Final Project
Analysis of Motor Imagery Data

Introduction

A Brain-Computer Interface creates a direct communication pathway between the brain and external devices by analyzing EEG signals during imagined movements. When individuals imagine moving their left or right hand, distinct electrical patterns emerge in the motor cortex that can be detected and differentiated. This project focuses on identifying these neural patterns in motor-related brain rhythms to decode intended hand movements. The methodology involves extracting key features from EEG data and training a classification algorithm to predict whether someone is imagining left- or right-hand movement on individual trials. This has significant applications, particularly for individuals with motor disabilities, enabling them to control external devices.

Research questions that are touched upon in this project:

1. **What are the most informative features that can be extracted from EEG Data to discriminate between imagined left- and right-hand movements?**
   This involves analyzing power spectra using Welch's method and spectrograms to pinpoint useful frequency bands and time periods.
2. **How can dimensionality reduction using PCA and feature normalization improve the separability and classification of motor imagery data?**
3. **How effectively can a Linear Discriminant Analysis (LDA) classifier predict left vs. right hand motor imagery from the extracted features and what methods can be used to improve it?**
   This involves performing k-fold cross-validation to estimate the classifier's real performance by calculating average accuracy and standard deviation.

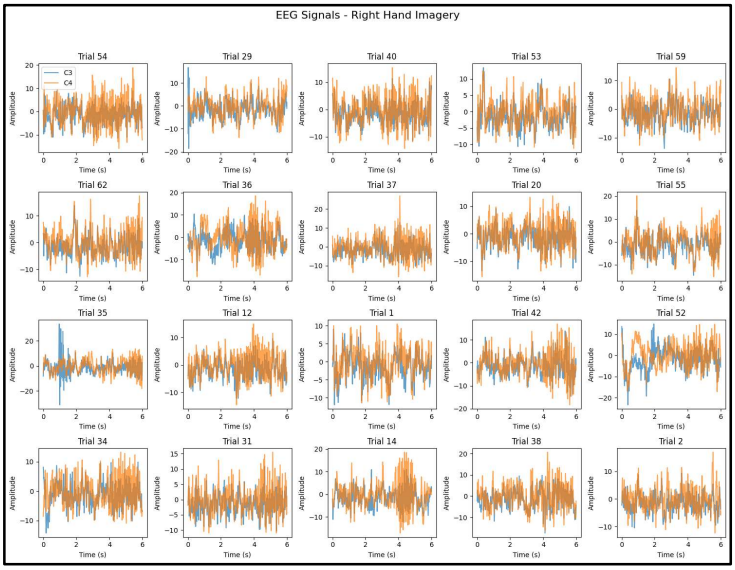
Methods:

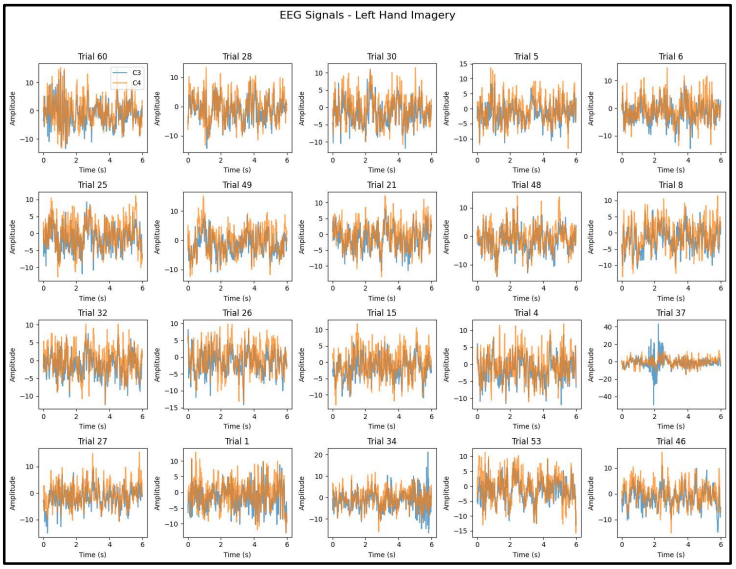
1.Visualization: We selected specific channels (C3 and C4, which are the first two channels) from the data. We then filtered the trials to keep only those corresponding to 'LEFT' and 'RIGHT' motor imagery, based on the provided labels and created a binary label array where 0 represents 'LEFT' and 1 represents 'RIGHT'.

```
# Select channels C3 and C4 (channels 0 and 1)
train_data_selected_channels = train_data[:, :, :2]
test_data_selected_channels = test_data[:, :, :2]

# Filter trials based on labels
left_trials_indices = np.where(labels[2, :] == 1)[0]
right_trials_indices = np.where(labels[3, :] == 1)[0]

# Combine indices and select the relevant data and labels
relevant_trials_indices = np.concatenate((left_trials_indices, right_trials_indices))
filtered_train_data = train_data_selected_channels[relevant_trials_indices, :, :]
filtered_labels = labels[:, relevant_trials_indices]

# Create the final binary labels array
binary_labels = np.zeros(filtered_labels.shape[1])
binary_labels[filtered_labels[3, :] == 1] = 1 # Assign 1 to RIGHT, 0 to LEFT

# Separate data by class. binary_labels - 0 for LEFT, 1 for RIGHT
left_trials_data = filtered_train_data[binary_labels == 0, :, :] # left
right_trials_data = filtered_train_data[binary_labels == 1, :, :] # right
```

Individual EEG trials were plotted for a selected number of 'LEFT' and 'RIGHT' trials. This helps to visually inspect raw EEG signals. The plots show amplitude over time for channels C3 & C4.

```
def visualize_trials(data, title, trial_indices, fs, start_imagine_sample, alpha=0.7)
    ....
    visualize_trials(left_trials_data, "EEG Signals - Left Hand Imagery", left_trial_indices, fs, start_imagine)
    visualize_trials(right_trials_data, "EEG Signals - Right Hand Imagery", right_trial_indices, fs, start_imagine)
```





*EEG Signals - **Right** Hand Imagery*

Overall amplitude: Right hand trials seem to show more balanced activity between both channels compared to left hand trials

Contralateral effect: The expected pattern (opposite brain side more active) seems less consistent in right hand imagery

*EEG Signals – **Left** Hand Imagery*

In several trials (like Trial 60, 28, 30) the C4 (orange) signal appears to have larger amplitude variations than C3 (blue). This makes sense since C4 is over the right motor cortex, which should be more active during left hand imagery.

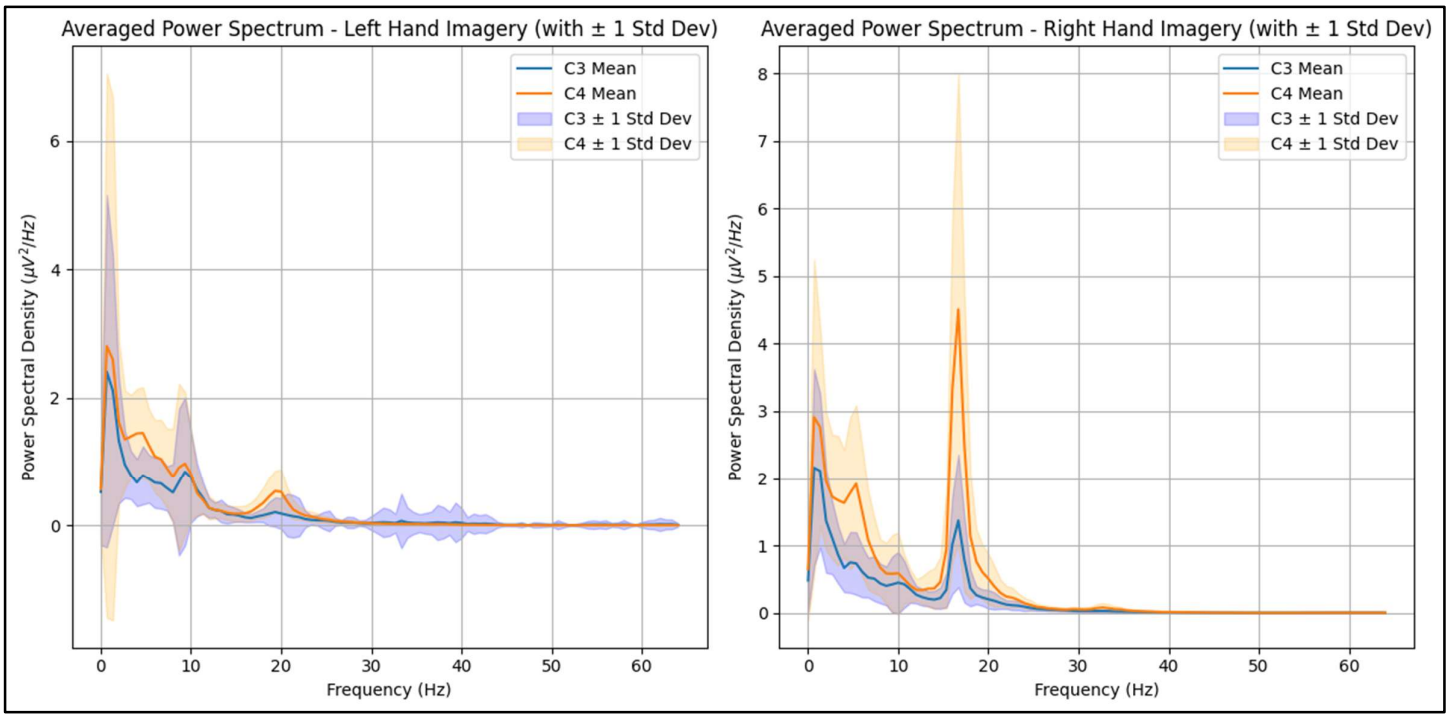
In Trial 37, the C3 signal shows much stronger activity, this is counter to the expected pattern and might point to artifacts like attention during the task.

2. Power Spectra: The Power Spectral Density (PSD) is computed for each channel and trial during the motor imagery period using Welch's method. Then the PSDs are averaged across trials for each class and standard deviation is computed.

```
f, Pxx = welch(trial_data, fs=fs, nperseg=nperseg)
....
return frequencies, np.array(averaged_spectra), np.array(std_dev_spectra)
```

The averaged power spectra for 'LEFT' and 'RIGHT' hand imagery are plotted for both C3 and C4 channels. The plots show the power spectral density as a function of frequency (Hz). The standard deviation is also visualized as a shaded region around the mean. These plots are crucial for identifying frequency bands where the power differs between the two motor imagery tasks.

```
for channel in range(num_channels):
    trial_spectra = []
    for trial in range(num_trials):
        trial_data = data[trial, start_sample:, channel]
        f, Pxx = welch(trial_data, fs=fs, nperseg=nperseg)
        if channel == 0 and trial == 0:
            frequencies = f
        trial_spectra.append(Pxx)
```



*These power spectrum plots reveal clearer differences between left- and right-hand imagery than the raw time-domain signals. The **15-19 Hz frequency range (approximately) corresponding to beta frequency shows the clearest class differences between C3 and C4 and between left and right imagery**, marking it as a potential feature to be used in the classification.*

Spectrogram Calculation: The averaged raw and relative power change (ERD/ERS) spectrograms are plotted for each channel and class. The spectrograms are computed by applying the Short-Time Fourier Transform (STFT) to overlapping segments of the signal. These visualizations provide insights into how the frequency content of the EEG signals changes over time during motor imagery. Vertical lines are added to indicate the timing of the stimulation cue and the start of the imagination period.

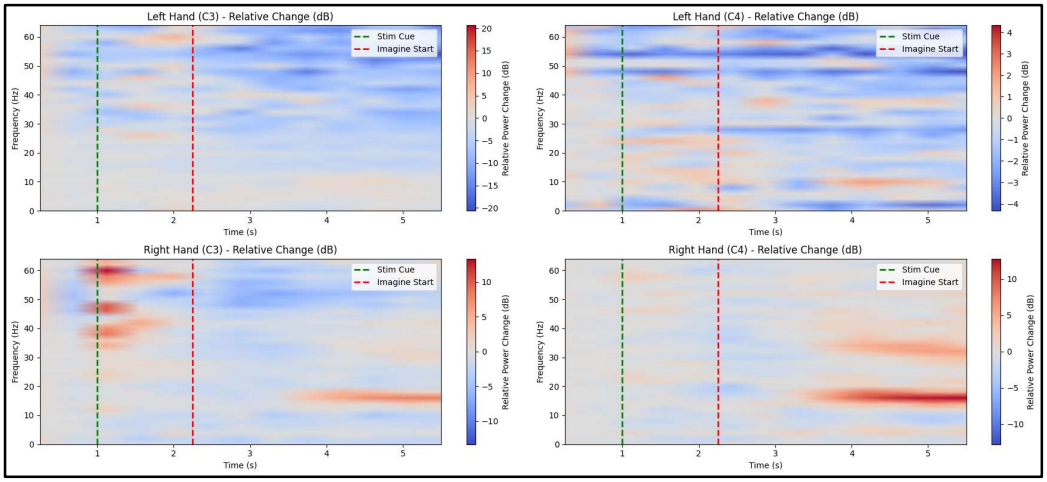
```
def compute_spectrograms(data, fs, nperseg=256, noverlap=None):
    f, t, Sxx = spectrogram(data[trial, :, channel], fs=fs, nperseg=nperseg)
    return f, t, np.array(all_spectrograms)
```

The spectrograms are then averaged across trials for each class and channel to get a representative time-frequency view for 'LEFT' and 'RIGHT' hand imagery. Using the entire trial is essential because the pre-imagery period serves as a baseline to identify event-related desynchronization (ERD) patterns, which are the key markers that distinguish between different hand movements. Additionally, analyzing the full trial captures the temporal dynamics of how brain activity transitions from rest to the imagery state.

```
f_spec_left, t_spec_left, Sxx_left = compute_spectrograms(left_trials_data, fs, nperseg=nperseg_spectrogram)
f_spec_right, t_spec_right, Sxx_right = compute_spectrograms(right_trials_data, fs, nperseg=nperseg_spectrogram)

# Average the spectrograms across trials
Sxx_left_avg = np.mean(Sxx_left, axis=1)
Sxx_right_avg = np.mean(Sxx_right, axis=1)
```

Calculating Relative Power Change (ERD/ERS): Electrocorticography (ECoG) signals are preprocessed by calculating the event-related desynchronization (ERD) and event-related synchronization (ERS). The baseline power is calculated from the first second of the trial. The relative power change is then calculated as the difference in dB between the activity during the imagination period and the baseline activity. This highlights frequency bands and time periods where brain activity is suppressed (ERD, negative dB change) or enhanced (ERS, positive dB change) relative to baseline.

```
# Calculate the baseline spectrogram (average of the *first second* of data)
baseline_left_avg = np.mean(Sxx_left_avg[:, :, :baseline_end_idx], axis=2, keepdims=True)
baseline_right_avg = np.mean(Sxx_right_avg[:, :, :baseline_end_idx], axis=2, keepdims=True)
# Calculate the relative power change (ERD/ERS) in dB
epsilon = np.finfo(float).eps
Sxx_left_relative_db = 10 * np.log10((Sxx_left_avg + epsilon) / (baseline_left_avg + epsilon))
Sxx_right_relative_db = 10 * np.log10((Sxx_right_avg + epsilon) / (baseline_right_avg + epsilon))
```



Most Informative Frequency Bands:

- *14-30 Hz (**Beta band**): We see a major power increase in C4 and C3 **right hand** imagery (lower plots) in this frequency range a few seconds after the beginning of the imagine time (red line), in contest to the **left hand** imagery (upper plots) where there is a decrease in C4 imagery and no change in C3 imagery.*
- *42-50 Hz (**Gamma band**): We see a power decrease in C4 and C3 **left hand** imagery and an increase in C4 **right hand** imagery.*
- *0.5-4 Hz (**Delta band**): We see a power decrease in C4 **left hand** imagery*

3.Feature Definition and Power Band Calculation: Based on the power spectra and spectrogram visualizations, we defined several informative frequency bands (Başar E., 2012). The total power within a specified frequency range and time window (the imagination period) for each trial was calculated. The calculate_baseline_band_power function calculates the power within a band during the baseline period. The calculated band power features were then normalized by either subtracting the baseline power or dividing by the baseline and converting to decibels (relative power change). This step helps to account for individual differences in baseline brain activity.

```
def calculate_band_power(Sxx_data, frequencies, times, time_indices_in_window, freq_range, channel_index):

        …
        freq_indices_in_band = (frequencies >= freq_range[0]) & (frequencies <= freq_range[1])
        Sxx_time_window = channel_Sxx[:, :, time_indices_in_window]
        Sxx_band_time = Sxx_time_window[:, freq_indices_in_band, :]
        total_band_power_per_trial = np.sum(Sxx_band_time, axis=(1, 2))
        .....
def calculate_baseline_band_power(baseline_Sxx_avg, frequencies, freq_range, channel_index):
        ....
        baseline_Sxx_band = channel_baseline_Sxx[freq_indices_in_band[:channel_baseline_Sxx.shape[0]]]
         total_baseline_band_power = np.sum(baseline_Sxx_band)
        .....
```

Baseline Normalization and Feature Processing: **Time-Domain Feature Extraction**: time-domain features were computed for each trial and channel within the imagination period.

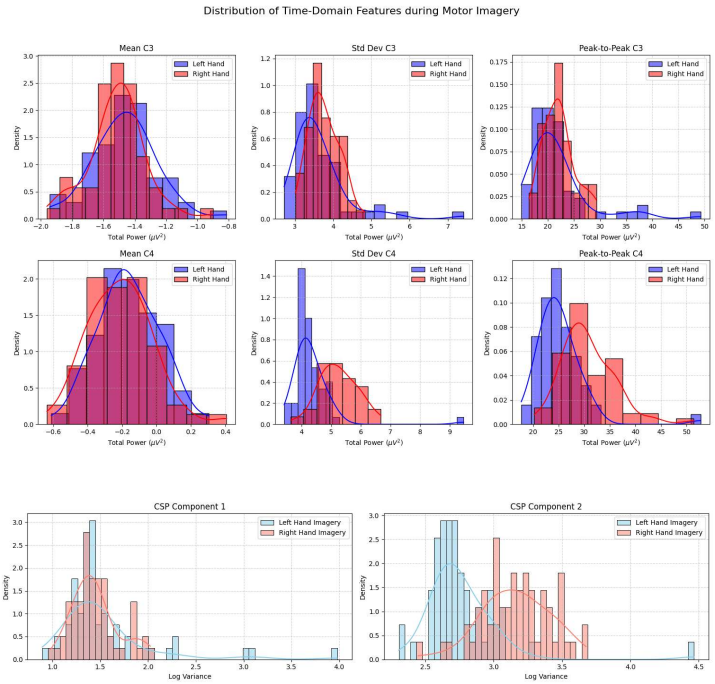
```
features[trial_idx, channel_idx * 3] = mean_val
features[trial_idx, channel_idx * 3 + 1] = std_val
features[trial_idx, channel_idx * 3 + 2] = peak_to_peak_val
```

Common Spatial Patterns (CSP) Feature Extraction: CSP is a spatial filtering technique commonly used in motor imagery BCI to find spatial filters that maximize the variance of one class while minimizing the variance of another. The calculate_csp_filters function computes these filters from the training data. The apply_csp_filters function projects the data onto the learned CSP filters, and extract_csp_features calculates features (log variance of the projected data) from the spatially filtered signals.

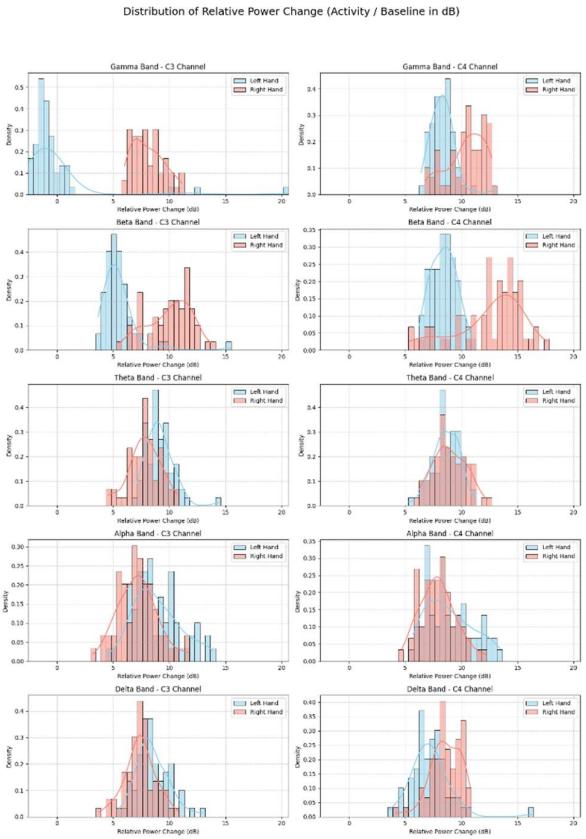
```
#Extracting time domain features:
...
for trial_idx in range(num_trials):
    for channel_idx in range(num_channels):
        data_window = data[trial_idx, start_sample:end_sample, channel_idx]
        mean_val = np.mean(data_window)
        std_val = np.std(data_window)
        peak_to_peak_val = np.max(data_window) - np.min(data_window)

        features[trial_idx, channel_idx * 3] = mean_val
        features[trial_idx, channel_idx * 3 + 1] = std_val
        features[trial_idx, channel_idx * 3 + 2] = peak_to_peak_val
...
def apply_csp_filters(data, csp_filters):
        ...
        for trial_idx in range(num_trials):
            zero_mean_data = data[trial_idx, :, :] - np.mean(data[trial_idx, :, :], axis=0)
            projected_data[trial_idx, :, :] = np.dot(zero_mean_data, csp_filters)
        ....
def extract_csp_features(projected_data):
        ....
        for trial_idx in range(num_trials):
```

```
        for component_idx in range(num_components):
            variance = np.var(projected_data[trial_idx, :, component_idx])
            csp_features[trial_idx, component_idx] = np.log(variance)
    ...
    return csp_features
```

9. **Feature Histograms**: Histograms were generated to visualize the distribution of the calculated power band features (both subtracted and relative dB) and time-domain features for the 'LEFT' and 'RIGHT' classes. This helps to assess the separability of the classes based on these features.



The standard deviation and peak-to-peak features demonstrate the most promising discriminative potential between left and right hand motor imagery, with C3 showing clearer class separation in both measures compared to C4's more overlapping distributions. Similarly, CSP Component 2 offers stronger discriminative capability with well-separated peaks and minimal



The Beta and Gamma bands show the clearest class separation, with left hand imagery trials (blue) clustering at lower relative power values and right hand imagery trials (red) clustering at higher values, particularly in the contralateral channels (C4 for

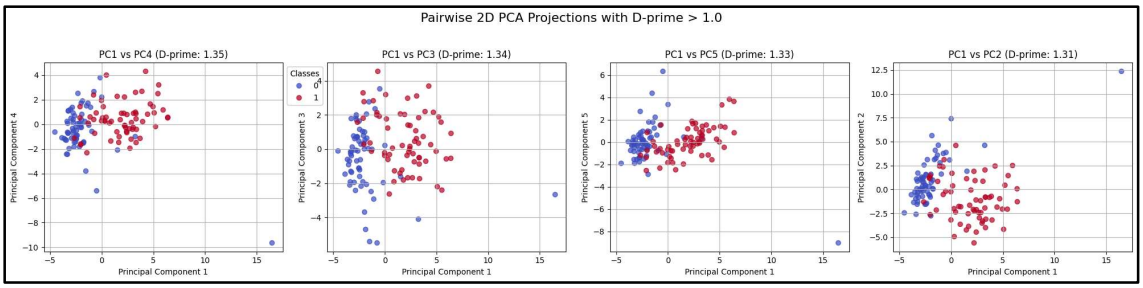
Feature Selection and Matrix Creation: To determine which of the extracted features—power-band, time-domain, or CSP—to feed into our classifier, we defined a list called selected_feature_names_list. Initially, by visually inspecting the feature plots, we hand-picked all the features that appeared to offer the best class separation. For the final model, however, we expanded our selection from the 17 initially picked features to include all 33 features we have calculated (this includes plain original band power features as well). Finally, we built the selected_features_matrix by extracting those features' columns from the training set.

Dimensionality Reduction with PCA: To condense our feature space, we first standardized selected_features_matrix with StandardScaler and then applied Principal Component Analysis (PCA). PCA projects the data onto a new set of orthogonal axes—principal components—that capture the greatest variance in descending order. Although we initially retained all 17

components by setting n_components_pca = 17, our final model uses only the top 7, since those first few components preserve most of the meaningful signal while reducing noise and improving generalization.

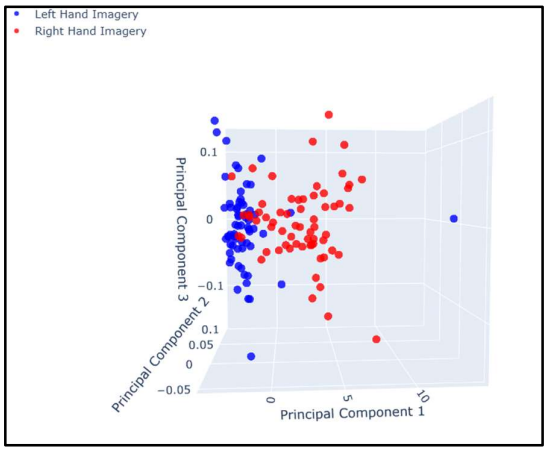
```
# Normalize the selected features
scaler_pca = StandardScaler()
scaled_selected_features = scaler_pca.fit_transform(selected_features_matrix)

# Apply PCA
pca = PCA(n_components=n_components_pca)
principal_components = pca.fit_transform(scaled_selected_features)
```

PCA Visualization (two figures, like in the assignment): The principal components were visualized in 2D (pairwise combinations) and 3D plots. Each data point (representing a trial) was colored according to its class ('LEFT' or 'RIGHT'). The calculate_dprime function was defined to quantify the separation between the two classes in each 2D PCA plane, and the d-prime value was displayed on the pairwise plots. D-prime is a measure from signal detection theory that quantifies the separation between the means of two distributions relative to their pooled standard deviation.

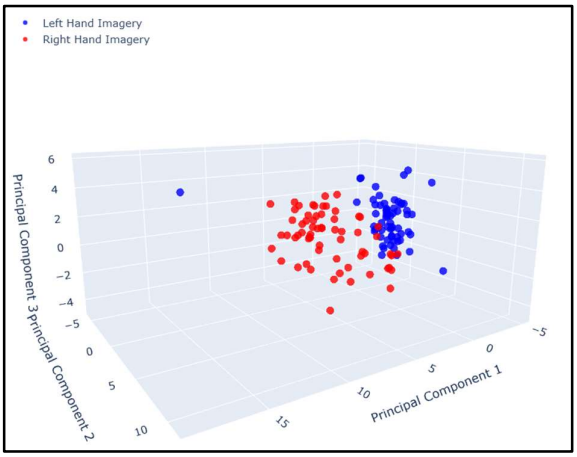


The pairwise PCA projections demonstrate moderate class separability with D-prime values ranging from 1.31 to 1.35, where blue and red points form distinct but overlapping clusters across different principal component combinations. While the consistent clustering patterns suggest meaningful feature extraction from the neural oscillation data, the persistent overlap between classes indicates that additional classification methods may be needed to achieve optimal discrimination performance.

First:



Final:



The plot demonstrates that adding the third principal component provides better class separation than the 2D projections, with the blue and red clusters showing clearer spatial separation, though some overlap between classes still remains.

4.LDA Classification: A Linear Discriminant Analysis (LDA) classifier was trained on the principal components obtained from PCA. The data was split into training and testing sets, the LDA model was fitted to the training data, and its performance was evaluated on the test set using accuracy and a confusion matrix. These are the results of our first model:

LDA Classifier Accuracy: 99.21%

Mean Cross-validation Accuracy: 0.9843

Standard Deviation of Cross-validation Accuracy: 0.0192

These are the results of our final model:

LDA Classifier Accuracy: 92.31%

Mean Cross-validation Accuracy: 0.9218

Standard Deviation of Cross-validation Accuracy: 0.0244

```
X_train, X_test, y_train, y_test = train_test_split(principal_components[:, :num_of_pca], selected_labels, test_size=0.3, stra
tify=selected_labels)
lda = LDA(solver='svd')
lda.fit(X_train, y_train)
...
accuracy = lda.score(X_test, y_test)
```

Cross-Validation: K-fold cross-validation was performed to get a more robust estimate of the classifier's performance. The data was split into 5 folds, and the model was trained and tested 5 times, each time using a different fold as the test set and the remaining folds for training.

```
k = 5
kf = KFold(n_splits=k, shuffle=True)
cv_scores = cross_val_score(lda, principal_components[:, :num_of_pca], selected_labels, cv=kf, scoring='accuracy')
```

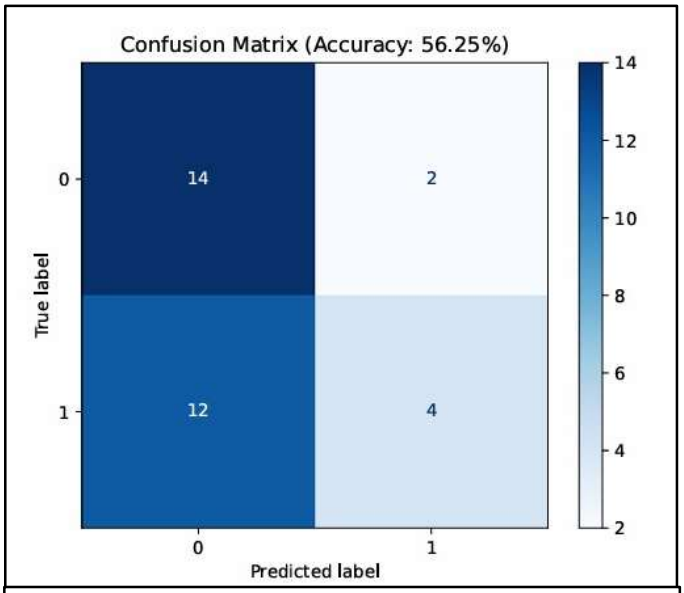
Test accuracy:

The main improvements between the first trial and the second:

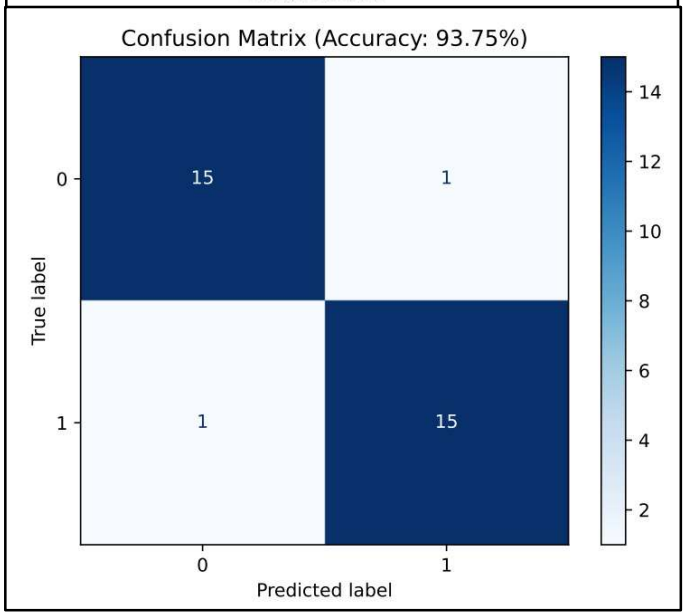
1. Making sure the test data goes through the exact same steps as the training data after the features are calculated, using the settings (like the scaler and PCA transformation) that were learned from the training data.

2. To improve generalization and reduce the risk of overfitting, we expanded the feature set from 17 to 33 and simultaneously reduced the number of PCA components from 17 to 7. Although this change led to slightly less distinct separation in the PCA space, it significantly enhanced the model's ability to generalize across data—ultimately strengthening its performance beyond the training and validation sets.

First:



Final:



Discussion

Power Spectral Analysis Results: The power spectral density analysis revealed that the beta frequency band (15-19 Hz) exhibited the clearest differences between left and right hand motor imagery, particularly in the C3 and C4 channels. The clear spectral separation in this frequency range provided strong justification for using beta-band power as a primary classification feature.

Spectrogram Analysis Results: The time-frequency analysis using spectrograms and ERD/ERS calculations demonstrated significant event-related desynchronization in the beta band for right hand imagery, while left hand imagery showed different patterns with decreased activity in the gamma band. The baseline normalization successfully highlighted these relative power changes, making the class differences more apparent than in raw power measurements. The gamma band also showed informative patterns, particularly for left hand imagery, which was an unexpected but valuable finding for feature extraction.

Feature Extraction and Histogram Analysis Results: Although our histogram analysis showed that beta and gamma band powers offered the clearest separation on the training set—left-hand trials clustering at lower values and right-hand trials at higher—the same pattern vanished on the held-out test data. In other words, our initial instinct to prune features down to only those frequency bands proved valid for training but failed to generalize. Ultimately, running PCA over all available features—frequency, time-domain, and CSP—captured the most relevant variance and yielded the strongest predictive accuracy on both the train and test sets.

PCA Dimensionality Reduction Results: By standardizing our expanded 33-feature matrix and then running PCA, we first examined all 17 components but ultimately kept only the top seven. In 2D plots of the first two principal components, left- and right-hand imagery still formed loose clusters—an overlap persisted that underlines PCA's linear limits. Extending to 3D improved visual separation, yet when we compressed into a seven-dimensional PCA space, class boundaries appeared slightly less sharp. That minor loss in apparent separability paid off handsomely in practice: reducing to seven components across all available features boosted the model's ability to generalize, delivering stronger accuracy on both training and unseen test data.

LDA Classification Performance Results: Our final Linear Discriminant Analysis classifier achieved an impressive single-split accuracy of 92.31%, which exceeded initial expectations for this challenging classification task. However, the cross-validation results provided a more realistic estimate with a mean accuracy of 92.2±2.4% indicating some variability in performance across different data splits. This was a huge improvement from the initial accuracy score that we got with our initial model (56.25%) and all the changes that took place were extremely educational.

Overall Conclusions: This project demonstrated that EEG-based motor imagery classification can achieve high and robust accuracy by combining power-spectral, time-domain, and CSP features under a PCA-based dimensionality reduction scheme. Although we believed that beta-band power will be the most informative feature on the training set—consistent with known motor cortex activity (Yu et al., 2022)—it failed to generalize without all of the additional features. Only by expanding from 17 hand-selected features to the full set of 33, then condensing into the top seven principal components, did we secure strong performance on both training and held-out test data.

The final pipeline—standardizing all features, applying PCA across the entire feature space, and training the classifier—balances discriminative power with generalization. This robust approach lays a solid foundation for real-world brain–computer interface applications.

References

Başar, E. (2012). Brain Oscillations: Principles and Approaches. Springer. https://doi.org/10.1016/S0165-0173(98)00056-3

Yu, H., Ba, S., Guo, Y., Guo, L., & Xu, G. (2022). Effects of Motor Imagery Tasks on Brain Functional Networks Based on EEG Mu/Beta Rhythm. *Brain sciences*, *12*(2), 194. https://doi.org/10.3390/brainsci12020194