

עבודה מסכמת ניתוח נתונים

מבוא:

מטלה זו עוסקת בניתוח נתוני EEG (Electroencephalography), רישום פעילות חשמלית של המוח) אשר נאספו במהלך ניסוי של דמיון תנועתי (Motor Imagery) של יד ימין ויד שמאל. נתונים אלו נאספו משני ערוצי EEG מרכזיים - C3 ו-C4, אשר תואמים לאזור השליטה בידיים בקורטקס המוחי¹. מטרתה של מטלה זו היא ביצוע של ניתוח של הנתונים, חילוץ תכונות משמעותיות מתוכם ובהתבסס על תכונות אלו בניית אלגוריתם למידת מכונה אשר יבצע סיווג של כל ניסיון (trial) לפי תוויתו (ימין או שמאל). תהליך זה משלב שלבים של ויזואליזציה, ניתוח ספקטרלי, הפקת תכונות, הפחתת ממדים (PCA) ולבסוף סיווג. המטלה מתבססת על עקרונות וכלים אשר נלמדו בקורס יחד עם שיפור הביצועים לצורך השגת רמת דיוק גבוהה ככל הניתן על מערך האימון ומערך המבחן.

שאלות המטלה:

שאלה 1- טעינת הנתונים, יצירת גרפים של 20 ניסיונות אקראיים מכל אחת משתי הקטגוריות (יד ימין ויד שמאל) יחד עם הצגת שני הערוצים (C3 ו-C4) בכל ניסיון בכדי לאבחן הבדלים ויזואליים ראשוניים בין הקבוצות.

שאלה 2- חישוב ספקטרום כוח ממוצע באמצעות שיטת Welch לכל הקטגוריות והערוצים יחד עם בדיקת תחומי תדר וזמן אשר מסייעים להפרדה בין הקבוצות. בניית ספקטוגרמות במטרה להשוות בין הקטגוריות (ביחס ל-baseline) כך שיהיה ניתן לזהות את טווחי הזמן והתדרים האינפורמטיביים בכל אחד מהמצבים.

שאלה 3- חילוץ תכונות כמותיות מאזורים בתדר ובזמן שנמצאו כאינפורמטיביים והוספת שתי תכונות נוספות לצורך שיפור הסיווג. בניית היסטוגרמות של התפלגות התכונות וביצוע הפחתת ממדים (PCA) עם הצגה גרפית הכוללת בתוכה הפרדה לינארית בין הקטגוריות.

שאלה 4- ביצוע אימון מסוג LDA על בסיס התכונות שנוצרו, ביצוע K – fold Cross Validation, בדיקת ביצועי הסיווג (דיוק, סטיית תקן) ושיפור הביצועים באמצעות הפחתת ממדים (PCA) או בחירת תכונות אופטימליות. החלת תהליך זהה על סט המבחן ודיווח תוצאות הסיווג.

שיטות:

הקוד הסופי חולק לארבעה קבצי py אשר מייצגים כל אחד בנפרד את השאלות (visualization.py, power_spectre.py, features.py, classification.py). בכל אחד מהקבצים הוגדרו הפונקציות הנדרשות והן מומשו בקובץ נפרד - main.py. בכדי להריץ את הקוד יש לשמור את כלל הקבצים בתיקיה משותפת ולהריץ את הקובץ main.py.

קבצי הנתונים כוללים הקלטות EEG משני ערוצים (C3 ו-C4) אשר נמדדו במהלך 160 חזרות של דמיון תנועה (יד ימין או יד שמאל). קובץ האימון (motor_imagery_train_data.npy) מכיל 128 חזרות עם תווית (left/right) וקובץ המבחן (motor_imagery_test_data.npy) מכיל 32 חזרות ללא תוויות עבור בדיקת המודל. כל חזרה נמשכת במשך 6 שניות ומכילה 768 דגימות.

טעינת הקבצים (שורות 62-36 בקובץ main.py):

יצירת הפונקציה (_load_npy_sequence()) בכדי לטעון את כלל המערכים (numpy arrays) שנשמרו ברצף בתוך קובץ npy אחד והחזרתם ברשימה. שימוש בלולאת while אשר קוראת את המערכים עד לסיום הקובץ (EOF). טעינת קובץ האימון (motor_imagery_train_data.npy) באמצעות פונקציה זו ולאחר הטעינה בדיקה שהקובץ מכיל ארבעה רכיבים: אותות EEG גולמיים, תוויות בינאריות לכל ניסוי, שמות התוויות ותדר הדגימה (Hz). רכיבים אלו מחולצים למשתנים מתאימים לצורך ניתוח ויזואלי וסטטיסטי בהמשך.

```

36 # — Utility —
37 def _load_npy_sequence(path: str | Path) -> list[np.ndarray]:
38     """Return *all* numpy arrays stored consecutively in a single .npy file."""
39     out: list[np.ndarray] = []
40     with open(path, "rb") as f:
41         while True:
42             try:
43                 out.append(np_load(f))
44             except EOFError:
45                 break
46     return out

```

```

49 # — Main Pipeline —
50 def main() -> None:
51     train_path = Path("motor_imagery_train_data.npy")
52     test_path = Path("motor_imagery_test_data.npy")
53
54     # === 0. Load TRAIN =====
55     trn = _load_npy_sequence(train_path)
56     if len(trn) != 4:
57         raise ValueError(
58             f"{train_path} must contain 4 arrays "
59             f"(data, labels, labels_name, fs) ⚡ found {len(trn)}."
60         )
61     train_data, labels, labels_name, fs_arr = trn
62     fs = int(fs_arr)

```

קובץ main.py (שורות 141-81) שימש להצאת כלל פונקציות הסיווג וההערכה אשר יתוארו בהמשך הקוד, הן על סט האימון והן על סט הבדיקה בהתאם למבנה הנתונים שהוזן.

קובץ visualization.py לשאלה 1:

בחירת הניסויים המתאימים (שורות 7-21):

יצירת הפונקציה `extract_valid_trails()` אשר מקבלת את אותות ה-EEG והתוויות ומחזירה את הניסויים המתאימים בלבד אשר תויגו ב- left או right. מתוך האותות הפונקציה שומרת את הערוצים הרלוונטיים (C3 ו-C4) לפעילות המוטורית. ניסוי יד שמאל בעל התווית '0' וניסוי יד ימין בעל התווית '1'. תוצר הפונקציה הוא מטריצת אותות בגודל $(n_trails, n_samples, 2)$ וקטור תוויות בגודל (n_trails) .

```

7 def extract_valid_trails(data: np.ndarray, labels: np.ndarray) -> Tuple[np.ndarray, np.ndarray]:
8     """
9     No artifacts: each trial belongs exclusively to LEFT or RIGHT.
10    Returns:
11    X (n_trials, n_samples, 2) ← channels C3,C4
12    y (n_trials,) ← 0=LEFT, 1=RIGHT
13    """
14    left_mask = labels[2] == 1
15    right_mask = labels[3] == 1
16    mask = left_mask | right_mask # All relevant trials
17
18    X = data[mask, :, :2] # First two channels
19    y = np.zeros(mask.sum(), dtype=int)
20    y[right_mask[mask]] = 1 # RIGHT → 1
21    return X, y

```

הצגת ניסויים אקראיים (שורות 24-43):

יצירת הפונקציה `plot_random_trails()` אשר מציגה בצורה גרפית 20 ניסויי EEG אקראיים ממחלקה נבחרת (יד ימין או יד שמאל). כל גרף מציג את האותות משני הערוצים הרלוונטיים כך שניתן לראות בצורה ברורה את דפוסי האותות בכל אחת מן המחלקות.

```

24 def plot_random_trails(X: np.ndarray, y: np.ndarray, label_value: int, title: str) -> None:
25     """Displays 20 random trials from the requested class (Left/Right)."""
26     np.random.seed(5)
27     idx = np.where(y == label_value)[0]
28     chosen = np.random.choice(idx, 20, replace=False)
29
30     fig, axes = plt.subplots(4, 5, figsize=(15, 10), sharex=True, sharey=True)
31     for i, ax in enumerate(axes.flat):
32         trial = X[chosen[i]]
33         ax.plot(trial[:, 0], label="C3", color="blue", linewidth=0.7)
34         ax.plot(trial[:, 1], label="C4", color="red", linewidth=0.7)
35         ax.set_xticks([]); ax.set_yticks([])
36         ax.set_title(f"Trial {chosen[i]}")
37     fig.suptitle(title, fontsize=16)
38     fig.text(0.5, 0.04, "Time (samples)", ha='center')
39     fig.text(0.04, 0.5, "Amplitude (µV)", va='center', rotation='vertical')
40     handles, labels_ = axes[0, 0].get_legend_handles_labels()
41     fig.legend(handles, labels_, loc='upper right')
42     plt.tight_layout(rect=[0.04, 0.04, 0.98, 0.94])
43     plt.show()

```

קובץ power_spectra.py לשאלה 2:

חילוץ אותות EEG (שורות 6-15):

יצירת הפונקציה `extract_trails()` לחילוץ אותות ה-EEG של ניסויי יד שמאל ויד ימין בטווח הזמן הרצוי לצורך איתור טווחי זמן אינפורמטיביים להשוואה בין התנאים.

```
6 # -----
7 # 1. Power Spectrum (Welch) with log10 stability
8 # -----
9 def extract_trails(data: np.ndarray, labels: np.ndarray,
10                  fs: int, start: float, end: float) -> Tuple[np.ndarray, np.ndarray]:
11     """Extract left/right trials using only samples between start and end (in seconds)."""
12     left = labels[2] == 1
13     right = labels[3] == 1
14     t0, t1 = int(start * fs), int(end * fs)
15     return data[left, t0:t1, :], data[right, t0:t1, :]
```

חישוב ספקטרום העוצמה (יחד עם הבונוס) (שורות 17-30):

יצירת הפונקציה `compute_avg_psd_db()` אשר מקבלת מטריצת ניסויים ומחשבת עבור כל ערוץ בכל ניסוי את ספקטרום העוצמה (Power Spectral Density) באמצעות שיטת Welch. הפונקציה מרכזת את העוצמה סביב הממוצע, ממירה את הערכים לסולם לוגריתמי (דציבלים) ושומרת את התוצאה עבור כל ערוץ בנפרד. לאחר מכן, מחזירה את מערך התדרים יחד עם ממוצע וסטיית התקן (בונוס) של הספקטרום לאורך כל הניסויים.

```
17 def compute_avg_psd_db(trials: np.ndarray, fs: int):
18     """Compute Welch-PSD in dB scale (log10), with mean & std across trials."""
19     psd_all_db = []
20     for trial in trials:
21         per_trial_db = []
22         for ch in range(trial.shape[1]):
23             sig = trial[:, ch] - np.mean(trial[:, ch])
24             nseg = min(fs, len(sig))
25             f, pxx = welch(sig, fs=fs, nperseg=nseg, detrend='constant')
26             pxx_db = 10 * np.log10(pxx + 1e-12)
27             per_trial_db.append(pxx_db)
28         psd_all_db.append(np.stack(per_trial_db, axis=1))
29     psd_all_db = np.stack(psd_all_db)
30     return f, psd_all_db.mean(axis=0), psd_all_db.std(axis=0)
```

חישוב ספקטוגרמת EEG (שורות 59-68):

יצירת הפונקציה `average_spectrogram()` אשר מחשבת את הספקטוגרמה של אות ה-EEG עבור כל אחד מהניסויים בערוץ שנבחר על ידי הסרת הרכיב הקבוע של האות (mean removal) וחישוב הספקטוגרמה על ידי `spectrogram` מספריית `scipy`. שמירת התוצאה מכל ניסוי וחישוב הממוצע של כלל הספקטוגרמות כך שמתקבלת ספקטוגרמה ממוצעת המייצגת את הפעילות התדרית לאורך זמן עבור כל הניסויים יחד.

```

59 # -----
60 # 2. Spectrograms – whole trial, +baseline correction
61 # -----
62 def average_spectrogram(data: np.ndarray, fs: int, chan: int):
63     specs = []
64     for trial in data:
65         f, t, Sxx = spectrogram(trial[:, chan] - trial[:, chan].mean(),
66                                 fs=fs, nperseg=fs // 2)
67         specs.append(Sxx)
68     return f, t, np.mean(specs, axis=0)

```

נרמול ספקטוגרמות (שורות 70-77):

יצירת הפונקציה `baseline_corrected()` אשר מחשבת ספקטוגרמות עבור כל ניסוי בנפרד תוך כדי נרמול ביחס לבסיס (baseline) המוגדר כעוצמת הספקטרום הממוצעת בשנייה הראשונה של הניסוי. המרת הספקטוגרמה לסולם דציבלים והחזרת ממוצע הספקטוגרמות המנורמלות.

```

70 def baseline_corrected(data: np.ndarray, fs: int, chan: int):
71     all_specs = []
72     for trial in data:
73         f, t, Sxx = spectrogram(trial[:, chan] - trial[:, chan].mean(),
74                                 fs=fs, nperseg=fs // 2)
75         baseline = Sxx[:, t < 1.0].mean(axis=1, keepdims=True)
76         all_specs.append(10 * np.log10(Sxx / (baseline + 1e-10)))
77     return np.mean(all_specs, axis=0), f, t

```

קובץ features.py לשאלה 3:חישוב עוצמת הספקטרום (שורות 20-23):

יצירת הפונקציה `_band_power()` לחישוב עוצמת הספקטרום בתחום תדרים המוגדר מראש באמצעות Welch על ידי סיכום הערכים בתחום התדרים הרלוונטיים והמרתם ליחידות דציבלים [dB].

```
20 def _band_power(sig: np.ndarray, fs: int, f_low: float, f_high: float) -> float:
21     f, Pxx = welch(sig, fs=fs, nperseg=fs)
22     band = Pxx[(f >= f_low) & (f <= f_high)].sum()
23     return 10 * np.log10(band + 1e-12)
```

חישוב עוצמת הספקטרום (שורות 29-32):

יצירת הפונקציה `_ensure_trial_first()` לוודא גודל תקין של המטריצות וביצוע transpose במידת הצורך יחד עם החזרת שגיאה במידה והמבנה לא תקין.

```
29 def _ensure_trial_first(labels: np.ndarray) -> np.ndarray:
30     if labels.ndim != 2 or 4 not in labels.shape:
31         raise ValueError("labels must be 2-D and include 4 columns (ARTIFACT, REMOVE, LEFT, RIGHT)")
32     return labels.T if labels.shape[0] == 4 else labels
```

חישוב מאפיינים (שורות 38-53):

יצירת הפונקציה `extract_band_power_features()` לחישוב מאפייני band – power לשלושת התחומים שהוגדרו מראש עבור כל ניסוי רלוונטי ולכל אחד מהערוצים. שמירת הניסויים של יד ימין ויד שמאל ועבורם חישוב הספק בתחום תדרים מוגדר לאחר הסרת הממוצע מכל ערוץ. הפונקציה מחזירה מטריצת פיצורים, וקטור תוויות מתאים ורשימת שמות הפיצורים אשר ישמשו להמשך הסינון.

```
38 def extract_band_power_features(data: np.ndarray, labels: np.ndarray, fs: int) -> Tuple[np.ndarray, np.ndarray, List[str]]:
39     labels = _ensure_trial_first(labels)
40     sel_idx = (labels[:, 2] == 1) | (labels[:, 3] == 1)
41     y = np.where(labels[sel_idx, 3] == 1, 1, 0)
42
43     X = []
44     for trial in data[sel_idx]:
45         feats = []
46         for t0, t1, ch, f_low, f_high, _ in BAND_DEFS:
47             s0, s1 = int(t0 * fs), int(t1 * fs)
48             sig = trial[s0:s1, ch] - trial[s0:s1, ch].mean()
49             feats.append(_band_power(sig, fs, f_low, f_high))
50         X.append(feats)
51
52     feature_names = [b[-1] for b in BAND_DEFS]
53     return np.array(X), y, feature_names
```

ספירת חציות האפס (שורות 59-62):

יצירת הפונקציה `zero_cross_rate()` לחישוב שיעור חציות האפס באות. ספירת כמות שינויי הסימן בדגימה לעומת הדגימה הקודמת. חלוקה במספר הדגימות ובחזרת שיעור החציות ביחס לאורך האות.

```

59 def zero_cross_rate(sig: np.ndarray) -> float:
60     """Zero crossing rate per sample (fraction)."""
61     crossings = np.where(np.diff(np.signbit(sig)))[0]
62     return len(crossings) / (len(sig) - 1 + 1e-12)

```

ZCR ויחס שונויות לוגריתמי (שורות 38-53):

יצירת הפונקציה `extract_additional_features()` לחישוב שני מאפייני זמן בכל ניסוי (קצב חציות האפס ZCR ויחס השונויות הלוגריתמי Log Variance Ratio) בין הערוצים C3 ו-C4. סינון הניסויים לפי דמיון התנועה, לקיחת חלון זמן בהתאם לניתוח נתוני הספקטוגרמה (4-6 שניות), הסרת הממוצע מכל ערוץ, חישוב קצב חציות האפס ב-C3 ויחס השונויות בין C3 ל-C4 תוך המרת הערכים ללוגריתמים. הפונקציה מחזירה את מטריצת הפיצ'רים ושמותיהם.

```

65 def extract_additional_features(data: np.ndarray, labels: np.ndarray, fs: int) -> Tuple[np.ndarray, List[str]]:
66     labels = _ensure_trial_first(labels)
67     sel_idx = (labels[:, 2] == 1) | (labels[:, 3] == 1)
68
69     feats_all = []
70     for trial in data[sel_idx]:
71         s0, s1 = int(4.0 * fs), int(6.0 * fs)
72         sig_c3 = trial[s0:s1, 0] - trial[s0:s1, 0].mean()
73         sig_c4 = trial[s0:s1, 1] - trial[s0:s1, 1].mean()
74
75         # Feature A: Zero-crossing rate in C3
76         zcr_c3 = zero_cross_rate(sig_c3)
77
78         # Feature B: log variance ratio C3/C4
79         var_ratio = np.var(sig_c3) / (np.var(sig_c4) + 1e-12)
80         feats_all.append([zcr_c3, np.log(var_ratio + 1e-12)])
81
82     return np.array(feats_all), ["ZCR_C3", "logVarRatio_C3/C4"]

```

הורדת ממדים עם PCA (שורות 105-107):

ביצירת הפונקציה `pca_visualization()` שימוש במחלקת PCA בכדי להוריד את הפיצ'רים לשלושה ממדים ובכך לבדוק את בחירת הפיצ'רים.

```

105 def pca_visualization(X: np.ndarray, y: np.ndarray, scaled: bool = False):
106     X_proc = StandardScaler().fit_transform(X) if scaled else X
107     comps = PCA(n_components=3).fit_transform(X_proc)

```

קובץ classification.py לשאלה 4 :הערכת LDA (שורות 28-57) :

יצירת הפונקציה `evaluate_lda_crossval()` אשר מבצעת הערכה מסוג LDA באמצעות שיטת K-fold cross-validation כך שברירת המחדל היא חלוקה ל-5 קיפולים. בכל איטרציה, אחת מהקבוצות משמשת כסט הולידציה והיתר כסט האימון. דיוק הוולידציה מחושב באמצעות `cross_val_score()` בעוד שדיוק האימון מחושב ידנית על ידי התאמת המודל על קבוצת האימון ובדיקתו עליה. הפונקציה מחזירה את תוצאות הדיוק בכל קיפול לשלב האימון ולשלב הולידציה.

```

28 def evaluate_lda_crossval(
29     X: np.ndarray,
30     y: np.ndarray,
31     k: int = 5,
32     random_state: int = 42,
33 ):
34     print("\n--- LDA Classification with k-fold Cross Validation ---")
35     print(f"Data shape: X = {X.shape}, y = {y.shape}")
36     print(f"Using k = {k} folds")
37
38     lda = LinearDiscriminantAnalysis()
39     kf = KFold(n_splits=k, shuffle=True, random_state=random_state)
40
41     val_scores = cross_val_score(lda, X, y, cv=kf, scoring="accuracy")
42     print(
43         f"Validation Accuracy: {val_scores.mean()*100:.2f}% ± {val_scores.std()*100:.2f}%"
44     )
45
46     train_scores = []
47     for train_idx, _ in kf.split(X):
48         X_train, y_train = X[train_idx], y[train_idx]
49         lda.fit(X_train, y_train)
50         y_train_pred = lda.predict(X_train)
51         train_scores.append(accuracy_score(y_train, y_train_pred))
52
53     train_scores = np.asarray(train_scores)
54     print(
55         f"Train Accuracy : {train_scores.mean()*100:.2f}% ± {train_scores.std()*100:.2f}%"
56     )
57     return val_scores, train_scores

```

הערכה נוספת (בנוסף) (שורות 199-241) :

יצירת הפונקציה `evaluate_classifier_crossval()` המבצעת הערכת ביצועים גנרית עבור כל מסוג מבוסס `sklearn` באמצעות שיטת K-fold cross-validation. הפונקציה מקבלת את המסווג כפרמטר (`clf`), יוצרת אובייקט מסוג `Kfold` שמחלק את הנתונים ל-k קבוצות לצורך אימון ובדיקה ומבצעת חישוב של דיוק הולידציה באמצעות `cross_val_score()`. בנוסף, החישוב מתבצע באופן ידני עבור קבוצת האימון גם כן. הפונקציה מחזירה את ציוני הדיוק לכל קיפול לשלב האימון ולשלב הוולידציה ומדפיסה את שם המסווג הנבחן.


```

199 def evaluate_classifier_crossval(
200     X: np.ndarray,
201     y: np.ndarray,
202     clf,
203     k: int = 5,
204     random_state: int = 42,
205     name: str | None = None,
206 ):
207     """
208     Generic K-fold evaluation tool for *any* sklearn classifier.
209
210     Parameters
211     -----
212     clf : sklearn.base.BaseEstimator
213         A classifier object (e.g., LogisticRegression()).
214     name : str
215         Name to print; default is the class name.
216     """
217     name = name or clf.__class__.__name__
218     print(f"\n--- {name} with {k}-fold Cross Validation ---")
219     print(f>Data shape: X = {X.shape}, y = {y.shape}")
220
221     kf = KFold(n_splits=k, shuffle=True, random_state=random_state)

```

```

223     # Validation
224     val_scores = cross_val_score(clf, X, y, cv=kf, scoring="accuracy")
225     print(
226         f"Validation Accuracy: {val_scores.mean()*100:.2f}% ± "
227         f"{val_scores.std()*100:.2f}%"
228     )
229
230     # Training
231     train_scores = []
232     for train_idx, _ in kf.split(X):
233         X_tr, y_tr = X[train_idx], y[train_idx]
234         clf.fit(X_tr, y_tr)
235         train_scores.append(accuracy_score(y_tr, clf.predict(X_tr)))
236     train_scores = np.asarray(train_scores)
237     print(
238         f"Train Accuracy : {train_scores.mean()*100:.2f}% ± "
239         f"{train_scores.std()*100:.2f}%"
240     )
241     return val_scores, train_scores

```

הגדרת מסווג Logistic Regression (שורות 28-57):

יצירת הפונקציה `evaluate_logreg_crossval()` אשר יוצרת את המסווג Logistic Regression עם מספר איטרציות מקסימלי של 1000 ומעבירה אותו יחד עם הדאטה לפונקציית ההערכה `evaluate_classifier_crossval()` הגנרית.

```

247 def evaluate_logreg_crossval(
248     X: np.ndarray,
249     y: np.ndarray,
250     k: int = 5,
251 ):
252     """K-fold CV for Logistic Regression (linear)."""
253     logreg = LogisticRegression(
254         penalty="l2",
255         solver="liblinear", # Suitable for small datasets
256         max_iter=1000,
257     )
258     return evaluate_classifier_crossval(X, y, logreg, k)

```

סיווג מבוסס PCA עם LDA (שורות 64-72):

יצירת הפונקציה `evaluate_with_pca()` המבצעת סיווג באמצעות LDA לאחר צמצום ממדים בשיטת PCA. נרמול הנתונים בעזרת `StandardScaler` ולאחר מכן ביצוע חישוב הרכיבים הראשיים באמצעות PCA תוך שמירה על מספר רכיבים מוגדר מראש עם ברירת מחדל של 3. התוצאה היא ייצוג מצומצם של הדאטה אשר משמר את רוב השונות. הערכה של הנתונים המצומצמים באמצעות `evaluate_lda_crossval()` בשיטת K-folds.

```

64     X: np.ndarray,
65     y: np.ndarray,
66     n_components: int = 3,
67     k: int = 5,
68 ):
69     print(f"\n--- PCA-based Classification with top {n_components} components ---")
70     X_scaled = StandardScaler().fit_transform(X)
71     X_pca = PCA(n_components=n_components).fit_transform(X_scaled)
72     return evaluate_lda_crossval(X_pca, y, k=k)

```

בחירת תכונות אופטימליות (שורות 78-136):

יצירת הפונקציה `evaluate_with_rfe()` כך שתבצע בחירת פיצ'רים בשיטת RFE (Recursive Feature Elimination) עבור מסווג מסוג LDA. עבור כל מספר אפשרי של פיצ'רים (עד למקסימום שנקבע), הפונקציה מחשבת את דיוק הוולידציה באמצעות התאמת מודל עם אותו מספר פיצ'רים נבחרים. לאחר סריקת כלל האפשרויות נבחרת תת – קבוצה אשר הניבה את הדיוק הגבוה ביותר ומוצגים שמות הפיצ'רים שנבחרו. ביצוע הרצת סיווג נוספת עם שלושת הפיצ'רים הטובים ביותר והשוואה נפרדת עם ארבעה פיצ'רים, לצורך בחינה נוספת של השפעת מספר הפיצ'רים על הביצועים.

```

78 def evaluate_with_rfe(
79     X: np.ndarray,
80     y: np.ndarray,
81     feature_names: list[str],
82     max_features: int | None = None,
83     k: int = 5,
84 ):
85     print("\n--- RFE-based Feature Selection ---")
86     if max_features is None:
87         max_features = X.shape[1]
88
89     scores: list[float] = []
90     features = list(range(1, max_features + 1))
91     best_selector: RFE | None = None
92
93     for n_feat in features:
94         lda = LinearDiscriminantAnalysis()
95         selector = RFE(lda, n_features_to_select=n_feat)
96         X_rfe = selector.fit_transform(X, y)
97         val_scores = cross_val_score(
98             lda, X_rfe, y, cv=KFold(n_splits=k, shuffle=True, random_state=42)
99         )
100         scores.append(val_scores.mean())
101         if n_feat == np.argmax(scores) + 1:
102             best_selector = selector

```

```

104 plt.figure(figsize=(6, 4))
105 plt.plot(features, scores, marker="o")
106 plt.title("Validation Accuracy vs. Number of Selected Features (RFE)")
107 plt.xlabel("Number of Features")
108 plt.ylabel("Validation Accuracy")
109 plt.grid(True)
110 plt.tight_layout()
111 plt.show()
112
113 best_n = np.argmax(scores) + 1
114 print(
115     f"Best number of features: {best_n} "
116     f"with accuracy: {scores[best_n - 1] * 100:.2f}%"
117 )
118
119 if best_selector is not None:
120     mask = best_selector.support_
121     selected = [n for n, keep in zip(feature_names, mask) if keep]
122     print("Selected features:", selected)
123
124     selected_idx = [i for i, n in enumerate(feature_names) if n in selected]
125     X_best = X[:, selected_idx]
126     print("\n[4.2c] LDA with only top 3 selected features:")
127     evaluate_lda_crossval(X_best, y, k=k)
128
129     if len(selected_idx) >= 3:
130         lda_full = LinearDiscriminantAnalysis()
131         rfe4 = RFE(lda_full, n_features_to_select=4)
132         X_rfe4 = rfe4.fit_transform(X, y)
133         print("\n[4.2d] LDA with top 4 features (for comparison):")
134         evaluate_lda_crossval(X_rfe4, y, k=k)
135
136     return best_n, scores

```

סיווג עם LDA לאחר נרמול (שורות 142-149):

יצירת הפונקציה `evaluate_on_normalized_features()` המבצעת הערכת סיווג באמצעות LDA לאחר נרמול הפיצורים בשיטת Z-score. לאחר הנרמול, העברת הנתונים לפונקציית ההערכה `evaluate_lda_crossval()` בשיטת K-fold. בפונקציה זו יש שימוש בנרמול בלבד, ללא הפחתת ממד או סינון תכונות.

```

142 def evaluate_on_normalized_features(
143     X: np.ndarray,
144     y: np.ndarray,
145     k: int = 5,
146 ):
147     print("\n[4.2e] LDA on Z-score Normalized Features (no PCA or RFE):")
148     X_scaled = StandardScaler().fit_transform(X)
149     evaluate_lda_crossval(X_scaled, y, k=k)

```

סיווג סופי על סט הבדיקה (שורות 142-149):

יצירת הפונקציה `evaluate_final_classifier_on_test()` המבצעת אימון סופי של מסווג LDA על כלל נתוני האימון והערכת ביצועים על סט הבדיקה. חילוף פיצורים מ-`train` באמצעות `build_all_features()`, בחירת שלושה פיצורים על ידי RFE, נרמול באמצעות Z-score ואימון המסווג על נתוני האימון. ביצוע RFE גם לסט הבדיקה ונרמול באמצעות האובייקטים שאומנו על סט האימון, ללא התאמה מחדש. חיזוי על סט הבדיקה, חישוב דיוק ושמירת קובץ תחזיות.

אילונה בר שדה 212366702

שירה אלטרץ 315262683

```
155 def evaluate_final_classifier_on_test(  
156     train_data,  
157     train_labels,  
158     test_data,  
159     test_labels,  
160     fs,  
161     feature_names,  
162 ):  
163     # ❶ Full feature extraction – Train  
164     X_train, y_train, _ = build_all_features(train_data, train_labels, fs)  
165  
166     # ❷ RFE for selecting 3 features  
167     lda = LinearDiscriminantAnalysis()  
168     rfe = RFE(lda, n_features_to_select=3).fit(X_train, y_train)  
169     X_train_sel = rfe.transform(X_train)  
170  
171     # ❸ Z-score normalization  
172     scaler = StandardScaler().fit(X_train_sel)  
173     X_train_scaled = scaler.transform(X_train_sel)  
174  
175     # ❹ Final LDA training  
176     clf = LinearDiscriminantAnalysis().fit(X_train_scaled, y_train)  
177  
178     # ❺ Test feature extraction  
179     X_test, y_test, _ = build_all_features(test_data, test_labels, fs)  
180     X_test_sel = rfe.transform(X_test)  
181     X_test_scaled = scaler.transform(X_test_sel)  
182  
183     # ❻ Prediction + accuracy  
184     y_pred = clf.predict(X_test_scaled)  
185     acc = accuracy_score(y_test, y_pred)  
186     print(f"Test Accuracy: {acc * 100:.2f}%")  
187  
188     np.save("test_predictions.npy", y_pred)  
189     print("Saved predictions to test_predictions.npy")
```

אילונה בר שדה 212366702

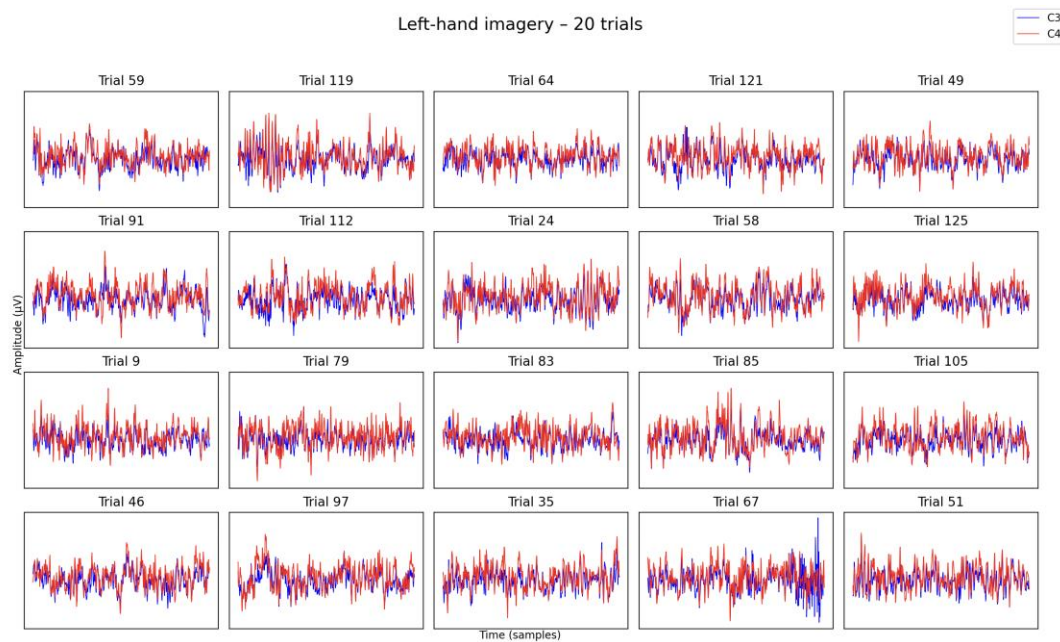
שירה אלטרץ 315262683

תוצאות:

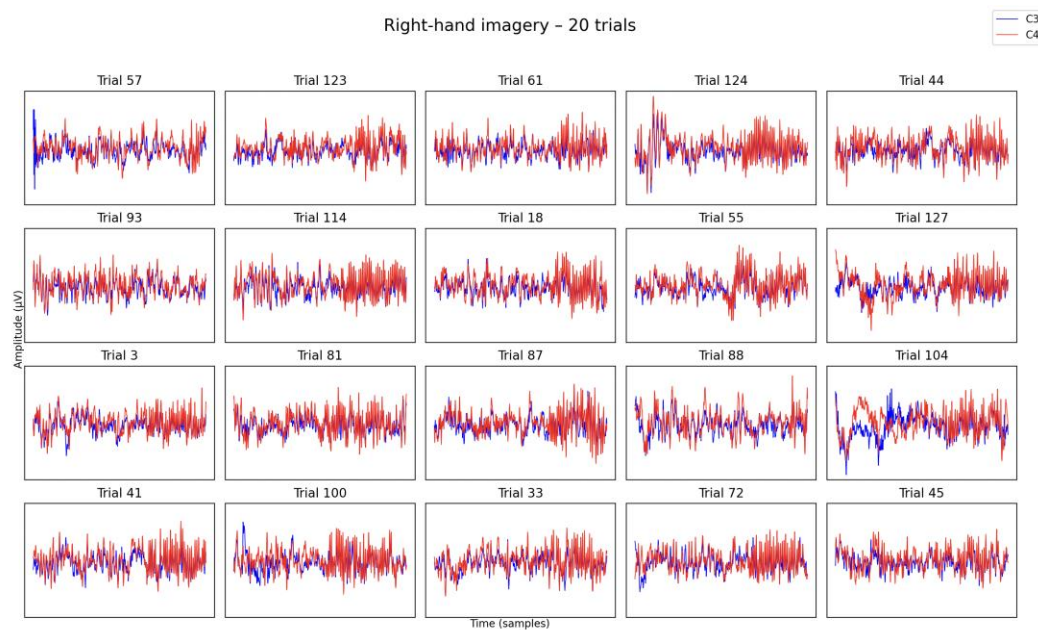
מוצגים 1,2-

20 חזרות של תנאי דמיון תנועה של יד שמאל (מוצג 1) ויד ימין (מוצג 2). הצגת אותות ה-EEG מהערוצים C3 (כחול) ו-C4 (אדום) בכל אחד מהגרפים.

מוצג 1-

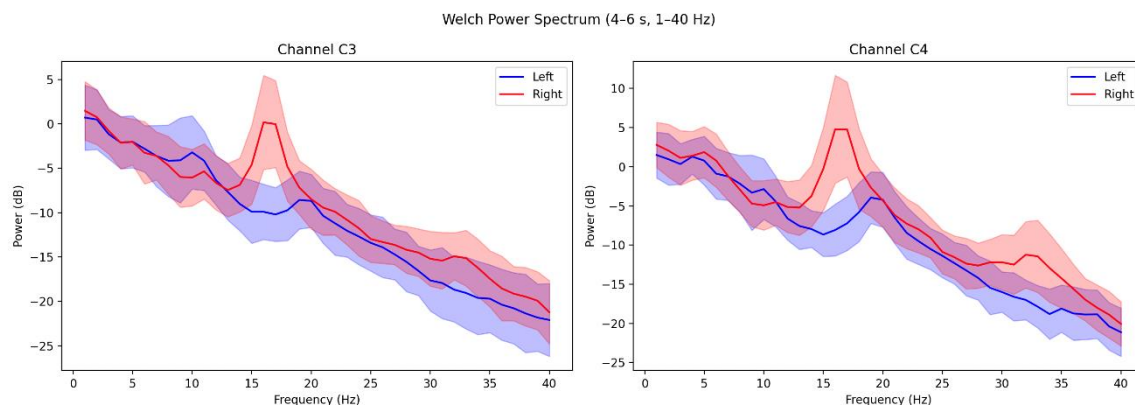


מוצג 2-



מוצג 3-

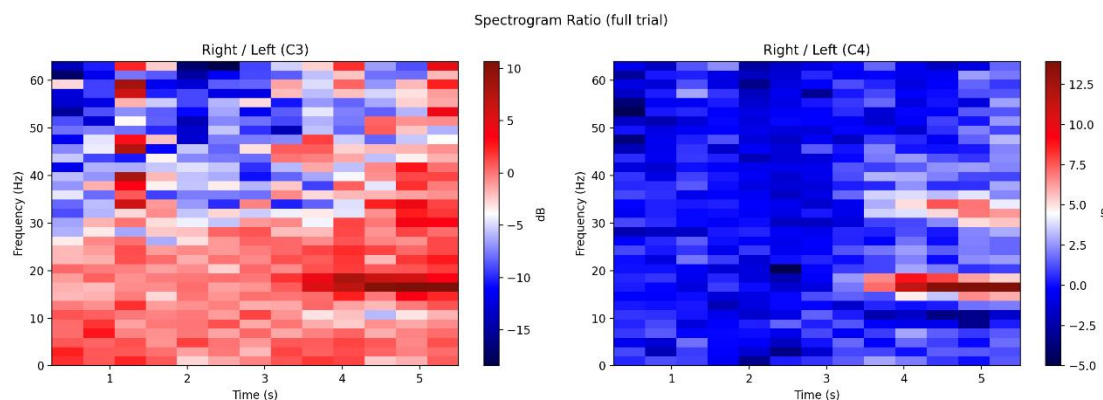
בגרף זה מוצגת עוצמת ההספק (Power) כפונקציה של תדר, כפי שחושבה בשיטת Welch עבור הערוצים C3 ו-C4. הקווים מייצגים את הממוצע עבור תנאי הדמיון של יד שמאל (כחול) ויד ימין (אדום) כשהשטח הצבוע מייצג את סטיית התקן.



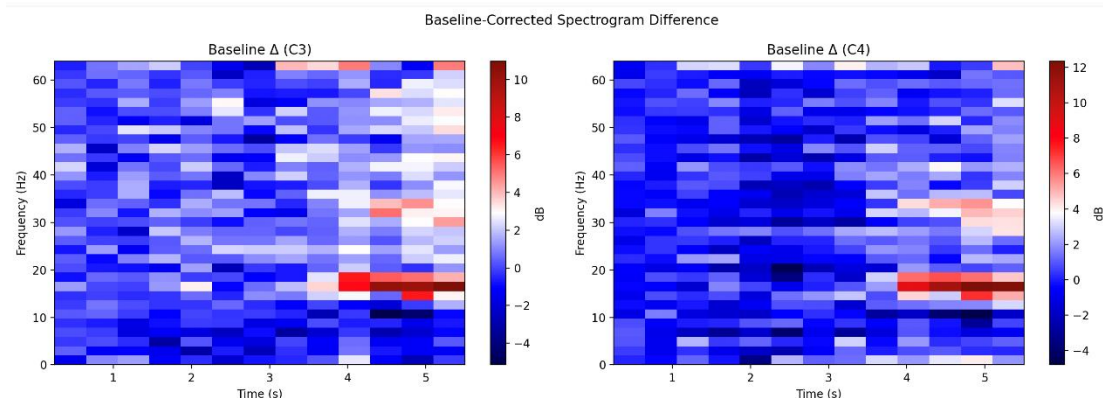
מוצגים 4,5-

בגרפים אלו מוצג היחס בין ספקטוגרמות של תנאי יד ימין לתנאי יד שמאל (Right/Left) עבור הערוצים C3 ו-C4 לאורך זמן ותדר. הערכים מוצגים בדציבלים כך שצבעים חמים (אדומים) מציינים עוצמה גבוהה יותר ביד ימין לעומת שמאל וצבעים קרים (כחול) להפך. מוצג 4 מציג את היחס הגולמי, בעוד מוצג 5 מציג יחס לאחר ניקוי קו בסיס (baseline).

מוצג 4-



מוצג 5-



מוצגים 6-8

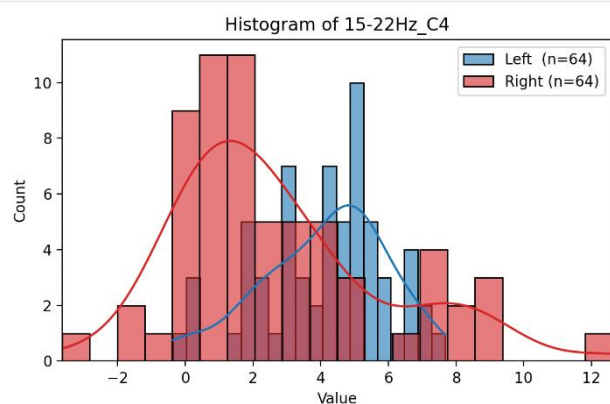
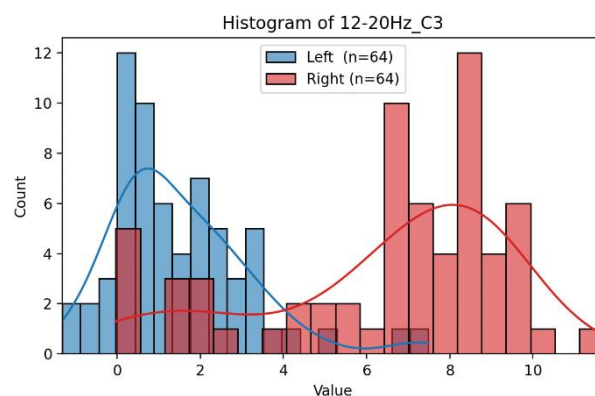
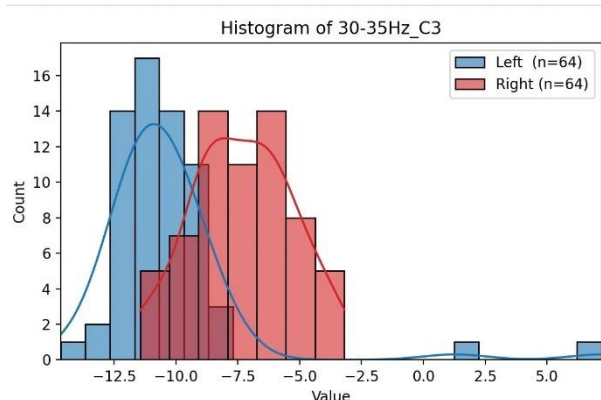
היסטוגרמות אלו מציגות את התפלגות ערכי ה-band-power בערוצים שונים עבור תדרים שנבחרו לשמש כפיצ'רים בתהליך הסיווג על סמך ניתוח ספקטרום מוקדם (Power Spectrum) (וספקטוגרמות) שבו נצפו הבדלים מובהקים בין התנאים. ישנה הפרדה בין ההתפלגויות עבור יד שמאל (כחול) ויד ימין (אדום).

מוצג 7

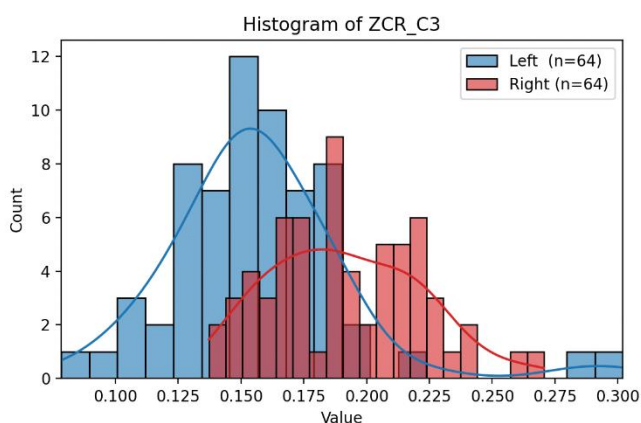
ערוץ C3 בתחום התדרים 30 – 35 [Hz] ובזמן 4 – 6[s]

מוצג 6

ערוץ C3 בתחום התדרים 12 – 20 [Hz] ובזמן 4 – 6[s]

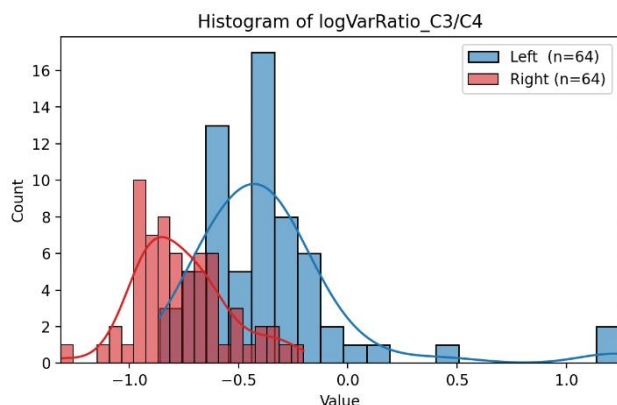
מוצג 8

ערוץ C4 בתחום התדרים 15 – 22 [Hz] ובזמן 1.8 – 4.1[s]

מוצג 9

היסטוגרמה המציגה את התפלגות ערכי הפיצ'ר ZCR (Zero Crossing Rate) שחושב עבור ערוץ C3 תוך השוואה בין תנאי הדמיון של יד שמאל (כחול) ויד ימין (אדום). ציר ה-X מתאר את השיעור בו האות חצה את קו האפס וציר ה-Y מראה כמה ניסיונות (trials) קיבל כל ערך.

מוצג 10-



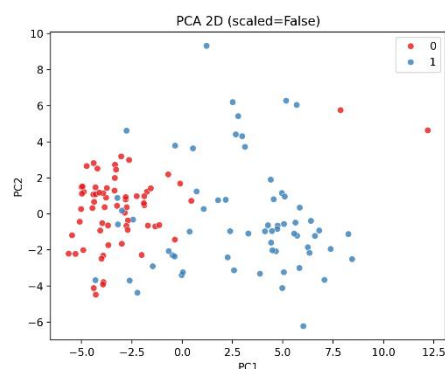
היסטוגרמה המציגה את התפלגות ערכי הפיצ'ר log-variance ratio (לוגריתם של יחס השונות בין הערוצים C3 ו-C4) עבור תנאי הדמיון של יד שמאל (כחול) ויד ימין (אדום). ציר ה-X מייצג את ערך הפיצ'ר וציר ה-Y מייצג את מספר הנסיונות (trials) עבור כל ערך.

מוצגים 11-14-

הגרפים הבאים מציגים את הדאטה לאחר ביצוע PCA עם הורדת הפיצ'רים לשלושה ממדים (PC1, PC2 ו-PC3) כך שהם מציגים את מירב השונות. גרפים המציגים את המידע בצורה דו-ממדית (הקרנה על PC1 ו-PC2) או בצורה תלת-ממדית (PC1, PC2 ו-PC3) עם צבע התווית ('0' = שמאל, '1' = ימין).

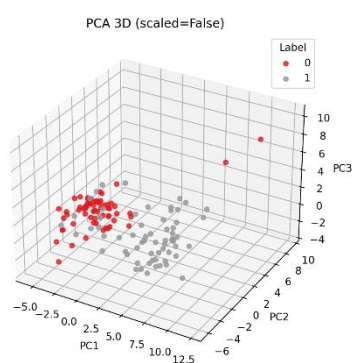
מוצג 11-

פיזור הנתונים על צירי PC1 ו-PC2, ללא נרמול מקדים של הפיצ'רים.



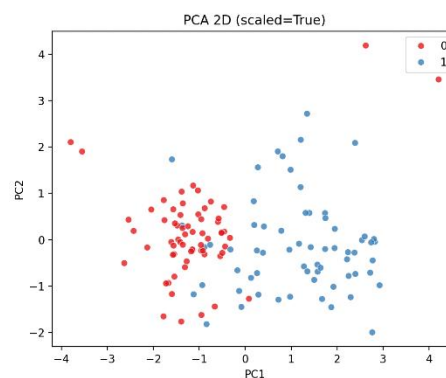
מוצג 12-

פיזור תלת-ממדי – ממדי על הצירים PC1, PC2 ו-PC3, ללא נרמול.



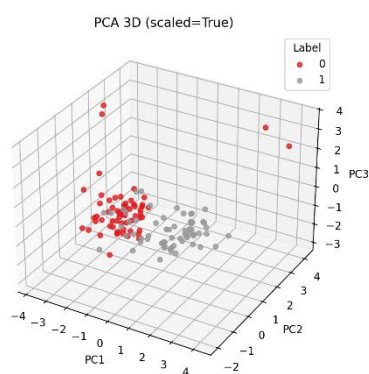
מוצג 13-

הקרנת הדאטה על צירי PC1 ו-PC2 לאחר נרמול Z-score.



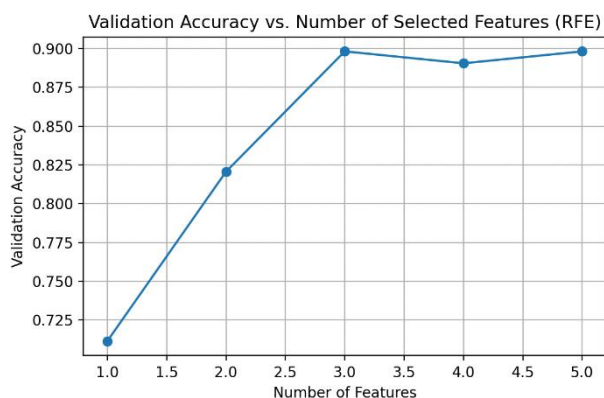
מוצג 14-

הקרנת הדאטה בתלת-ממד על PC1, PC2 ו-PC3 לאחר נרמול.



מוצג 15-

הגרף מציג את הקשר בין מספר הפיצורים שנבחרו לבין דיוק הוולידציה של המודל כפי שנמדד בשיטת RFE (Recursive Feature Elimination). ציר ה-X מייצג את מספר הפיצורים שנבחרו, ציר ה-Y מייצג את הוולידציה וכל נקודה מייצגת את הדיוק שהתקבל עבור מספר מסוים של פיצורים.



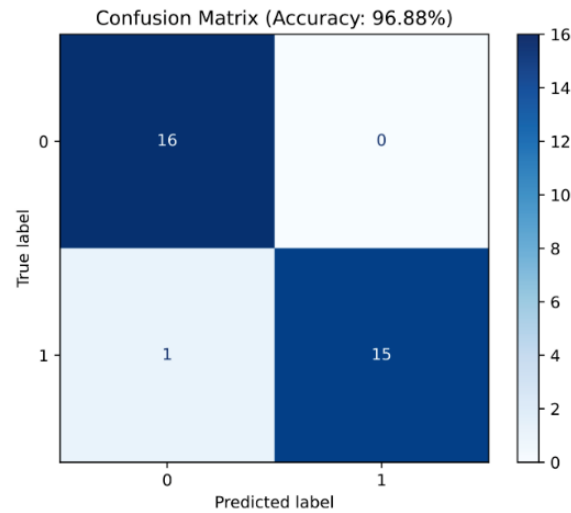
מוצג 16-

במוצג זה ניתן לראות את תוצאות הסיווג שהתקבלו ממספר מודלים וגישות שונות, תוך השוואת ביצועים לפי דיוק ממוצע וסטיית תקן. עבור כל שיטה מוצגים דיוקי אמון וולידציה שנמדדו באמצעות K-fold cross-validation על בסיס קבוצות פיצורים שונות.

מודל LDA לאחר נרמול פיצורים	מודל LDA עם ארבעה פיצורים נבחרים	מודל LDA עם שלושה פיצורים נבחרים	מודל LDA לאחר PCA	בונס – מודל רגרסיה לוגיסטית	מודל LDA	
X = (128,5) Y = (128)	X = (128,4) Y = (128)	X = (128,3) Y = (128)	X = (128,2) Y = (128)	X = (128,5) Y = (128)	X = (128,5) Y = (128)	Data Shape
5	5	5	5	5	5	Using K
89.82% ± 6.34%	89.05% ± 5.72%	89.82% ± 4.73%	89.82% ± 4.73%	89.78% ± 4.13%	89.82% ± 6.34%	Validation Accuracy
90.42% ± 1.92%	90.43% ± 1.46%	90.82% ± 1.19%	89.26% ± 1.24%	90.04% ± 1.16%	90.42 ± 1.92%	Train Accuracy

מוצג 17-

במוצג זה ניתן לראות את מטריצת Confusion Matrix המתקבלת לאחר הפעלת המודל על סט הבדיקה שכלל 32 דגימות. המספרים מייצגים את כמות הסיווגים הנכונים והשגויים לפי תוויות אמת ותחזית כך שהדיוק הכולל עמד על 96.88% (31 מתוך 32 הדגימות סווגו נכונה עם טעות אחת בלבד מסוג False Negative).



דיון:

מטרתו של תרגיל זה הינה סיווג תנועות דמיוניות של יד ימין ויד שמאל מתוך אותות EEG אשר נמדדו משני ערוצים (C3 ו-C4) באמצעות אלגוריתם של למידת מכונה, כלומר, לזהות האם נבדק דמיון תנועת יד ימין או יד שמאל רק מתוך אות המוח החשמלי ובאופן אוטומטי (ממוחשב) בהתבסס על ניסיון בודד. במהלך הדיון יובא פירוט על שלבי תהליך העבודה ועל התוצאות שהוצגו בחלק הקודם.

תחילה התבצעה **הכרה ויזואלית** עם האותות עצמם באמצעות ויזואליזציה של ניסויי EEG גולמיים. תוצאות אלו (מוצגים 1 ו-2) מציגים 20 ניסיונות אקראיים לכל תנאי (דמיון תנועת יד שמאל או ימין) בשני הערוצים C3 ו-C4 לאורך זמן המדידה. ניתן לראות שהנתונים רועשים ומכך שקשה להסיק תובנות ברורות מהתבוננות חזותית בהם, אך בשליש האחרון של מספר ניסויים ביד ימין ניכרת פעילות מוגברת בערוץ C4, דבר אשר עשוי להעיד על הבדל רלוונטי לצורך הסיווג.

מהפקת **גרף ספקטרום העוצמה, Power Spectral Density** (מוצג 3) שחושב על ידי שיטת Welch, בו נראים ממוצעי תנאי הדמיון של יד שמאל וימין בערוצים השונים, ניתן לראות כי קיימת הפרדה ברורה בין יד ימין ליד שמאל בעיקר בטווח התדרים של $20 - 15$ [Hz] בזמנים $4 - 6$ שניות שנלקחו מניתוח נתוני ההכרה הוויזואלית. מכאן שתדרים אלו עשויים להיות אינפורמטיביים להמשך הסיווג. הבדלים בין התנאים בטווח התדרים המוזכר לעיל עולים בקנה אחד עם ממצאים מהספרות, המעידים על ירידה בעוצמת גלי בטא β ($20 - 15$ [Hz]) באזורים מוטוריים במהלך דמיון תנועה, תופעה הידועה כ- **Event-Related Desynchronization (ERD)**ⁱⁱ. תופעה זו משקפת ירידה בסנכרון העצמי ונחשבת למדד נירופיזיולוגי מהימן לפעולה מוטורית ממשית או מדומינת ומכאן כי היא תואמת את הנתונים ממטלה זו.

בהמשך נבנו **ספקטוגרמות הממחישות את ההבדלים בין תנאי דמיון יד ימין לבין דמיון יד שמאל כפונקציה של זמן ותדר**. שלא כמו ב- PSD, בספקטוגרמות אלו נעשה שימוש באות המלא מאחר ששיטה זו מאפשרת זיהוי שינויים גם לאורך הזמן (כולל תחילת הפעילות המוטורית וההסתגלות). הספקטוגרמות (מוצג 4) חושבו באמצעות לוגריתם של יחס ההספק בין תנאי יד ימין ליד שמאל, כך שערכים חיוביים מצביעים על עוצמה גבוהה יותר ביד ימין ולהפך, דבר אשר אפשר זיהוי ברור של אזורים מבדילים בין התנאים. הממצאים מצביעים על הבדל מובהק בטווחי התדרים $20 - 12$ [Hz] ובטווח זמנים של $4 - 6$ שניות ומכאן שפערים אלו יכולים לשמש לאפיון תכונות אפקטיביות לצורך סיווג. בכדי לאשש ממצאים אלו בוצע **נרמול** של הספקטוגרמות (מוצג 5) לפי השנייה הראשונה בניסוי המוגדרת כ- baseline. הנרמול סייע בהבהרת אזורים מבדילים בין התנאים, לפיו ניתן לראות כי קיים שינוי נוסף באות של C3 גם בין $35 - 30$ [Hz] באותם הזמנים ($4 - 6$ שניות) ומכאן שיש לנרמול משמעות חשובה בבחירת התכונות לסיווג. ההבדלים הנראים בטווחי התדרים מתאימים גם במקרה זה לתופעת ה- ERD, שכן מדובר על אותם טווחי התדרים של גלי הבטא β וזוהי עוד אינדיקציה לכך שישנה חשיבות לבחירת התכונות בהקשר לתופעה זו.

לשם בניית מערך תכונות אפקטיבי לסיווג תנאי דמיון יד ימין מול יד שמאל, נבחרו חמישה פיצ'רים מרכזיים (מוצגים 6-10). שלושת הפיצ'רים הראשונים מתבססים על **ממוצע עוצמה**

ספקטרלית (PSD) בטווחי תדרים וזמנים אשר נמצאו אינפורמטיביים מתוך ניתוחי גרף הספקטרום והספקטוגרמות שכן בטווחים אלו קיימים הבדלים משמעותיים בין האותות:

הראשון מערוץ C3 בטווחי תדר $12 - 20$ [Hz] וזמן $4 - 6$ [s] (מוצג 6), השני מאותו הערוץ בטווחי תדר $30 - 35$ [Hz] וזמן $4 - 6$ [s] (מוצג 7) והשלישי מערוץ C4 בטווחי תדר $15 - 22$ [Hz] וזמן $1.8 - 4.1$ [s], שנמצא כטווח זמנים המשקף הפרדה טובה בין דמיון יד ימין לדמיון יד שמאל (מוצג 8). הפיציר הרביעי, **ZCR (Zero Crossing Rate)**, אשר מוגדר כתכונת אות בולטת המשקפת את קצב המעברים של האות דרך חציית האפסⁱⁱⁱ והוא משקף את רמת התנודתיות וקצב השינוי של האות בערוץ C3 לאורך הזמן. הוא נבחר מתוך הנחה כי הוא מוסיף מידע נוסף למערך הפיצירים, כיוון שכך ניתן לבצע מעקב על רמת התנודתיות של האות המוחי אשר עשויה להשתנות בהתאם לרמת הפעילות העצבית באזור המוטורי המושפעת מסוג הדמיון המוטורי, בכך היא יכולה לספק אינדיקציה שונה בין התאים (מוצג 9). לבסוף, נבחר הפיציר החמישי (מוצג 10) המתאר את **היחס הלוגריתמי בין שונות האות ב-C3 לעומת C4** (שיטה בה יחס שונות ערוצים משמש להבחנה בין מצבי מוח^{iv}) ומשווה בין שני צידי המוח, מכך ניתן לקבל מידע על פעילות או שינוי בהם. היחס הלוגריתמי משקף את האסימטריה בפעילות החשמלית בין שני חצאי המוח אשר עלולה להופיע במצבי דמיון תנועה. מכיוון ש-C3 ו-C4 ממוקמים מעל ההמיספרות המוטוריות השמאלית והימנית בהתאמה, דמיון תנועת יד ימין או שמאל עשוי להפעיל את אחד הערוצים יותר מהשני ובכך להוות אינדיקציה טובה לסיווג התנאים.

לכל אחד מהפיצירים שתואר לעיל מצורפת היסטוגרמה המתאימה לו (מוצגים 10 - 6) כך שניתן לראות בהן הפרדות בין תנועת יד ימין ותנועת יד שמאל ומכאן להסיק כי הם מהווים פיצירים המתארים בצורה טובה את הסיווג, בעיקר בפיציר הראשון, הרביעי והחמישי.

לאחר מכן בוצעה הפחתת ממדים באמצעות שיטת **PCA (Principal Component Analysis)** אשר נועדה לצמצם את מספר הפיצירים תוך שמירה על מירב השונות במידע. מתוך חמשת הפיצירים שנבחרו קודם לכן בוצעה הורדה לשלושה ממדים. בצורה זו הרכיבים החדשים שנוצרו ($PC1$, $PC2$, $PC3$) מהווים צירופים לינאריים של הפיצירים המקוריים ובכך מתאפשרת הדמיה נוחה יותר של מבנה הנתונים. התוצאות הוצגו בגרף דו-ממדי (מוצג 11), אשר מתבסס על שני הרכיבים הראשיים עם מירב השונות ובגרף תלת-ממדי (מוצג 12) אשר מראה את הייצוג לפי שלושת הרכיבים. ניתן לראות שבשני המקרים קיימת הפרדה כמעט מלאה בין הקבוצות, דבר המצביע על כך שהפיצירים שנבחרו **אכן נושאים מידע רלוונטי לסיווג בין התנאים** (כלומר, בין יד ימין ליד שמאל), וכן כי רוב השונות מיוצגת על ידי שני הרכיבים הראשונים ($PC1$ ו- $PC2$), דבר אשר יהיה רלוונטי להמשך הסיווג. בשלב שלאחר מכן בוצע **נרמול** לפיצירים שיהיו בעלי אותן יחידות מידה, דבר אשר אפשר להשוות בניהם בצורה הוגנת כיוון שהוא ביטל הבדלים בגודל וביחידות בין הפיצירים השונים. מוצגים אלו (מוצגים 13-14) ממחישים כי גם במצב בו כל הפיצירים בעלי משקל שווה, **קיימת מגמת הפרדה כמעט מלאה בין התנאים** ובכך ישנו חיזוק לתרומה המשולבת של כלל הפיצירים למבנה הנתונים ועל כך ש-PCA עם הורדה ל-3 ממדים (ואף שניים, כיוון שרוב השונות נשמרת בתצוגה דו-ממדית) מהווה בסיס טוב להמשך מהלך הסיווג.

באמצעות שיטת **RFE (Recursive Feature Elimination)**^v מודגם הקשר בין מספר הפיצירים הנבחרים לבין דיוק המודל. RFE מדרגת את הפיצירים לפי תרומתם לסיווג ומסירה בכל

איטרציה את הפיצ'ר בעל ההשפעה הפחותה ביותר. מהגרף (מוצג 15) ניתן לראות כי הדיוק משתפר עם הוספת הפיצ'רים ומתייצב לאחר הוספת שלושה פיצ'רים בערך של כ- 90%. מכאן ניתן להסיק כי עיקר התרומה לסיווג נובעת משלושת הפיצ'רים המרכזיים והפיצ'רים האחרונים מוסיפים מידע מועט ואינם תורמים רבות לסיווג.

כדי לבחור את המודל הטוב ביותר נבחנו שישה סוגי מודלים: מודל LDA (עם חמשת הפיצ'רים), מודל רגרסיה לוגיסטית (לשאלת הבונוס), מודל LDA לאחר PCA ל-2 ממדים (המייצגים את מרב השונות כפי שניתן לראות במוצגים 11-14), מודל LDA עם שלושה פיצ'רים נבחרים, מודל LDA עם 4 פיצ'רים נבחרים ומודל LDA לאחר נרמול כלל הפיצ'רים. כל אחד מהמודלים נבחן בשיטת **K-fold cross-validation**^{vi}, שיטה נפוצה להערכת ביצועי מודל למידת מכונה בה באמצעות חלוקה לתתי קבוצות מאמנים את המודל כך שבכל פעם קבוצה אחרת משמשת כקבוצת הבדיקה והשאר כקבוצות האימון ולבסוף מודדים את ממוצע הביצועים. ערך ה-K הנבחר לקיפול הצולב הוא $K=5$, מאחר וערך זה מהווה איזון אידיאלי בין חלוקה מאוזנת של הנתונים המאפשרת יציבות סטטיסטית, זמן ריצה סביר והערכת ביצועים מהימנה. ערך קטן עלול להוביל להערכה שאינה מייצגת עקב מספר ריצות נמוך להתאמת המודל, בעוד שערך גדול עשוי לגרום להתאמת יתר לדגימות ולפגוע ביכולת ההכללה של המודל ולזמן ריצה ארוך. מכאן שהשיטה מספקת הערכה יציבה ומהימנה ליכולת ההכללה של המודל תוך שימוש מיטבי בכלל המידע. תוצאות הבחינה מוצגות בטבלה (מוצג 16).

האלגוריתם של **LDA (Linear discriminant Analysis)**^{vii} נועד למצוא גבול לינארי המפריד בצורה מיטבית בין שתי קבוצות על ידי חישוב יחס בין שונות בין – קבוצתית לבין שונות תוך – קבוצתית. הוא פשוט ומתאים למבנה נתונים לינארי יחסית בדומה למבנה במטלה זו. מודל ה**רגרסיה הלוגיסטית (logistic regression)**^{viii} הוא אלגוריתם המחשב הסתברות להשתייכות לדגם מסוים באמצעות פונקציית סיגמואיד ומתאים אף הוא לבעיית הסיווג הבינארי, לכן הוא נבחר כמודל מתאים. כיוון שהמודלים מתבססים על הפרדה בצורה שונה ניתן לבחון בצורה זו את הצלחת הפיצ'רים וההפרדה הלינארית בצורה המיטבית. במודלים בהם ייתכנו שילובי פיצ'רים שונים (קבוצות של שלושה או ארבעה פיצ'רים) נעשה שימוש בשיטת **RFE** כך שיבחרו הפיצ'רים בעלי הדיוק המקסימלי. תוצאותיה של חקירה זו מראות כי המודל הטוב ביותר לסיווג דמיון התנועות הוא המודל עם שלושת הפיצ'רים הנבחרים שהם ZCR, log variance-ratio ו-C3 בתחום שבין 12 – 20 [Hz] ו- 4 – 6 [s] (לפי RFE), מודל זה מניב תוצאה של $89.82\% \pm 4.73\%$ של Validation Accuracy (אחוז דיוק המודל על קבוצת הנתונים ששימשה לאימון) ותוצאה של $90.82 \pm 1.19\%$ של Train Accuracy (אחוז הדגימות שהמודל סיווג נכון במהלך האימון), אשר יחד מהווים את התוצאה הטובה ביותר בהשוואה למודלים האחרים. הבחירה במודל זה מתיישבת עם הציפיות אשר נראו במוצגים קודמים במטלה כפי שנאמר קודם לכן (מוצגים 6, 9 ו-10 אשר מהווים סיווג טוב של תנועות הידיים).

ביצוע ההערכה הסופית על סט הבדיקה ($n=32$) הניב מטריצת Confusion Matrix (מוצג 17) עם דיוק גבוה של 96.88%. מתוך 32 דגימות, 31 סווגו נכונה ודגימה אחת בלבד סווגה כשגיאה מסוג False Negative. מהדיוק הגבוה ניתן להסיק כי ביצועי הכלליים של המודל ויכולת ההכללה שלו טובים.

אילונה בר שדה 212366702

שירה אלטרץ 315262683

לסיכום, מהלך העבודה כלל ניתוח תדרים, זמנים גזירת פיצ'רים רלוונטיים, הפחתת ממדים
ובחינת מודלים שונים אשר הובילי בסופו של דבר ליצירת מודל סיווג מדויק ואפקטיבי. ממצאי
העבודה מצביעים על כך ששילוב בין עיבוד אותות EEG, בחירת תכונות מושכלת ומימוש
אלגוריתמים הניב ביצועים מרשימים במבחני ההכללה ונוצר מודל בעל יכולות סיווג מוצלחות.

-
- ⁱ Luo, Z., Jin, R., Shi, H., & Lu, X. (2021). Research on recognition of motor imagination based on connectivity features of brain functional network. *Neural Plasticity*, 2021, 6655430.
<https://pmc.ncbi.nlm.nih.gov/articles/PMC7895585/>
- ⁱⁱ Putzolu, M., Samogin, J., Cosentino, C., Mezzarobba, S., Bonassi, G., Lagravinese, G., Pelosin, E., & Avanzino, L. (2022). Neural oscillations during motor imagery of complex gait: An HdEEG study. *Scientific Reports*, 12, Article 4314.
<https://www.nature.com/articles/s41598-022-07511-x>
- ⁱⁱⁱ Stancin, I., Cifrek, M., & Jovic, A. (2021). A review of EEG signal features and their application in driver drowsiness detection systems. *Sensors*, 21(11), 3786.
https://www.researchgate.net/publication/351995829_A_Review_of_EEG_Signal_Features_and_Their_Application_in_Driver_Drowsiness_Detection_Systems
- ^{iv} Mouri, F. I., Valderrama, C. E., & Camorlinga, S. G. (2023). Identifying relevant asymmetry features of EEG for emotion processing. *Frontiers in Psychology*, 14, 1217178.
<https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2023.1217178/full>
- ^v Yin, Z., Liu, L., Chen, J., Zhao, B., & Wang, Y. (2020). Locally robust EEG feature selection for individual-independent emotion recognition. *Expert Systems with Applications*, 162, 113768.
<https://www.sciencedirect.com/science/article/abs/pii/S0957417420305923>
- ^{vi} Wilimitis, D., & Walsh, C. G. (2023). Practical considerations and applies examples of cross-validation for model development and evaluation in health care: Tutorial. *JMIR AI*, 2, e49023.
<https://pmc.ncbi.nlm.nih.gov/articles/PMC11041453/>
- ^{vii} Prasan, N. H. (2024, January 6). Linear Discriminant Analysis (LDA) in Classification. *Medium*. <https://medium.com/@prasanNH/linear-discriminant-analysis-lda-in-classification-354aca40ac57>

^{viii} Sperandei, S. (2013). Understanding logistic regression analysis. *Biochemia Medica*, 23(1), 12–18.

<https://pmc.ncbi.nlm.nih.gov/articles/PMC3936971/>