# פרויקט סיכום

תמיר שילוני 313328924

עופר אבין 204517924

**הקדמה**

ממשקי מוח מחשב –(BCI) brain computer interface עושים שימוש באותות על מנת ליצור תקשורת בין מוח או מצבו המוחי של בן אדם לבין מכונה חיצונית [1]. המכונה מפרשת את האותות לרוב על ידי שיטות של למידת מכונה (ML) machine learning ועושה בהם שימוש כמו ביצוע פעולות שונות [1, 2]. BCI מספק דרך חלופית לבני אדם לתקשר ללא צורך במערכת העצבים הפריפריאלית ושימוש בשרירים [2].

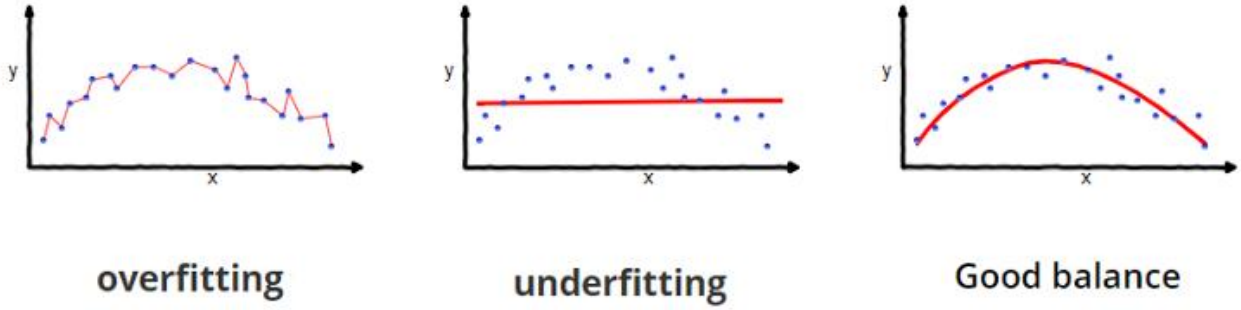
(MI) Motor Imagery מוגדר כתהליך קוגנטיבי בו אדם מדמיין את התנועה של גופו מבלי להניע אותו באמת [3] . שימוש ב BCI המבוסס על MI מוצע כפתרון לאנשים בעלי מוגבלויות מוטוריות בין אם פיזיות כגון קטיעות ובין אם כתוצאה מבעיות נוירונליות. מערכות כאלו יכולות לעזור בפעולות כמו הזזת גפיים תותבות, שליטה על כסא גלגלים, שימוש במציאות מדומה ועוד דרכים בהן ניתן לסייע לאנשים עם מגבלויות תנועה[1,3,4].

אלקטרואנצפלוגרם (EEG)Electroencephalogram הינה מערכת לרישום פעילות חשמלית מהמוח על ידי אלקטרודות המוצמדות לקרקפת של הנבדק. מדידה כזאת אינה פולשנית בניגוד לאלקטרודות המוכנסות לתוך הגולגולת, בנוסף מספקת את המידע בזמן קצר מאוד. יתרונות אלו הופכות את מערכות ה BCI המבוססות EEG למעשיות ולכן נפוצות מאוד[3,4]. לרוב כאשר הניסוי עוסק ב MI האלקטרודות המעניינות יהיו האלקטרודות הנמצאות מעל האונה הפריאטלית שם נמצא הקורטקס המוטורי הראשוני (primary motor cortex) [4].

שימוש ב EEG מערב חילוץ מאפיינים (features) וסיווג [3]. המאפיינים המחולצים מן האות יכולים להיות עוצמת התדרים המרכיבים את הגל, העוצמות היחסיות של תדרים שונים ביחס לעוצמה הכללית, המתח המקסימלי של הגל או הממוצע שלו ולמעשה כמעט כל פעולה העולה על הדעת. כמובן חלק מהמאפיינים יועילו יותר חלק יועילו פחות או אפילו יפגעו בפעולת הסיווג.

מטרת הסיווג היא תרגום מאפייני האות המחולצים בשלב חילוץ המאפיינים לפקודות הממשות את כוונת המשתמש[3]. המסווג (classifier) למעשה ממיר את המאפיינים המבדילים בין האותות השונים המגיעים מן המוח למידע על כוונת המשתמש כמו הזזת יד ימין או רגל שמאל וכדומה. בבסיס המסווג נמצא אלגוריתם המסוגל ללמוד מתוך סט נתונים. בתחילה המסווג מקבל את נתוני האימון הנקראים סט האימון (training set) באמצעותם הוא בונה מודל חיזוי. לאחר שלב האימון, תפקיד המסווג הוא לתייג את הכוונה של המשתמש כאשר הוא מקבל נתונים חדשים. אלגוריתמים הנמצאים בשימוש במערכות BCI שונות יכולים להיות רשתות נוירונים, הפרדה ליניארית (LDA) linear discriminate analysis, support vector machines ואלגוריתמים נוספים מתחום ML [3].

שתי בעיות הקשורות זו בזו העולות כאשר משתמשים באלגוריתמים של ML הן תת התאמה (underfitting) והתאמת יתר (overfitting). בעית תת התאמה נוצרת כאשר המודל מנסה לנבא על סמך מעט מדי מאפיינים, לדוגמה אם נרצה לאמן מודל אשר מנבא את ממוצע התואר על סמך ציוני הבגרות של הסטודנט בלבד. מודל כזה כנראה לא יראה ביצועים טובים כל כך. בעיית ה overfit קוראת כאשר המודל מנסה לנבא על סמך יותר מדי מאפיינים [5]. נמשיך באותה דוגמה, אם הנתונים שלנו כוללים מאפיינים על סטודנטים כמו, ציוני בגרות, פסיכומטרי, שעות למידה ביום, ומלבד זאת גם מין, גובה וצבע שיער, ונניח בנוסף שרבים מבעלי הממוצע הגבוה בסט האימון הם בעלי שיער חום, המודל שיתקבל עלול לקחת בחשבון את צבע השיער כמנבא לממוצע אף על פי שכנראה אין כלל קשר בין השניים. למודל כזה יהיה קשה להכליל, כלומר להראות ביצועים טובים כאשר יופעל על נתונים חדשים וזאת מכיוון שהוא למעשה מותאם מידי לסט האימון.




בתמונה, הקו האדום הוא המודל המותאם לנקודות שהן סט האימון. ניתן ליראות כי במצב של overfitting המודל מותאם לנתוני האימון בצורה מושלמת. במצב של underfitting המודל לא יכול להתעכל בצורה מספקת על מנת להתאים לנתונים. המצב הרצוי הוא איזון בין שני המקרים.

על מנת להעריך את ביצועי המודל ולנסות להמנע ממצב של overfitting אנו מחלקים את נתוני האימון שלנו לשניים, סט עליו נתאמן – ה training set וסט נתונים נוסף זר לנתוני האימון אשר ידמה את העולם האמיתי לו נקרא סט האימות (validation set) לאחר האימון נבצע בדיקה על ה validation set ונקבל הערכה לשגיאת ההכללה של המודל כלומר השגיאה של המודל על נתונים שלא ראה לפניכן. החלוקה ל training set ו validation set נעשת לרוב כך ש $70\% - 80\%$ מהנתונים ישמשו לאימון והשאר לאימות[6]. מכיון שבדרך זו אנו "מבזבזים" נתונים עליהם יכולנו להתאמן קיימת שיטה נוספת המשתמשת בכל הנתונים בשלב האימון - k fold cross validation . בשיטה זו אנו מחלקים את הנתונים ל K קבוצות זרות. כעת כל קבוצה $i$ כאשר $1 \geq i \geq k$ תשמש פעם אחת כ validation set ובשאר הפעמים תהיה חלק מה training set. בצורה כזאת למעשה מתאמנים על כל הנתונים ואפשר לקבל הערכה טובה יותר של טיב המודל.

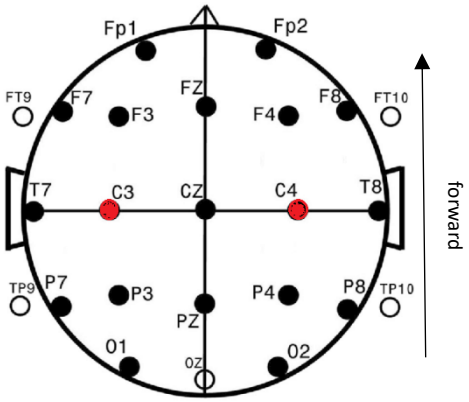
בעבודה זאת קיבלנו נתוני EEG של 128 ניסויים (trails) משתי אלקטרודות c3 ו c4 הנמצאות מעל האיזור המוטורי באונה הפריאטלית. הנבדק ממנו מגיעים הנתונים קיבל בתחילת כל ניסוי הוראה לדמיין שהוא מזיז את אחת הידיים, ימין או שמאל. על ידי חילוץ מאפיינים מתוך אות ה EEG הנתון ואימון מסווג המשתמש באלגוריתם LDA היה עלינו לחזות את הפעולה האמיתית שאותה דמיין הנבדק כלומר ימין או שמאל. את בדיקת המודל ביצענו על סט נתונים אחר המיועד לבחינת המודל ונקרא סט המבחן (test set) הנמצא במודל.

מלבד חילוץ המאפיינים ואימון המודל על מנת להמנע מ overfitting היה עלינו לבחור רק חלק מהמאפיינים אשר חולצו מתוך נתוני ה EEG, על מנת לעשות זאת השוונו בין שני כלים אשר שמשו אותנו לבחירת המאפיינים. הכלי הראשון לבחירת המאפיינים הוא פונקציה של מטלב fsnca העושה שימוש באלגוריתם neighborhood component analysis (NCA) והכלי השני אשר אנחנו כתבנו המבצע מבחן קולמוגורוב – סמירנוב לשני מדגמים (kolmogorov–Smirnov test(KST. בנוסף את המודל אימנו באמצעות k fold cross validation אשר איפשר לנו הערכה של המודל לפני בדיקתו על סט המבחן.

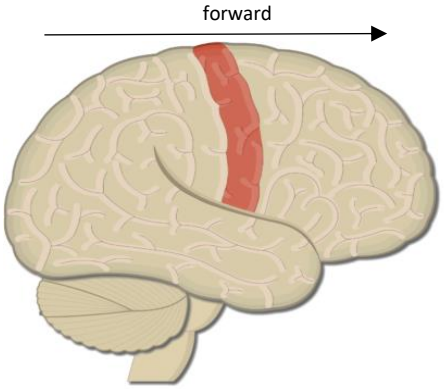
**שיטה**

<u>שלב הניסוי</u>

הנתונים אותם קיבלנו הם נתוני EEG המגיעים משתי אלקטרודות C3 ו C4. ההתמקדות בשתי אלקטרודות אלו נעשית מכיון שהן נמצאות במרכז הקרקפת מעל לקורטקס המוטורי הראשוני.




סידור אלקטרודות EEG על הראש במבט מלמעלה כאשר c3 ו c4 צבועות באדום




מסומן באדום - הקורטקס המוטורי הראשוני

בסך הכל קיבלנו 128 שלבי ניסוי (trials) עבור כל אחד מהם ישנו תיוג האם הנבדק התבקש לדמין שהוא מניע את יד ימין או את יד שמאל ועוד 32 שלבים נוספים ללא תיוג אשר ישמשו בשלב בדיקת המודל עבור ה test set.

<u>שלב הניתוח</u>

על מנת להבין איך הנתונים שלנו ניראים ואילו מאפיינים מעניינים נוכל להוציא מתוך הנתונים ביצענו החזייה (Visualization) של הנתונים במספר צורות. נציג את הנתונים תחת שני התנאים, ימין ושמאל ובאמצעות התבוננות עם העין נחפש שינויים אשר יעזרו לנו לחלץ מאפיינים שבתורם יעזרו לבצע הפרדה בניהם.

<u>חילוץ פיצ'רים:</u>

על מנת להחליט איזה מאפיינים כדאי לחלץ מאות ה EEG נבצע מספר החזיות של המידע כאשר נבחין בהבדל בין הנתונים המתוייגים כשמאל לבין הנתונים המתוייגים כימין. הגרפים בהם נשתמש יהיו 20 גרפים אקראיים עבור כל קטגוריה בהם נציג את המידע משתי האלקטרודות, Power spectrum באמצעות פונקציית Pwelch ו spectograms של כל שילוב בין האלקטרודות לבין ימין ושמאל. בנוסף גם spectogam של ההפרש בין ימין לשמאל עבור שתי האלקטרודות. מלבד מאפיינים בהם ניתן לחזות עם העיין בגרפים נוסיף מאפיינים הנמצאים בשימוש רחב בתחום.

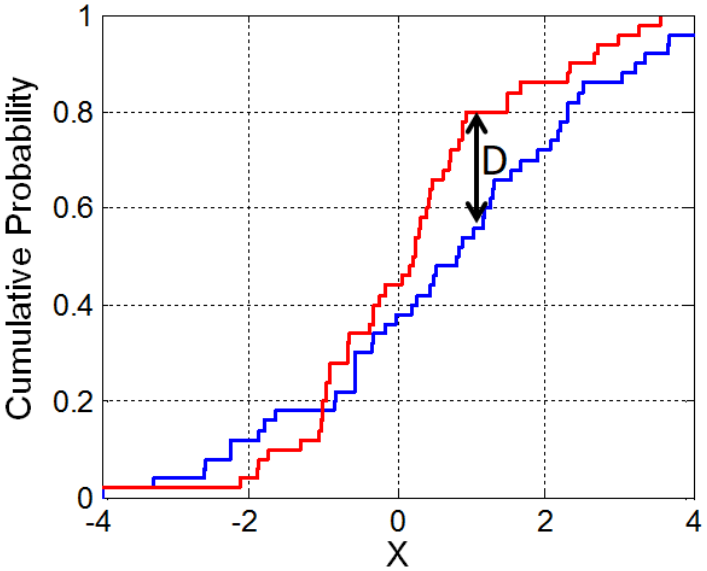
בחירת פיצ'רים:

על מנת לסנן את המאפיינים אשר יועילו ביותר בשלב הסיווג נשתמש בשני כלים NCA ו KST הפונקציות פולטות את המשקולות אשר אותן הן חישבו עבור כל מאפיין. משקולת גבוהה עבור NCA משמעה מאפיין עם פוטנציאל הפרדה גבוה יותר בין שתי הקטגוריות המנובאות ימין ושמאל, ואילו עבור KST המצב הפוך, משקולת נמוכה תנבא טוב יותר.

neighborhood component analysis(NCA)

פונקציית מטלאב המשתמשת באלגוריתם זה על מנת לבחור מאפיינים היא fsnca. אלגוריתם NCA מנסה לנבא כל תצפית בנתונים על ידי התצפיות הקרובות אליה ביותר. עבור כל תצפית $i$, מקבלות שאר התצפיות הסתברויות שהן תוצאה של פונקציה התלויה במרחק מתצפית $i$ . הסתברויות אילו למעשה מהוות משקל לפיו ינבא האלגוריתם את התוצאה עבור התצפית $i$ . בשלב הבא משתמש האלגוריתם בפונקצית שגיאה העוזרת לו לעדכן את ההסתברויות עבור התצפיות האחרות ובמקרה הצורך לבחור תצפית אחרת אשר תעזור לנבא את תצפית $i$.

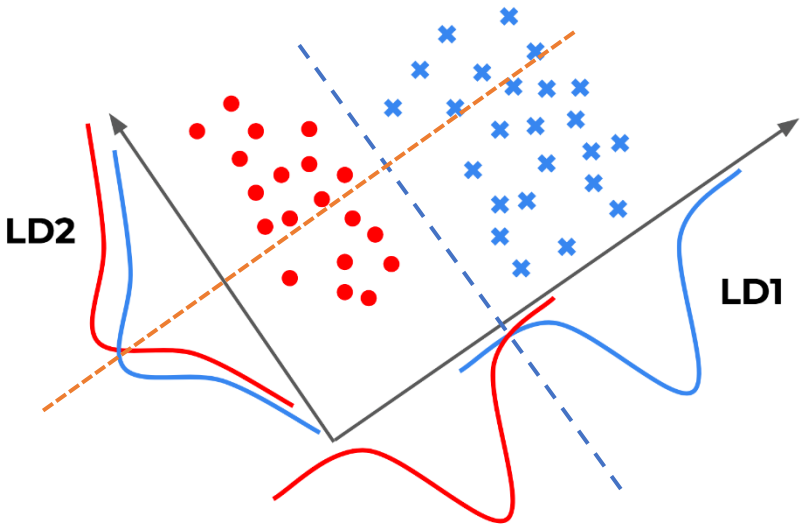
_מבחן קולמוגורוב – סמירנוב לשני מדגמים kolmogorov–Smirnov test(KST)




השיטה השנייה אותה בדקנו לבחירת המאפיינים היא מבחן קולמוגורוב – סמירנוב לשני מדגמים. מבחן קולמוגורוב – סמירנוב הוא מבחן הבודק האם שני מדגמים מגיעים מאותה התפלגות. הנחת האפס של המבחן היא כי המדגמים אכן מגיעים מאותה התפלגות והוא עובד על פונקציית הצפיפות המצטברת של התפלגויות המדגמים. באמצעות הסטטיסטי D (ראה תמונה) בודקים את המרחק בין ההתפלגויות ומחשבים ערך $p$ המציין את ההסתברות לקבל את המרחק תחת הנחת האפס. ערך $p$ קטן יותר – הסתברות נמוכה יותר לקבל את המרחק D תחת הנחת האפס וישנו סיכוי גבוהה יותר ששתי המדגמים מגיעים מהתפלגויות שונות. אלגוריתם המשתמש בקולמוגורוב – סמירנוב נוסה בעבר על משימת בחירת מאפיינים והראה תוצאות טובות [7] לכן בעבודה זאת כתבנו פונקציה הבודקת האם מאפיין אותו חילצנו משרה שתי התפלגויות שונות על הנתונים המתוייגים כימין ועל הנתונים המתויגים כשמאל באמצעות אותו מבחן. ככול שההתפלגויות שונות אחת מהשניה ישנה תיקווה כי המסווג יוכל להשתמש באותו מאפיין ולסווג את הנתונים בצורה טובה יותר. לכן עבור כל מאפיין נעשה מבחן קולמוגורוב – סמירנוב לשני מדגמים. עבור כל מאפיין ערך ה $p$ יהיה המשקל אותו ניתן למאפיין, משקל קטן יותר עבור מאפיין יתן עדיפות לאותו מאפיין על פני מאפיינים עם משקל גבוה יותר. האלגוריתם אותו כתבנו הוא אלגוריתם נאיבי, כלומר הוא בודק כל מאפיין בפני עצמו מבלי להתייחס לשאר המאפיינים שכבר נבחרו.

## סיווג באמצעות אלגוריתם LDA

זהו אלגוריתם העושה סיווג בצורה ליניארית של הנתונים. הלאגוריתם מחשב את הצירים אשר התלת התצפיות עליהם תמקסם את היחס בין המרחק בין הממוצעים של הקטגוריות אותן הוא צריך לנבא לשונות הפנימית בתוך הקטגוריות. במרחב כזה ישנה אפשרות להעביר מישור מפריד בין התצפיות השיכות לקטגוריות שונות.




בתמונה מופיעים שני צירים אשר חושבו על ידי אלגוריתם LDA. ניתן ליראות כי הפרש הממוצעים (נקודת הקיצון של ההתפלגויות) גדול יותר כאשר מתילים את הנתונים על LD1 מאשר על LD2 ובנוסף השונות הנוצרת מהתלת הנתונים (רוחב ההתפלגויות) קטנה יותר בהתלה על LD1. כתוצאה מכך המישור המפריד המחושב באמצעות LD1 (מקוקו בכחול) מפריד בצורה טובה יותר מאשר המשור המפריד המחושב באמצעות LD2 (מקוקו בכתום).

## אימון המודל והערכת המודל באמצעות cross validation:

מספר הקבוצות אליהן בחרנו לחלק את התצפיות, trials הוא 5, זאת מכיון ש $\frac{128}{5} \approx 26$ וכך כל סט אימון יהווה כ $80\%$ מן הנתונים [6]. על ידי מעבר על מספרי הקבוצות נבחר סט האימות ושאר הקבוצות מוקצות לסט האימון עליהם מתאמן המסווג. לאחר האימון המסווג פולט את הניבואים שלו על סט האימות אותן נשווה עם התגיות האמיתיות. נחשב את את ממוצע הדיוק – accuracy עבור כל אחד מחמשת השלבים ונחשב את הממוצע וסטיית התקן של ה accuracy אשר יתנו לנו הערכה לטיב המודל. המסווג פולט גם את גודל הטעות על נתוני האימון אשר ניתנים גם הם להמרה לאחוזי accuracy. הפרש גדול בין הדיוק על סט האימון לדיוק על סט האימות יצביע על overfit של המודל. בנוסף למדד ה accuracy נבדוק גם את ה confusion matrix (CM).

| | Predicted 0 | Predicted 1 |
|---|---|---|
| Actual 0 | TN | FP |
| Actual 1 | FN | TP |

TN – true negative, FT – false positive, FN – false negative, TP – true positive

CM מאפשר לנו לבחון את המודל בצורה מדויקת יותר. מלבד אחוזי דיוק היא מאפשרת לנו לבדוק את כמות ה false positive ו false negative (במקרים בהם הסיווג הוא בין true ל false) שהם התצפיות המתויגות כfalse שסווגו כ true והתצפיות המתויגות כ true אך סווגו כ false בהתאמה. במקרה שלנו אלה יהיו ערכי ימין שסווגו כשמאל וערכי השמאל שסווגו כימין. אצלינו שמאל ישמש כtrue וימין ישמש כ false. מדדים כאילו יכולים לתת לנו הערכה איפה המודל מתקשה יותר. האם רוב הטעויות שלו הן שמאל אשר מתוייגים כימין או להפך. זהו מדד חשוב אם קיימת העדפה לדוגמה למציאת כל הtrials המתויגיום כשמאל אבל אנו מוכנים להתפשר על מציאת תגיות הימין. אפשר לחשוב על העדפה כזאת כאשר אנו רוצים ליצר יד תותבת והתאמתה לאנשים ימנים או שמאליים.

על ידי ה CM ניתן לחשב מדד נוסף לאיכות המודל הנקרא f1 score. זהו מדד מדוייק הרבה יותר מ accuracy [8]. מדד ה f1 מחושב על ידי שני ערכים precision ו recall : $recall = \frac{TP}{TP+FN}$ $precision = \frac{TP}{TP+FP}$ כלומר אצלינו אילו יהיו היחסים בין סיווג שמאל נכון מתוך כל כל הtrials המתויגים כשמאל והיחס בין סיווג שמאל אמיתי מתוך כל מה שסווג כשמאל בהתאמה. הנוסחה ל f1 היא $f1\ score = 2 * \frac{precision*recall}{precision+recall}$ . מכיון שהמדד הזה לוקח בחשבון את ה precision וה recall הוא מאפשר לנו להעריך יותר טוב את המודל מאשר accuracy.

על מנת להבין כמה מאפיינים עלינו להכניס למודל על מנת לקבל את התוצאות הטובות ביותר נעשה שימוש גם ב מדד error אותו נגדיר כ $error = 100 - accuracy$ וגם במדד $f1$ עבור כל כמות מאפיינים. המודל אשר יתן את המדדים הטובים ביותר אותו נבחר על מנת לבדוק את ה *test set*.

**קוד**

פרמטרים:

נטען את הנתונים לאימון 'motor_imagery_train_data.mat' ונחלץ מהם את הפרמטרים הרלוונטים אלינו. נחלץ את תדר הדגימה ואת אורך הניסוי ובעזרתם נבנה את וקטור הזמנים כל שכל נקודה בזמן הניסוי הינה דגימה שונה. לאחר מכן נגדיר את זמן תחילת הדימיון של הנבדק(2.25) ובכך ניצור את וקטור זמני הדימון miPeriod בשביל התמקדות בו בזמן הויזואליזציה וחישוב המאפיינים. בנוסף נחלץ את מספר הניסויים ונגדיר את וקטור התדרים הרלוונטים עבורנו בניסוי- f = 0.5:0.1:40. נגדיר את גודל החלון וגודל החפיפה הרצויה ובעזרתם נחשב את מספר החלונות שנקבל כאשר נחשב את עוצמת התדרים-numOfWindows.
את מספר האלקטרודות בהן נשתמש נגדיר בcell array בשם chans ובנוסף את שמות האלקטרודות נגדיר במערך chansName כאשר המספר המייצג את האלקטרודה ב chans תואם לשם של האלקטרודה במערך chansName. כמו כן נחלץ את קטגוריות הדימיון האפשריים שרלוונטים לניסוי ונשמור את הכמות שלהם - nclass, ואת השורות שמייצגות אותם בנתונים– clasRow במשתנים ואת השמות שלהם במערך(classes). נסכום את מספר הניסויונות עבור כל קטגוריית דימיון ntrailsPerClass ואת כל הנתונים נשמור בתוך מבנה בשם Prmtr.

```
load('motor_imagery_train_data.mat');       % trainig data
testFileName = 'motor_imagery_test_data.mat';
fs = P_C_S.samplingfrequency;               %sampling frequency, Hz
dt = 1/fs;                                  %time step [sec]
nTrials = size(P_C_S.data,1);               % num of trails
trialLen = size(P_C_S.data,2);              % num of sample
timeVec = dt:dt:trialLen/fs;                %create time vec according to parameters
f = 0.5:0.1:40;                             % relevant freq range
window = 1.1;                               %window length in secs
windOverlap = 1;                           %window overlap length in secs
numOfWindows = floor((size(P_C_S.data,2)-window*fs)/...
    (window*fs-floor(windOverlap*fs)))+1;   %number of windows
miStart = 2.25;                             %motor imagery start in sec
miPeriod = timeVec(timeVec >= miStart);     %motor imagery period
edgePrct = 90;                              %spectral edge percentaile

chans = cell2mat(P_C_S.channelname(1:2));   %channels in use
chans = str2num(chans);
chansName = ["C3" "C4"];                    %channels names should corresponds to
chans
nchans = length(chans);                     % num of channels
```

```
nclass = 2;                                  %this project support two classes only
classes = P_C_S.attributename(3:end);        %extract classes assuming rows 1,2 are
artifact and remove
classes = string(classes);
clasRow = cellfun(@(x) find(ismember(P_C_S.attributename,x)), classes, 'un',false);
%extartct classes rows
ntrialsPerClass = [sum(P_C_S.attribute(clasRow{1},:)==1),...
    sum(P_C_S.attribute(clasRow{2},:)==1)];

% creating struct for all relevant parameters
Prmtr =
struct('fs',fs,'time',timeVec,'freq',f,'nTrials',nTrials,'winLen',floor(window*fs),...

'winOvlp',floor(windOverlap*fs),'miPeriod',miPeriod,'nclass',nclass,'classes',classes,
...
    'clasRow',cell2mat(clasRow),'ntrialsPerClass',ntrialsPerClass,...
    'chans',chans,'chansName',chansName,'nchans',nchans,'edgePrct',edgePrct);


isTrainMode = 1; % if set to 1 than the script will ran a loop in order to
%check the best num of features to select using analyzeNumOfFeat function
%when is set to 0 a normal run will occur
```

נתונים:

את המידע נארגן ונשמור במבנה חדש Data שיכיל את כל סוגי הנתונים הרלוונטים שנתונים ואלו שנחשב עבור כל שלבי המודל. את המידע הגולמי מין האלקטרודות נשמור בשדה Data.allData ונייצר את שמות סוגי התנאים הרלוונטים לניסוי-Data.combLables לפי הפרמטרים שהוגדרו(מספר ערוצים כפול מספר קטגוריות דימיון). נבדוק מהם האינדקסים של הנסיונות עבור דימיון לצד שמאל ועבור דמיון לצד ימין Data.indexes ונשמור את התווית של כל ניסיון Data.lables.

```
Data.allData = P_C_S.data;
Data.combLables = cell(1,nchans*nclass);        %lables for channels*class
combinations
Data.lables = strings(nTrials,1);               %lable each trail for his class
k = 1;
for i = 1:nclass
    currClass = classes(i);
    Data.indexes.(classes{i}) = find(P_C_S.attribute(Prmtr.clasRow(i),:)==1);
%finding the indexes for each class
    Data.lables(Data.indexes.(classes{i})) = currClass;                        %lable
each trail for his class
    for j = 1:nchans
        chanCls = char(currClass + chansName(j));
%creating combination name
        Data.(chanCls) = Data.allData(Data.indexes.(classes{i}),:,chans(j));
%arrange data by combination
        Data.combLables{1,k} = chanCls;
        k = k+1;
    end
end
```

מאפיינים(features):

קביעת- bands:

לאחר הסתכלות בגרפים שקיבלנו מהתהליך הויזואליזציה ושימוש בפונקציות השונות ניצור cell array בשם bandPower שישמש כשדה של המבנה Features.
כל תא בו מכיל cell בפני עצמו כך שהאינדקס הראשון שלו מציין את תחום התדרים ביחידות של Hz והאינדקס השני את הזמנים הרלוונטים(בשניות) לאותו תחום תדרים. כל cell כזה מייצג הבדל שנראה לנו כמשמעותי בין התנאים השונים, והעוצמה בו תשמש כמאפיין שינסה לעזור למסווג להבדיל בין ה class במטלת הסיווג.

את כל הפרמטרים והמידע שקשור למאפיינים שנחלץ נשמור בתוך מבנה Features. נשמור את מספר המאפיינים שנרצה להשתמש בסוף כדי לאמן את המודל Features.nFeatSelect, ואת תחומי הזמנים והתדרים שנמצאו רלוונטים לאחר הצגת הנתונים בצורה ויזואלית במערך הנקרא bandPower. בנוסף נגדיר פרמטרים הרלוונטים למאפיינים האמפליטודים של הנתונים הגולמיים:מתח שישמש כ-סף mVthrshold ומערך עם שמות האלקטרודות להשוואה diffBetween. נחשב את מספר המאפיינים שנחלץ לאורך המודל על ידי מספר המאפיינים עבור כל תחום רלוונטי שנבחר ב Features.bandPower (4 טווחים כפול 2 מאפיינים- bandpower ו relative bandpower ) + מספר המאפיינים השונים שבחרנו לחלץ (10). כל מאפיין כזה מחושב עבור כל ערוץ ולכן יחד עם המאפיין האחרון של הפרש מתח בין 2 האלקטרודות ישנם סך הכל 37 מאפיינים. כמו כן בשלב זה נבחר בתור פרמטר את השיטה שלפיה נבצע את בחירת המאפיינים הטובים יותר Features.sfMethod.

```
%band power features 1st arr - band, 2nd arr - time range
Features.bandPower{1} = {[15,20],[3.5,6]};
Features.bandPower{2} = {[32,36],[4,6]};
Features.bandPower{3} = {[9,11],[5.5,6]};
Features.bandPower{4} = {[17,21],[1.2,2.7]};
nBandPowerFeat = length(Features.bandPower)*2;  %bandpower and relative bandpower for
each relevant range

Features.mVthrshld = 4;                         %threshold feature in muV

Features.diffBetween = ["C3","C4"];             % choose two elctrode to calc diff

generalFeat = 10;                               %number of general features should be
10!
%Total Power,Root Total Power,Slope,Intercept,Spectral Moment,Spectral Entropy
%Spectral Edge,Threshold Pass Count,Max Voltage,Min Voltage

nDifFeat = 1;                   %number of diffs between chanle features should be 1!
Features.nFeat = ((nBandPowerFeat+generalFeat)*nclass)+ nDifFeat; %num of total
features feature selection method

Features.nFeatSelect = 8 ;      %number of features to select for classification
Features.fsMethod = "ks";      %choose feature selection method between cna  and ks
Features.distPrecision = 5;     %only in case you run ks for feature selection
```

מודל:
נבחר את k=5 בשביל ליישם את k fold cross validation ונייצר cell 2 בשביל תוצאות הפרדיקציה של המודל – results ובשביל חישוב טעות האימון – trainErr וכן מטריצה לחישוב הaccuracy עבור כל fold בתהליך.

```
k = 5;                          %k fold parameter
results = cell(k,1);
trainErr = cell(k,1);
acc = zeros(k,1);
```

ויזואליזציה ראשונית:

עבור כל קטגוריית דימיון(classes) תחילה נבצע ויזואליזציה של הסיגנל בעזרת הפונקציה signalVisualization המקבלת כקלט את המבנים Data ו Prmtr ואת הclass שעבורו מבצעים את הויזואליזציה. הפונקציה תייצר תרשים של signalPerFig (פרמטר שהוגדר בתחילת הקוד) תתי גרפים בהם הסיגנל הגולמי של כל אחת מהאלקטרודות עבור 20 ניסיונות רנדומלים. בעזרת השדה Data.indexes.(Prmtr.classes{class}) ניגש לאינדקסים הרלוונטים ובעזרת הפונקציה randperm נבחר בצורה אקראית signalPerFig מהאינדקסים האלו. לכל אינדס של ניסיון נחלץ את את הסיגנל התואם לו מתוך המבנה Data עבור כל אחד מהאלקטרודות כאשר כל אלקטרודה תופיע בצבע שונה.

```
%visualization of the signal in Voltage[muV] for rand co-responding trails
for i = 1:nclass
      signalVisualization(Data,i,Prmtr)
end
```

```
function signalVisualization(Data,clss,Prmtr)
% this function plot the Voltage[mV] for a random co-responding trails for each chanle
on the same plot
% the number of trails depend on the pramter
% generic function for all classes
    timeVec = [0:length(Prmtr.time)-1]/Prmtr.fs; %time in sec
    randIndex_trails = Data.indexes.(Prmtr.classes{clss})...
       (randperm(length(Data.indexes.(Prmtr.classes{clss})),...
       (Prmtr.Vis.plotPerRow*Prmtr.Vis.plotPerCol)));   %choose rand trails
    figure('Units','normalized','Position',Prmtr.Vis.globalPos);
    sgtitle(Prmtr.classes{clss} + " " + "Imagery")
    for i = 1:length(randIndex_trails)
        subplot(Prmtr.Vis.plotPerCol,Prmtr.Vis.plotPerRow,i)
        C3 = plot(timeVec,Data.allData(randIndex_trails(i),:,1),'r');   %c3 signal
        hold on
        C4 = plot(timeVec,Data.allData(randIndex_trails(i),:,2),'b');   %c4 signal
        ylim([-15,15])
        if(i>=17)
            xlabel('time [sec]');
        end
        toLable = [1,5,9,13,17];
```

```
        if(ismember(i,toLable))
            ylabel('Amplitude[\muV]');
        end
    end
    hold on
    Leg = legend([C3,C4],Prmtr.classes(1),Prmtr.classes(2));
    set(Leg,'Position',[0.848091556210113 0.925793650793651 0.113690476190476
0.071031746031746],'Units', 'normalized');
end
```

לאחר מכן נחשב את עוצמת התדרים עבור כל אחד מהתנאים בשדה Data.combLables בעזרת הפונקציה של מטלב pwelch. כאן נעשה שימוש ב miPeriod וקטור זמני הדימיון לאורך הניסו כדי לנסות להבין את הנתונים דווקא בשלב שהם אמורים "לספר לנו משהו" , להבדיל בין הקטגוריות. נעבור בלולאה על סוגי התנאים ולכל תנאי נחלץ את הנתונים התואמים לו בזמן הדימיון(Data.(chanCls)(:,(Prmtr.miPeriod*fs))
ונשלח אותם לפונקציה pwelch יחד עם הפרמטרים של גודל החלון, גודל החפיפה, וקטור התדרים וקצב הדגימה. את התוצאה שחוזרת מפונקציית ה pwelch נשמור בשדה Data.PWelch.

```
% calculating PWelch for all condition
for i = 1:nclass
    for j = 1:length(Prmtr.chans)
        currClass = Prmtr.classes(i);
        chanCls = char(currClass + chansName(j));
        Data.PWelch.(chanCls) = pwelch(Data.(chanCls)(:,(Prmtr.miPeriod*fs))',...
            Prmtr.winLen,Prmtr.winOvlp,Prmtr.freq,Prmtr.fs);
    end
end
```

לאחר מכן נציג את הנתונים בעזרת הפונקציה: plotPwelch שמקבלת כפרמטרים את המבנים Data ו Prmtr ומשתמשת גם היא בפונקציה comparePowerSpec עם אותם ארגומנטים. הפונקציה מייצרת 2 תרשימים: הראשון מתבצע בפונקציה עצמה והוא תרשים עם Prmtr. chans* Prmtr.nclass תתי תרשימים עבור כל אחד מהתנאים השונים בו הוא מציג את ממוצע עוצמת התדרים כפונקציה של התדר. הפונקציה עוברת בלולאה על סוגי התנאים ובכל אחת מהאיטרציות מציגה בתת גרף את ממוצע עוצמת התדרים שחושב מקודם בעזרת הפונקציה pwelch כתלות בתדר של הסיגנל. בנוסף הפונקציה מייצרת תת כותרת תאומת לכל תת תרשים ולבסוף קוראת לפונקציה comparePowerSpec . התרשים השני הינו ההפרש של ממוצע עוצמת התדרים בין ה class השונים בתוך כל אלקטרודה כפונקציה של התדר. הפונקציה עוברת בלולואה מספר האלקטרודות ומייצרת תתי גרפים כמספר האלקטרודות בהם היא מציגה את עוצמת התדרים הממוצעת של 2 ה class השונים באותו תת גרף ומציגה כותרת תואמת.

```
function plotPwelch(Data,Prmtr)
%this function plot the mean spctral power by freq for each condition returned by
pwelch commend
    figure('Units','normalized','Position',Prmtr.Vis.globalPos);
    %sgtitle('Power Spect by PWelch')
    for i = 1:length(Data.combLables)   %looping all condition
        subplot(Prmtr.nchans,Prmtr.nclass,i)
        plot(Prmtr.freq,mean(Data.PWelch.(Data.combLables{i}),2))
```

```
        title(Data.combLables{i});hold on
        if(mod(i-1,Prmtr.nchans)==0)
            ylabel('Power Spectrom')
        end
        if(i>=3)
          xlabel('Frequency[Hz]')
        end
    end
    hold off
    comparePowerSpec(Data,Prmtr)
end
```

```
function comparePowerSpec(Data,Prmtr)
%this function compare the chanle for each class for the PWelch results
    figure('Units','normalized','Position',Prmtr.Vis.globalPos);
    %sgtitle('Compare Power Spec by chanle')
    for i =1:length(Prmtr.chansName)
        subplot(Prmtr.nchans,1,i)
        plot(Prmtr.freq,mean(Data.PWelch.(Data.combLables{i}),2),'b')   %left
        hold on
        plot(Prmtr.freq,mean(Data.PWelch.(Data.combLables{i+2}),2),'r') %right
        title(strcat('Power Spectrom diff',{'  '}', (Prmtr.chansName{i})))
        ylabel('Power Spectrom')
        if(i>1)
            xlabel('Frequency[Hz]')
        end
        legend(Prmtr.classes{1},Prmtr.classes{2})
    end
end
```

בשלב הבא נחשב ספקטוגרמה עבור כל אחד מהתנאים בשדה Data.combLables. תחילה נקצה מקום עבור כל תנאי על ידי יצירת מטריצה 3-מימדית בגדול (ntrailsPerClass,size(f,2),numOfWindows) .
כעת נעבור בלולאה על מספר הניסיונות של כל תנאי ונחשב את הספקטוגרמה על ידי הפונקציה של מטלב spectrogram שמקבלת כקלט את הסיגנל הגולמי המתאים עבור הניסיון הספציפי (לאורך כל הניסוי) ואת הפרמטרים של גודל החלון, גודל החפיפה, וקטור התדרים וקצב הדגימה. את התוצאה שחוזרת נכניס למטריצה התואמת שבנינו עבור החזרה הספציפית. בסוף הלולאה הפנימית נסיים לבנות את מטריצת הספקטוגרמה של התנאי הראשון וכך נמשיך כבור כל התנאים. נשים לב שבשונה מחישוב ה- power spectrum מקודם בעזרת הפונקציה pwelch לה שלחנו רק את הנתונים מזמן הדימיון בחישוב הספקטוגרמה שלחנו את כל הסיגנל(לאורך כל ה trail) , וזאת מכיוון שניתן לראות בו גם הבדלים בעוצמה בכל תדר כפונקציה של הזמן. הספקטוגרמה מוסיפה לנו את מימד הזמן דבר שלא ניתן לראות בחישוב ה- power spectrum .
עבור כל מטריצה של ספקטוגרמות שאנו בונים נמיר בה את הערכים לעוצמה בדציבלים ונמצא מעבר לכל החזרות על ידי הפעולות הבאות הבאות: ערך מוחלט על המטריצה והעלתה בריבוע,הכפלה ב 10*log_10 ולבסוף מיצוע על כל החזרות והורדת מימד על ידי הפונקציה squeeze. הסיבה שאנו מבצעים את הפעולות בסדר זה היא שממוצע לאחר טרנספורמציה שאינה לינארית שונה מטרנספורמציה של הממוצע.

```
%calculating spectrogram for all conditions

for i =1:length(Data.combLables)    %looping all condition

   for j = 1:size(Data.(Data.combLables{i}),1)
      Data.spect.(Data.combLables{i})(j,:,:) =
spectrogram(Data.(Data.combLables{i})(j,:)',...
          Prmtr.winLen,Prmtr.winOvlp,Prmtr.freq,Prmtr.fs,'yaxis');
   end
% convert the units to dB and average all spect for each cindition
    Data.spect.(Data.combLables{i}) =
squeeze(mean(10*log10(abs(Data.spect.(Data.combLables{i}))).^2));
end

%visualization spectogram
plotSpectogram(Data,Prmtr)
plotSpectDiff(Data,Prmtr)
```

לאחר שסידרנו והתאמנו את התוצאות עבור כל סוגי התנאים של הניסוי ושמרנו אותם בשדה Data.spec נוכל להציג אותם תוך התייחסות למימד הזמן.נציג 2 תרשימים של הספקטוגרמות:

1. תרשים ובו תתי גרפים כאשר כל תת גרף מציג את תוצאת הסקפוגרמה עבור תנאי שונה של הניסוי כאשר מימד הזמן [sec] הוא ציר האיקס, התדר ביחידות של [Hz] מופיע על ציר ה – *Y* ועוצמת ההתדר בדציבלים תופיע על ידי הצבע בתרשים, כך שככל שהצבע יותר אדום עוצמת התדר חזקה יותר וככל שהצבע כחול יותר עוצמת התדר חלשה יותר. תרשים זה אנו מקבלים על ידי הפעלת הפונקציה plotSpectogram שמקבלת כקלט את Data ו- Prmtr . הפונקציה עוברת בלולואה על כל תנאי הניסוי(Data.combLables), מייצרת תת גרף ובעזרת הפונקציה של מטלב imagesc שמקבלת את וקטור הזמן, וקטור התדרים ואת מטריצת העוצמה הממוצעת שחישבנו מקודם מציגה את הספקטוגרמה.

```
function plotSpectogram(Data,Prmtr)
%this function plot the spctral power for each condition returned by
%spectogram commend
    figure('Units','normalized','Position',Prmtr.Vis.globalPos);
    %sgtitle('Power Spectogram');
    for i = 1:length(Data.combLables)   %looping all condition
        subplot(Prmtr.nclass,Prmtr.nchans,i)
        imagesc(Prmtr.time,Prmtr.freq,Data.spect.(Data.combLables{i}))
        set(gca,'YDir','normal')
        colormap(jet);
        axis square
        title(Data.combLables{i});
        if(mod(i-1,Prmtr.nchans) == 0)
            ylabel ('Frequency [Hz]');
        end
        if(i>2)
            xlabel ('Time [sec]');
        end
    end
    cb = colorbar;
    cb.Label.String = 'Power diff [dB]';
    cb.FontSize = 10;
    pos = [0.9,0.3,0.02,0.35];
    set(cb,'units','Normalized','position',pos);
```

```
    caxis('auto');
end
```

2. תרשים ובו תתי גרפים כמספר האלקטרודות כאשר בכל תת גרף נציג את הפרש העומצה הממוצעת בין שני ה class השונים. כלומר עבור כל אלקטרודה נציג את ההפרש בין ממוצע העוצמה של ימין לבין ממוצע העוצמה של שמאל. הפונקציה plotSpectDiff שמקבלת את Data ו- Prmtr יוצרת את התרשים הנל בכך שתחילה היא מחשבת עבור כל ערוץ את ההפרש בין ממוצע העוצמה של שמאל לבין ימין. היא מחשבת אותם על ידי חיסור של המטריצות שיצרנו עבור כל תנאי ולאחר מכן מציגה אותם בתתי גרפים שוב בעזרת הפונקציה imagesc כמו בתרשים הקודם. שוב הצגה כזו של ההפרשים יותר נוחה לעין כדי לראות את ההבדלים של ה class השונים לאורך הניסוי.

```
function plotSpectDiff(Data,Prmtr)
%this function calculate mean diff between class for each chanle from the spectogram
results
    diff1 = Data.spect.(Data.combLables{1})-Data.spect.(Data.combLables{3});
    diff2 = Data.spect.(Data.combLables{2})-Data.spect.(Data.combLables{4});
    spectDiffCond = {diff1,diff2};
    diffTitle = {'C3Diff','C4Diff'};
    figure('Units','normalized','Position',Prmtr.Vis.globalPos);
    sgtitle('Spectogram Diff')
    for i = 1:length(spectDiffCond)
        subplot(1,length(spectDiffCond),i)
        imagesc(Prmtr.time,Prmtr.freq,spectDiffCond{i})
        set(gca,'YDir','normal')
        colormap(jet);
        axis square
        title(diffTitle{i})
        if(i == 1)
            ylabel ('Frequency [Hz]');
        end
        xlabel ('Time [sec]');
    end
    cb2 = colorbar;
    cb2.Label.String = 'Power diff [dB]';
    pos = [0.91868,0.324253,0.0117708,0.340874];
    lablPos = [3.9142857,0.54263613,0];
    set(cb2,'units','Normalized','position',pos,'FontSize',12);
    set(cb2.Label,'units','Normalized','position',lablPos,'FontSize',12);
    caxis('auto');
end
```

חילוץ פיצרים:

לאחר שהצגנו את הנתונים בדרכים השונות נעבור לשלב בו נייצר מאפיינים שאנו חושבים שיועילו בצורה הטובה ביותר לפעולת הסיווג. תחילה נייצר את המטריצה שתשמש אותנו לשמירת המאפיינים, נקצה לה מקום כך שהמימדים שלה הינם (nTrails*Features.nFeat). את מספר המאפיינים חישבנו בתחילת הניסוי ואנו נחלץ את המאפיין עבור כל אחד מהחזרות של הניסוי. כמו כן נתחזק אינדקס שמסמן תמיד מה היא העמודה הפנוייה במטריצה עבור המאפיין הבא – fIdx=1 . בנוסף אנו נתחזק מערך שישמור את שמות המאפיינים כך שלכל עמודה i במטריצת

המאפיינים(Features.featMat) תאומת עמודה i במערך השמות עם השם המתאים(Featurs.featLables). כעת נשתמש בפונקציה extractFeatures שמקבלת כקלט את:

1) הנתונים של האמפליטודה מכל האלקטרודות עבור כל הניסויים – Data.
2) מבנה הכולל את כל הפרמטרים של הניסוי - Prmtr.
3) מבנה המכיל את הפרמטרים וההשדות של המאפיינים – Features.
4) שם המטריצה אליה אנו מכניסים את המאפיינים שמחלצים – Metrix.
5) האינדקס הפנוי עבור המאפיין הבא.

```
Features.featMat = zeros(nTrials,Features.nFeat);       %allocate space
fIdx = 1;                                               % index for the co-responding
feature to  col
Features.featLables = cell(1,Features.nFeat);           %allocate space to features
name
Features = extractFeatures(Data.allData,Prmtr,Features,'featMat',fIdx);   %calc and
extract all features
[Features.featMat,meanTrain,SdTrain] = zscore(Features.featMat);            % scale
all features
```

אנו שלחנו לפונקציה את כל הארגומנטים שתואמים לסט האימון: Data.allData , Prmtr, Features, 'featMat' ו fIdx. הפונקציה מחזירה כפלט את המבנה Features כך שמטריצת המאפיינים שתואמת לארגומנט הרבעי 'Matrix' (במקרה הספציפי פה מייצרת את מטריצת המאפיינים של סט האימון - 'featMat') כוללת בתוכה את כל המאפיינים שחולצו וגם מעדכנת את מערך שמות המאפיינים.
הפונקציה עוברת בלולאה ראשית על מספר האלקטרודת ועבור כל אלקטרודה היא מחשבת 18 מאפיינים שונים(nBandPowerFeat+generalFeat). בנוסף היא מחשבת בסוף הפונקציה בעזרת פונקציית עזר chanDiffAmp את המאפיין של הפרש אמפליטודי בין 2 ערוצים:

Band Power - חישוב העוצמה בכל רצועת תדרים בזמן שקבענו לה, בעזרת הפונקציה bandpower. נמצא את התחום שתואם לכל cell ב Features. bandPower ונפעיל את הפונקציה bandPower שמקבלת את הסיגנל בטווח הזמן שחילצנו, את קצב הדגימה, ואת טווח התדרים שתואם לה לפי Features. bandPower . הפונקציה תחזיר כפלט את העוצמה הממוצעת ברצועת התדרים בזמן שקבענו.

Relative power- נחלק את ה Band Power שחילצנו עבור ה cell הספציפי בעוצמה הכללית של של התדרים. את העוצמה הכוללת נחשב על ידי שימוש בפונקציה bandPower שתקבל רק את הסיגנל בטווח הזמן שחילצנו. הפלט של המאפיין הזה הינו העוצמה היחסית של ה band מתוך כל הספקטרום.

כעת נחשב את ה spect power על הזמן בו הנבדק דמיין את תנועת היד,שכן בזמן זה נצפה הבדלים בעוצמת התדרים בין התנאים השונים - PW

Total Power and Root Total Power – נחשב את העוצמה הכוללת בעזרת הפונקציה sum שתקבל את עוצמת התדרים – PW, ואת שורש עוצמת התדרים בעזרת הפונקציה root.

Slope and Intercept – נחשב Spectral fit ונקבל את השיפוע והחותך של המשוואה הלינארית לעוצמה:
לשם כך נשתמש בפונקציה specSlopeInter שמקבלת כקלט את עוצמת וקטור התדרים – PW ואת וקטור התדרים ומחזירה את השיפוע והחותך. פונקציה זו עוברת על מספר ה trails ועבור כל trail היא מחשבת את השיפוע והחותך בעזרת הפונקציה polyfit עבור 10 log 10 של התדרים ממעלה ראשונה.

בשביל המאפיינים הבאים נמיר את את הערכים של PW – עוצמת התדרים כפונקציה של התדר לפונקציית הסתברות, כלומר נחלק את עוצמת התדרים ב Total Power והיא תשמש כפונקציית ההסתברות של העוצמה לכל חלון(probability). פונקציה זו מציינת את ההסתברות לקבלת תדר מסויים כפונקציה של העוצמה. בכך נקבל כי סכום כלל הערכים הינו 1.

Spectral Moment – בשביל לחשב את המאפיין נכפול את וקטור התדרים ב probability, כלומר נכפול כל תדר בהסתברות לקבל אותו.

Spectral Entropy – עבור מאפיין זה השתמשנו בנוסחה:

spectral entropy=−Σpnorm(f)log2(pnorm(f))

חישבנו את אנטרופיית עוצמת התדרים המנורמלת על ידי סכימה של כפל מטריצת התדרים המנורמלת בכפל אלמנטים של: פונקציית התדרים מנורמלת $\log_2$ והוספת מינוס כדי להישאר עם ערכים חיוביים.

Spectral Edge – כדי לחשב את המאפיין הזה נשתמש בפונקציה spectralEdge שמקבלת את פונקציית הסתברות התדרים, את וקטור התדרים ואת האחוזון הרצוי שקבענו בהגדרת המשתנים. פונקציה זו מוצאת את התדר שמתחתיו נמצאים 90 אחוז מהעוצמה של אותה אלקטרודה: בעזרת הפונקציה cumsum אנו עושים סכימה מצטברת של העוצמה ובודקים היכן היא מעל סף ה90 אחוז. לאחר מכן בעזרת הפונקציה diff אנו מקבלים את האינדקס בו העוצמה חוצה את הסף ולבסוף אנו מחלצים את התדר הנכון מוקטור התדרים.

Threshold Pass Count – נחשב את מספר הדגימות שהמתח הנמדד בהם היה מעל סף שנקבע בהגדרת המשתנים. נעשה זאת בעזרת יצירת וקטור לוגי על כל הדגימות שמעל הסף ואז בעזרת הפונקציה diff נקבל את האינדקסים בהם המתח חוצה את הסף לכיוון מסויים, נפעיל עליהם ערך מוחלט כדי לתפוס את חציית הסף בכיוון הנכון ולבסוף נסכום אותם.

Max Amplitude – נחשב את המתח המקסימלי בעזרת הפונקציה max על כל הסיגנל עבור כל ה trails.

Min Amplitude – נחשב את המתח המינמלי בעזרת הפונקציה min על כל הסיגנל עבור כל ה trails.

Diff Amplitude Between Channels – נחשב את הפרש האמפליטודה בין 2 אלקטרודות: נשתמש בפונקציה chanDiffAmp שמקבלת את Prmtr Data ואת שני האלקטרודות שנקבעו בהגדרת המשתנים(Features.diffBetween). פונקציה זו מחזירה את וקטור ההפרשים עבור כל trail ואת התיאור של שם המאפיין בהתאם לאלקטרודות בחיסור.

```
function [Features] = extractFeatures(Data,Prmtr,Features,Matrix,fIdx)
% function that extract all fearures and arrange it in a matrix
% Data - all data for extracting the features
% Prmtr - all parameter of the Data
% Features - struct that containing all features data lables and parameters
% Matrix - train or test matrix
% fIdx - curr index for insert fearure and lable
for i = 1:length(Prmtr.chans)
    for j = 1:length(Features.bandPower)    %looping over relevant range
        tRange = (Prmtr.time >= Features.bandPower{j}{2}(1) & ...
            Prmtr.time <= Features.bandPower{j}{2}(2));
        %raw bandpower
        Features.(Matrix)(:,fIdx) = ...
            (bandpower(Data(:,tRange,i)',Prmtr.fs,Features.bandPower{j}{1}))';
        Features.featLables{fIdx} =char("Bandpower - "+Prmtr.chansName(i)+newline+...
            " "+Features.bandPower{j}{1}(1)+"Hz - "+Features.bandPower{j}{1}(2)+"Hz");
        fIdx = fIdx + 1;
        %relative bandpower
```

```
        totalBP = bandpower(Data(:,tRange,i)')';
        Features.(Matrix)(:,fIdx) = Features.(Matrix)(:,fIdx-1)./totalBP;
        Features.featLables{fIdx} = char("Relative Bandpower -
"+Prmtr.chansName(i)+newline+...
            " "+Features.bandPower{j}{1}(1)+"Hz - "+Features.bandPower{j}{1}(2)+"Hz");
        fIdx = fIdx + 1;
    end
    PW = pwelch(Data(:,(Prmtr.miPeriod*Prmtr.fs),i)',...
            Prmtr.winLen,Prmtr.winOvlp,Prmtr.freq,Prmtr.fs);    %calc PWelch for
features
    %total power and root total power
    power = sum(PW);
    Features.(Matrix)(:,fIdx) = power';
    Features.featLables{fIdx} = char("Total Power " + Prmtr.chansName{i});  %update
the feature name
    RTP = sqrt(power);
    Features.(Matrix)(:,fIdx+1) = RTP';
    Features.featLables{fIdx+1} = char("Root Total Power "+
Prmtr.chansName{i});%update the feature name
    %Spectral fit
    [slope,intercept] = specSlopeInter(PW,Prmtr.freq);
    Features.(Matrix)(:,fIdx+2) = slope;
    Features.featLables{fIdx+2} = char("Slope "+ Prmtr.chansName{i});%update the
feature name
    Features.(Matrix)(:,fIdx+3) = real(log(intercept)); %return the intercept scale
    Features.featLables{fIdx+3} = char("Intercept "+ Prmtr.chansName{i});%update the
feature name
    probability = PW./power;    %normalize the power by the total power so it can be
treated as a probability
    %spectralMoment
    Features.(Matrix)(:,fIdx+4) = (Prmtr.freq*probability)';
    Features.featLables{fIdx+4} = char("Spectral Moment "+ Prmtr.chansName{i});%update
the feature name
    %Spectral entropy
    Features.(Matrix)(:,fIdx+5) = (-sum(probability .* log2(probability),1))';
    Features.featLables{fIdx+5} = char("Spectral Entropy "+
Prmtr.chansName{i});%update the feature name
    %Spectral edge
    Features.(Matrix)(:,fIdx+6) =
(spectralEdge(probability,Prmtr.freq,Prmtr.edgePrct))';
    Features.featLables{fIdx+6} = char("Spectral Edge "+ Prmtr.chansName{i});%update
the feature name
    fIdx = fIdx + 7;
    %Threshold Pass count
    Features.(Matrix)(:,fIdx) = sum(abs(diff(Data(:,:,i) >=
Features.mVthrshld,[],2)),2); %get sum of passed th
    Features.featLables{fIdx} = char("Threshold Pass Count "+ Features.mVthrshld+"mV "
+ Prmtr.chansName{i});%update the feature name
    %Max Voltage
    Features.(Matrix)(:,fIdx+1) = max(Data(:,:,i),[],2);
    Features.featLables{fIdx+1} = char("Max Voltage " + Prmtr.chansName{i});%update
the feature name
    %Min Voltage
    Features.(Matrix)(:,fIdx+2) = min(Data(:,:,i),[],2);
    Features.featLables{fIdx+2} = char("Min Voltage " + Prmtr.chansName{i});%update
the feature name
    fIdx = fIdx + 3;
end
```

```
% diff Amplitude between channels
channle1 = find(Prmtr.chansName == Features.diffBetween(1));
channle2 = find(Prmtr.chansName == Features.diffBetween(2));
[diffAmpSum,description] = chanlDiffAmp(Data,Prmtr,channle1,channle2);
Features.(Matrix)(:,fIdx) = diffAmpSum;
Features.featLables{fIdx} = description;
end
```

לאחר חילוץ כל המאפיינים ננרמל את התוצאות בעזרת הפונקציה zscore ונשמור את וקטור הממוצעים וסטיות התקן.

היסטוגרמות:

נייצר את ההיסטוגרמות לכל אחד מהמאפיינים על ידי הפונקציה makeFeaturesHist שמקבלת את Data, Featuers ו Prmtr ויוצרת עבור כל מאפיין תרשים ובו 2 תתי גרפים בהם ההיסטוגרמה של כל אלקטרודה(מלבד המאפיין האחרון שכבר לוקח בחשבון התייחסות ל2 האלקטרודות ביחד).
הפונקציה מחלצת את המספר של המאפיין האחרון ומחשבת את גודל הקפיצה בכדי להתאים לכל מאפיין את המאפיין התואם להאלקטרודה השנייה. לאחר מכן הפונקציה עוברת בלולאה על מחצית מספר המאפיינים(לא כולל המאפיין האחרון), מייצרת תרשים ואז בלולאה פנימית על מספר הערוצים מייצרת תת גרף,מתאימה את הכותרת הרלוונטית למאפיין ועבור כל class מחשבת את ההיסטוגרמה על המידע המתאים בעזרת ה bins שהוגדרו בחלק הגדרת המשתנים. לבסוף הפונקציה מבצעת את ההיסטוגרמה גם עבור המאפיין האחרון.
היסטוגרמות מציגות את ההפרדה בין התנאים של שמאל וימין עבור כל מאפיין בו השתמשנו. בדרך זו נוכל לבחון מאפיין באופן נקודתי ולדעת האם הוא מייצר הפרדה טובה בין התנאים.

```
function makeFeaturesHist(Prmtr,Features,Data)
%this function loops through all features and creats histogram for each one
% of them.
% each figure plot using subplot the same feat for all channles
    lastFeat = size(Features.featMat,2);        %should be diff between elec feature
    jump = (size(Features.featMat,2) - 1)/2;    %co-responding feature for second
electrode
    hist = cell(1,Prmtr.nclass);                %allocate space
    for i = 1:(size(Features.featMat,2))/2      %loops through all features
        titlCell = cell(1,length(Prmtr.chans)); % co-responding title for all chans
for each elec
        figure('Units','normalized','Position',Prmtr.Vis.globalPos);
        hold on;
        for k = 1:Prmtr.nchans
            subplot(length(Prmtr.chans),1,k)    %subplot the same fearure for each
elect
            if(k~=1)        % check if use jump for co-responding feature for diff
elec
                i = i + (jump*(k-1));
            end
            titlCell{k} = char(Features.featLables{i}); % select the co-responded
feature name
            for j = 1:Prmtr.nclass
                %making the hist for each class
                hist{j} =
histogram(Features.featMat(Data.indexes.(Prmtr.classes{j}),i),'nor','pr');
```

```
                hist{j}.BinEdges = Prmtr.Vis.binEdges;
                hold on;
                alpha(Prmtr.Vis.trnsp);
            end
            if(k == 1)
                legend(Prmtr.classes{1},Prmtr.classes{2})
            end
            title(titlCell{k}, 'Units','normalized','Position', Prmtr.Vis.globTtlPos);
            ylabel('probability');
            if(k == length(Prmtr.chans))
                xlabel('Standad Deviation');
            end
        end
        xlim(Prmtr.Vis.xLim);
        hold off;
    end
    %plot the last feature
    figure('Units','normalized','Position',Prmtr.Vis.globalPos);
    ttlast = char(Features.featLables{lastFeat}); % select the co-responded feature
name for the last feature
    for j = 1:Prmtr.nclass
        hist{j} =
histogram(Features.featMat(Data.indexes.(Prmtr.classes{j}),lastFeat),'nor','pr');
        hist{j}.BinEdges = Prmtr.Vis.binEdges;
        hold on;
        alpha(Prmtr.Vis.trnsp);
    end
    hold on;
    xlim(Prmtr.Vis.xLim);
    title(ttlast, 'Units','normalized','Position', Prmtr.Vis.globTtlPos);
    xlabel('Standard Deviation');
    ylabel('probability');
    legend(Prmtr.classes{1},Prmtr.classes{2});
    hold off;
end
```

בחירת מאפיינים:

בהגדרות הפרמטרים אנו מגדירים משתנה isTrainMode כאשר אם הוא 1 אנו מבצעים אנליזה מתקדמת של בחירת המאפיינים. אנו עושים זאת בכך שאנו עוברים בלולאה על מספר המאפיינים ועבור כל מספר שונה של מאפיינים אנו מאמינים מודל, מבצעים תהליך של cross-validation ומחשבים את ממוצע וסטיית התקן של ה accuracy ואת המדד של F1 Score. את הנתונים המחשובים אנו מציגים בצורה גרפית בכדי להבין מהו מספר המאפיינים שישיג את התוצאות הטובות ביותר.
לשם כך נאתחל 5 וקטורים באורך של מספר המאפיינים בכדי לשמור את תוצאות ה:
F1,accMean,accTrainMean,accSD,accTrainSD .
כעת עבור כל מספר שונה של מאפיינים נשתמש בפונקציה selectFeat שמקבלת את Features ואת התיוג של כל trail והיא מחזירה לנו את האינדקסים של המאפיינים שהיא בוחרת לנכון לקחת, כלומר אלו שלטענתה יביאו אחוז סיווג טוב יותר.

```
if isTrainMode == 1     %if true than check best num of features by loop through all
features
    numOfIter = Features.nFeat;
    f1 = zeros(numOfIter,1);
```

```
    accAvg = zeros(numOfIter,1);
    trAccAvg = zeros(numOfIter,1);
    accSD = zeros(numOfIter,1);
    trAccSD = zeros(numOfIter,1);
else
    numOfIter = 1;      %if false than train model once with nFeatSelect features
end

for iter = 1:numOfIter  %in case that train mode is on this loop will finde the best
num of features
```

```
%   select only the nFeatSelect best features
    if isTrainMode == 1
        Features.nFeatSelect = iter;
    end
    [featIdx,selectMat,featOrder] = selectFeat(Features,Data.lables);
```

הפונקציה selectFeat בודקת את הפרמטר שנקבע בשלב הפרמטרים והוא Features.sfMethod ולפי הערך שלו יודעת באיזה שיטה של בחירת מאפיינים להשתמש: nca או kst. שתי הפונקציות האלו מקבלות את מטריצת המאפיינים ואת התיוג של כל trail ומחזירות כפלט את המשקל שהם נותנות לכל מאפיין:
הפונקציה fsks מקבלת כקלט גם את יחידות ה bins עבור ההיסטוגרמה שהיא מייצרת:
הפונקצייה הופכת את התיוגים ממערך של string למערך קטגוריאלי בעזרת הפונקציה categorical.
לאחר מכן בעזרת הפונקציה categories היא מייצרת cell ובו סוגי הקטגוריות בלבד. כעת אנו בונים שני cell לכל class ובו הכל תא יש את מאפיין. בשלב הבא ניקח מכל מאפיין את המקסימום ואת המינימום שלו בכדי שכשנפעיל את הפונקציה histcount נוכל לחשב את ההתפלגות של כל class.
לבסוף בעזרת הפונקציה kstest2 נבצע את מבחן קולגורוב סמירנוף ונחזיר את המשקולות של כל מאפיין.
אחרי שקיבלנו את המשקולות של כל המאפיינים נמיין אותם בסדר יורד אם אנו עובדים עם nca או סדר עולה אם עובדים עם ks. נבחר את ה Features.nFeatSelect המאפיינים הראשונים בסידור וניצור את מטריצת המאפיינים הנבחרים. פונקציה זו מחזירה את המטריצה הנבחרת ואת האינדקסים של המאפיינים שנבחרו. בנוסף פונקציה זו מחזירה את הרשימה של כל סדר המאפיינים בשביל ניתוחי המשך ותצוגה.

```
function [featIdx,selectMat,featOrder] = selectFeat(Features,labels)
% this function extract the n best fearures using one of 2 method
% Features - struct that containing all features data lables and parameters
% nFeat2Reduce = the num of feature to slect
    %Selecting features by method
    if Features.fsMethod == "nca"
        Selection = fscnca(Features.featMat,labels);
        weights = Selection.FeatureWeights;
        order = 'descend';
    else
        weights = fsks(Features.featMat,labels,Features.distPrecision);
        order = 'ascend';
    end
    %Decsending order of importence
    [~ , featOrder] = sort(weights, order);
    %Taking the most importent features
    featIdx = featOrder(1:Features.nFeatSelect);
```

```
    selectMat = Features.featMat(:,(featIdx));
end
```

```
function p = fsks(features,lables,precision)
%this function gives each features a weight which is the p value of the
%two-sample Kolmogorov-Smirnov test. the function finds all trials belongs
%to each class, finds the max and min value in order to calculate the range where
% the distributions will be calculated. the precision of the distributions
% is by multiplying the precision parameter with the range.
%lastly it runs the Kolmogorov-Smirnov test for all distribitions
    catVec = categorical(lables);
    cat = categories(catVec);
    celFeat1 = num2cell(features(catVec == cat{1},:),1);    %only 1st class trails
    celFeat2 = num2cell(features(catVec == cat{2},:),1);    %only 2nd class trails
    maxVal = num2cell(max(features,[],1)); %get total max for each feature
    minVal = num2cell(min(features,[],1)); %get total min for each feature
    hist1 = cellfun(@(x,y,z) histcounts(x,linspace(y,z,(z-y)*precision),'no','pr'),...
        celFeat1,minVal,maxVal,'UniformOutput',false);  %calculate distribution for
1st class
    hist2 = cellfun(@(x,y,z) histcounts(x,linspace(y,z,(z-y)*precision),'no','pr'),...
        celFeat2,minVal,maxVal,'UniformOutput',false);  %calculate distribution for
2nd class
    [~,p] = cellfun(@(x,y) kstest2(x,y), hist1, hist2); %two-sample Kolmogorov-Smirnov
test
end
```

לאחר מכן אנו מבצעים תהליך של cross-validation :
באופן רנדומלי ניתן לכל trail מספר בין 1 ל-k (פרמטר של הניסוי שנקבע בהתחלה ומשמש למספר הפולדים) שיסמן לאיזה קבוצה הוא שייך. נעשה זאת בעזרת סידור ה trails כפרמוטציה של המספרים בין 1 למספר ה- trails ואז גיבוב בעזרת הפונקציה mod(5) . נקצה מקום בשביל ה- confusion matrix – מטריצה ריבועית לפי מספר ה class-ים השנים וכעת בלולאה על מספר הפולדים(K) נייצר כל פעם את קבוצת האימות וקבוצת האימון. כאשר כל פעם מספר אחר ישמש לקבוצת האימות וכל השאר לקבוצת האימון. נפעיל את המסווג בעזרת הפונקציה של מטלב classify שמקבלת כקלט את סט המבחן,סט האימון, את התיוגים של סט האימון ואת אלגוריתם הסיווג. אנו נשלח לפונקציה את הפרמטרים הנל כאשר אלגוריתם הסיווג הינו 'linear' בכדי לעבוד עם אלגוריתם LDA. פונקציה זו מחזירה כפלט את הפרדיקציה שלה ומחשבת את הטעות על סט המבחן. תוצאות אלו יכנסו cell של results ו trainErr בהתאמה. כעת כדי לחשב את הדיוק נסכום את הפעמיים בהם הפרדיקציה הייתה נכונה וממיר את זה לאחוזים על ידי חלוקה באורך הפרדיקציה והכפלה ב100. בשלב הבא נייצר את ה- confusion matrix בעזרת הפונקציה של מטלב confusionmat שמקבלת כקלט את התיוגים של סט המבחן ואת הפרדיקציה שיצרנו ומייצרת את המטריצה מחולקת לכל סוגי ההצלחות והכישלונות של הפרדיקציה. בסוף הלולאה על מספר הפולדים נוכל לחשב את ה precision ו recall בכדי לחשב את F1 Score ולמצא את הדיוק ולחשב את סטיית התקן של סט המבחן. לבסוף נהפוך את טעות האימון שחזרה מהמסווג למידת דיוק על השלמה ל 1 וכפל ב100 כדי לקבל אחוז ושוב נחשב את הממוצע וסטיית התקן של דיוק האימון.

```
idxSegments = mod(randperm(nTrials),k)+1;   %randomly split trails in to k groups
    cmT = zeros(nclass,nclass);                 % allocate space for confusion matrix
    for i = 1:k
    % each test on 1 group and train on the else
        validSet = logical(idxSegments == i)';
```

```
        trainSet = logical(idxSegments ~= i)';
        [results{i},trainErr{i}] =...

classify(selectMat(validSet,:),selectMat(trainSet,:),Data.lables(trainSet),'linear');
        acc(i) = sum(results{i} == Data.lables(validSet));  %sum num of correct
results
        acc(i) = acc(i)/length(results{i})*100;
        %build the confusion matrix
        cm = confusionmat(Data.lables(validSet),results{i});
        cmT = cmT + cm;
    end
    %calculate accuracy and f1
    percision = cmT(1,1)/(cmT(1,1) + cmT(1,2));
    recall = cmT(1,1)/(cmT(1,1) + cmT(2,1));
    f1(iter) = 2*((percision*recall)/(percision+recall));

    accAvg(iter) = mean(acc);
    accSD(iter) = std(acc);

    trainAcc = (1-cell2mat(trainErr))*100;
    trAccAvg(iter) = mean(trainAcc);
    trAccSD(iter) = std(trainAcc);

end
```

אחרי שעברנו על כל מספרי המאפיינים נציג בצורה ויזואלית את התוצאות בעזרת הפונקציה analyzeNumOfFeat שמקבלת כקלט את Prmtr ומערך שכולל את התוצאות של F1,accMean,accTrainMean,accSD,accTrainSD ואת מספר המאפיינים הנבחרים ומוציאה כפלט את המספר שמביא לרמת מודל גבוהה ביותר לפי תוצאת ה F1 Score. הפונקציה בונה וקטור עם מספרי המאפיינים ומייצרת תרשים ובו השגיאה באחוזים עבור כל בחירה של מספר מאפיינים שונה יחד עם 2 קווים שמלווים את השגיאה כפונקציה של סטיית התקן שלהם. בכל תרשים כזה מציגים את התוצאות האלו עבור סט האימון וסט המבחן יחד. כל הנתונים האלו חושבו כבר מקודם ולכן הפונקציה רק מבצעת גישה למידע יחד עם חיבור וחיסור של וקטורים מתאימים. בנוסף הפונקציה מייצרת תרשים ובו ציון ה F1 Score כפונקציה של כמות המאפיינים שנבחרו. לבסוף הפונקציה מחזירה את את האינדקס בו התוצאה של F1 Score היא הגבוה ביותר.

```
function maxF1Idx = analyzeNumOfFeat(Prmtr,analyzeMath,Features)
% this function make an advance analysis to detarmain what is the best num
% of feature which will perform bast at the test.
% function use F1 Score and error of accuracy
% Prmtr - all parameter of the Data
    numFeatVec = 1:1:Features.nFeatSelect;
    figure('Units','normalized','Position',Prmtr.Vis.globalPos);
    suptitle(char('Validation Vs Train Accuracyfor '+Features.fsMethod+' method'));
    % validation plot
    ValErrorVec = 100 - analyzeMath(:,2);   %turning the acc to error
    [minErr,minErIdx] = min(ValErrorVec);
    maxErr = max(ValErrorVec);
    posSDVal = ValErrorVec + analyzeMath(:,3);
    negSDVal = ValErrorVec - analyzeMath(:,3);
    V = plot(numFeatVec,ValErrorVec,'b'); hold on;
    upV = plot(numFeatVec,posSDVal,'color',[0 0.4470 0.7410],'LineStyle',':'); hold
on;
```

```
    underV = plot(numFeatVec,negSDVal,'color',[0 0.4470 0.7410],'LineStyle',':'); hold
on;
    minV = plot(numFeatVec(minErIdx),minErr,'or','Color','k','LineWidth',3);
    % train plot
    TrainErrorVec = 100 - analyzeMath(:,4); %turning the acc to error
    posSDTrain = TrainErrorVec + analyzeMath(:,5);
    negSDTrain = TrainErrorVec - analyzeMath(:,5);
    T = plot(numFeatVec,TrainErrorVec,'r'); hold on;
    upT = plot(numFeatVec,posSDTrain,'color',[0.8500 0.3250 0.0980],'LineStyle',':');
hold on;
    underT = plot(numFeatVec,negSDTrain,'color',[0.8500 0.3250
0.0980],'LineStyle',':'); hold on;

    ylabel('Error [%]')
    xlabel('Num Of Select Feature')
    ylim([0 maxErr+10])
    Leg2 = legend([V,upV,underV,T,upT,underT,minV],...
        {'Validation','Validation+SD','Validation-SD','Train','Train+SD','Train-
SD','Min Error'});
    set(Leg2,'location','northeast');
    %F1 score plot
    figure('Units','normalized','Position',Prmtr.Vis.globalPos);
    suptitle(char('F1 Score by Num Of Features for '+Features.fsMethod+' method'));
    P = plot(numFeatVec,analyzeMath(:,1),'b');
    hold on;
    xlabel('Num Of Selected Feature')
    ylabel('F1 Score')
    Leg3 = legend('F1 Score');
    set(Leg3,'location','southeast');
    [maxF1,maxF1Idx] = max(analyzeMath(:,1));
    Pmax = plot(numFeatVec(maxF1Idx),maxF1,'or','LineWidth',3);
    Leg3 = legend([P,Pmax],{'F1 score','max score'});
    set(Leg3,'location','southeast');
end
```

כעת נדפיס את התוצאות בעזרת הפונקציה printAcc שמקבלת כקלט את ממוצע הדיוק,סטיית התקן ופרמטר שמציין האם זה תוצאות האימון או המבחן ומדפיסה למסך את תוצאת הדיוק מלווה בסטיית התקן שלה עם התיאור המתאים(אימון או מבחן). נשלח לפונקציה בהתאמה את התוצאות של סט האימון עם הפרמטר 1 ופעם שנייה את תוצאות סט המבחן עם הפרמטר 0.
לאחר הצגת התוצאות נציג את ה – confusion matrix בעזרת הפונקציה confusionchart שמקבלת כקלט את מטריצת ה confusion ואת סוגי ה class-ים.

```
if isTrainMode == 1             %plots for checking best features number
    disp(char("optimal number of features is: "+...
        analyzeNumOfFeat(Prmtr,[f1,accAvg,accSD,trAccAvg,trAccSD],Features)));
else                            %in regular mode, print accuracy and confusion matrix
    printAcc(accAvg,accSD,1);
    printAcc(trAccAvg,trAccSD,0);
    figure('Units','normalized','Position',globalPos)
    cmChart = confusionchart(cmT,[classes(1) classes(2)]);
    cmChart.Title = char(classes(1)+" "+classes(2)+" classification");
end
```

בעזרת הפונקציה plotPCA שמקבלת את מטריצת המאפיינים, Data ו Prmtr נציג 2 גרפים שמייצגים הפרדה של הנתונים.
ראשית נשתמש בפונקציה pca כדי לקבל את הוקטורים העצמיים שמתאימים למטריצת המאפיינים, ניקח את השלושה המתאימים לערכים העצמיים הגדולים ביותר, ונכפיל בכפל מטריצות עם מטריצת המאפיינים.
המטריצה component המקודדת מייצגת את מטריצת ה- מאפיינים לאחר שהופחתה לשלושה ממדים. ההפחתה היא לשלושה ממדים מכיוון שוויזואליזציה של הנתונים מתאפשרת עד 3 ממדים (דו ותלת ממד).
כעת נייצר 2 גרפים אחד עבור הצגה בדו מימד ואחד בשביל הצגה בתל מימד. כדי להציג את הנתונים נשתמש בפונקציה scatter ו scatter3 בהתמה כאשר קוראים לכל פונקציה פעמיים, קריאה אחת עבור כל סוג של class. פונקציות אלו מקבלות ארגומנטים כמספר המימדים שאנו מציגים כאשר כל ארגומנט הוא מאפיין אחד המציג את תוצאות המאפיין עבור ה class הספציפי(כלומר ניקח רק את האינדקסים המתאימים לכל class).

```
function plotPCA(featMet,data,Prmtr)
% function that compute PCA and Ploting the seperation of 2 and 3 dimensions PCA
    comp = pca(featMet);
    comp = comp(:,1:3);             %choose the best 3 PCA
    component = (featMet * comp)';  % coded matrix
    figure('Units','normalized','Position',Prmtr.Vis.globalPos);
    sz = 11;    %Scatter dots size
    %2 dimentional
    sgtitle("PCA Plots");
    subplot(1,2,1)
    scatter(component(1,(data.indexes.LEFT)),...
        component(2,(data.indexes.LEFT)),sz,'b','filled');hold on;
    hold on;
    scatter(component(1,(data.indexes.RIGHT)),...
        component(2,(data.indexes.RIGHT)),sz,'r','filled')
    title('2 Dimentional PCA')
    axis square
    xlabel('PC1'); ylabel('PC2');
    set(gca,'YDir','normal')
    %3 dimentional
    subplot(1,2,2)
    scatter3(component(1,(data.indexes.LEFT)),...

component(2,(data.indexes.LEFT)),component(3,(data.indexes.LEFT)),sz,'b','filled')
    hold on;
    scatter3(component(1,(data.indexes.RIGHT)),...

component(2,(data.indexes.RIGHT)),component(3,(data.indexes.RIGHT)),sz,'r','filled')
    title('3 Dimentional PCA')
    axis square
    xlabel('PC1'); ylabel('PC2'); zlabel('PC3');
    set(gca,'YDir','normal')
    leg3 = legend('Left','Right');
    pos2 = [0.01203125,0.839907407407407,0.062239583333333,0.078148148148148];
    set(leg3,'position',pos2)
end
```

– Test Data

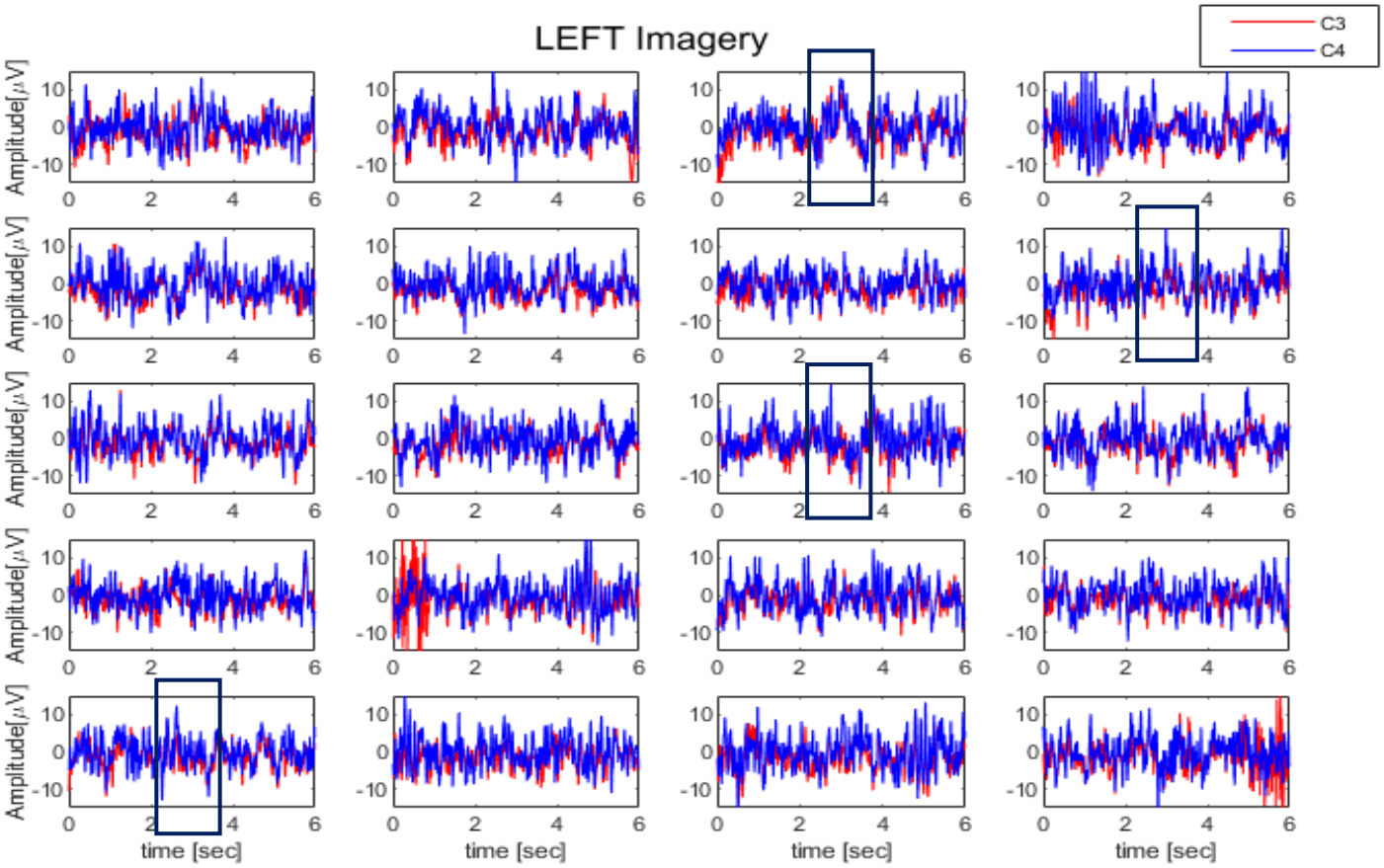
בשלב זה נטען את שם קובץ הנתונים עבור ה- test (testFileName) ונשמור את כל הנתונים התואמים לסט האימון,כלומר את האלקטרודות שעבדנו איתם במהלך הניסוי. לאחר מכן נייצר מטריצה עבור מאפייני סט ה test כאשר ממדיה הם מספר הtrail של סט הtest * מספר המאפיינים שחילצנו בסט האימון. בשלב הבא נשתמש בפונקציה extractFeatuers כאשר הפעם נשלח לך את הסיגנל של סט הtest ואת שם המטריצה אליה הם יחלצו את ה- features. לאחר מכן נבחר רק את המאפיינים שבחרנו עבור המסווג של סט האימון ונבצע תהליך של נרמול למאפיינים בעזרת וקטור הממוצעים וסטיות התקן של סט האימון. לבסוף באמצעות הפונקציה classify שמקבלת הפעם את מטריצת המאפיינים של סט ה test, מטריצת המאפיינים של סט האימון, התיוגים של סט האימון ושוב 'linear' נקבל את הניבויים לסט ה test על ידי מסווג בשיטת LDA.

```
%Load test data

load(testFileName)
testData = data(:,:,1:length(Prmtr.chans));

%Test feature extraction

Features.TestFeatMat = zeros(size(testData,1),Features.nFeat);              %allocate
space
Features = extractFeatures(testData,Prmtr,Features,'TestFeatMat',fIdx);     %calc and
extract all features

%Test feature selection
Features.TestFeatMat = Features.TestFeatMat(:,featIdx);                 %choosing the
same features
Features.TestFeatMat = (Features.TestFeatMat -
meanTrain(:,featIdx))./SdTrain(:,featIdx);      % scale according to train data

%Test classifier - output is the classifier predictions for test data.
testPredict = classify(Features.TestFeatMat,selectMat,Data.lables,'linear');
```

## תוצאות

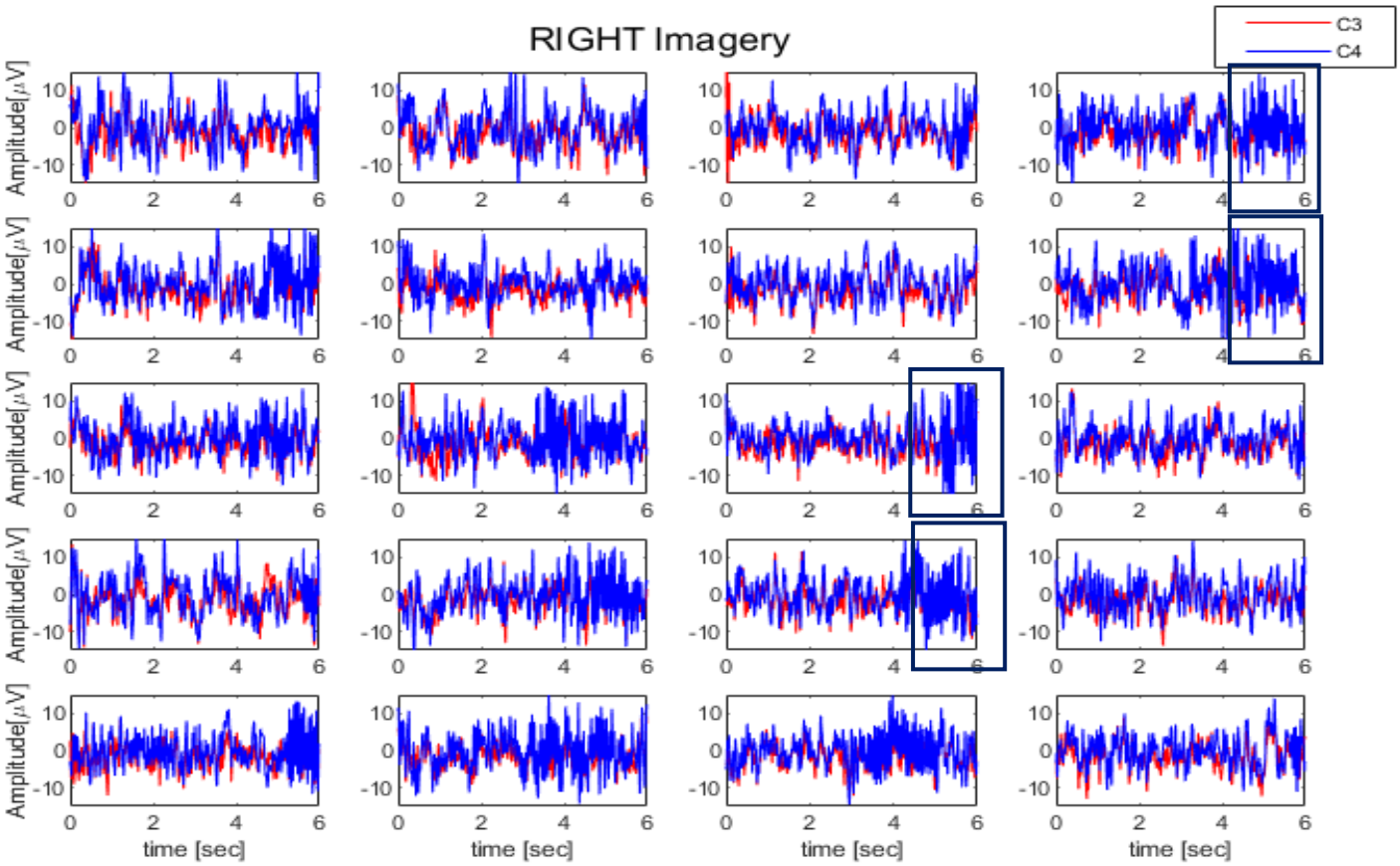
20 *trials* של אמפליטודה כתלות בזמן:

גרף 1 – שתי אלקטרודות כאשר הנבדק התבקש לחשוב על יד שמאל

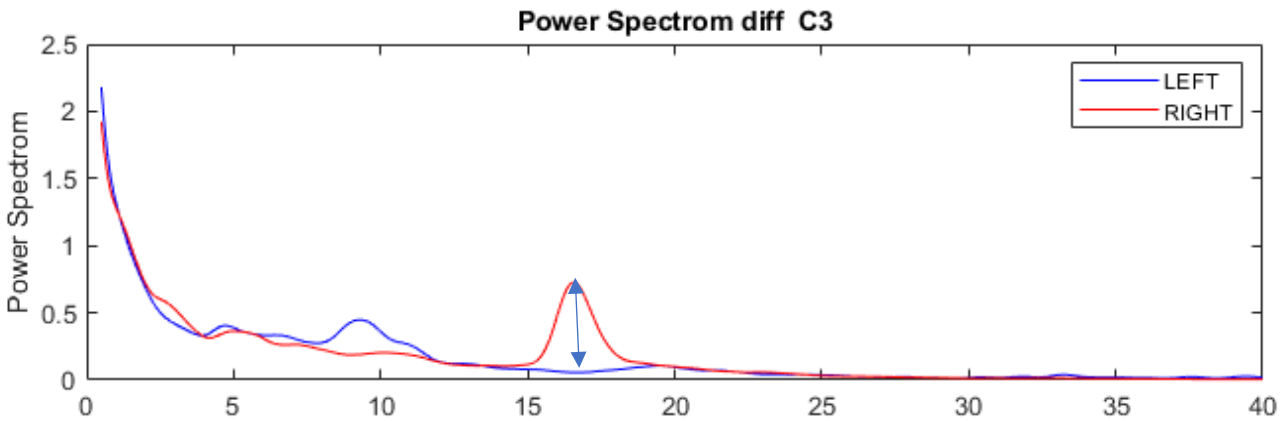


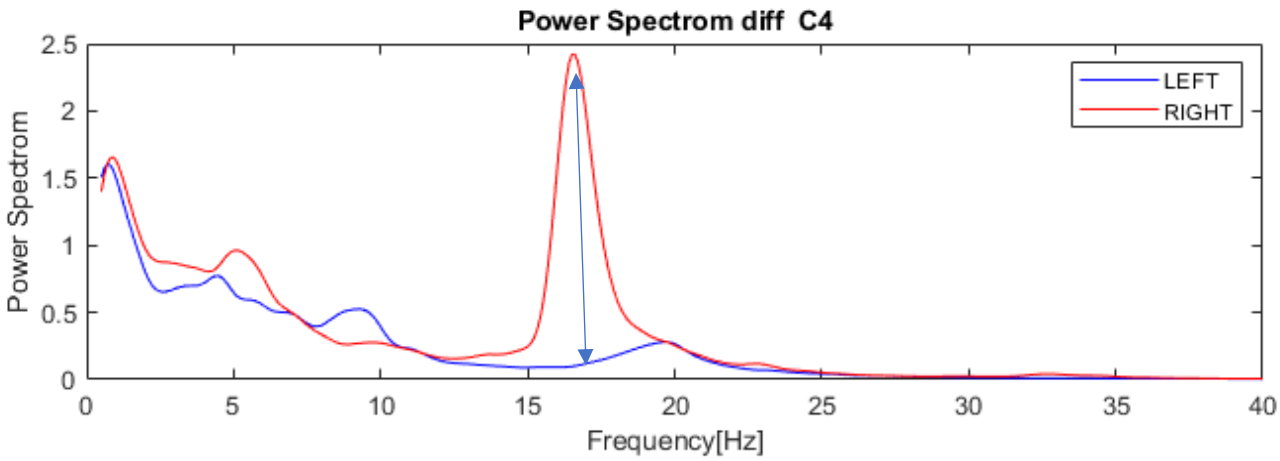

בהרבה מן ה *trials* בתחילת זמן דמיון התנועה ישנה עליה באמפליטודה.

גרף 2 – שתי אלקטרודות כאשר הנבדק התבקש לחשוב על יד ימין



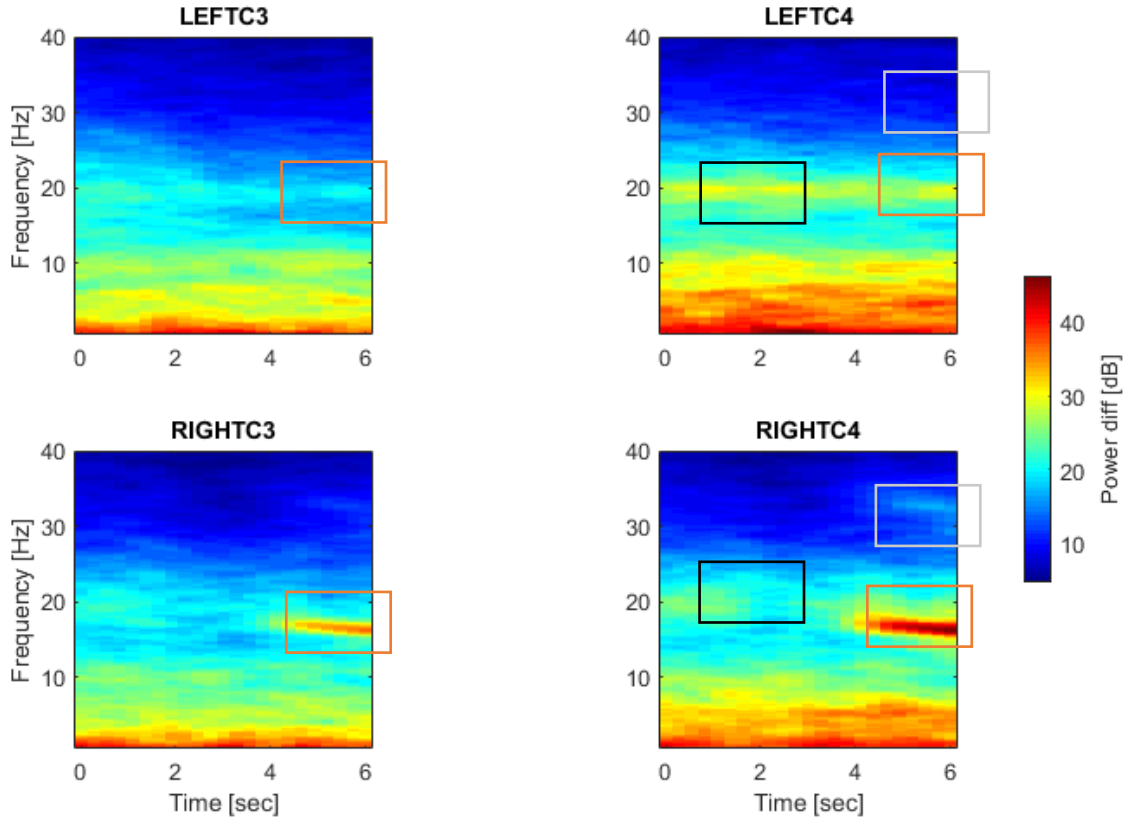

בהרבה מן ה *trials* תדירות ושונות האמפליטודה עלה לקראת סוף ה *trial*



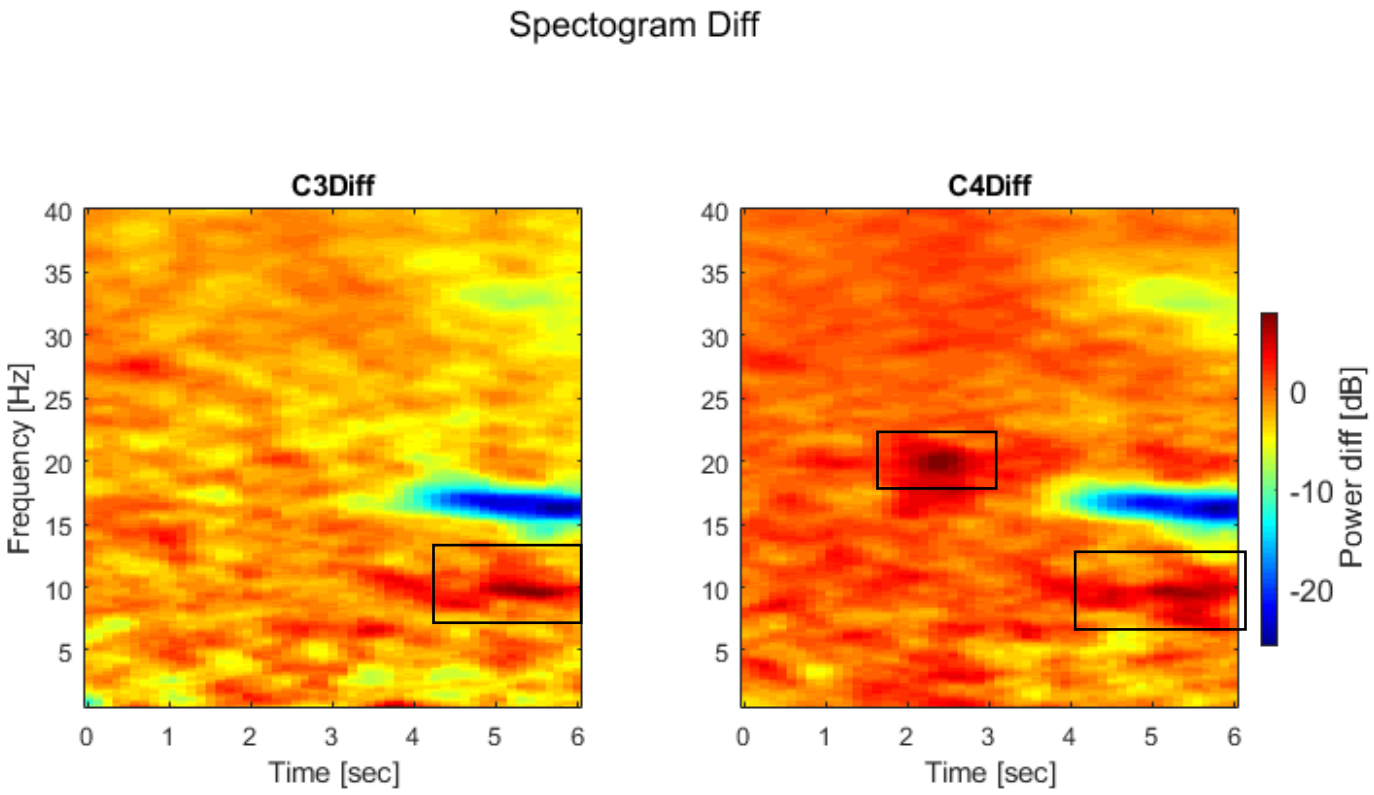



בגרף למעלה רואים את ה *power spectrum*, גרף זה מחושב רק על הזמן מאז התחיל הנבדק לדמיין את התנועה.

גרף 4 – ספקטוגרמה עבור כל אחד מן התנאים.



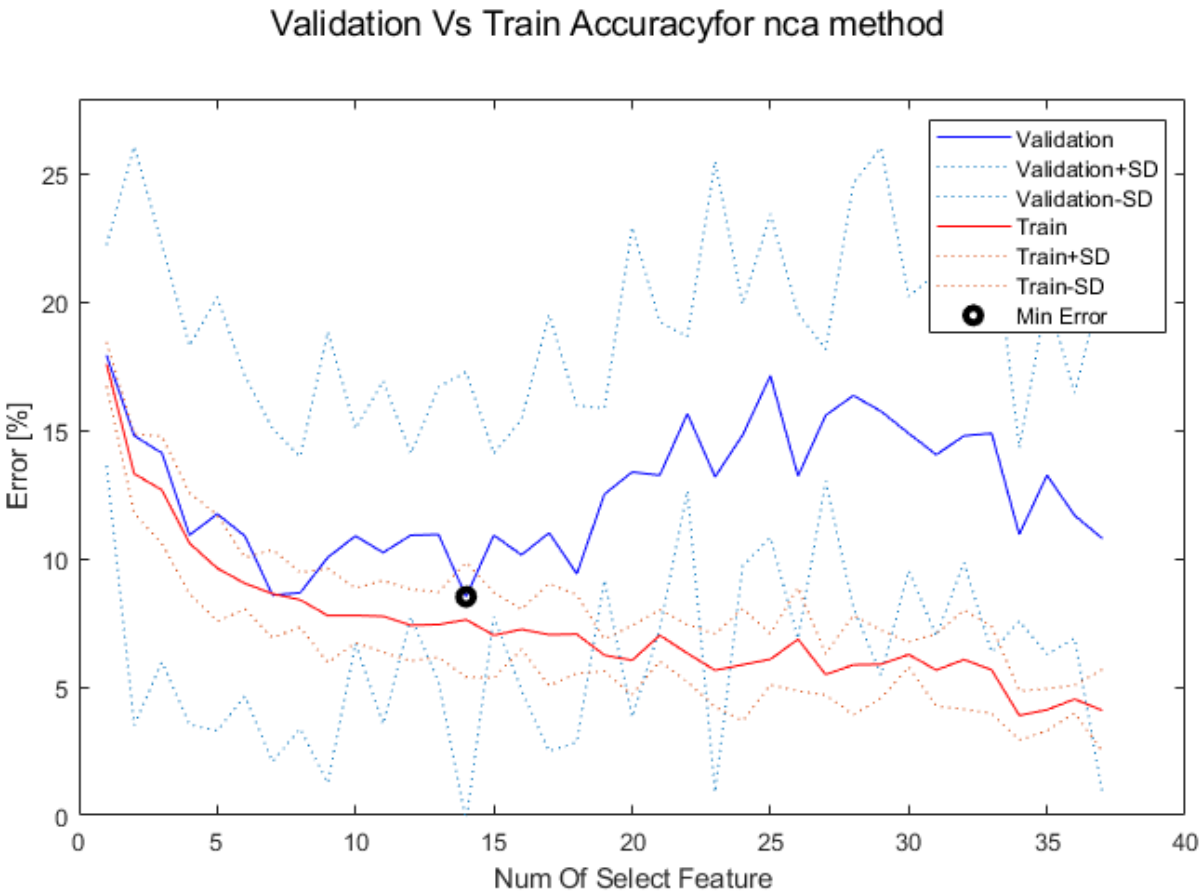

גרף 5 – ספקטוגרמת ההפרש בין ימין לשמאל עבור כל אלקטרודה.



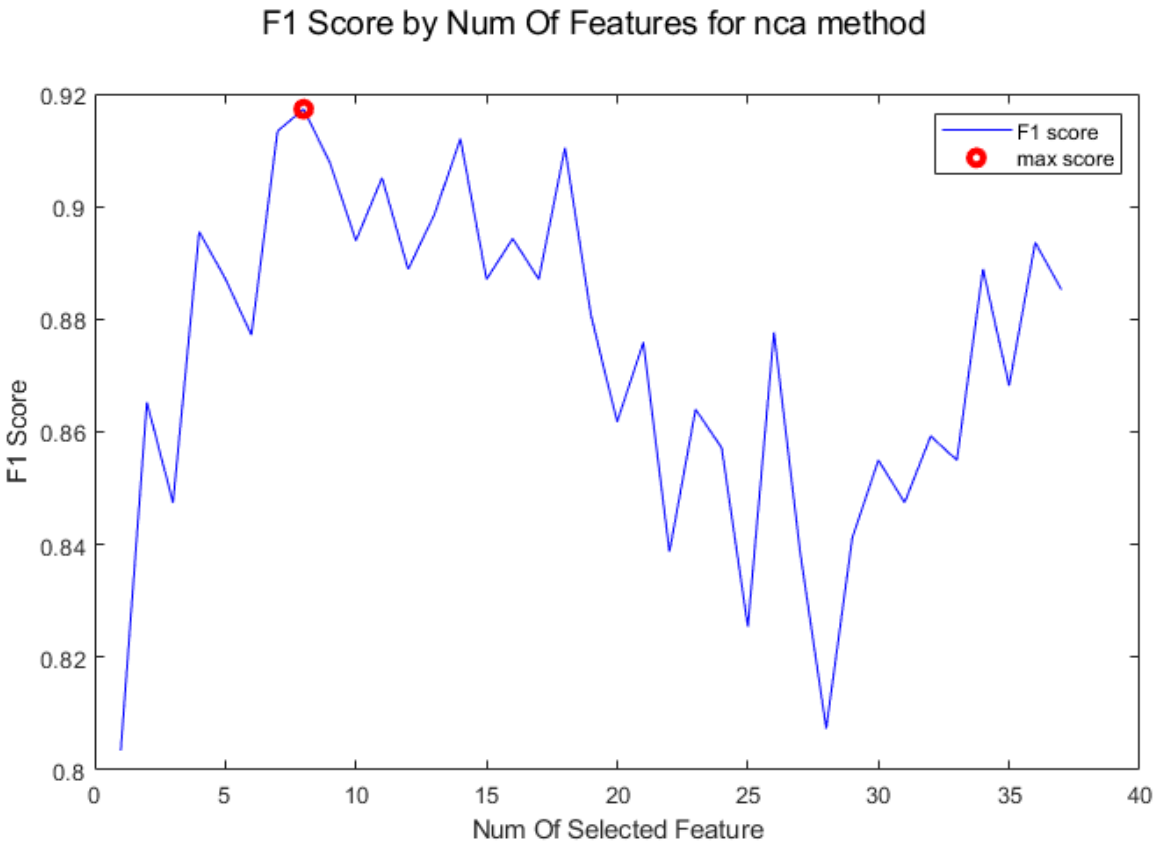
| | NCA_Order | KST_ORDER |
|---|---|---|
| 1 | Spectral Edge C4 | Relative Bandpower - C3 15Hz - 20Hz |
| 2 | Relative Bandpower - C3 17Hz - 21Hz | Bandpower - C4 32Hz - 36Hz |
| 3 | Relative Bandpower - C3 15Hz - 20Hz | Bandpower - C3 15Hz - 20Hz |
| 4 | Bandpower - C4 32Hz - 36Hz | Relative Bandpower - C4 32Hz - 36Hz |
| 5 | Relative Bandpower - C3 9Hz - 11Hz | Bandpower - C4 15Hz - 20Hz |
| 6 | Root Total Power C4 | Relative Bandpower - C4 15Hz - 20Hz |
| 7 | Threshold Pass Count 4mV C4 | Relative Bandpower - C3 9Hz - 11Hz |
| 8 | Spectral Entropy C3 | Spectral Moment C4 |
| 9 | Bandpower - C3 17Hz - 21Hz | Total Power C4 |
| 10 | Spectral Moment C4 | Relative Bandpower - C4 9Hz - 11Hz |
| 11 | Spectral Entropy C4 | Root Total Power C4 |
| 12 | Bandpower - C4 15Hz - 20Hz | Relative Bandpower - C3 32Hz - 36Hz |
| 13 | Relative Bandpower - C4 15Hz - 20Hz | Max Amplitude C3 |
| 14 | Bandpower - C3 15Hz - 20Hz | Relative Bandpower - C3 17Hz - 21Hz |
| 15 | Max Amplitude C3 | 4\muV Threshold Pass Count C4 |
| 16 | Relative Bandpower - C3 32Hz - 36Hz | Bandpower - C4 17Hz - 21Hz |
| 17 | Relative Bandpower - C4 32Hz - 36Hz | Intercept C4 |
| 18 | Min Amplitude C3 | Spectral Moment C3 |
| 19 | Total Power C4 | Slope C4 |
| 20 | Relative Bandpower - C4 9Hz - 11Hz | Spectral Edge C3 |
| 21 | Relative Bandpower - C4 17Hz - 21Hz | Spectral Edge C4 |
| 22 | Slope C4 | Bandpower - C3 32Hz - 36Hz |
| 23 | Threshold Pass Count 4mV C3 | Max Amplitude C4 |
| 24 | Bandpower - C4 17Hz - 21Hz | Min Amplitude C4 |
| 25 | Slope C3 | Bandpower - C3 9Hz - 11Hz |
| 26 | Bandpower - C3 32Hz - 36Hz | Bandpower - C4 9Hz - 11Hz |
| 27 | Min Amplitude C4 | Bandpower - C3 17Hz - 21Hz |
| 28 | Bandpower - C4 9Hz - 11Hz | Root Total Power C3 |
| 29 | Amplitude Diff: C3-C4 | Spectral Entropy C3 |
| 30 | Spectral Edge C3 | Total Power C3 |
| 31 | Intercept C4 | Slope C3 |
| 32 | Max Amplitude C4 | Relative Bandpower - C4 17Hz - 21Hz |
| 33 | Bandpower - C3 9Hz - 11Hz | Amplitude Diff: C3-C4 |
| 34 | Intercept C3 | Intercept C3 |
| 35 | Root Total Power C3 | Spectral Entropy C4 |
| 36 | Spectral Moment C3 | Min Amplitude C3 |
| 37 | Total Power C3 | 4\muV Threshold Pass Count C3 |

סדר המאפיינים אינו זהה  אך בשמנייה הראשונה ישנם 3 מאפיינים חופפים לשני הכלים

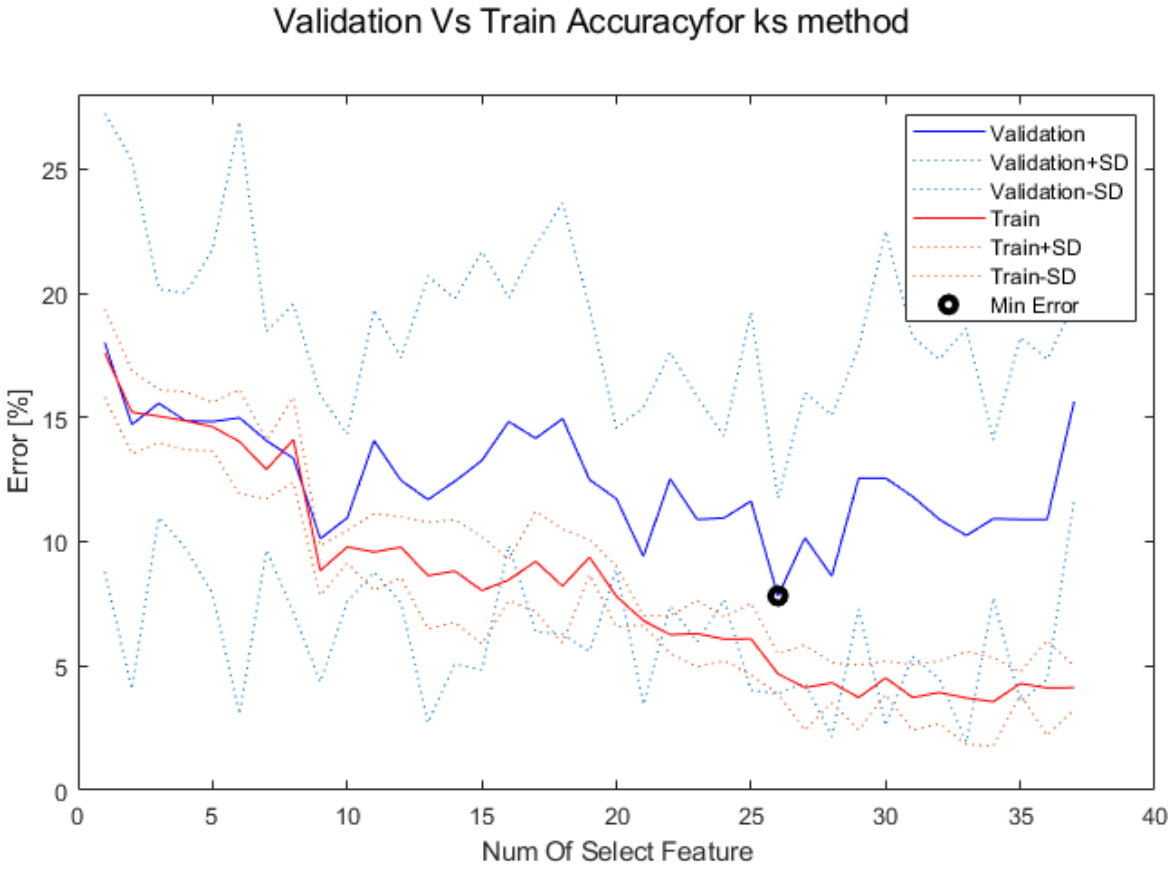
גרף 6 – השגיאה כתלות במספר המאפיינים עבור בחירת מאפיינים על ידי *NCA*



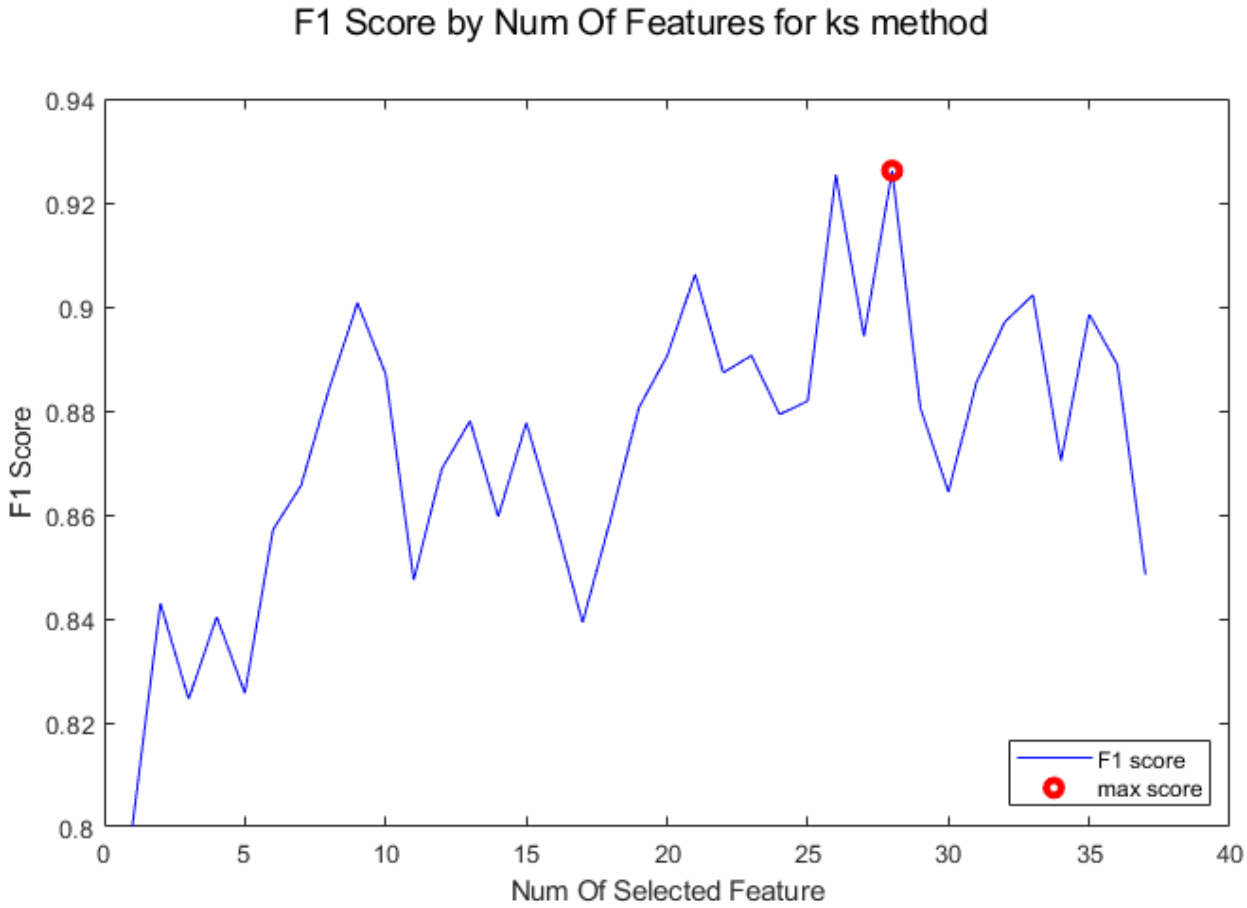

גרף 7 – ציון *f1* כתלות במספר המאפיינים עבור בחירת מאפיינים על ידי *NCA*



גרף 8 - השגיאה כתלות במספר המאפיינים עבור בחירת מאפיינים על ידי *KS*

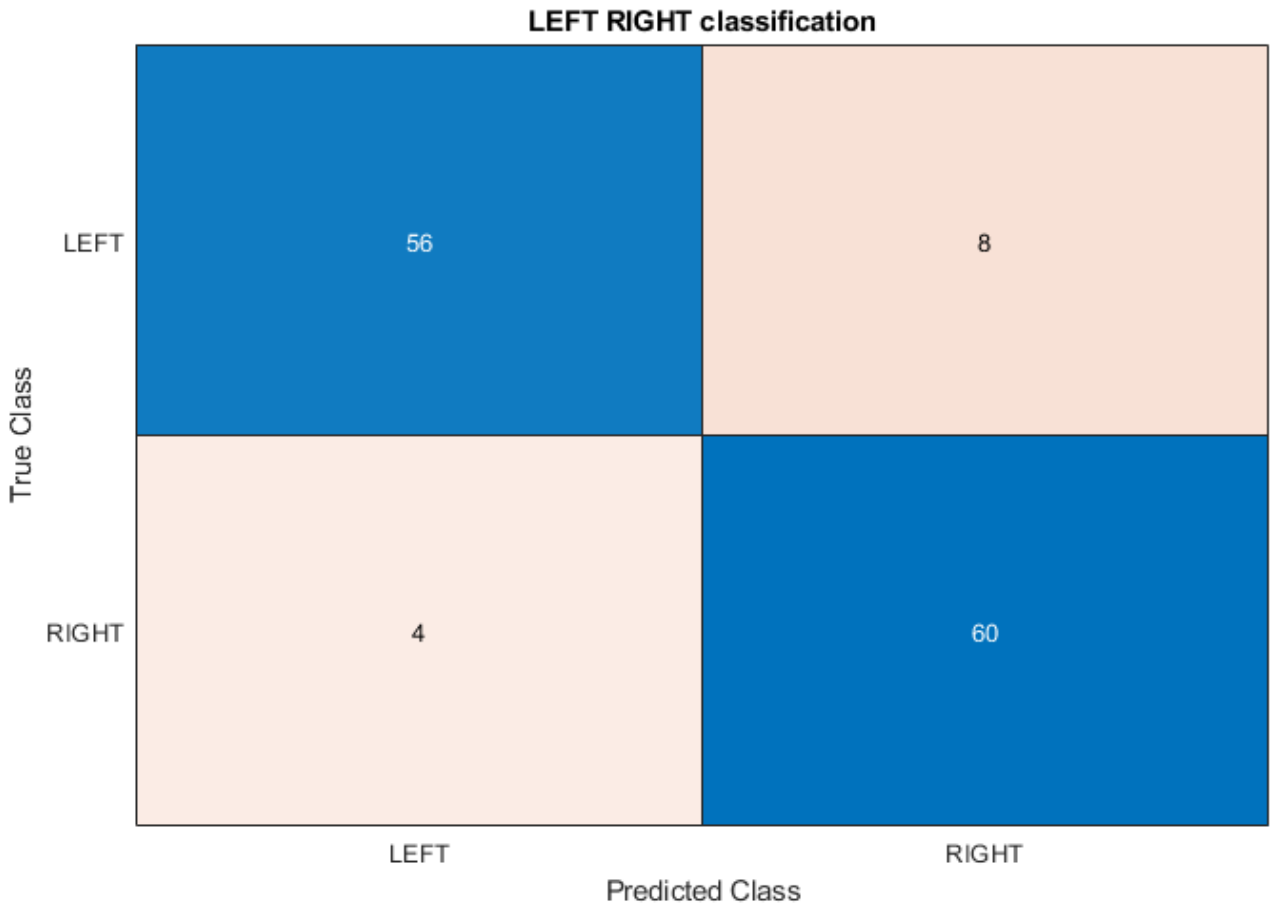



גרף 9 – ציון *f1* כתלות במספר המאפיינים עבור בחירת מאפיינים על ידי *KS*

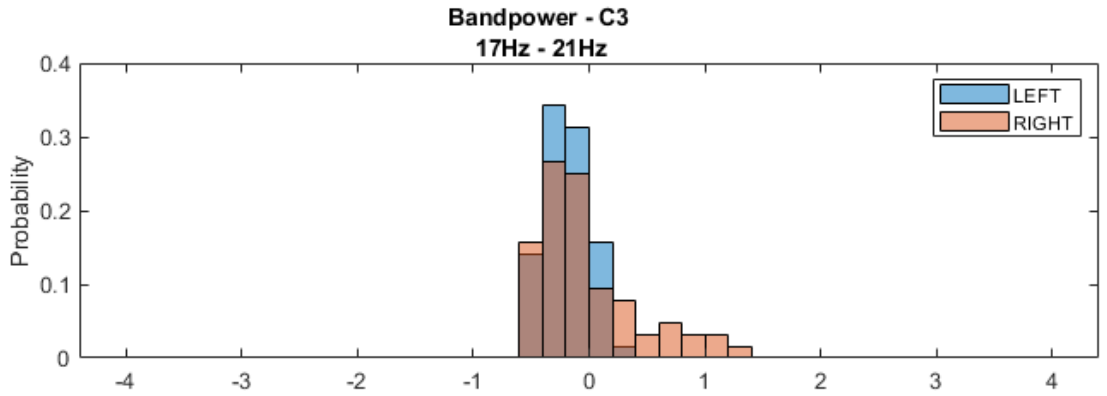


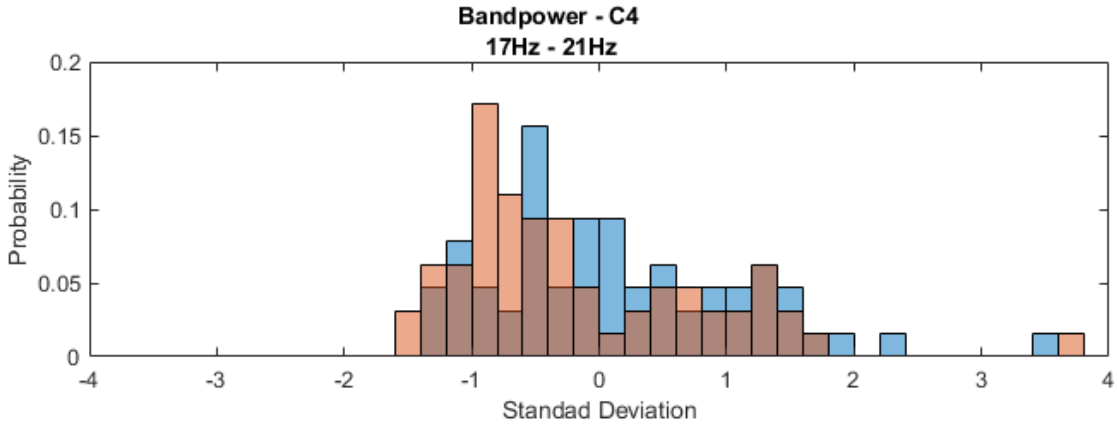
אפשר לראות כי ה *f1 score* בסביבת 8 מאפיינים הוא הגבוה ביותר לפי אלגוריתם *NCA* מספר קטן במעט ממינימום השגיאה הנראה בגרף המציג את אחוז השגיאה גם בו נראית השגיאה הנמוכה באיזור 15 מאפיינים ואילו לפי אלגוריתם *KST* ה *f1 score* ממשיך לעלות עד איזור 27 מאפיינים ונמצא בהתאמה לשגיאה המינימאלית.

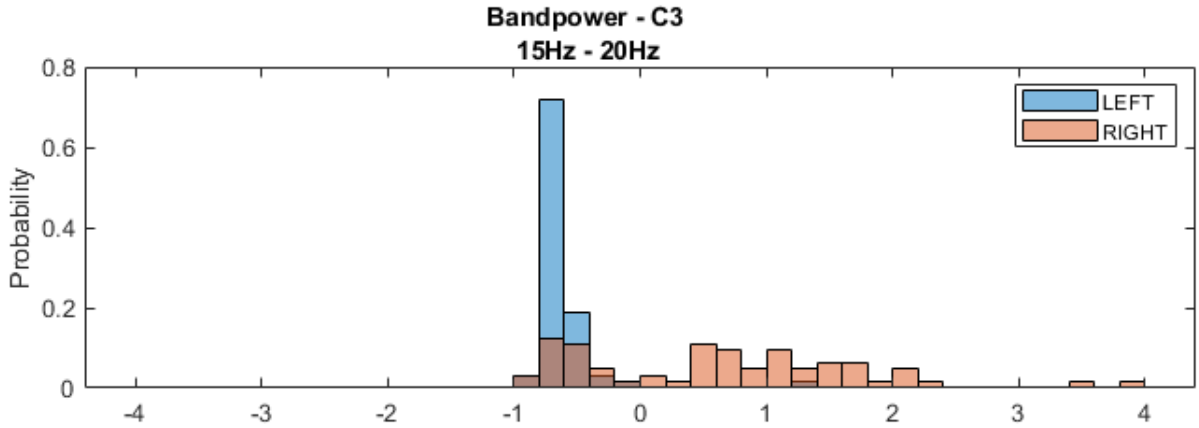
*Confusion matrix*

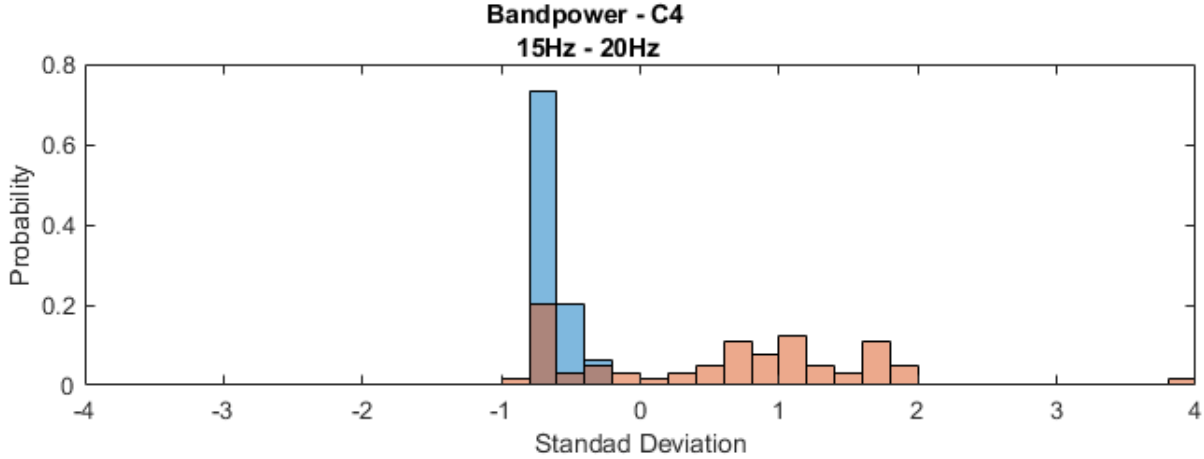


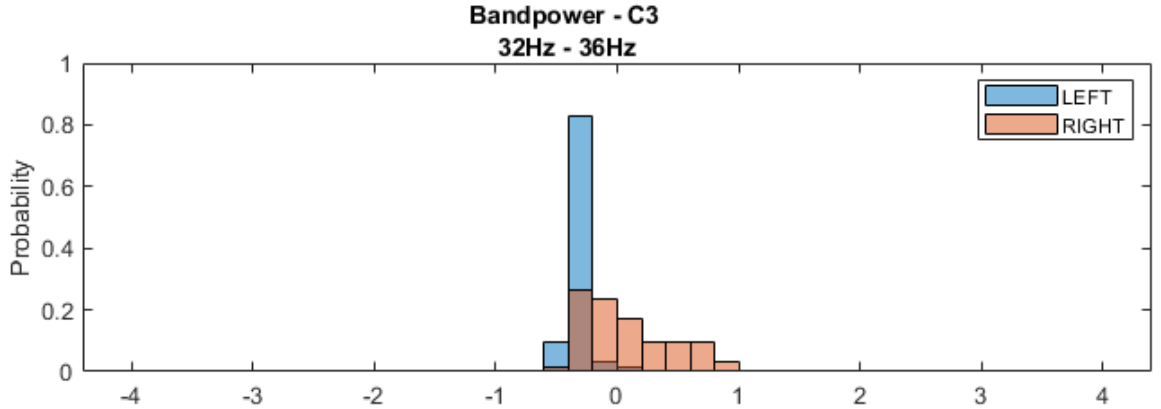
היסטוגרמות *Bandpower*

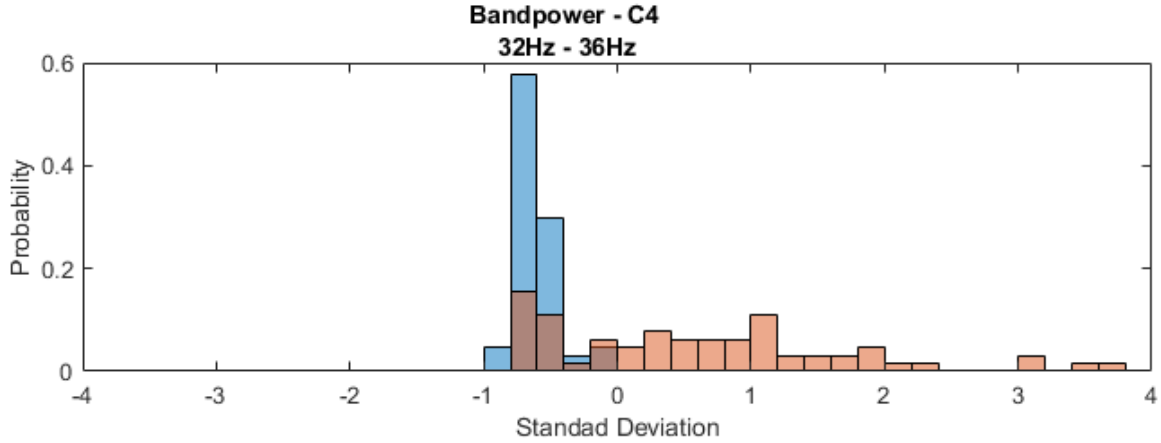


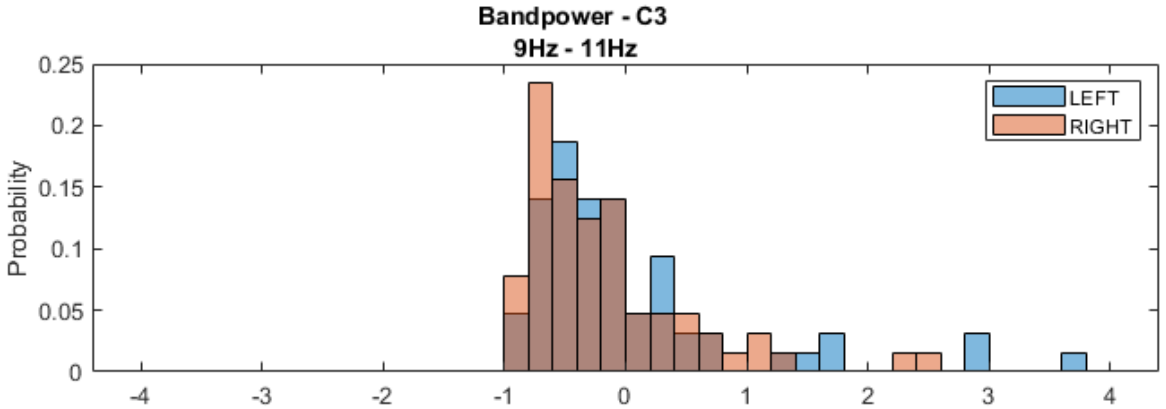



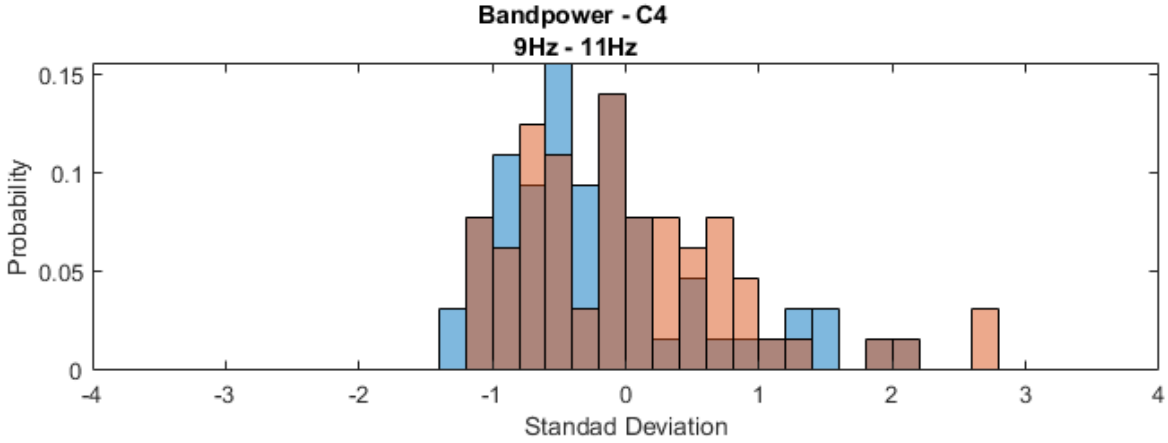




Bandpower - C3
32Hz - 36Hz
Probability
LEFT
RIGHT

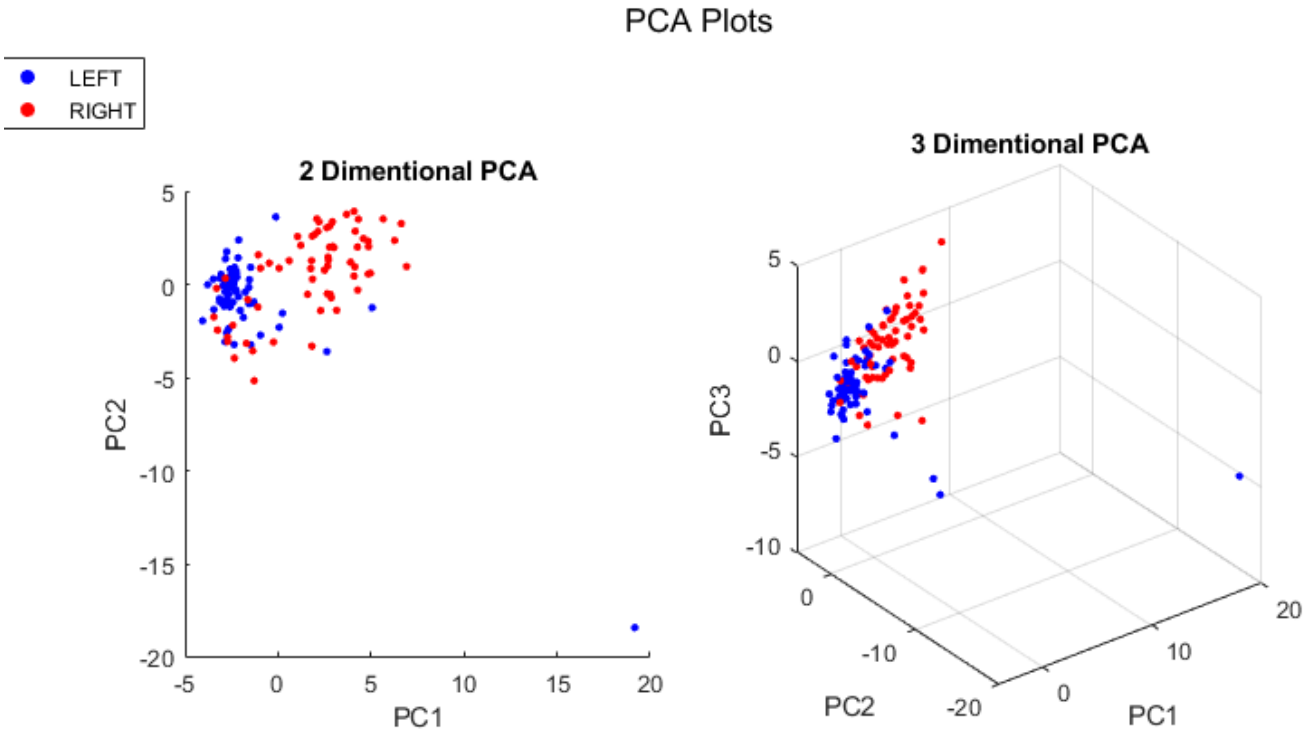

Bandpower - C4
32Hz - 36Hz
Probability
Standad Deviation


Bandpower - C3
9Hz - 11Hz
Probability
LEFT
RIGHT


Bandpower - C4
9Hz - 11Hz
Probability
Standad Deviation

ההיסטוגרמות מציגות את ההתפלגות של ערכי המאפיינים לפי כל קטגוריה ימין ושמאל. ההתפלגויות מנורמלות ומצגות כהסתברויות כתלות בסטיית התקן.

גרף 10 - *PCA*




תצאות הדיוק על סט הנתונים אותו קיבלנו נעו סביב 91% דיוק על סט האימות וכך גם על סט האימון. לבסוף התוצאות על נתוני המבחן – *test set* הינם 90.63%

**דיון ומסקנות:**

בעבודה זאת התבקשנו לנבא איזו יד דימיין הנבדק כי הוא מזיז על ידי נתוני EEG משתי אלקטרודות ובאמצעות מסווג LDA. על מנת לבצע זאת היינו צריכים לחלץ מאפיינים מתוך הנתונים ומתוכם לבחור את הטובים ביותר אשר יעזרו במטלת הסיווג. כבר במבט ראשון על הגרפים ניתן ליראות כי ישנם הבדלים יחסית בולטים בין שני התנאים ימין ושמאל גם באמפליטודת המתח ובעיקר ב power spectrum.

בגרפים המציגים את האות עצמו בולטת בעיקר העובדה כי אל אלקטרודה C4 מגיע מתח גבוה יותר וכי בזמן הדימיון המתח משתנה בתדירות גבוה ובעל שונות גבוהה. לכן חילצנו מאפיין של כמות הפעמים בהם נחצה סף של $4\mu V$ . סף זה נבחר לאחר בדיקת מספר ספים אחרים ומצאנו כי הוא הביא לאחוזי הדיוק הגבוהים יותר. על ידי אלגוריתם NCA מאפיין זה כמדד השביעי להפרדה בין ובמקום ה15 על ידי KST לכן אפשר להניח כי ישנו קשר בין דמיון התנועה לשונות האמפליטודה. למרות שישנם הבדלים גם בגובה האמפליטודה באיזור הזמן בו מתחיל הנבדק לדמיין ניראה כי המאפיינים הקשורים לאמפליטודה מינימאלית ומקסימאלית אינם מנבאים טוב את כיוון התנועה.

ההבדלים הבולטים ביותר הקופצים לעיין נמצאים בגרפים של ה power spectrum וה spectrogram בעוצמה בטווח התדרים HZ 15-20. השוני בטווח זה מופיע קצת בתחילת אך בעיקר לקראת סוף ה trial. תדרים אילו הנמצאים בטווח תדרי הבטא אשר בקורטקס המוטורי מקושרים לתנועה [9]. מאפיינים של bandpower באותם טווחי תדרים נבחרו על ידי שני הכלים לבחירת במקומות גבוהים. על ידי NCA במקומות 2-4 ועל ידי KST במקומות 2 ו 3 ובנוסף ההיסטוגרמות של אותם טווחי תדרים מראים כי הבדלים יחסית גדולים בהתפלגויות של ה trials המסומנים כימין וכשמאל. אילו מאששים כי ההבדל אשר ניתן לראות בעניים מתוך הגרפים אכן קשור לדמיון התנועה.

עם זאת הניבוי אינו מדויק ותוצאת הדיוק אליה הגענו היא 90.63% כלומר המאפיינים אשר חילצנו לא הצליחו לחזות ולהסביר את כל ה trials. אפשר ליחס את זה למאפיינים לא מושלמים ואולי ניתן למצוא מאפיינים נוספים על מנת לחסות על הפער. אך ישנם גם חוקרים הטוענים כי המידע המגיע מן הקורטקס המוטורי אינו מספיק על מנת לחזות תנועה וכי ישנם איזורים אחרים במוח אשר יכולים לעזור בחיזוי דמיון תנועה, כמו ה posterior parietal cortex המשחק תפקיד חשוב בתכנון תנועה [4].

למרות שהניבוי אינו מדוייק במאת האחוזים, הדיוק אותו קיבלנו על ה test set כמעט זהה לזה שקיבלנו על ה validation set כלומר המודל אינו סבל מ overfitting. אנו מיחסים זאת לכך שהשתמשנו מלבד ב accuracy גם f1 score אשר מאפשר לקבל הערכה מדוייקת יותר של המודל. בנוסף למדד f1 ואולי אף חשוב מכך השתמשנו בכלים לבחירת מאפיינים אשר הביאו לשימוש במעט מאפיינים במודל (8 מתוך 37). בגרף 6 אפשר ליראות כי לאחר שמתקבלת השגיאה הנמוכה ביותר השגיאה על סט האימון ממשיכה לרדת בעוד השגיאה על סט האימות נשארת קבועה ואף עולה, מצב כזה ממחיש את הסכנה ב overfit כאשר לא משתמשים בסט אימות. אנו חושבים כי השגיאה של המודל שלנו הולכת וקטנה אך למעשה אם נבדוק אותו על נתונים חדשים הוא יהיה מדוייק פחות. ציוני ה f1 בגרף 7 לאחר 8 מאפיינים מחזקים מאוד את המסקנה הזאת.

כאשר אנו משווים בין שני הכלים NCA ו KST, NCA מראה תוצאות יותר טובות הן מבחינת accuracy והן מבחינת f1 score אפשר לראות זאת בגרפים 8 ו 9. עם זאת ניתן גם לשים לב כי לאחר כ 8 מאפיינים KST מגיע ל0.9 בציון ה f1 ושגיאה קרובה ל 10% לא רחוק מ nca. אך הדבר הבולט ביותר שמראה כי KST אינו מצליח למצות את המאפיינים הוא שככול שנוספים מאפיינים למודל ציון ה f1 עולה והשגיאה יורדים והוא מגיעה לשיא רק באיזור 27. כלומר הוא אינו מצליח למצוא את המאפיינים הטובים ביותר בהתחלה.

תוצאות כאלו אינן מפתיעות מכיוון שהאלגוריתם בו השתמשנו ב KST בדק כל מאפיין לבדו מבלי לבדוק כמה הוא מוסיף על מאפיינים קודמים שכבר נבחרו, כך עושה למשל האלגוריתם המשתמש ב NCA. אפשר לצפות לתוצאות טובות יותר מאלגוריתם דומה המוצא מאפיינים אשר יוסיפו אחוזי דיוק למאפיינים שכבר נבחרו. בנוסף להיותו של האלגוריתם נאיבי, מבחן קולמוגורוב – סמירנוב לשני מדגמים בודק האם שני המדגמים הגיעו מהתפלגות זהה בעוד אלגוריתם הסיווג LDA משתמש במרחק בין הממוצעים ובשונות של המדגמים דבר שאומנם יכול להלום את KST אך לא בצורה גורפת כך שיכול להיות ש KST לבדו אינו מתאים כאשר עובדים עם מסווג מסוג LDA.

לסיכום, מכיון שהגענו לאחוז דיוק של 90.63% אפשר לומר שניבוי ברמה גבוהה של דמיון תנועה על ידי נתונים משתי האלקטרודות C3 ו C4 ושימוש במסווג ליניארי מסוג LDA הוא אפשרי ביותר. ניראה כי המשך המאמץ בתחום יאפשר לקחת כלים אילה בחשבון כאשר אנו באים לתכנן BCI על מנת לעזור לאנשים בעלי מוגבלויות מוטוריות. בנוסף כאשר אנו באים לעסוק במשימות ניבוי כאלו כדאי להשתמש במדדים אשר יעזרו להעריך את המודל בצורה מדוייקת ובכלים אשר יעזרו לבחור את המאפיינים הטובים ביותר מתוך כמות הנתונים הגדולה שלנו.

## ביבליוגרפיה


[1] M. O. Miah, M. M. Rahman, R. Muhammod, and D. M. Farid, “Prediction of Motor Imagery Tasks from Multi-Channel EEG Data for Brain-Computer Interface Applications,” *bioRxiv*, 2020.

[2] J. K. Feng *et al.*, “An Optimized Channel Selection Method Based on Multifrequency CSP-Rank for Motor Imagery-Based BCI System,” *Comput. Intell. Neurosci.*, vol. 2019, 2019.

[3] S. Aggarwal and N. Chugh, “Signal processing techniques for motor imagery brain computer interface: A review,” *Array*, vol. 1–2, no. August, p. 100003, 2019.

[4] Y. Wang and S. Makeig, “Predicting intended movement direction using EEG from human posterior parietal cortex,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5638 LNAI, no. July, pp. 437–446, 2009.

[5] D. M. Hawkins, “The Problem of Overfitting,” *J. Chem. Inf. Comput. Sci.*, vol. 44, no. 1, pp. 1–12, 2004.

[6] G. Afendras and M. Markatou, “Optimality of training/test size and resampling effectiveness in cross-validation,” *J. Stat. Plan. Inference*, vol. 199, pp. 286–301, 2019.

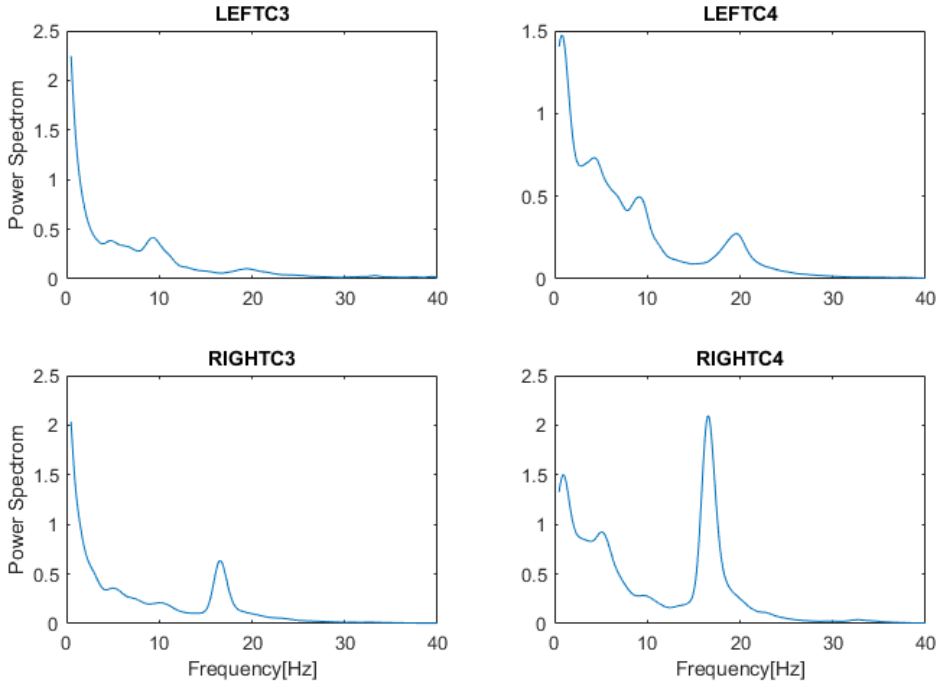
[7] A. Ivanov and G. Riccardi, “KOLMOGOROV-SMIRNOV TEST FOR FEATURE SELECTION IN EMOTION RECOGNITION FROM SPEECH,” pp. 5125–5128, 2012.

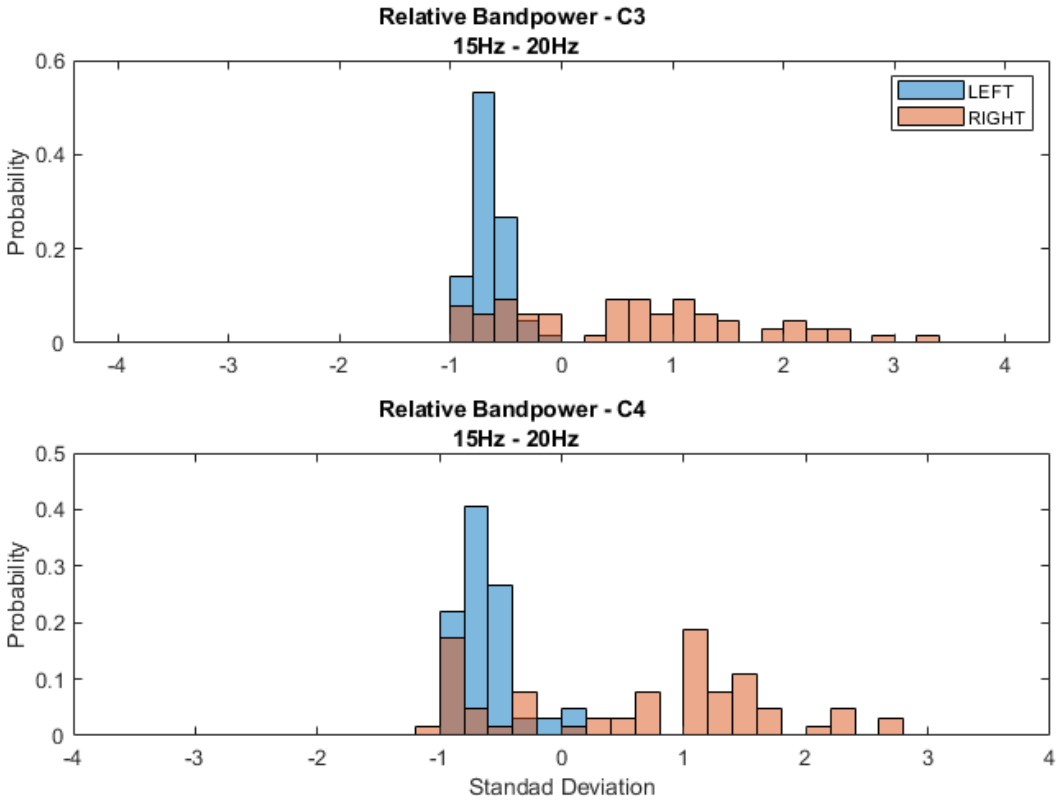
[8] Z. C. Lipton, C. Elkan, and B. Naryanaswamy, “Optimal thresholding of classifiers to maximize F1 measure,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8725 LNAI, no. PART 2, pp. 225–239, 2014.

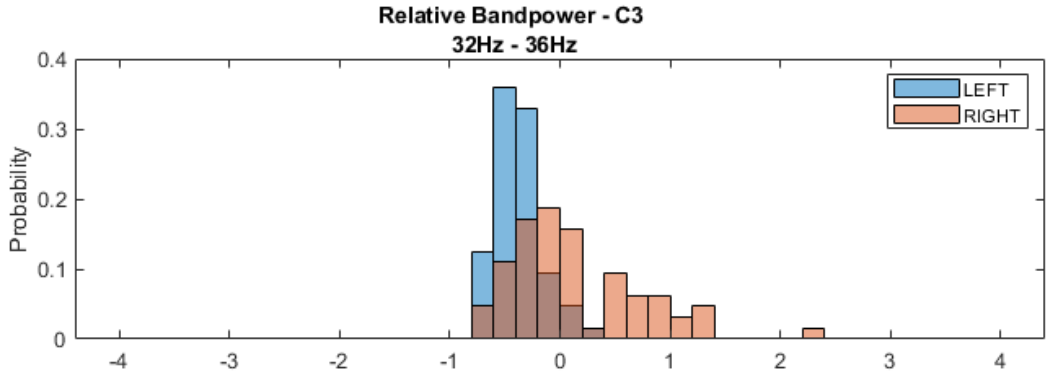
[9] M. T. Jurkiewicz, W. C. Gaetz, A. C. Bostan, and D. Cheyne, “Post-movement beta rebound is generated in motor cortex: Evidence from neuromagnetic recordings,” *Neuroimage*, vol. 32, no. 3, pp. 1281–1289, 2006.

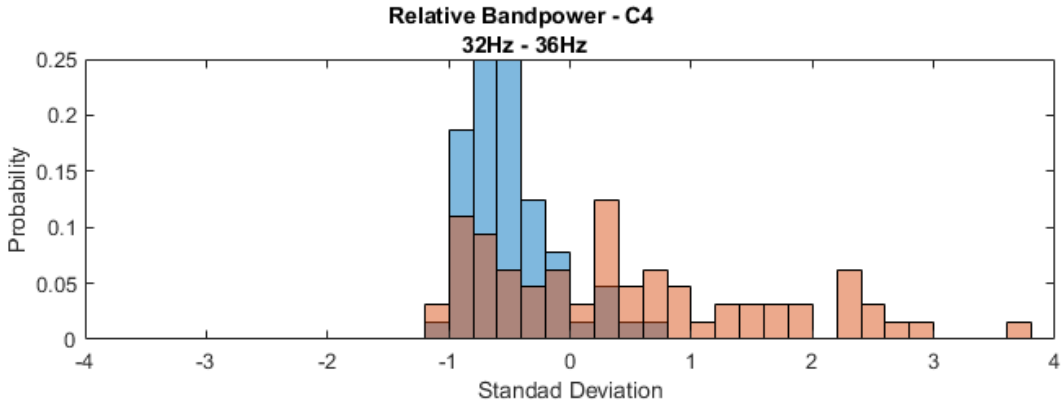
**נספחים**

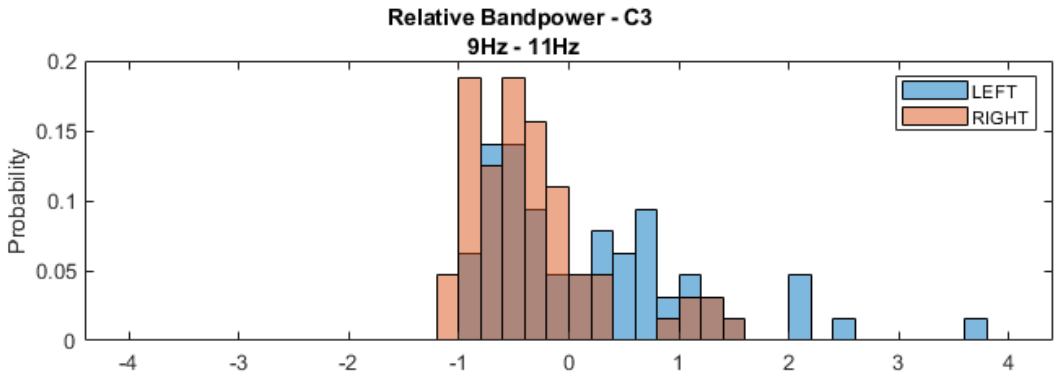
Pwelch power spectrum עבור כל אחד מהתנאים

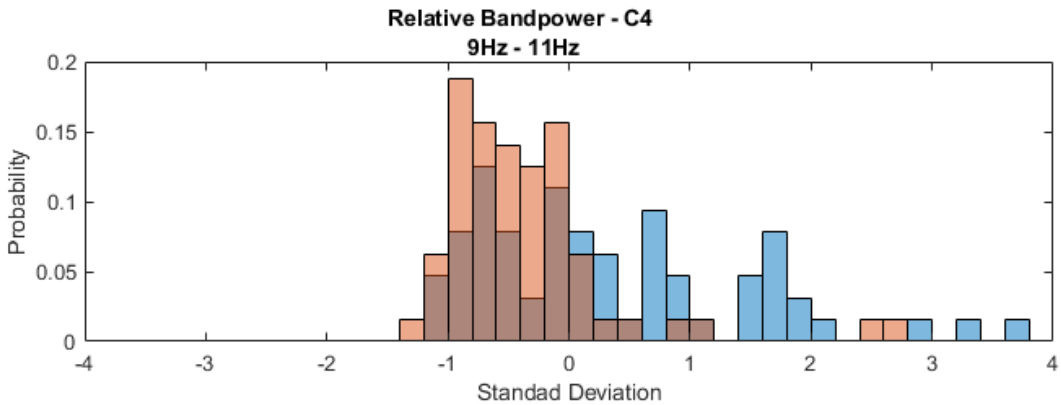


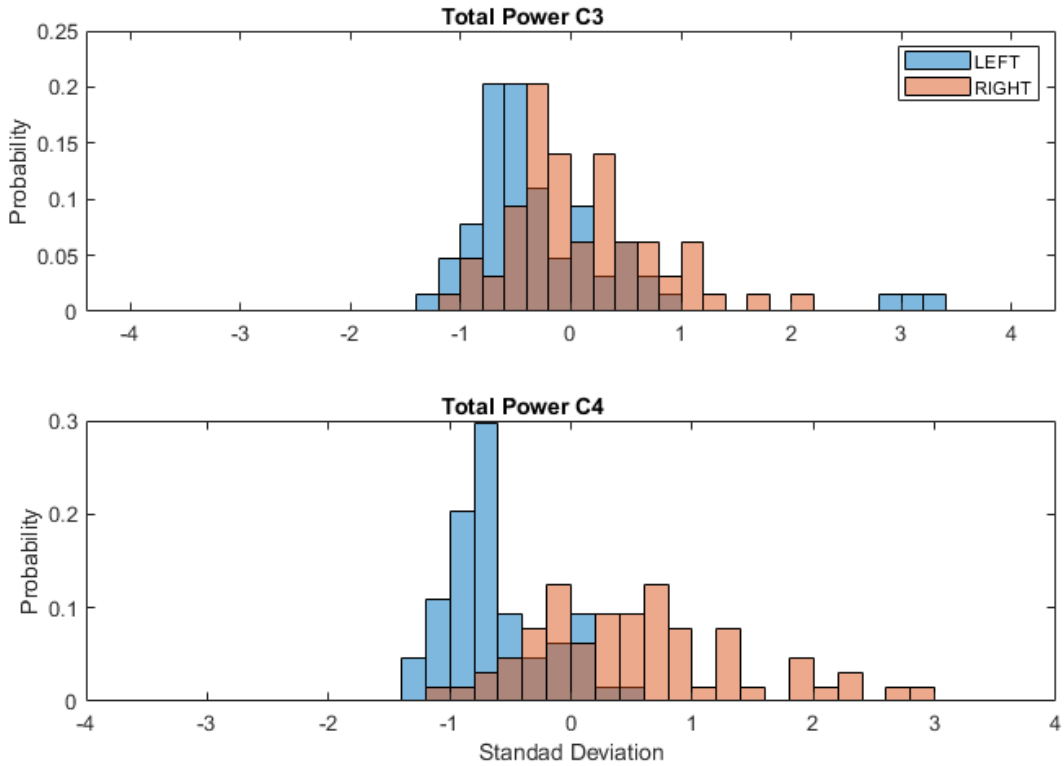

Relative Bandpower - C3
32Hz - 36Hz
Probability
LEFT
RIGHT

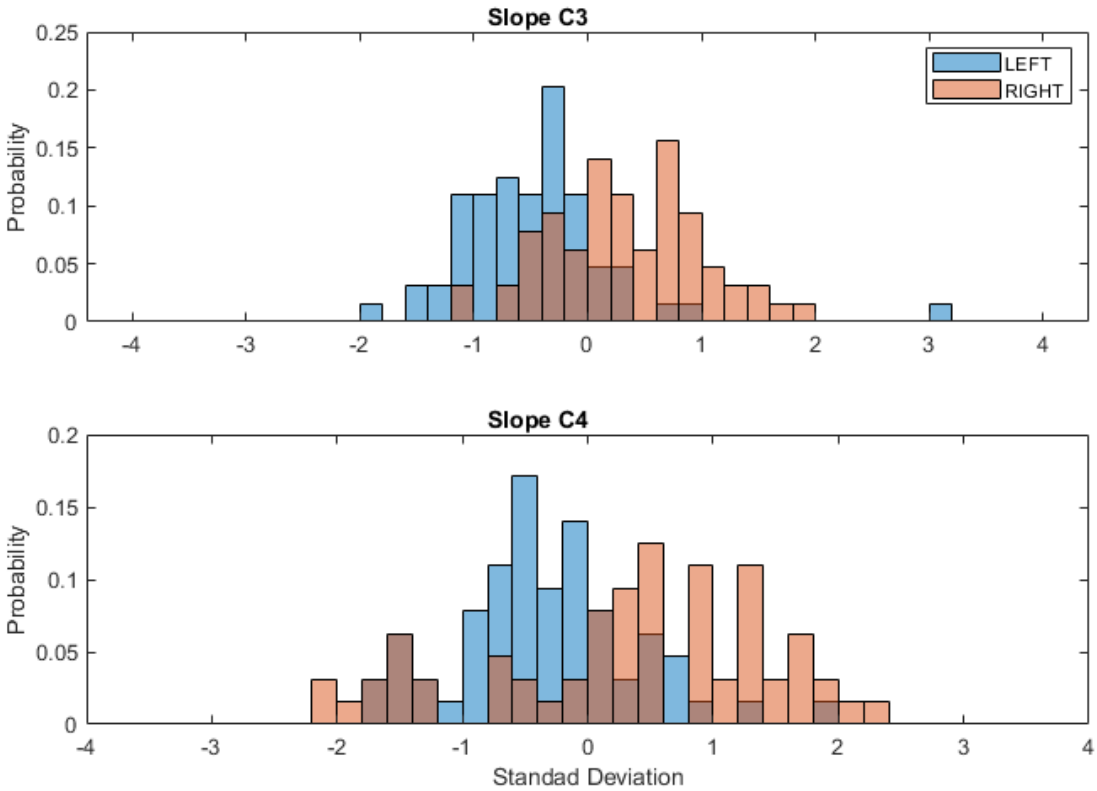

Relative Bandpower - C4
32Hz - 36Hz
Probability
Standad Deviation

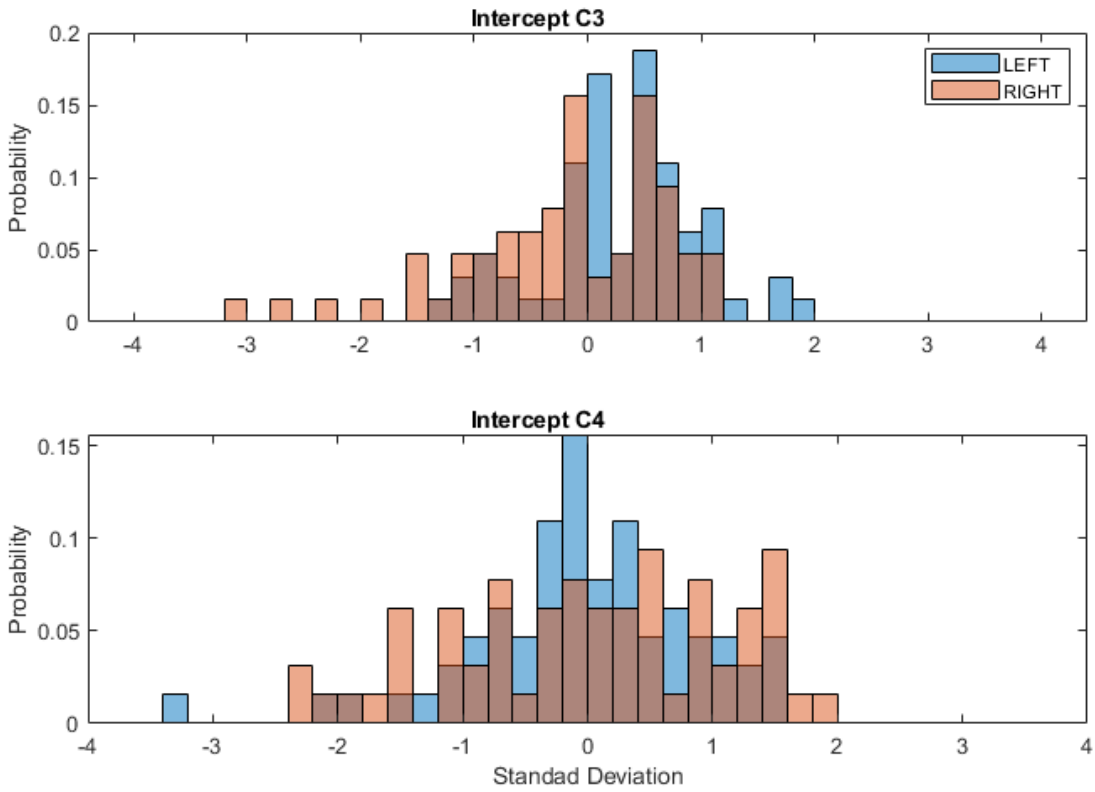

Relative Bandpower - C3
9Hz - 11Hz
Probability
LEFT
RIGHT

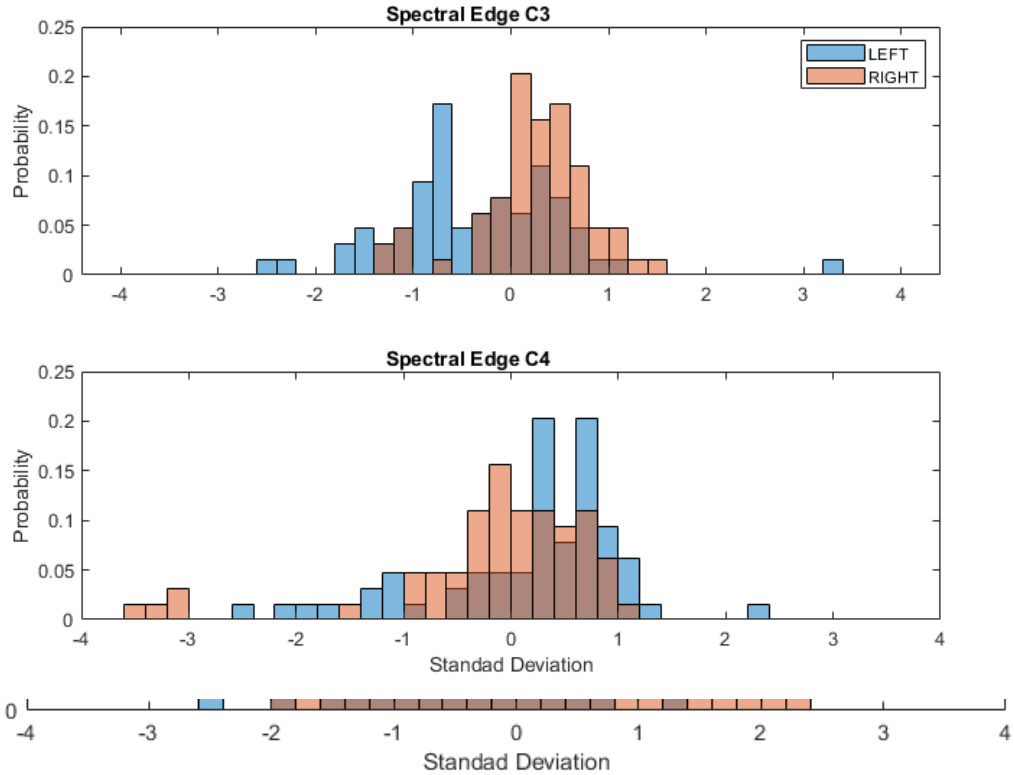

Relative Bandpower - C4
9Hz - 11Hz
Probability
Standad Deviation

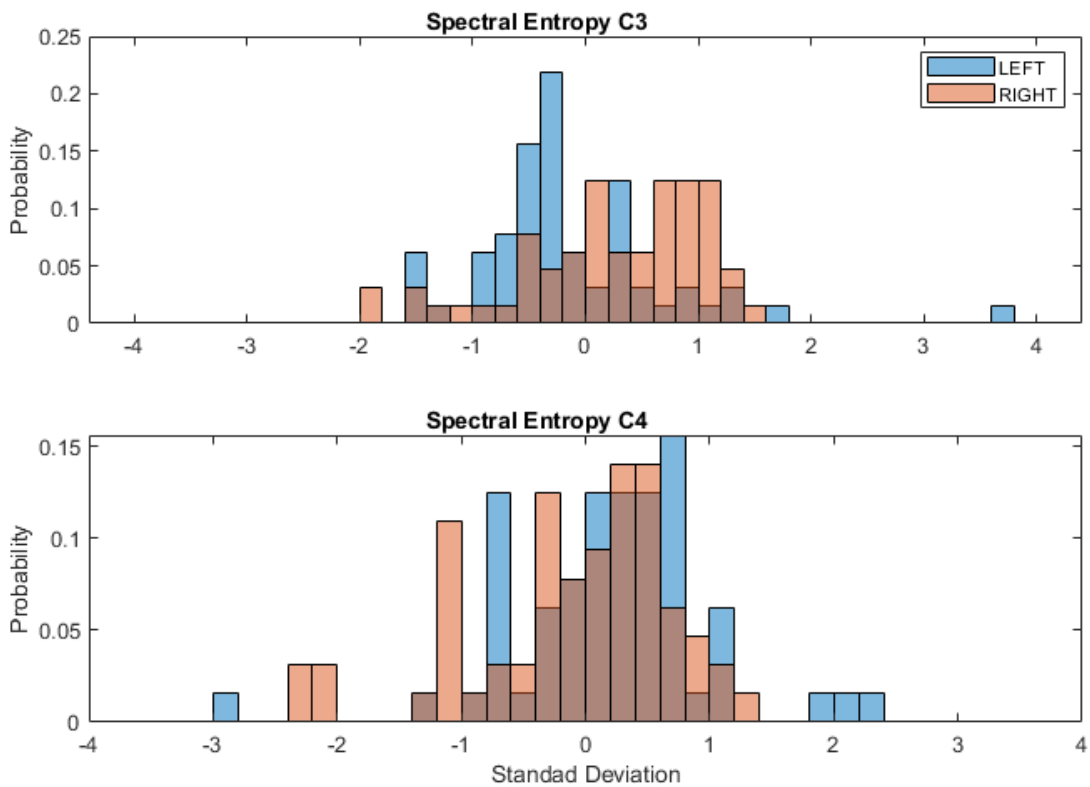
**Relative Bandpower - C3**
**17Hz - 21Hz**

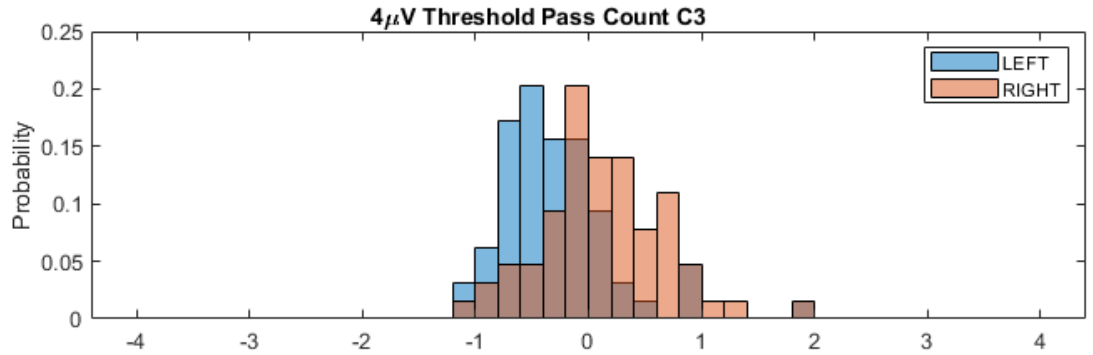


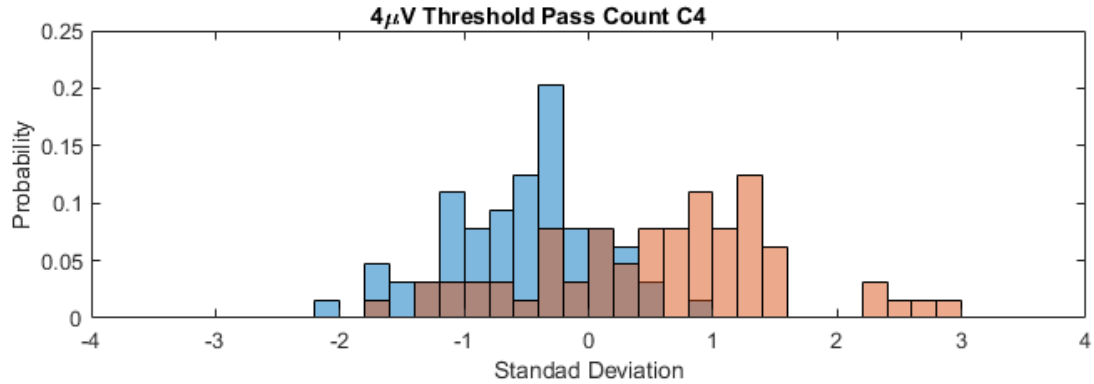

Slope C3
LEFT
RIGHT
Probability
Slope C4
Probability
Standad Deviation

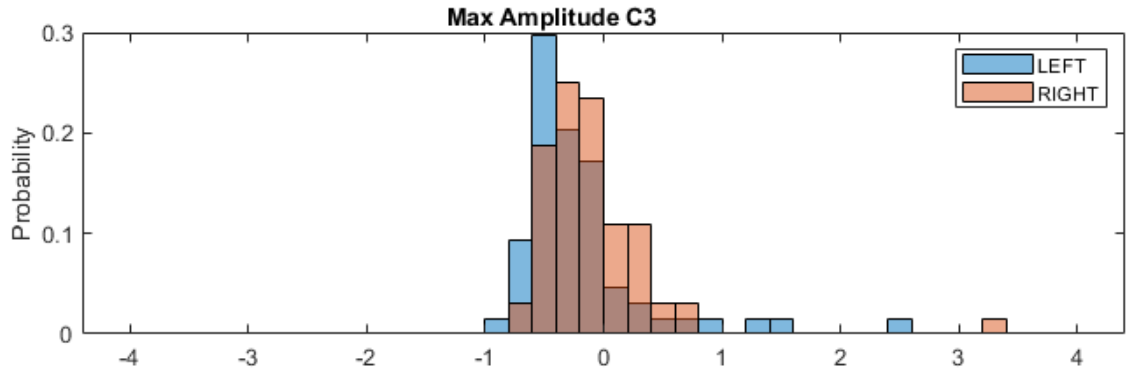

Intercept C3
LEFT
RIGHT
Probability
Intercept C4
Probability
Standad Deviation

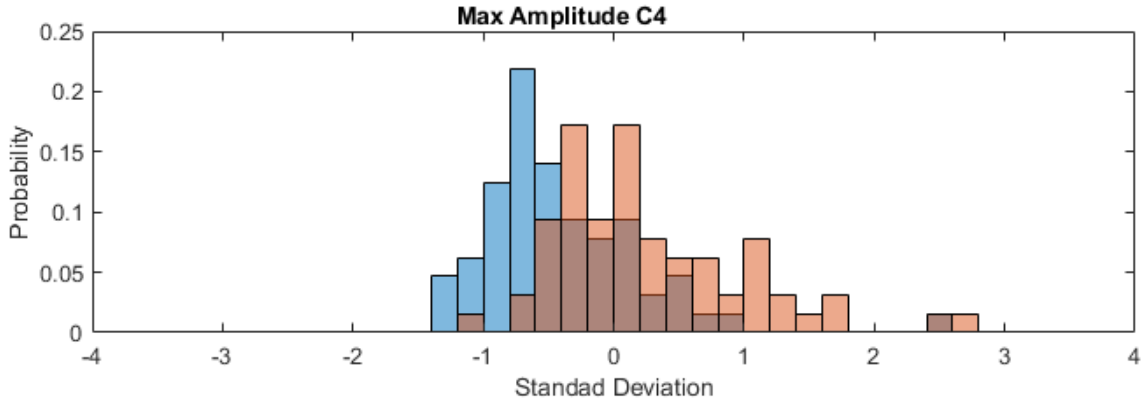

Spectral Edge C3
LEFT
RIGHT
Probability
Spectral Edge C4
Probability
Standad Deviation
Standad Deviation

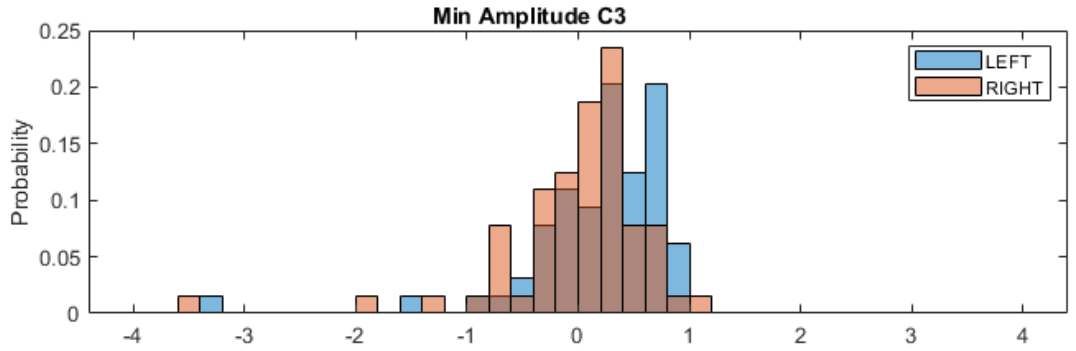

Spectral Entropy C3
LEFT
RIGHT
Probability
Spectral Entropy C4
Probability
Standad Deviation

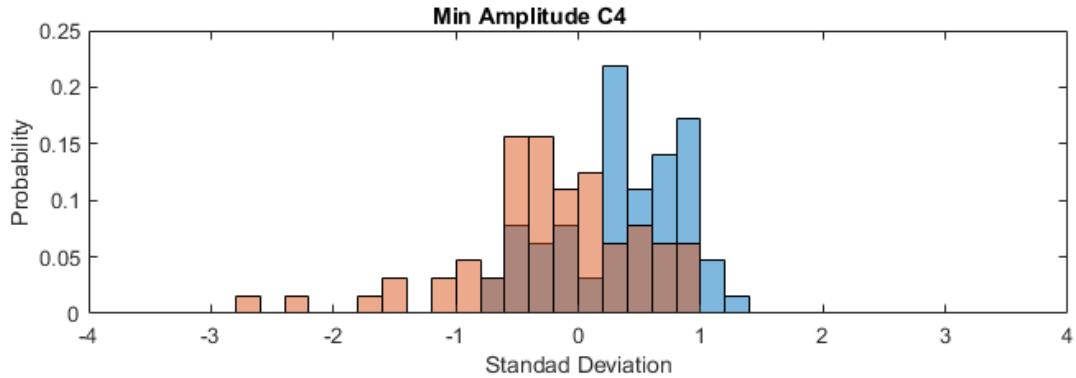

4μV Threshold Pass Count C3
LEFT
RIGHT
Probability
0.25
0.2
0.15
0.1
0.05
0
-4
-3
-2
-1
0
1
2
3
4

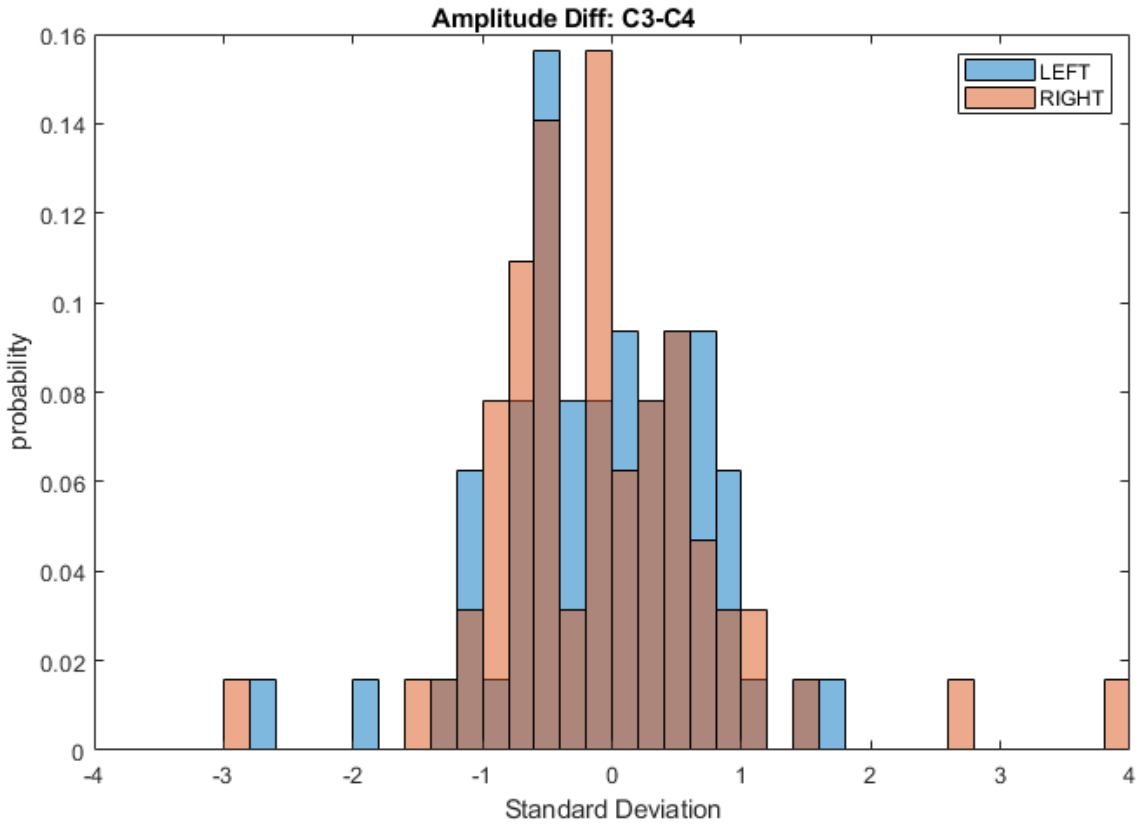

4μV Threshold Pass Count C4
Probability
0.25
0.2
0.15
0.1
0.05
0
-4
-3
-2
-1
0
1
2
3
4
Standad Deviation


Max Amplitude C3
LEFT
RIGHT
Probability
0.3
0.2
0.1
0
-4
-3
-2
-1
0
1
2
3
4


Max Amplitude C4
Probability
0.25
0.2
0.15
0.1
0.05
0
-4
-3
-2
-1
0
1
2
3
4
Standad Deviation


Min Amplitude C3
Probability
LEFT
RIGHT
0.25
0.2
0.15
0.1
0.05
0
-4
-3
-2
-1
0
1
2
3
4


Min Amplitude C4
Probability
0.25
0.2
0.15
0.1
0.05
0
-4
-3
-2
-1
0
1
2
3
4
Standad Deviation


Amplitude Diff: C3-C4
probability
LEFT
RIGHT
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
-4
-3
-2
-1
0
1
2
3
4
Standard Deviation