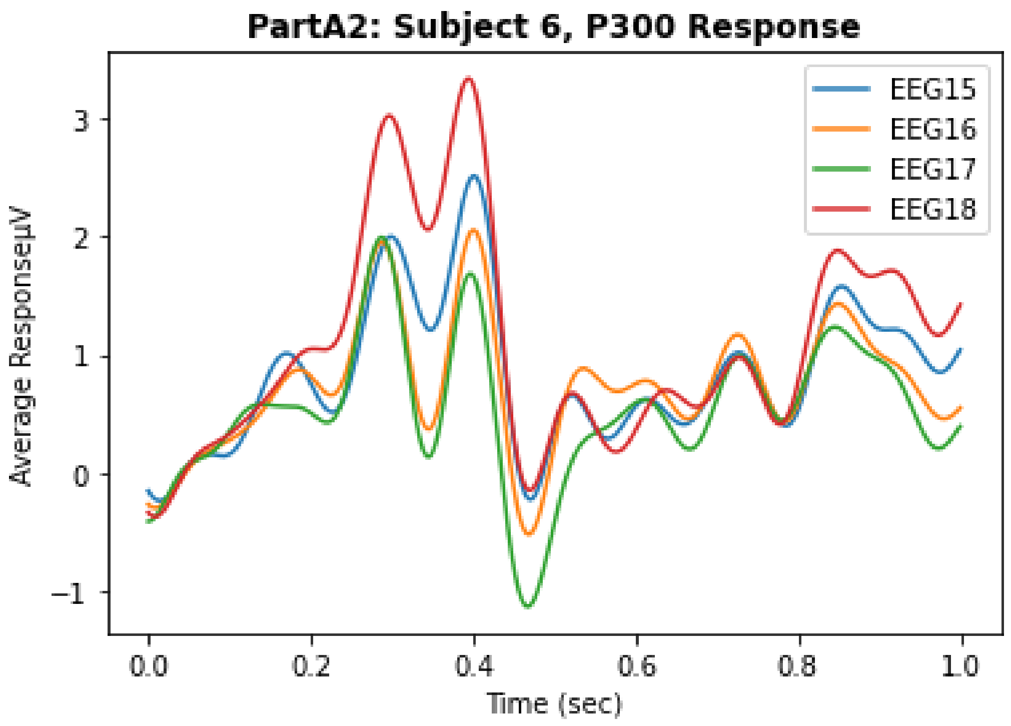

Figure 2.2: EEG response for non-responding EEGs

3. Taking from the list of subjects in figure 3.1, subject 4 has a brain injury and spinal injury that may inhibit brain activity. When comparing Figures 2.1 and 3.2, there are obvious p300 spikes on EEG 31 for both subjects, but the control subject has much higher magnitude. The timing of spikes is nearly the same but the Subject with brain trauma has a weaker response.

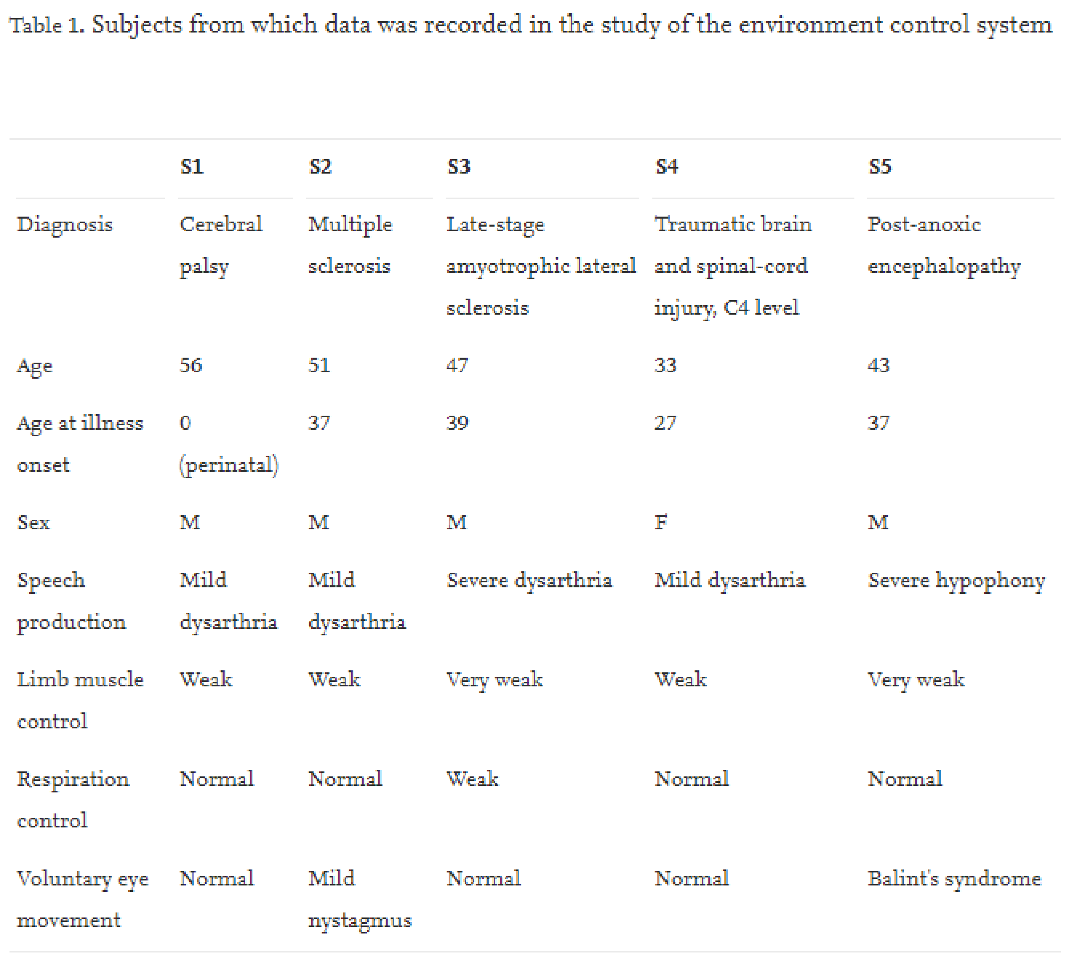

Table 1. Subjects from which data was recorded in the study of the environment control system

	S1	S2	S3	S4	S5
Diagnosis	Cerebral palsy	Multiple sclerosis	Late-stage amyotrophic lateral sclerosis	Traumatic brain and spinal-cord injury, C4 level	Post-anoxic encephalopathy
Age	56	51	47	33	43
Age at illness onset	0 (perinatal)	37	39	27	37
Sex	M	M	M	F	M
Speech production	Mild dysarthria	Mild dysarthria	Severe dysarthria	Mild dysarthria	Severe hypophony
Limb muscle control	Weak	Weak	Very weak	Weak	Very weak
Respiration control	Normal	Normal	Weak	Normal	Normal
Voluntary eye movement	Normal	Mild nystagmus	Normal	Normal	Balint's syndrome

Figure 3.1: List of Subject's descriptions for Data set (subject 6-9 are controls)

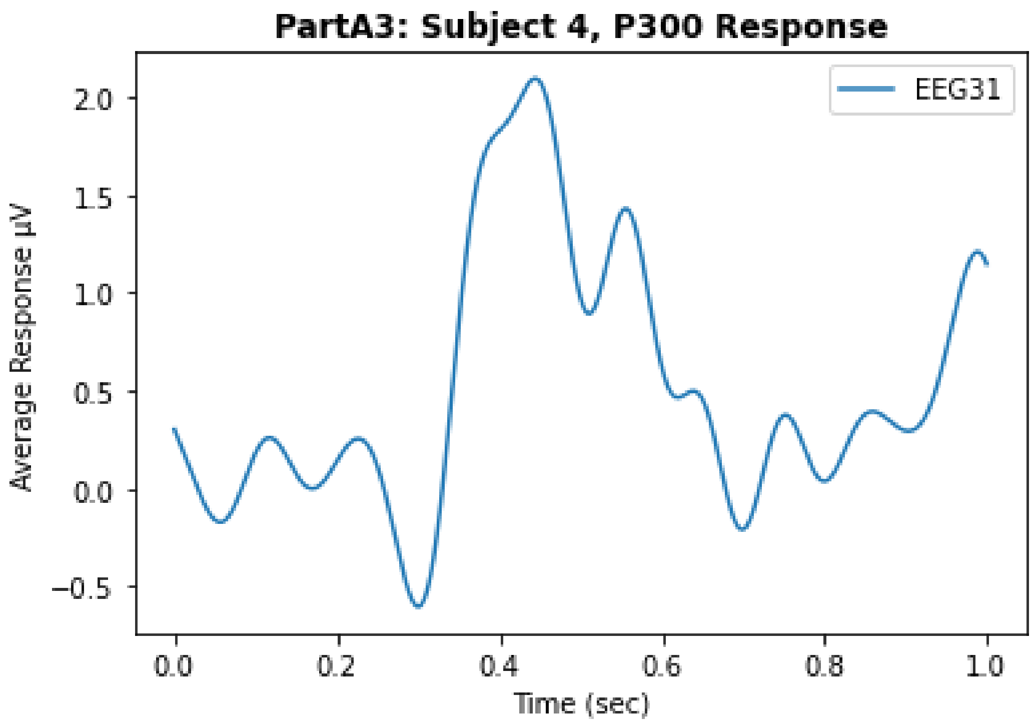

Figure 3.2: Subject 4, EEG 31 P300 Response

Part B

1. In part B1, the Null Data is compared to the actual difference in target and non-target responses for subject 6 EEG 31. Using the vertical lines, it can be seen in Figure 4.1 there is an apparent difference in the actual waveform from the null data around 0.2-0.5ms. This window will be

used in the future questions in dimensionality reduction.

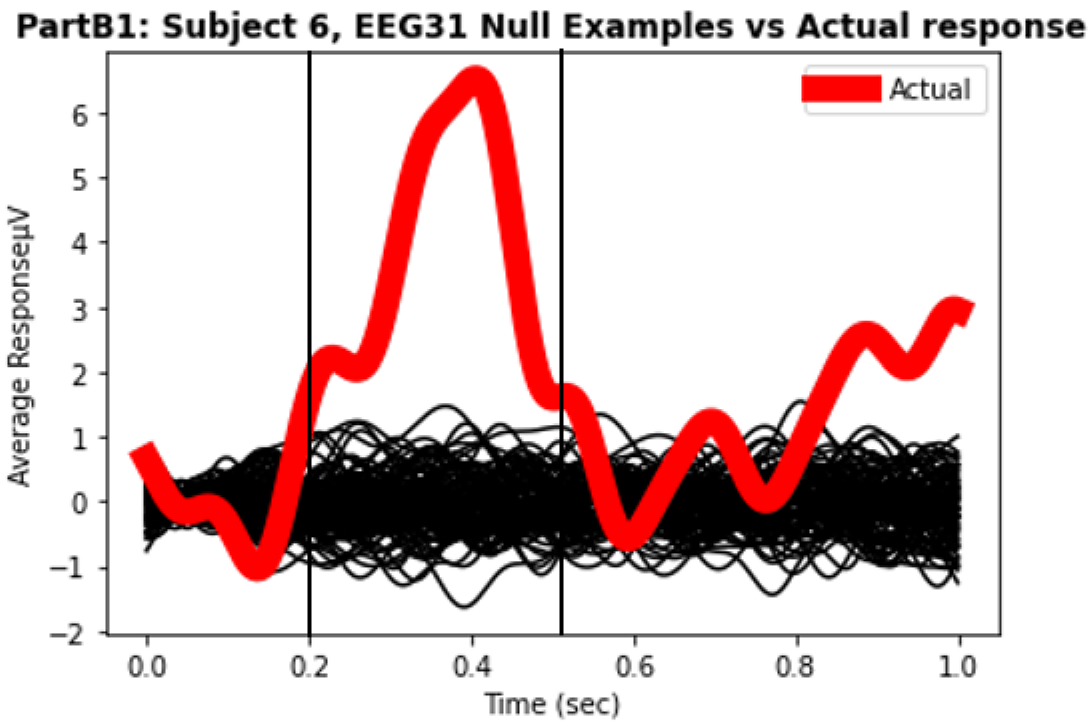

Figure 4.1: Permutation of Null Data (N=100) compared to Actual Difference

2. *From Print Statement* "B2: The Pvalue of channel Fz for subject 6 is 0.0." This is given 1000 Null examples Permutations. This is because in all of the null data, the max height of the difference of targets of the actual target peak is always higher than the Null. This was calculated using all 24 trials. And can be seen below in figure 4.2

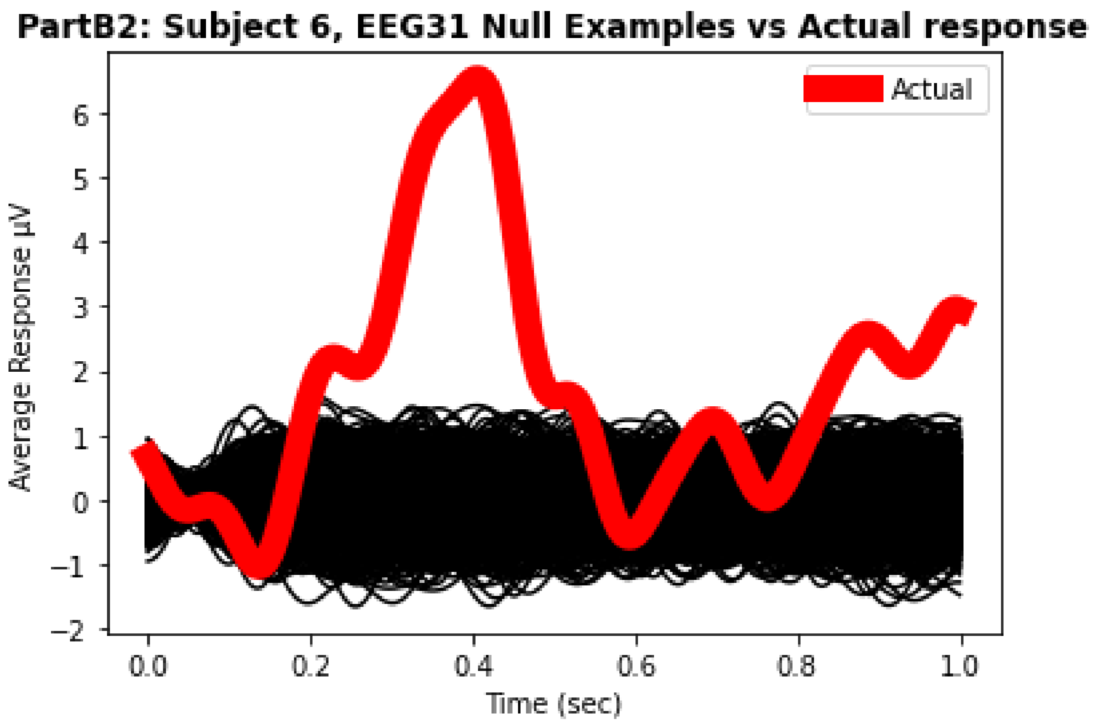

Figure 4.2: Permutation of Null Data (N=1000) compared to Actual Difference

3. *From Print Statement* "B3: the P Values for Cz, Pz, and Oz are 0.0, 0.0, and 0.0 at their peak heights"
   Seen in graphs 4.3, 4.4, 4.5 for electrodes Cz, Pz, and Oz, respectively, the actual difference in target vs non-target data is plotted against the null data. For each electrode on the control, the P-value is still 0 but it should be noted that as the electrodes move further back on the head, the difference in target and nontarget become less drastic moving from over 6microvolts to only 2microvolts.
   *From Print Statement "*B3: The peak height occurs at 405.76171875 ms for channel Fz, 384.27734375 ms for Cz, 389.6484375 ms for Pz, 290.0390625 ms for Oz". The timepoints of the peaks all appear in similar time points from around 250ms to 500ms. This was the selected window from part B1. This can be seen by where the red bar escape from the black null examples on each figure below.

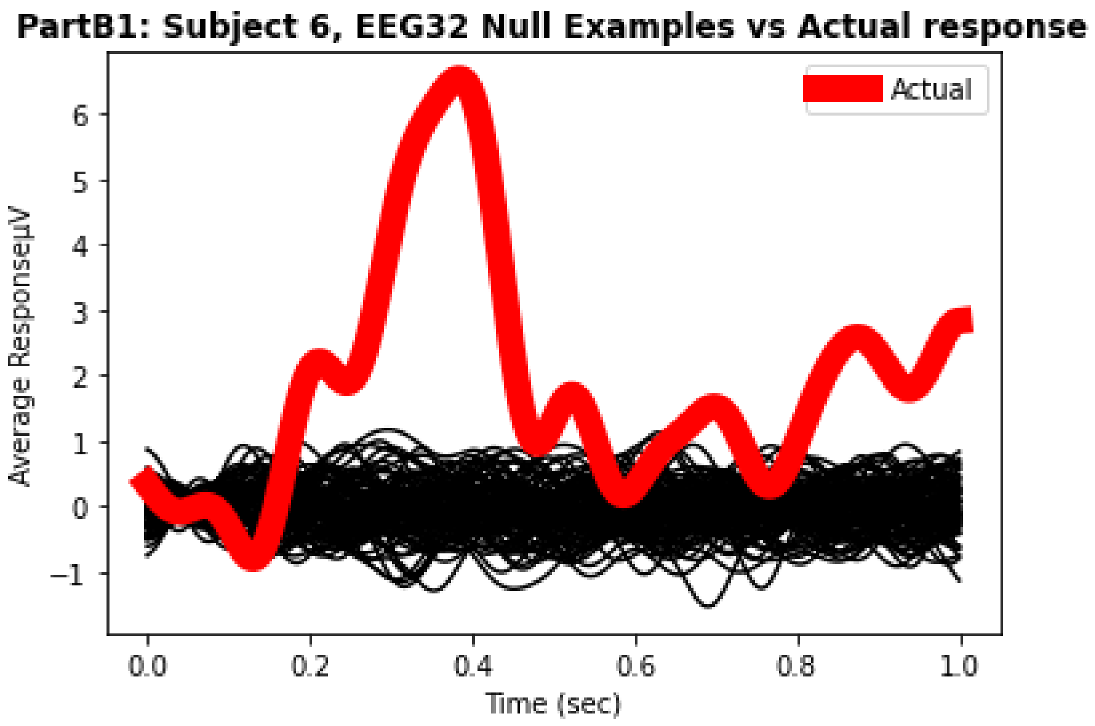

Figure 4.3: Permutation on Electrode 32 vs actual Difference

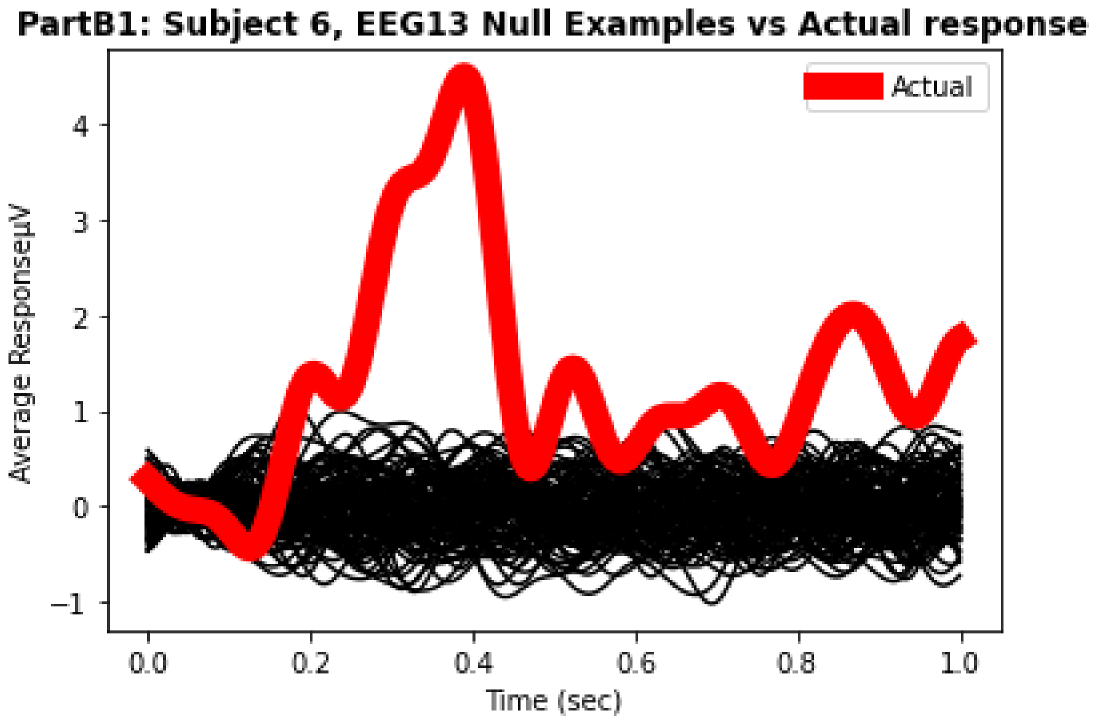

Figure 4.4: Permutation on Electrode 13 vs actual Difference

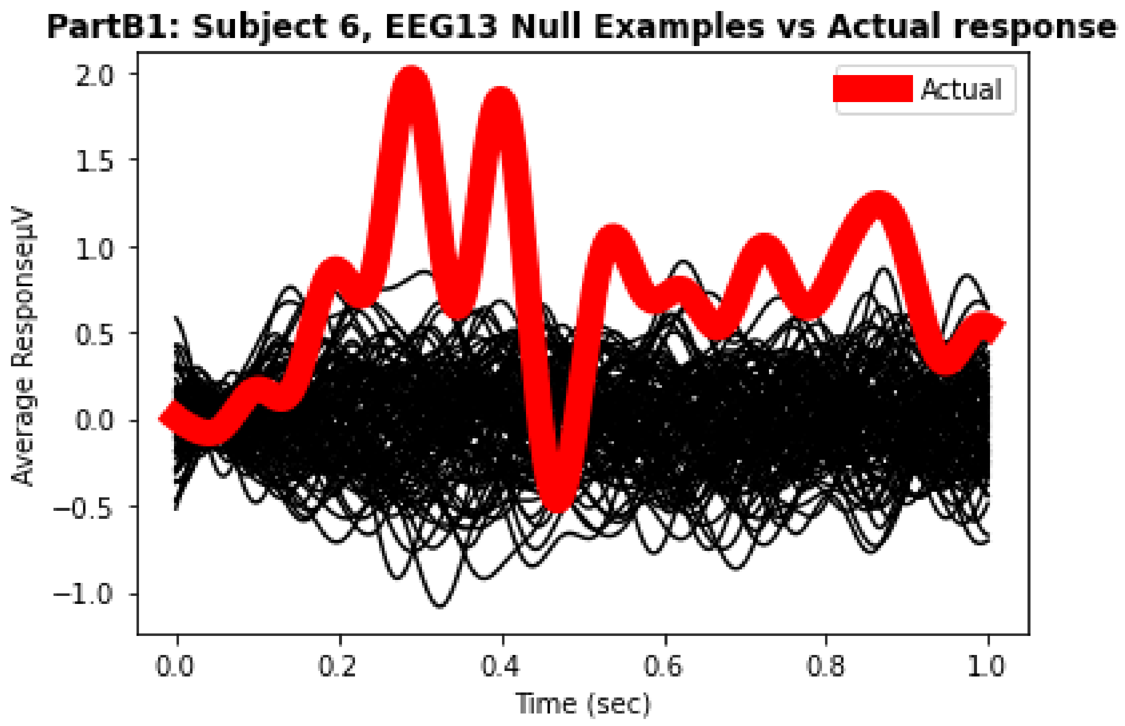

Figure 4.5: Permutation on Electrode 16 vs Actual Difference

Part C

1. For Subject 6…
   - For a full 1st session of data, N/4. The Pvalue of channel Fz is 0.0
   - For a full 2nd session of data, N/4. The Pvalue of channel Fz is 0.0
   - For a full 3rd session of data, N/4. The Pvalue of channel Fz is 0.0
   - For a full 4th session of data, N/4. The Pvalue of channel Fz is 0.0

   - For Session 1 Trial 1, N/24, the p value is 0.004
   - For Session 1 Trial 2, N/24, the p value is 0.01
   - For Session 1 Trial 3, N/24, the p value is 0.034
   - For Session 1 Trial 4, N/24, the p value is 0.019
   - For Session 1 Trial 5, N/24, the p value is 0.003
   - For Session 1 Trial 6, N/24, the p value is 0.022
   - For Session 2 Trial 1, N/24, the p value is 0.0
   - For Session 2 Trial 2, N/24, the p value is 0.0
   - For Session 2 Trial 3, N/24, the p value is 0.0
   - For Session 2 Trial 4, N/24, the p value is 0.541
   - For Session 2 Trial 5, N/24, the p value is 0.012
   - For Session 2 Trial 6, N/24, the p value is 0.092
   - For Session 3 Trial 1, N/24, the p value is 0.017

- For Session 3 Trial 2, N/24, the p value is 0.28
- For Session 3 Trial 3, N/24, the p value is 0.001
- For Session 3 Trial 4, N/24, the p value is 0.008
- For Session 3 Trial 5, N/24, the p value is 0.002
- For Session 3 Trial 6, N/24, the p value is 0.0
- For Session 4 Trial 1, N/24, the p value is 0.001
- For Session 4 Trial 2, N/24, the p value is 0.0
- For Session 4 Trial 3, N/24, the p value is 0.0
- For Session 4 Trial 4, N/24, the p value is 0.0
- For Session 4 Trial 5, N/24, the p value is 0.056
- For Session 4 Trial 6, N/24, the p value is 0.002
    - Average Pvalue per trial was 0.0439

The overall P values from session to session are still over 100% confident that the target data was true. Additionally, the p values of each trial individually tend to be very low with the acceptation of two or three trials. The average P value per trial was still under 0.05 which means that it is more than 95% confident that the max of the target is more than the Nulls Generated. This means that there could be a reduction in trial time, or the number of trials and a confident result could still be drawn. The sessions could be reduced from 6 trials to 2 or 3 trials and still receive a P Value close to 0.0. The target acquisition is in very high confidence for subject 6.

2. Seen in figure 5.1, with each P value for all 24 trials plotted out, there are 3 trials, 10, 12, and 14, that had abnormally high p-values. Otherwise, this data hosted very consistent and low p-values. This is significant as it proves that the BCI in a real-world performance could yield consistent results. With the pairing of two trials worth of data, it could be assumed that nearly all p-values would be within an acceptable range and the target selection of the device would be accurate and reliable.

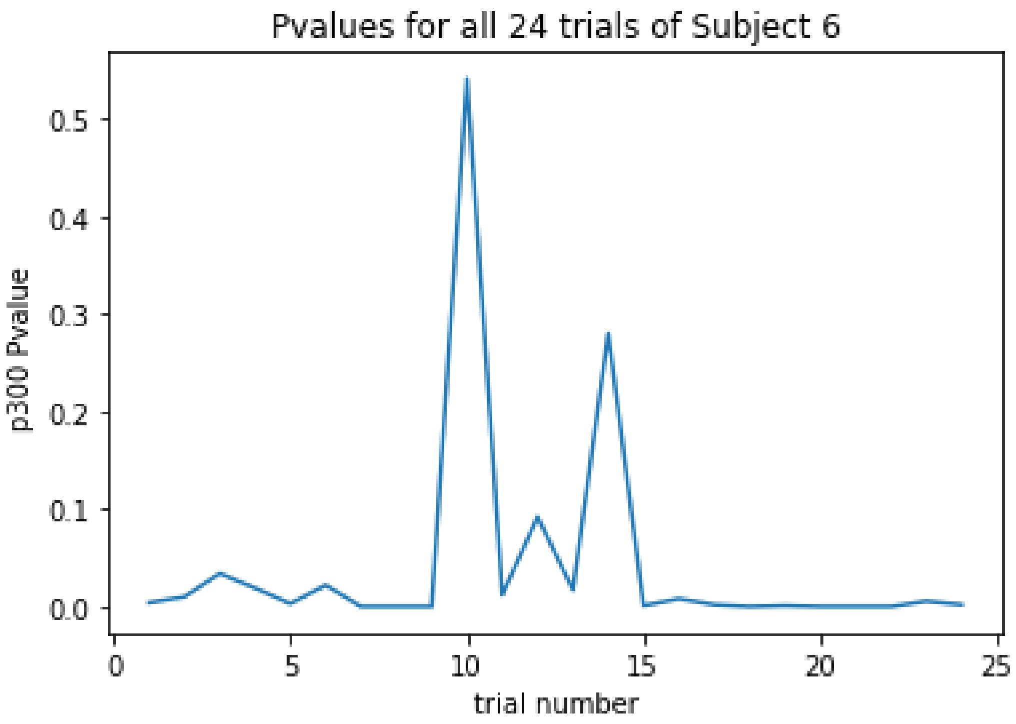

Figure 5.1 P values for each trial on Subject 6

3. For Subject 4 with a C4 spinal cord injury and traumatic brain injury
    - For a full 1st session of data, N/4. The Pvalue of channel Fz is 0.348
    - For a full 2nd session of data, N/4. The Pvalue of channel Fz is 0.302
    - For a full 3rd session of data, N/4. The Pvalue of channel Fz is 0.038
    - For a full 4th session of data, N/4. The Pvalue of channel Fz is 0.0
        - Session average pvalue = 0.172

    - For Session 1 Trial 1, N/24, the p value is 0.986
    - For Session 1 Trial 2, N/24, the p value is 0.048
    - For Session 1 Trial 3, N/24, the p value is 0.755
    - For Session 1 Trial 4, N/24, the p value is 0.678
    - For Session 1 Trial 5, N/24, the p value is 0.07
    - For Session 1 Trial 6, N/24, the p value is 0.071
    - For Session 2 Trial 1, N/24, the p value is 0.529
    - For Session 2 Trial 2, N/24, the p value is 0.79
    - For Session 2 Trial 3, N/24, the p value is 0.518
    - For Session 2 Trial 4, N/24, the p value is 0.794
    - For Session 2 Trial 5, N/24, the p value is 0.638
    - For Session 2 Trial 6, N/24, the p value is 0.325
    - For Session 3 Trial 1, N/24, the p value is 0.157
    - For Session 3 Trial 2, N/24, the p value is 0.094
    - For Session 3 Trial 3, N/24, the p value is 0.205

- For Session 3 Trial 4, N/24, the p value is 0.485
- For Session 3 Trial 5, N/24, the p value is 0.462
- For Session 3 Trial 6, N/24, the p value is 0.006
- For Session 4 Trial 1, N/24, the p value is 0.485
- For Session 4 Trial 2, N/24, the p value is 0.396
- For Session 4 Trial 3, N/24, the p value is 0.042
- For Session 4 Trial 4, N/24, the p value is 0.001
- For Session 4 Trial 5, N/24, the p value is 0.035
- For Session 4 Trial 6, N/24, the p value is 0.482
    - Average pvalue per trial was 0.393

The overall P values from session to session averaged to 0.172. This alone is not a very high p-value and would pose many errors in target acquisition confidence. Additionally, the p values of each trial individually tend to be very high as seen in Figure 6.1. This means that if this subject were trained on one trial with an average p-value of 0.393, then tested on the other twenty-three trials, they would perform very bad. This is likely because of their existing brain and spinal injury inhibiting brain function and overall action potential propagation. Their injuries are also the likely reason as to why they performed so much worse than subject 6 when comparing the magnitude of P values in Figures 5.1 and 6.1.

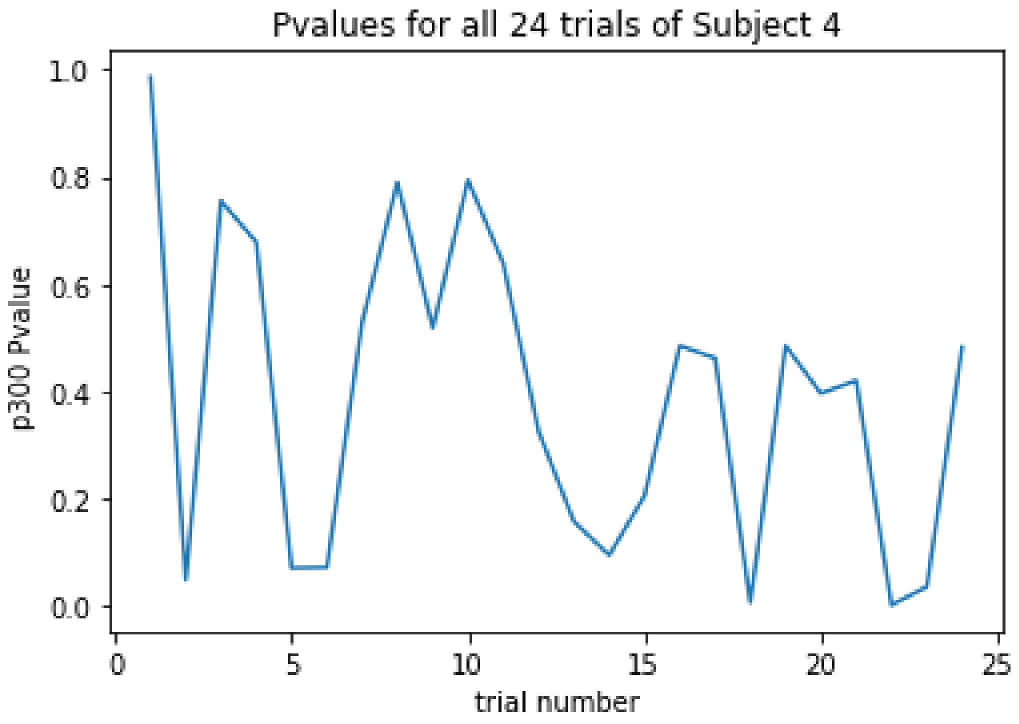

Figure 6.1 P values for each trial on Subject 4

4. Using the Window of 250-500ms determined in part B, and PCA, the targets and non-targets were reduced to 2 smaller principle components. The window was used in order to show the maximum difference and statistically significant difference between targets and non-targets. PCA can be used because gaussian distribution is assumed. The built-in pca function was used to reduce the dimensions as it can sort the orthogonal dimensions that are minimally correlated and maximally variable. It can calculate the covariance, eigen vectors, and depict the largest possible difference between the two sets of data. Then the principle components were graphed below in Figure 7.1. Before PCA could be calculated, the 4 EEG data had to be concatenated into one long set of target and non-target data.

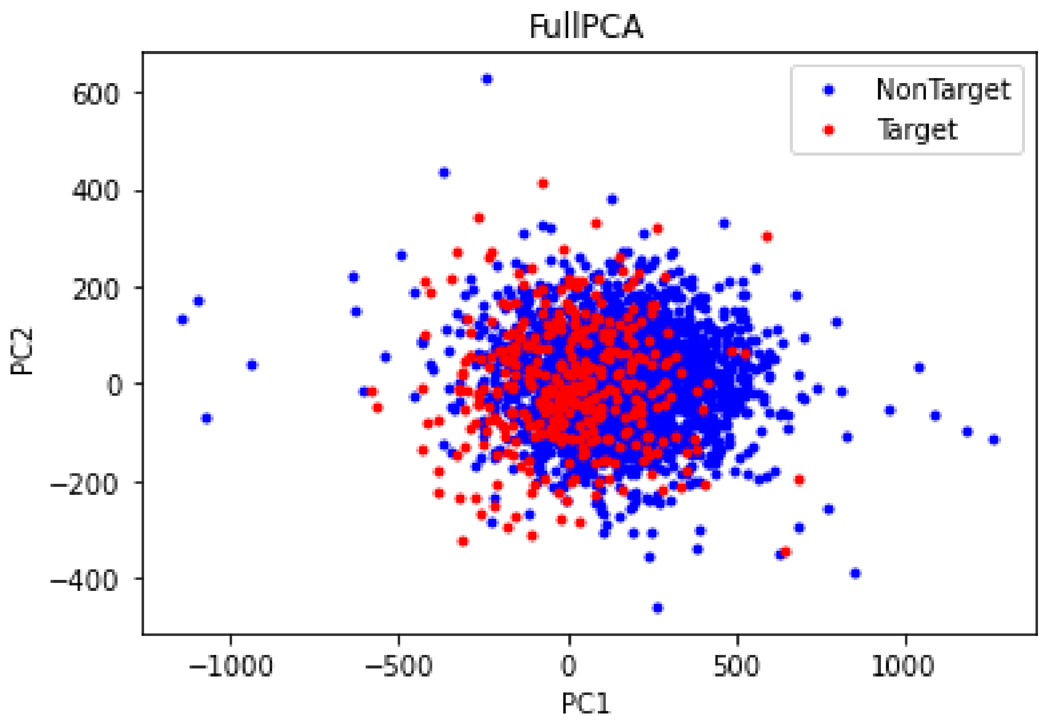

Figure 7.1: PCA of Target and Non-Target Data

5. As seen in Figure 7.1 Full PCA leaves too many data points to accurately classify two clusters with normal boundaries. Therefore, a k10, k25, and k50average of the data were graphed in Figures 7.2, 7.3, and 7.4 below. Here it is possible to see clusters begin to emerge. In k10, there are still a significant amount of data points to observe, but there is also a decent amount of intertwining and therefore creating boundaries would introduce some false positives and false negatives. The K25 data set has less data and significantly less overlap between the two clusters. A single false positive here would be much more important than a false positive of k10 however. The k50 set does not have many data points to observe and is rather sparce. It may be too far averaged to reliably pull information from. Very clear clusters are seen though.

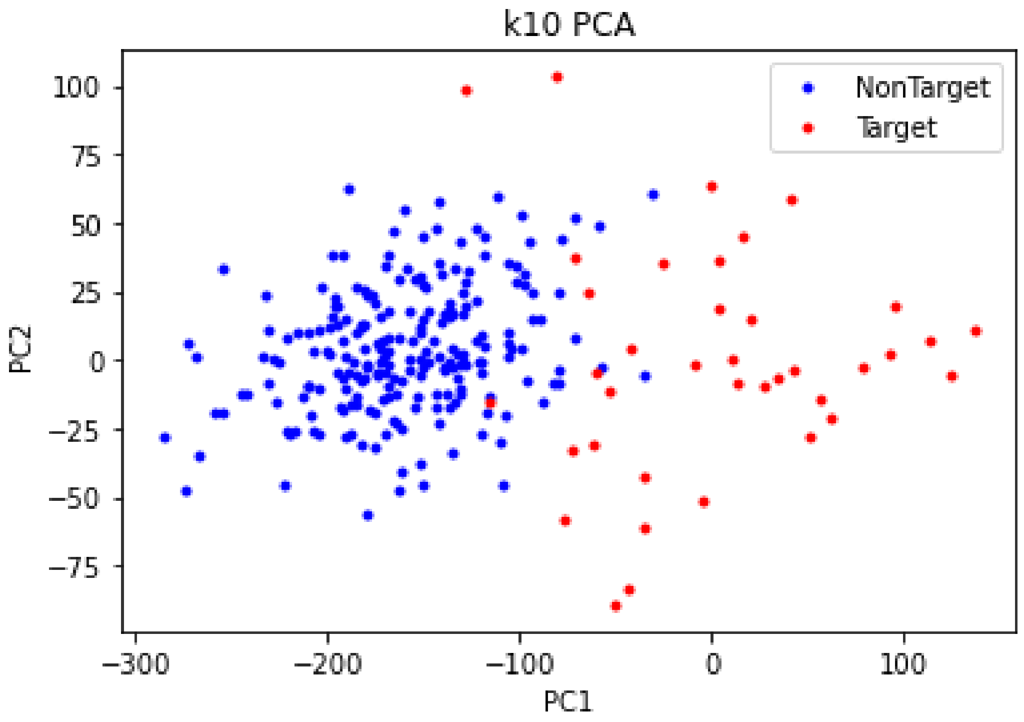

Figure 7.2: PCA of Target and Non-Target Data Averaged by Every 10 Points

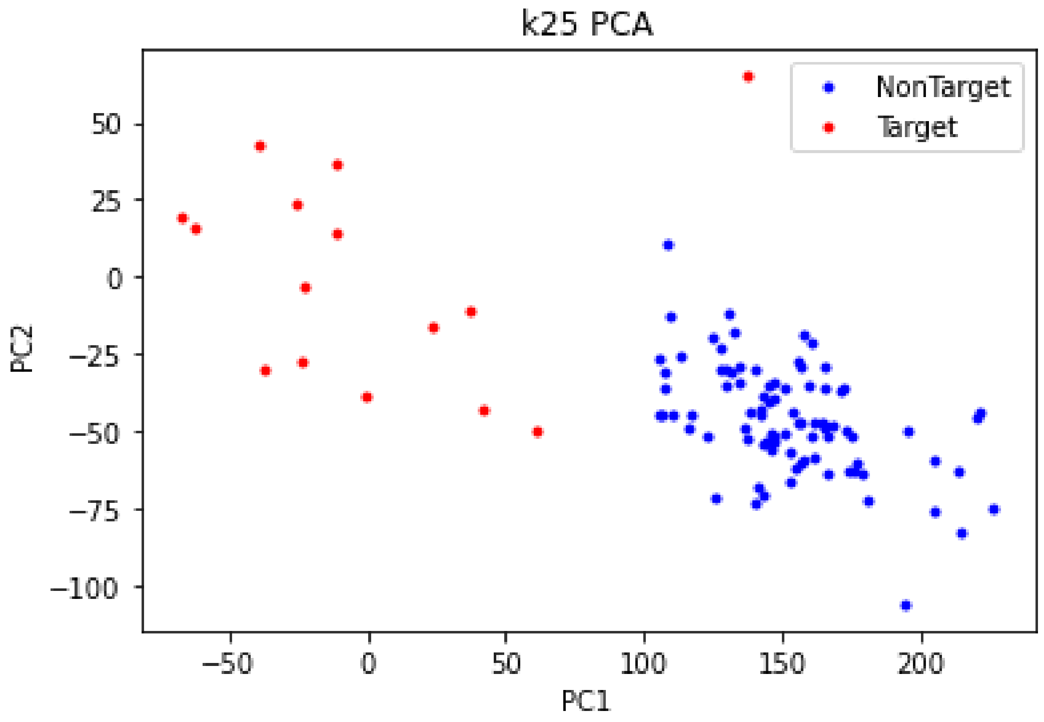

Figure 7.3: PCA of Target and Non-Target Data Averaged by Every 25 Points

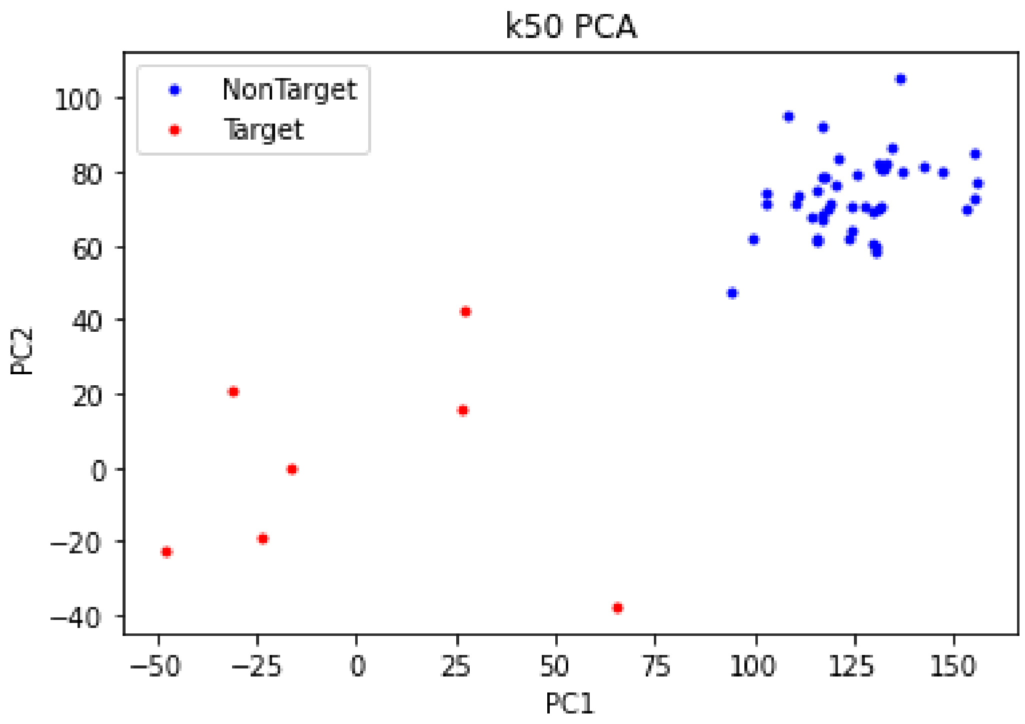

Figure 7.4: PCA of Target and Non-Target Data Averaged by Every 50 Points

6. After drawing a vertical boundary line through PCA1 with the formula -130 + (i)*18 for "i" iterating from 1 through 8, the sensitivity and specificity was calculated and plotted below in Figure 8.1. I personally would choose a boundary at around -100 as it would yield the highest specificity and sensitivity. It should be noted that the axis on the figure distort the values but would not show up if axis limits were presented. A boundary at PCA = -100 would yield a Specificity around 98.5% and a sensitivity around 95%. In real world applications of this BCI, I would like to evaluate its performance on speed of target acquisition as well as accuracy. There is already some data to support this evaluation in this study. Other ways to evaluate this BCI would be more qualitative as a device: ease of use, durability, practicality. The other main quantitative evaluation I would suggest is the introduction of more non-targets. Instead of 6 pictures, how would this BCI work with 12 pictures?

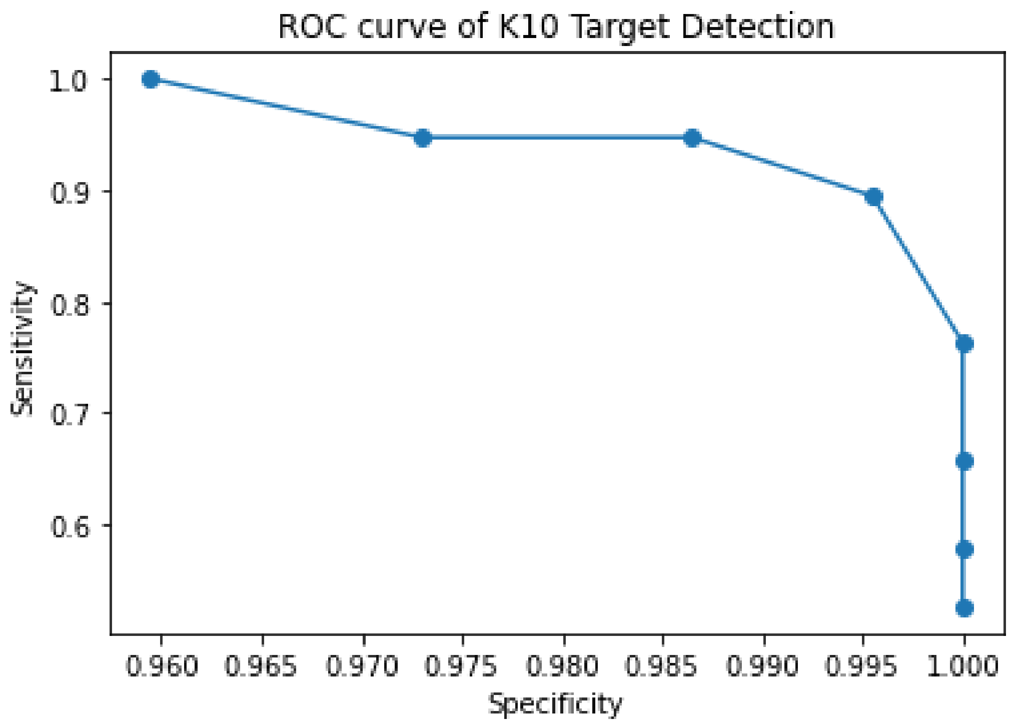

Figure 8.1: ROC Curve given 8 boundary lines

All CODE:

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 11 17:31:50 2021

@author: Ben
"""

#Ben McAteer Midterm Project BME 511

import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
import scipy.signal as signal
from sklearn.decomposition import PCA
import itertools
#import pandas as pd

#from sklearn.decomposition import PCA
#from scipy import signal
from scipy.io import loadmat
```

```
#from numpy.fft import fft, ifft
#from scipy.signal import find_peaks
#from scipy import linalg

import glob
import os

NumTrials = 6
fs = 2048 #Hz sample rate

data1 = {}
events1 = {}
stimuli1 = {}
target1 = {}
targets_counted1 = {}

def SST (Subject,Session,Trial,count): # pulls all matlab files for a specific subject in order from
session 1 first run to the final session

    os.chdir('/Users\Ben\OneDrive - purdue.edu\BME
511\MidtermProject\AllSubjsEEG\subject{}\session{}'.format(Subject,Session))
    mat_files = glob.glob('*.mat')
    #print(mat_files) #troubleshoot
    currentTrial = loadmat(mat_files[Trial], squeeze_me = True)



    currentfile = 'data' + str(count)

    data1[currentfile] = currentTrial['data'] #raw data of the current trial
    events1[currentfile] = currentTrial['events']
    stimuli1[currentfile] = currentTrial['stimuli']
    target1[currentfile] = currentTrial['target']
    targets_counted1[currentfile] = currentTrial['targets_counted']

    # print('data size', np.shape(Trial['data']))
    # print('events size', np.shape(Trial['events']))
    # print('stimuli size', np.shape(Trial['stimuli']))
    # print('target ', (Trial['target']))
    # print('targetcounted ', (Trial['targets_counted']))

    # for key, value in data.items():
    #    print(key)
    return(data1,events1,stimuli1,target1,targets_counted1)

def unpackStamp(x):
    y = np.int32(x[0])
    mo = np.int32(x[1])
    d = np.int32(x[2])
    h = np.int32(x[3])
    mi = np.int32(x[4])
    s = x[5]
    s_new = np.int32(np.floor(s))
    micros = np.int32((s - s_new) * 1e6)
    unpacked = datetime.datetime(y, mo, d, h, mi, s_new, micros)
    return unpacked


def events2samps(events, fs):
    firsteve_time = 0.4
    Nevents = events.shape[0]
    evesamps = np.zeros(Nevents)
    for k in range(Nevents):
        td =  unpackStamp(events[k, :]) - unpackStamp(events[0, :])
        evesamps[k] = np.int32(np.round(td.total_seconds()*fs + firsteve_time*fs + 1))

    return evesamps

#1A
#from numpy.fft import fft, ifft
from scipy import signal
import datetime

def
sorting_epoch(epochs_data,epochs_events,epochs_stimuli,epochs_target,epochs_targets_coun
ted,count, targetcount, nontargetcount, noisytargetcount):

    curfile = 'data' + str(count)

    #print(np.shape(data))

    # for key, value in epochs_data.items():
    #     print(key)

    datacur = epochs_data[curfile]
    eventscur = epochs_events[curfile]
    stimulicur = epochs_stimuli[curfile]
    targetcur = epochs_target[curfile]
    targets_countedcur = epochs_targets_counted[curfile]


    RefT = []
    NewData = []

    lowfreq = 1 #Hz low frequency of the filter
    highfreq = 12 #Hz high frequency of the filter

    RefT = datacur[32,:] * .5 + datacur[33,:] * 0.5

    for i in range (0,32):
        NewData.append(datacur[i] - RefT) #subtracts the data by the average of the two reference
EEG leads


    #create and filter Data
    filt_len = int(np.ceil(1./.5 * fs)) #filter length with 0.5 being the transition band
    h_filter = signal.firwin(filt_len, [lowfreq,highfreq], fs=fs, pass_zero=False) #compute filter
    FilterData = signal.filtfilt(h_filter,1,NewData,axis = -1) #Filtered Data with baseline filtered out


    Eventsdata = events2samps(eventscur,fs)#index where event occurs in the session

    epochs = []#np.zeros(len(Eventsdata))
    #time = np.arange(0,1,1/fs) #0 to 1 second at sample freq
    targetindex = np.zeros(len(Eventsdata))
    nontargetindex = np.zeros(len(Eventsdata))

    for i in range(len(Eventsdata)):#Eventsdata.astype('int'):
        epochs.append(FilterData[:,int(Eventsdata[i]):int(Eventsdata[i]+2048)]) #events +1000ms of
data afterwards
        if stimulicur [i] == targetcur:
            targetindex[i] = 1
        else:
            nontargetindex[i] = 1

            #need to create the difference beweteen targets vs non
    target_epoch = []
    nontarget_epoch = []
    for i in range(len(Eventsdata)): #currently, instead of making the target epoch 34x whatever, it
is just appending and making it all one long line
        if targetindex[i] == 1:
            target_epoch.append(epochs[i])
        else:
            nontarget_epoch.append(epochs[i])




 ##find avg of the first 100 of each epoch, then subtract avg from each epoch

    targetData[curfile] = []
    nontargetData[curfile] = []
    BStarget_epoch = np.zeros_like(target_epoch)
    BSnontarget_epoch = np.zeros_like(nontarget_epoch)
    #subtracting the baseline DC from the target epochs
    for i in range(0,len(target_epoch)):
        # targetData[curfile].append(target_epoch[i][:][:] - (np.mean(target_epoch[i][:][0:205])))
        for j in range(0, 32):

            #epochmean = np.mean(target_epoch[i][j][0:205])
            BStarget_epoch[i][j][:] = target_epoch[i][j][:] - np.mean(target_epoch[i][j][0:205])

        targetData[curfile].append(BStarget_epoch[i][:])

    for i in range(0,len(nontarget_epoch)):
        #nontargetData[curfile].append(nontarget_epoch[i][:] -
np.mean(nontarget_epoch[i][:][0:205])) #removes the drifting baseline from all epochs
        for j in range(0, 32):
             #epochnonmean = np.mean(nontarget_epoch[i][j][0:205])
             BSnontarget_epoch[i][j][:] = nontarget_epoch[i][j][:] -
np.mean(nontarget_epoch[i][j][0:205])

        nontargetData[curfile].append(BSnontarget_epoch[i][:])

## REMOVING NOISE OVER 40uV
    target_epoch_max[curfile] = []
    nontarget_epoch_max[curfile] = []
    good_targetepoch[curfile] = []
    good_nontargetepoch[curfile] = []

    for i in range(0,len(targetData[curfile])):
        target_epoch_max[curfile].append(np.amax(abs(targetData[curfile][i])))
        noisytargetcount  += 1
        if target_epoch_max[curfile][i] <= 40:
            good_targetepoch[curfile].append(targetData[curfile][i]) #should creat a dictionary of 
only good epochs in target
            targetcount += 1
            #targetData[curfile][i] = 0 #use np.nanmean to avoid nan's in the future
    #filttargetData[curfile] = targetData[~np.isnan(targetData)]    #use np.nanmean to avoid nan's 
in the future
        #if target_epoch_max[curfile][i] == 0:
            #targetData[curfile][i]= np.delete(targetData[curfile][i],np.nan) #Currently need to 
remove the rows but heeccccc
            #targetData[curfile][i] 
=targetData[curfile][np.logical_not(np.isnan(targetData[curfile][i]))]

    for i in range(0,len(nontargetData[curfile])):
        nontarget_epoch_max[curfile].append(np.amax(abs(nontargetData[curfile][i])))

        if nontarget_epoch_max[curfile][i] <= 40:
            good_nontargetepoch[curfile].append(nontargetData[curfile][i])
            nontargetcount += 1
            #nontargetData[curfile][i] = np.nan
        #     nontargetData[curfile][i] = 0 #use np.nanmean to avoid nan's in the future
        # if nontarget_epoch_max[curfile][i] == 0:
        #     nontargetData[curfile].remove(nontargetData[curfile][i],0)


    #set = 0, if x = 0.remove(array, obj)(targetdata, 0)

    #300targetand nontarget epochs per eeg electrode for all trials per one subject

    #print(len(FilterData[1,:])) # two different ways to show either rows or columns length
    #print(FilterData.shape[1])

    return targetData,nontargetData, good_targetepoch, good_nontargetepoch, targetcount, 
nontargetcount, target_epoch, NewData, noisytargetcount  #subtracts the data by the average 
of the two reference EEG leads


Subjects = 6
Session = 4
Trial = 6
TotalTarget = 0
TotalNonTarget = 0
```

```
FzEEG = 31   - 1
CzEEG = 32   - 1
PzEEG = 13   - 1
OzEEG = 16   - 1
count = 1

targetcount = 0
nontargetcount = 0
noisytargetcount = 0

targetData = {}
nontargetData = {}
target_epoch_max = {}
nontarget_epoch_max = {}
Overalltarget = []
Overallnontarget = []

good_targetepoch = {}
good_nontargetepoch = {}

for i in range(0,Session):
   print('youre doing great sweetie')

   for j in range(0,Trial):

      print('Session Number',i+1,'Trial Number',j+1)
      Edata,Eevents,Estimuli,Etarget,Etargets_counted = SST(Subjects,i+1,j, count) #TRIALS START
AT 0- Subject#, Session#, Trial#-1
      #Edata,Eevents,Estimuli,Etarget,Etargets_counted = SST(Subjects,4,j, count) #TRIALS START
AT 0- Subject#, Session#, Trial#-1

      targetData,nontargetData,good_targetepoch,good_nontargetepoch,targetcount,
nontargetcount, target_epoch, NewData, noisytargetcount =
sorting_epoch(Edata,Eevents,Estimuli,Etarget,Etargets_counted,count,targetcount,
nontargetcount, noisytargetcount) #

      count += 1


def avg_epochs(epoch_targetData,epoch_nontargetData,ct): #need to combine all 24 trial
epochs, best if kept seperate per EEG lead to be able to reference.

   curfile = 'data' + str(ct)
   avgtargetepoch[curfile] = []
   avgnontargetepoch[curfile] = []
   #print(len(epoch_targetData[curfile]))
  # for i in range(0,len(epoch_targetData[curfile])):
       #for j in range(0,32):
       #avgtargetepoch[curfile] = np.nanmean(targetData[curfile][:],axis = 0)
   avgtargetepoch[curfile] = np.mean(epoch_targetData[curfile][:],axis = 0) #works but doesnt
ignore NAN numbers and therefore returns NAN for everything
   avgnontargetepoch[curfile] = np.mean(epoch_nontargetData[curfile][:],axis = 0)


   Overalltarget.append(avgtargetepoch[curfile]) # converts from a list
   Overallnontarget.append(avgnontargetepoch[curfile]) #converts from a list
   #avgnontargetepoch[curfile] = np.nanmean(epoch_nontargetData[curfile][:])

   return Overalltarget,Overallnontarget#avgtargetepoch,avgnontargetepoch)


def PermutationSetup(good_targetepoch,good_nontargetepoch,ct):
   #Need to create target data and non target data OUT of dictionarys like i did withOverall
Targets. Then use his code
   curfile = 'data' + str(ct)

   Targetlist.append(good_targetepoch[curfile])
   nonTargetlist.append(good_nontargetepoch[curfile])

   NullData = np.concatenate( Targetlist + nonTargetlist)

   return NullData, Targetlist, nonTargetlist



Targetlist = []
nonTargetlist = []
avgtargetepoch = {}
avgnontargetepoch = {}
NullEEG = []
ct = 1

for i in range(0,Session):
   for j in range(0,Trial):

      Overalltarget,Overallnontarget = avg_epochs(good_targetepoch,good_nontargetepoch,ct)
      NullData, Targetlist, nonTargetlist = PermutationSetup(good_targetepoch,
good_nontargetepoch, ct)
```

```
        print('Averaging Epochs Session',i+1,'Trial',j+1)
        ct += 1

Targetarray = np.concatenate(Targetlist)
nonTargetarray = np.concatenate(nonTargetlist)

TargetFzEEG = Targetarray[:,FzEEG,:]
nonTargetFzEEG = nonTargetarray[:,FzEEG,:]
NullFzEEG = NullData[:,FzEEG,:]

TargetCzEEG = Targetarray[:,CzEEG,:]
nonTargetCzEEG = nonTargetarray[:,CzEEG,:]
NullCzEEG = NullData[:,CzEEG,:]

TargetPzEEG = Targetarray[:,PzEEG,:]
nonTargetPzEEG = nonTargetarray[:,PzEEG,:]
NullPzEEG = NullData[:,PzEEG,:]

TargetOzEEG = Targetarray[:,OzEEG,:]
nonTargetOzEEG = nonTargetarray[:,OzEEG,:]
NullOzEEG = NullData[:,OzEEG,:]


avgEEGtarget = np.arange(0,32)
avgnonEEGtarget = np.arange(0,32)
#for i in range(0,len(avgtarget)):
Totalepochs = Overalltarget + Overallnontarget
avgEEGtarget = np.mean(Overalltarget, axis = 0)
avgnonEEGtarget = np.mean(Overallnontarget, axis = 0)
#print(noisytargetcount,'noisy targets')


def Permutation(NullFzEEG, TargetFzEEG, nonTargetFzEEG, NullCzEEG, TargetCzEEG,
nonTargetCzEEG, NullPzEEG, TargetPzEEG, nonTargetPzEEG ,NullOzEEG, TargetOzEEG,
nonTargetOzEEG):

    Nperms = 1000
    TargetFzEEGavg = np.mean(TargetFzEEG, axis = 0)
    nonTargetFzEEGavg = np.mean(nonTargetFzEEG, axis = 0)
    constantDifFz = (TargetFzEEGavg - nonTargetFzEEGavg)
    actualdifFz = (TargetFzEEGavg - nonTargetFzEEGavg).max()
    maxtimepointFz= np.argmax((TargetFzEEGavg - nonTargetFzEEGavg)) / fs * 1000
    TargetCzEEGavg = np.mean(TargetCzEEG, axis = 0)
    nonTargetCzEEGavg = np.mean(nonTargetCzEEG, axis = 0)
    constantDifCz = (TargetCzEEGavg - nonTargetCzEEGavg)
    actualdifCz = (TargetCzEEGavg - nonTargetCzEEGavg).max()
    maxtimepointCz = np.argmax((TargetCzEEGavg - nonTargetCzEEGavg)) / fs * 1000

    TargetPzEEGavg = np.mean(TargetPzEEG, axis = 0)
    nonTargetPzEEGavg = np.mean(nonTargetPzEEG, axis = 0)
    constantDifPz = (TargetPzEEGavg - nonTargetPzEEGavg)
    actualdifPz = (TargetPzEEGavg - nonTargetPzEEGavg).max()
    maxtimepointPz= np.argmax((TargetPzEEGavg - nonTargetPzEEGavg)) / fs * 1000

    TargetOzEEGavg = np.mean(TargetOzEEG, axis = 0)
    nonTargetOzEEGavg = np.mean(nonTargetOzEEG, axis = 0)
    constantDifOz = (TargetOzEEGavg - nonTargetOzEEGavg)
    actualdifOz = (TargetOzEEGavg - nonTargetOzEEGavg).max()
    maxtimepointOz= np.argmax((TargetOzEEGavg - nonTargetOzEEGavg)) / fs * 1000


#null hypothesis
    Fzpval = 0
    Czpval = 0
    Pzpval = 0
    Ozpval = 0

    maxFzvals = np.zeros(Nperms)  # To store the peak we get under the null
    maxCzvals = np.zeros(Nperms)  # To store the peak we get under the null
    maxPzvals = np.zeros(Nperms)  # To store the peak we get under the null
    maxOzvals = np.zeros(Nperms)  # To store the peak we get under the null

    nsamps = len(NullFzEEG)
    #print('length of Nsamps',nsamps)
    for k in range(Nperms):
        order = np.random.permutation( nsamps)

        FzFperm = NullFzEEG[order[:len(TargetFzEEG)], :].mean(axis=0)
        FzTperm = NullFzEEG[order[len(TargetFzEEG):], :].mean(axis=0)
        CzFperm = NullCzEEG[order[:len(TargetCzEEG)], :].mean(axis=0)
        CzTperm = NullCzEEG[order[len(TargetCzEEG):], :].mean(axis=0)
        PzFperm = NullPzEEG[order[:len(TargetPzEEG)], :].mean(axis=0)
        PzTperm = NullPzEEG[order[len(TargetPzEEG):], :].mean(axis=0)
        OzFperm = NullOzEEG[order[:len(TargetOzEEG)], :].mean(axis=0)
        OzTperm = NullOzEEG[order[len(TargetOzEEG):], :].mean(axis=0)
        difvalsFz.append(FzFperm - FzTperm)
        maxFzvals[k] = (FzFperm - FzTperm).max()
        #need to find the index where this happens and dvide by fs and mult by 1000 to get the
time point in ms

        difvalsCz.append(CzFperm - CzTperm)
        maxCzvals[k] = (CzFperm - CzTperm).max()
        difvalsPz.append(PzFperm - PzTperm)
        maxPzvals[k] = (PzFperm - PzTperm).max()
        difvalsOz.append(OzFperm - OzTperm)
        maxOzvals[k] = (OzFperm - OzTperm).max()

        if maxFzvals[k] > actualdifFz:
            Fzpval += 1

        if maxCzvals[k] > actualdifCz:
            Czpval += 1

        if maxPzvals[k] > actualdifPz:
            Pzpval += 1

        if maxOzvals[k] > actualdifOz:
            Ozpval += 1


    Pzpval = Pzpval/Nperms
    Czpval = Czpval/Nperms
    Fzpval = Fzpval/Nperms
    Ozpval = Ozpval/Nperms


    return Fzpval,maxFzvals, actualdifFz, constantDifFz, difvalsFz, Czpval,maxCzvals, actualdifCz,
constantDifCz, difvalsCz, Pzpval,maxPzvals, actualdifPz, constantDifPz,
difvalsPz,Ozpval,maxOzvals, actualdifOz, constantDifOz, difvalsOz, maxtimepointFz,
maxtimepointCz,maxtimepointPz,maxtimepointOz



# for i in range(0,Session):
#     for j in range(0,Trial):

avgSingleFzEEG= avgEEGtarget[FzEEG,:]#Average Filtered target epochs for a single electrode
avgnonSingleFzEEG= avgnonEEGtarget[FzEEG,:]# Average Filtered non target epochs for a single
electrode
```

```
#goodtargetSingleEEG = Targetlist[:][EEG,:]#Filtered target epochs for a single electrode
#goodnontargetSingleEEG = nonTargetlist[:,EEG,:] #Filtered non target epochs for a single
electrode
maxFzvals = []
actualdifFz = []
constantDifFz = []
difvalsFz = []
maxCzvals = []
actualdifCz = []
constantDifCz = []
difvalsCz = []
maxPzvals = []
actualdifPz = []
constantDifPz = []
difvalsPz = []
maxOzvals = []
actualdifOz = []
constantDifOz = []
difvalsOz = []


Fzpval,maxFzvals, actualdifFz, constantDifFz, difvalsFz, Czpval,maxCzvals, actualdifCz,
constantDifCz, difvalsCz, Pzpval,maxPzvals, actualdifPz, constantDifPz,
difvalsPz,Ozpval,maxOvals, actualdiOz, constantDifOz, difvalsOz,maxtimepointFz,
maxtimepointCz,maxtimepointPz,maxtimepointOz = Permutation(NullFzEEG, TargetFzEEG,
nonTargetFzEEG, NullCzEEG, TargetCzEEG, nonTargetCzEEG, NullPzEEG, TargetPzEEG,
nonTargetPzEEG ,NullOzEEG, TargetOzEEG, nonTargetOzEEG)

print('B2: The Pvalue of channel Fz for subject',Subjects,' is',Fzpval)
print('B3: the Values for Cz, Pz, and Oz are', Czpval, Pzpval, Ozpval)
print('B3: The peak height occurs at ',maxtimepointFz,'ms for channel Fz,',maxtimepointCz,'ms
for Cz,',maxtimepointPz,'ms for Pz,',maxtimepointOz, 'ms for Oz')




def PCA_EEG(Target4EEG,nonTarget4EEG, CombTarget4EEG,pc1, pc2 ):

    electrodibois = [0,1,2,3]

    # Do PCA to get 2 dimensions
    for i in range(0,len(Target4EEG)):
        #print('this first loop is running?')
        #for j in range(0,3):
```

```
        CombTarget4EEG = np.concatenate([Target4EEG[i][ele] for ele in electrodibois])
        CombTargets.append(CombTarget4EEG)

    pc1 = PCA(n_components=2)
    pc1.fit(CombTargets)
    Target_pc = pc1.transform(CombTargets)

    for i in range(0,len(nonTarget4EEG)):
        CombnonTarget4EEG = np.concatenate([nonTarget4EEG[i][ele] for ele in electrodibois])
        CombnonTargets.append(CombnonTarget4EEG)

    #pc2 = PCA(n_components=2)
    #pc2.fit(CombnonTargets)
    nonTarget_pc = pc1.transform(CombnonTargets)

    return Target_pc, nonTarget_pc, CombTargets, CombnonTargets

Target_pc = []
nonTarget_pc = []
CombTargets = []
CombnonTargets = []
CombTarget4EEG =  []
CombnonTarget4EEG = []

pc1 = []
pc2 = []

#need time window of .25-.45 ish


LowTime = int(.25*fs)
HighTime = int(.5*fs)
TimeWindowArray = np.arange(LowTime,HighTime)
Target4EEG = Targetarray[:,(FzEEG,CzEEG,PzEEG,OzEEG),LowTime:HighTime]
nonTarget4EEG = nonTargetarray[:,(FzEEG,CzEEG,PzEEG,OzEEG),LowTime:HighTime]

Target_pc, nonTarget_pc, CombTargets, CombnonTargets = PCA_EEG(Target4EEG,
nonTarget4EEG, CombTarget4EEG, pc1, pc2 )


def avgEpochs(CombTargets, CombnonTargets, k10,k25,k50,
k10targetepoch,k10nontargetepoch,k25targetepoch,k25nontargetepoch,k50targetepoch,k50no
ntargetepoch):
```

```
#instead of 280 total epochs, we would simply have 28 avg epochs to perform PCA on.
tk10 = round(len(CombTargets) / k10)
tk25 = round(len(CombTargets) / k25)
tk50 = round(len(CombTargets) / k50)

nk10 = round(len(CombnonTargets) / k10)
nk25 = round(len(CombnonTargets) / k25)
nk50 = round(len(CombnonTargets) / k50)

#print(tk10,tk25,tk50,nk10,nk25,nk50)

k10runningepoch = []
k25runningepoch = []
k50runningepoch = []

for i in range(0,len(CombTargets)):
   if i%k10 == 0 and i > 0:
      k10targetepoch.append(np.mean(k10runningepoch,axis = 0))
      k10runningepoch = []
      k10runningepoch.append(CombTargets[i])

   else:
      k10runningepoch.append(CombTargets[i])

   if i%k25 == 0 and i > 0:
      k25targetepoch.append(np.mean(k25runningepoch,axis = 0))
      k25runningepoch = []
      k25runningepoch.append(CombTargets[i])

   else:
      k25runningepoch.append(CombTargets[i])

   if i%k50 == 0 and i > 0:
      k50targetepoch.append(np.mean(k50runningepoch,axis = 0))
      k50runningepoch = []
      k50runningepoch.append(CombTargets[i])

   else:
      k50runningepoch.append(CombTargets[i])

k10runningepoch = []
k25runningepoch = []
k50runningepoch = []
```

```
    for i in range(0,len(CombnonTargets)):
        if i%k10 == 0 and i > 0:
            k10nontargetepoch.append(np.mean(k10runningepoch,axis = 0))
            k10runningepoch = []
            k10runningepoch.append(CombnonTargets[i])

        else:
            k10runningepoch.append(CombnonTargets[i])

        if i%k25 == 0 and i > 0:
            k25nontargetepoch.append(np.mean(k25runningepoch,axis = 0))
            k25runningepoch = []
            k25runningepoch.append(CombnonTargets[i])

        else:
            k25runningepoch.append(CombnonTargets[i])

        if i%k50 == 0 and i > 0:
            k50nontargetepoch.append(np.mean(k50runningepoch,axis = 0))
            k50runningepoch = []
            k50runningepoch.append(CombnonTargets[i])

        else:
            k50runningepoch.append(CombnonTargets[i])



    return (
k10targetepoch,k10nontargetepoch,k25targetepoch,k25nontargetepoch,k50targetepoch,k50no
ntargetepoch)

k10targetepoch = []
k10nontargetepoch = []
k25targetepoch = []
k25nontargetepoch = []
k50targetepoch = []
k50nontargetepoch = []

k10 = 10
k25 = 25
k50 = 50

k10targetepoch,k10nontargetepoch,k25targetepoch,k25nontargetepoch,k50targetepoch,k50no
ntargetepoch = avgEpochs(CombTargets, CombnonTargets, k10,k25,k50,
k10targetepoch,k10nontargetepoch,k25targetepoch,k25nontargetepoch,k50targetepoch,k50no
ntargetepoch)


def
PCA_K(k10targetepoch,k10nontargetepoch,k25targetepoch,k25nontargetepoch,k50targetepoch
,k50nontargetepoch,  k10tpc,k25tpc, k50tpc, k10npc,k25npc,k50npc):

   k10tpc = PCA(n_components=2)
   k10tpc.fit(k10targetepoch)
   Target_pck10 = k10tpc.transform(k10targetepoch)

   k25tpc = PCA(n_components=2)
   k25tpc.fit(k25targetepoch)
   Target_pck25 = k25tpc.transform(k25targetepoch)

   k50tpc = PCA(n_components=2)
   k50tpc.fit(k50targetepoch)
   Target_pck50 = k50tpc.transform(k50targetepoch)

   #k10npc = PCA(n_components=2)
   #k10npc.fit(k10nontargetepoch)
   nonTarget_pck10 = k10tpc.transform(k10nontargetepoch)

   #k25npc = PCA(n_components=2)
   #k25npc.fit(k25nontargetepoch)
   nonTarget_pck25 = k25tpc.transform(k25nontargetepoch)

   #k50npc = PCA(n_components=2)
   #k50npc.fit(k50nontargetepoch)
   nonTarget_pck50 = k50tpc.transform(k50nontargetepoch)


   return (Target_pck10, nonTarget_pck10, Target_pck25, nonTarget_pck25,Target_pck50,
nonTarget_pck50)

Target_pck10 = []
nonTarget_pck10 = []
Target_pck25 = []
nonTarget_pck25 = []
Target_pck50 = []
nonTarget_pck50 = []

k10tpc = []
```

```
k25tpc = []
k50tpc = []
k10npc = []
k25npc = []
k50npc = []

Target_pck10, nonTarget_pck10, Target_pck25, nonTarget_pck25,Target_pck50,
nonTarget_pck50 =
PCA_K(k10targetepoch,k10nontargetepoch,k25targetepoch,k25nontargetepoch,k50targetepoch
,k50nontargetepoch, k10tpc,k25tpc, k50tpc, k10npc,k25npc,k50npc)


def ROC(xbound,  Target_pck10, nonTarget_pck10,Sens, Spec):
  #Thits = 0
  for i in range(0,8):
    xbound = -130 + i*18
    #print(xbound)
    Tmisses = 0
    Nhits = 0

    for i in range(0,len(Target_pck10)):
      # if Target_pck10[i][0] > xbound:
      #    Thits =+1
      if Target_pck10[i][0] < xbound:
        Tmisses += 1
      if nonTarget_pck10[i][0] > xbound:
        Nhits += 1
      # if nonTarget_pck10[i][0] > xbound:
      #    Nmisses =+ 1
    # print(Tmisses)
    # print(Nhits)

    Sens.append((len(Target_pck10) - Tmisses) / (len(Target_pck10)))
    Spec.append((len(nonTarget_pck10) - Nhits) / (len(nonTarget_pck10))) #x axis

  return(Sens, Spec)

xbound = 0
Sens = []
Spec = []

Sens, Spec = ROC(xbound, Target_pck10, nonTarget_pck10,Sens, Spec)
```

```
##PLOTS
#time = np.arange(0,len(NewData[1])/fs,1/fs)
t = np.arange(0,1,1/fs)

#ROC Curve
plt.figure()
plt.plot(Spec,Sens, 'o-')
plt.xlabel('Specificity')
plt.ylabel('Sensitivity')
plt.title('ROC curve of K10 Target Detection')




#1st epoch
# plt.figure()
# for i in range(0,data.shape[0]):
#     plt.plot(time[820:820+2048],data[0][820:+2048])
# plt.xlabel('Time (sec)')
# plt.ylabel(u'\u03bcV')
# plt.title('EEG 1st Epoch',fontweight='bold')

# #FilteredData
# plt.figure()
# for i in range(0,NewData.shape[0]):
#     plt.plot(time[820:820+2048],FilterData[i][820:820+2048]) #Plot the signal
# plt.xlabel('Time (sec)')
# plt.ylabel(u'\u03bcV')
# plt.title('EEG Filtered Data',fontweight='bold')

#TARGET AND NONTARGET DATA partA
plt.figure()
plt.plot(t,avgEEGtarget[FzEEG], label= "Target (" + str(targetcount) + " trials)") #Plot the signal
plt.plot(t,avgnonEEGtarget[FzEEG], label= "Non-Target (" + str(nontargetcount) + " trials)") #Plot
the signal) #Plot the signal
plt.xlabel('Time (sec)')
plt.ylabel('Average Response 'u'\u03bcV')
plt.title('PartA2: Subject '+str(Subjects)+', EEG31 Target vs Nontargets',fontweight = 'bold')
plt.legend()

#Difference of Target and nonTarget P300
plt.figure()
#for i in range(0,32):
plt.plot(t,(avgEEGtarget[30] - avgnonEEGtarget[30]), label = "EEG31")#+str(i+1)) #Plot the signal
plt.xlabel('Time (sec)')
plt.ylabel('Average Response 'u'\u03bcV')
plt.title('PartA3: Subject '+str(Subjects)+', P300 Response',fontweight = 'bold')
plt.legend()


#100 NULLPERMUTATIONS part b2
plt.figure()
for i in range(0,1000):
    plt.plot(t,difvalsFz[i], 'k') #Plot the signal) #Plot the signal
plt.plot(t,constantDiffFz, 'r',linewidth = 10, label = "Actual") #Plot the signal
plt.xlabel('Time (sec)')
plt.ylabel('Average Response 'u'\u03bcV')
plt.title('PartB2: Subject '+str(Subjects)+', EEG31 Null Examples vs Actual response',fontweight = 
'bold')
plt.legend()

# #Full Value PCA
# plt.figure()
# for i in range(0,len(nonTarget4EEG)):
#     plt.plot(nonTarget_pc[i,0],nonTarget_pc[i,1],'b.')
# for i in range(0,len(Target4EEG)):
#     plt.plot(Target_pc[i,0],Target_pc[i,1],'r.')
# plt.plot([],[],'b.',label = 'NonTarget')
# plt.plot([],[],'r.',label = 'Target')
# plt.xlabel('PC1')
# plt.ylabel('PC2')
# plt.legend()
# plt.title('FullPCA')

# #k10 PCA
# plt.figure()
# for i in range(0,len(nonTarget_pck10)):
#     plt.plot(nonTarget_pck10[i,0],nonTarget_pck10[i,1],'b.')
# for i in range(0,len(Target_pck10)):
#     plt.plot(Target_pck10[i,0],Target_pck10[i,1],'r.')
# plt.plot([],[],'b.',label = 'NonTarget')
# plt.plot([],[],'r.',label = 'Target')
# plt.xlabel('PC1')
# plt.ylabel('PC2')
# plt.legend()
# plt.title('k10 PCA')

# #k25 PCA
# plt.figure()
# for i in range(0,len(nonTarget_pck25)):
#     plt.plot(nonTarget_pck25[i,0],nonTarget_pck25[i,1],'b.')
# for i in range(0,len(Target_pck25)):
#     plt.plot(Target_pck25[i,0],Target_pck25[i,1],'r.')
# plt.plot([],[],'b.',label = 'NonTarget')
# plt.plot([],[],'r.',label = 'Target')
# plt.xlabel('PC1')
# plt.ylabel('PC2')
# plt.legend()
# plt.title('k25 PCA')

# #k50 PCA
# plt.figure()
# for i in range(0,len(nonTarget_pck50)):
#     plt.plot(nonTarget_pck50[i,0],nonTarget_pck50[i,1],'b.')
# for i in range(0,len(Target_pck50)):
#     plt.plot(Target_pck50[i,0],Target_pck50[i,1],'r.')
# plt.plot([],[],'b.',label = 'NonTarget')
# plt.plot([],[],'r.',label = 'Target')
# plt.xlabel('PC1')
# plt.ylabel('PC2')
# plt.legend()
# plt.title('k50 PCA')

# plt[2].plot(t,constantDifCz, 'r',linewidth = 10, label = "Actual") #Plot the signal
# for i in range(0,100):
#     plt[2].plot(t,difvalsCz[i], 'k') #Plot the signal) #Plot the signal
# #plt[2].xlabel('Time (sec)')
# #plt[2].ylabel('Average Response'u'\u03bcV')
# plt[2].title('PartB1: Subject '+str(Subjects)+', EEG32 Null Examples vs Actual
response',fontweight = 'bold')
# plt[2].legend()
# plt[3].plot(t,constantDifPz, 'r',linewidth = 10, label = "Actual") #Plot the signal
# for i in range(0,100):
#     plt[3].plot(t,difvalsPz[i], 'k') #Plot the signal) #Plot the signal
# #plt[3].xlabel('Time (sec)')
# #plt[3].ylabel('Average Response'u'\u03bcV')
# plt[3].title('PartB1: Subject '+str(Subjects)+', EEG13 Null Examples vs Actual
response',fontweight = 'bold')
# plt[3].legend()
# plt[4].plot(t,constantDifOz, 'r',linewidth = 10, label = "Actual") #Plot the signal
# for i in range(0,100):
#     plt[4].plot(t,difvalsOz[i], 'k') #Plot the signal) #Plot the signal
# #plt[4].xlabel('Time (sec)')
# #plt[4].ylabel('Average Response'u'\u03bcV')
# plt[4].tile('PartB1: Subject '+str(Subjects)+', EEG16 Null Examples vs Actual
response',fontweight = 'bold')
# plt[4].legend()
#print(nontargetcount,'nontarget',targetcount,'target')
# plt.figure()
# for i in range(0,NewData.shape[0]):
#     plt.plot(t,epochs[i]) #Plot the signal
# plt.xlabel('Time (sec)')
# plt.ylabel(u'\u03bcV')
# plt.title('EEG All Epochs',fontweight='bold')

# plt.figure()
# for i in range(0,NewData.shape[0]):
#     plt.plot(t,targetData[i]) #Plot the signal
# plt.xlabel('Time (sec)')
# plt.ylabel(u'\u03bcV')
# plt.title('Single Trial EEG Epochs Target',fontweight='bold')

# plt.figure()
# for i in range(0,NewData.shape[0]):
#     plt.plot(t,nontargetData[i]) #Plot the signal
# plt.xlabel('Time (sec)')
# plt.ylabel(u'\u03bcV')
# plt.title('Single Trial EEG Non-Target Epochs',fontweight='bold')


# #Filter freq domain
# # [w, hf_filter] = signal.freqz(h_filter,a=1,fs=fs, worN=h_filter.size)
# # plt.figure()
# # plt.plot(w.squeeze(),abs(hf_filter))
# # plt.xlim([0,13])
# # plt.xlabel('Frequency (Hz)',fontsize=12,)
# # plt.ylabel('Magnitude',fontsize=12)
# # plt.title('H(f)',fontsize=15,fontweight='bold')
# # plt.show()

# #for i in range(34): #plots all data lines
# #    plt.plot(data[i,:], '.')

# #plotting Pvalues of all 24 trials
# z =
[0.004,0.01,.034,.019,.003,.022,0,0,0,.541,.012,.092,.017,.28,.001,.008,.002,0,0.001,0,0,0,0.0056
,.002] #subject 6
# y =
[0.986,0.048,0.755,0.678,0.07,0.071,0.529,0.79,0.518,0.794,0.638,0.325,0.157,0.094,0.205,0.48
5,0.462,0.006,0.485,0.396,0.42,0.001,0.035,0.482]
# x = np.arange(1,25)
# plt.figure()
# plt.plot(x,y)
# plt.title('Pvalues for all 24 trials of Subject 4')
# plt.xlabel('trial number')
# plt.ylabel('p300 Pvalue ')
# avgpval4 = np.mean(y)
# print(avgpval4)
# avgpval6 = np.mean(z)
# print('sub 6',avgpval6)
```