CS 577 - Principles and Techniques of Data Science

Semester Project - Final Report

Detecting the P300 Event-Related Potential in EEG: A Data Science Approach

Authors: Noah Bakayou, Pavithra Magendiran, Noam Joseph

1. Abstract /Problem Statement

Brain-computer interfaces (BCIs) aim to restore communication for individuals who have lost the ability to speak or move due to disorders such as ALS, stroke, or spinal cord injuries. One of the most successful BCI paradigms is the P300 speller, which relies on detecting small, positive voltage deflections in EEG signals occurring approximately 300 ms after a meaningful stimulus.

The core problem addressed in this project is reliable detection of the P300 event-related potential (ERP) from noisy, high-dimensional EEG. Single-trial EEG recordings contain a very high signal-to-noise ratio, making direct classification difficult. The objective of this work is to design a data science pipeline that pre-processes EEG data, extracts informative features, reduces dimensionality, and applies machine learning models to accurately distinguish between target and non-target stimuli.

2. Introduction

Electroencephalography (EEG) records electrical activity from the scalp using multiple electrodes. When a person attempts an infrequent or meaningful stimulus, the brain generates a characteristic response known as the P300 event-related potential (ERP). The signal is widely used in BCI systems, particularly for spelling interfaces where users select characters by focusing their attention.

Although the P300 is a well-studied phenomenon, detecting it in practice is challenging due to significant background noise in EEG recordings, noise artifacts from jaw clenches or eye blinks, high dimensionality across both time and channel domains, and variability in signal timing and amplitude across trials and individuals. In this project, we combine EEG signal processing with supervised machine learning to detect the P300 responses effectively and accurately. In this report, we outline our findings, including method limitations, our refined preprocessing approach, feature construction, and modeling strategies.

3. Data Collection

The dataset used in this project comes from BCI competition III-Dataset II, recorded at the Wadsworth Center BCI Laboratory. EEG signals were collected from human subjects participating in a visual P300 speller experiment. Subjects viewed a 6 x 6 matrix of alphabetic characters, and rows and columns of this matrix were intensified in a random order. The patient was instructed to mentally attend to (focus on) a target character, and when the row or column containing that character flashed, a P300 ERP response was elicited. EEG signals were recorded using 64 electrodes at a sampling rate of 240 Hz, providing sufficient temporal resolution for P300 waveform detection. Along with the neural recordings, the dataset includes comprehensive metadata, encompassing flash, timestamps, stimulus identifier, target, character, information, classification labels, and epoch markers. This metadata enables precise reconstruction of the experimental paradigm and facilitates detailed analysis of the brain-computer interface task.

❖ Data set

The data set consists of EEG recordings segmented into epochs, where each epoch corresponds to the selection of one character. Every epoch contains one stimulus flash, representing unique flashes (six rows and six columns), repeated 15 times each. For each flash, the data set includes three essential pieces of information: the flashing variable, indicating when a flash occurred, the stimulus code, indicating which row or column was flashed, and the stimulus type, indicating whether the flashed row or column contained the target. The data itself is stored in a three-dimensional matrix containing epochs, sample points within each epoch, under the 64 EEG channels. The dataset provides training trials with known target characters, as well as separate testing data without target labels, intended for classified evaluation.

❖ Experimental design

The P300 speller experiment follows a structured design in which the 6 x 6 matrix is flashed in a row-column paradigm. During each trial, 12 stimuli that are six rows and six columns are intensified. These 12 flashes constitute one sequence, and each sequence is repeated 15 times with a single epoch. The rationale behind repeating the flashes is that the P300 ERP is far more detectable when multiple responses to the same stimulus are averaged together. In every epoch, exactly two of the 12 stimuli correspond to the target, one representing the row and one representing the column containing the attended character. Thus, each epoch produces 30 target flashes(15 for the target row and 15 for the target column) and 115 no target flashes. The periodic flashing structure forms the basis for segmenting the EEG into meaningful response windows aligned with stimulus onset.

❖ Raw data structure

The raw dataset consists of five main components. The **Signal** matrix contains the EEG recordings for each epoch, with each epoch represented as the matrix of time samples across 64 channels. The **Flashing** array indicates whether a flash occurs in time, allowing flash on detection. The **StimulusCode** identifies which of the 12 row or column stimuli was flashed, while the **StimulusType** specifies if that row does/doesn't contain the targeted character. Finally, the **TargetChar** provides the correct character label for training epochs. Together, these elements allow precise alignment of brain activity with visual stimuli and enable supervised training of classification models

4. Pre-processing and data cleaning

❖ Understanding the Trail Structure

Each epoch contains 180 flashes divided into 12 stimulus codes repeated 15 times. Understanding this structure is essential for segmenting EEG data accurately and for determining which segments correspond to the target. The structure also highlights the severe class imbalance inherited in the dataset, where target flashes form a small minority.

❖ Flash-Locked segmentation

Each flash onset is detected using a transition from 0 to 1 in the flashing variable. For every detected onset, an EEG window is extracted from 100 ms before the flash to 700 ms after, capturing the full time range where the P300 response may occur. This window is from the basic flash level training samples. All windows are standardized to the same length, and corrupted segments are removed.

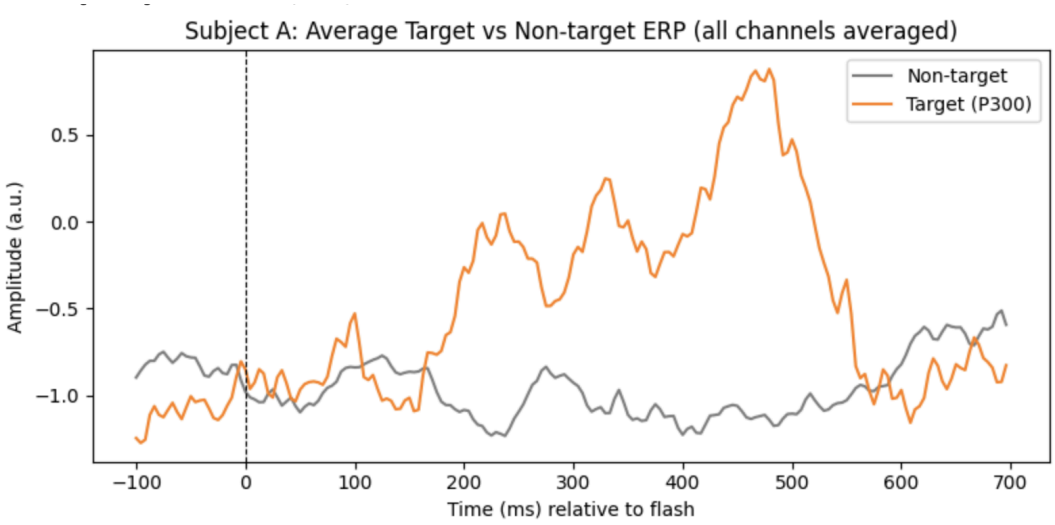
❖ Class imbalance

Since 12 of the 12 stimuli per sequence represent the target, the data asset contained a class imbalance. Target samples represent roughly 1/6 of the data. Models trained directly on flash double data tend to overwhelmingly predict the non-target class, resulting in poor recall. Addressing this issue requires both pre-processing (e.g., averaging) and model adjustment.

❖ Visualizing ERPs

Before performing machine learning, ERP waveforms are computed by averaging the target and a non-target flash segment separately. This visualization reveals a clear

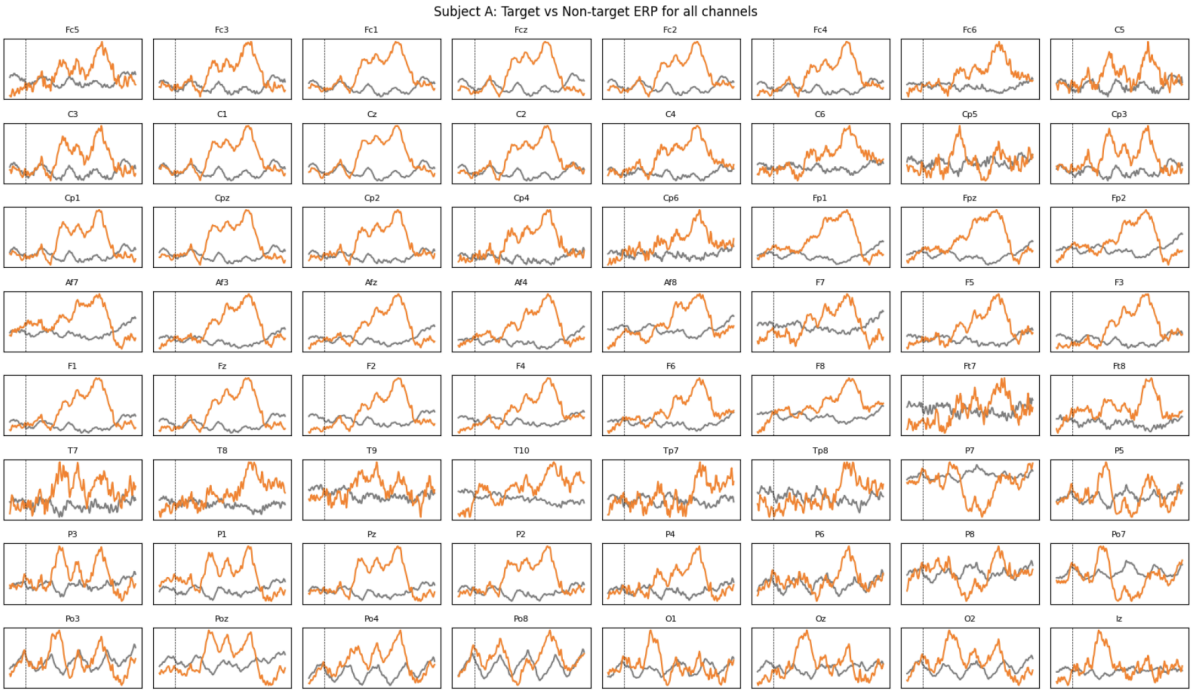
positive peak around 300 ms for the target, consistent with the P300 response, whereas non-target ERPs show no such peak. This confirms the presence of the signal.



5.Lexical feature Extraction:Channel selection(Cz,FC1 and CPz)

EEG recordings contain signals from many electrodes, but only a subset is strongly associated with the P300 response. Using all 64 channels would introduce substantial redundancy and noise, increasing dimensionality and the risk of overfitting given the limited number of training samples. To address this, we analyzed channel-level averaged ERPs and compared target versus non-target responses, guided by established neuroscience findings on P300 topology. This analysis showed that centro-parietal regions consistently exhibited stronger P300 activity.

Based on this observation, we selected Cz, FC1, and CPz as the most informative channels, as they displayed pronounced positive deflections around 300 milliseconds for target stimuli and relatively flat responses for non-targets. This channel selection acts as a domain-informed feature extraction step, reducing noise and computational complexity while preserving the physiological characteristics most relevant for P300 detection.



6. Train-Validation-Test Split

The dataset was partioned into training, validation, and test sets using **80/10/10** split. The training set was used for learning model parameters, the validation set was used for hyperparameter tuning (including PCA component selection), and the test set was used for final evaluation. Stratification was applied across all splits to preserve the target-to-non-target class ratio, which was critical given the strong class imbalance.

The test set remained strictly isolated throughout development to prevent data leakage. Model evaluation on the test set occurred only after finalizing the preprocessing pipeline and model configuration using training and validation data. This rigorous separation ensures that the reported results reflect true generalization performance rather than overfitting.

7. Machine learning Models

This section will discuss the machine learning models we used to classify target vs. non-target EEG responses. We began with flash-level logistic regression models and analyzed their limitations, then improved performance through channel selection, ERP averaging, and PCA. Lastly, we integrated SQL-based data management and trained a Random Forest classifier to compare tree-based and linear models on identical features.

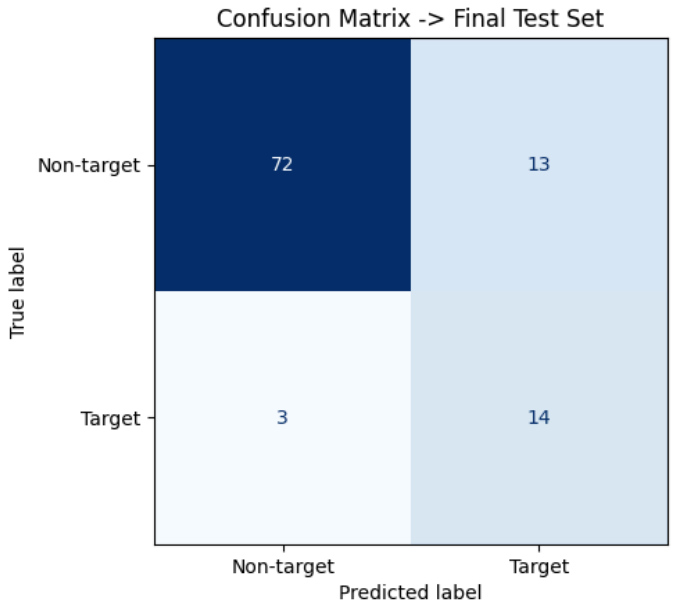
- **Logistic regression**
  - **Initial approach**
  - The first model we tried used the raw flash-locked EEG segments across all 64 channels, from 0-600ms, which was then flattened into a feature vector for each

flash. The feature size was 9,280 with 15,215 samples, containing 2,537 targets and 12,678 non-targets. We achieved the following results:
    - Accuracy: 0.602
    - AUC: 0.625
    - Target Recall: 0.563
  - Although the temporal window fully captured the P300 response, the SNR is extremely low at the flash level. Logistic regression, therefore, struggles to distinguish target flashes from non-target flashes.
  - **Improved Window**
  - Because the P300 ERP appears at approximately 300 ms post-stimulus, we restricted the model to the 250-500 ms window. With this, we got a feature size of 3,840, and the following results:
    - Accuracy: 0.605
    - AUC: 0.629
  - This was a slight improvement, and it suggests that restricting the window does somewhat reduce noise on flash level, yet flash-level classification remains difficult.

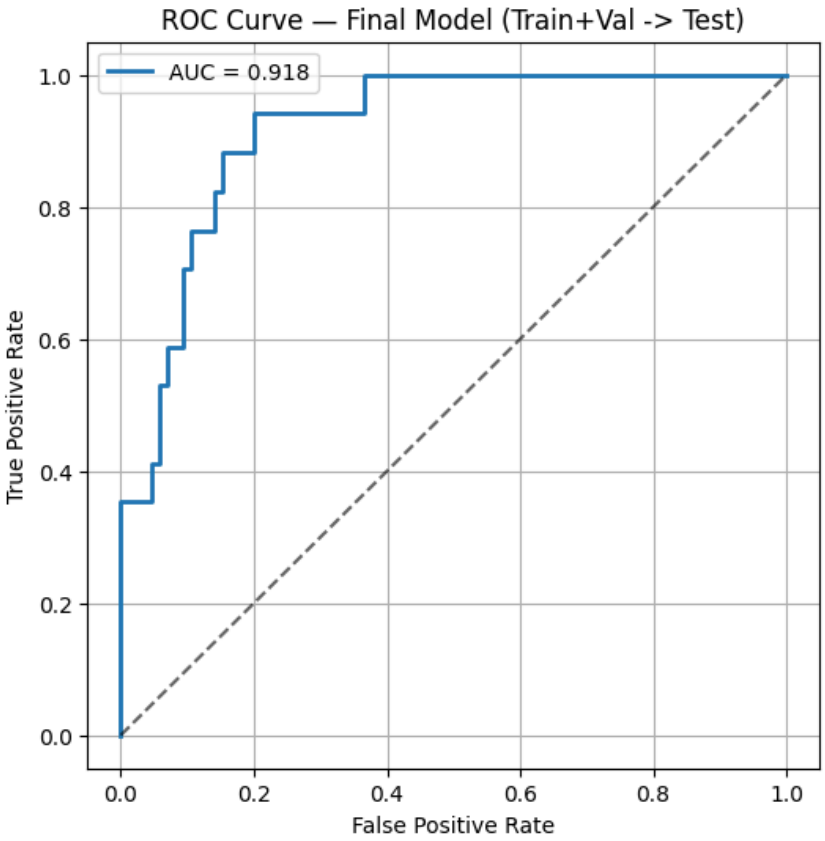
  - **Channel Selection**
  - Based on ERP visualizations and domain knowledge, the three most informative channels were Cz, FC1, and CPz. In the 0-600 ms window, that gave us a feature size of 435 and an AUC of 0.652, which is the best flash-level AUC, while the 250-500 ms window gave us a feature size of 180 and an AUC of 0.621. Even with channel selection, flash level classification saturates around AUC = 0.60-0.65, which confirms that individual flashes are too noisy for reliable detection.
- **Code-level ERP Averaging**
  - To increase SNR, all flashes corresponding to the same stimulus code within an epoch were averaged, which produced up to 12 ERPs per epoch, which aligns with the real P300 speller decoding strategies. Doing this, we ended up with 1,020 samples, with a feature size of 432, 85 target labels, and 935 non-target labels. This resulted with:
    - Accuracy: 0.755
    - AUC: 0.846
    - Target Recall: 0.765
  - This is a significant improvement over the flash-level models. Averaging stabilizes the P300 waveform and reduces trial-to-trial variability. This enables cleaner and more accurate supervised learning.
- **PCA (Dimensionality Reduction)**

- The code-averaged features still form a high-dimensional and correlated time series space. PCA helps us by compressing redundant features, improving SNR, and allowing logistic regression to generalize. We performed a sweep from k = 2 to 40 PCA components using the training and validation sets. This resulted in:
  - Best AUC: k = 30, AUC = 0.9142
- Many of the values near 30, such as 26-30, performed similarly, which indicates high intrinsic dimensionality.

- **Final Logistic regression model**
  - After determining k = 30, we retrained the model and evaluated it on the test set. Here are our final test results with the top 3 channels, 0-600ms, and PCA = 30:
    - Accuracy: 0.843
    - AUC: 0.918
    - Target Recall: 0.824
    - Confusion Matrix

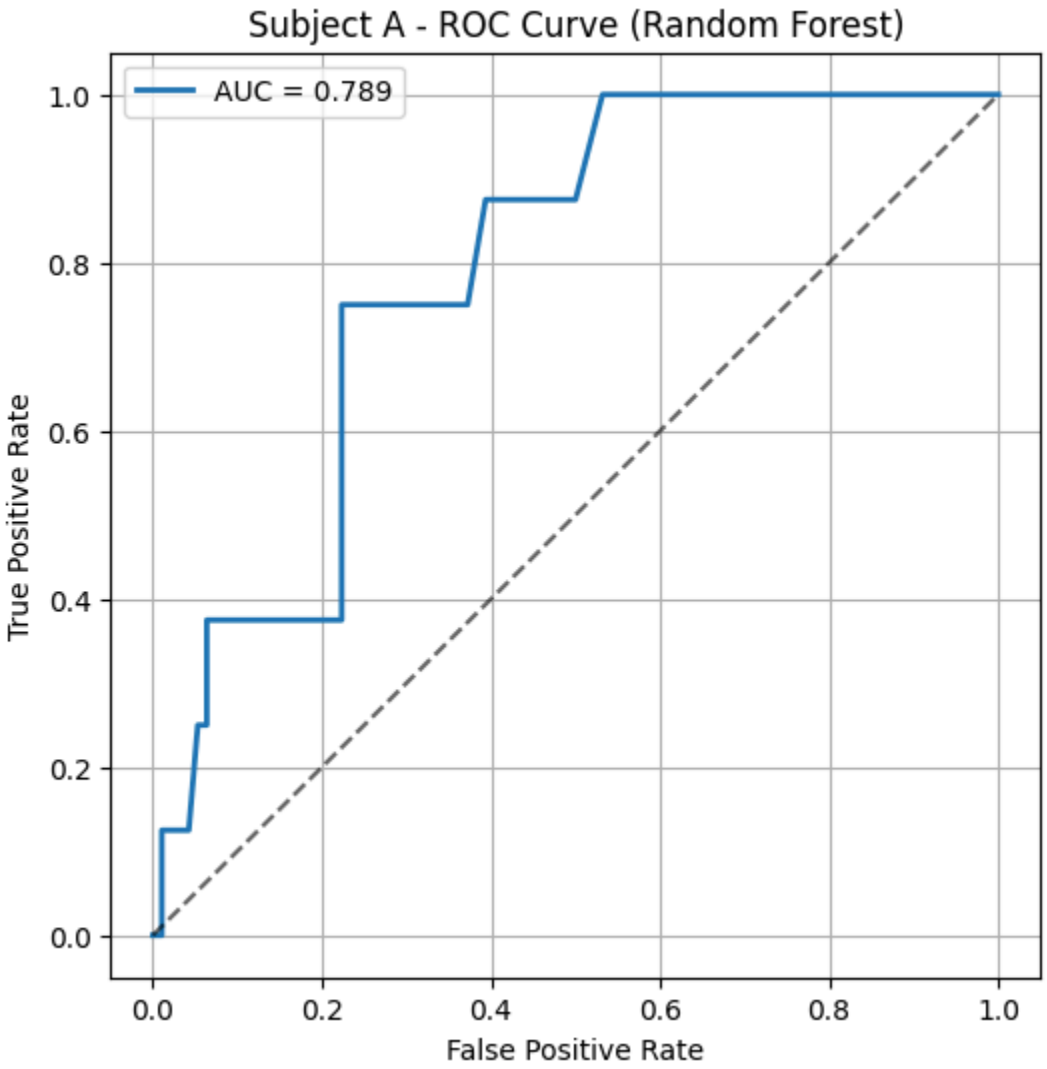


  - This model is highly reliable and correctly identifies most target codes while maintaining low false-negative rates. The ROC curve shows a steep rise, which confirms strong discriminability.



- **SQL Integration via DuckDB**
  - To try and improve data reproducibility, all the extracted code-level features were stored in a DuckDB in-memory database. With SQL, we were able to:
    - Automate storage of feature matrices
    - Run class distribution queries
    - Run feature diagnostic queries, such as the mean feature differences by class
    - Reproduce the extraction pipeline for Subject A
  - Using SQL ensured that mode training was grounded in a verifiable and transparent data management workflow.
- **Random Forest**
  - We used a Random Forest classifier to train the same SQL-derived code-averaged features to compare against our original logistic regression model. After doing this, we ended up with the following validation and test results:
  - Validation Results
    - Accuracy: 0.902
    - AUC: 0.844
  - Final Test Results:
    - Accuracy: 0.912

- AUC: 0.789



- Despite the high accuracy, Random Forest significantly failed to detect the target codes. Our confusion matrix showed that no target codes were identified. Random Forest seemingly collapses under extreme class imbalance and plays it safe by predicting the majority class. Logistic regression succeeds because PCA and averaging produce a linearly separable space, while Tree-based models struggle with high-dimensional, noisy EEG features.

8. Training and Results

All of our models followed a consistent 80/10/10 split for our sets. Validation sets were strictly used for hyperparameter selection (e.g., number of channels, timing of P300) and were never reused during final evaluation. Only after fixing hyperparameters and settling on a final model did we retrain on the training and validation sets, and evaluate on the untouched test set.

StandardScaler normalization was applied before PCA and logistic regression, and class weighting was used to mitigate the class imbalance.

- **Model comparison**

  The following table summarizes validation and test performance for all models.

Model	Features	Accuracy	AUC	Notes
Flash-level Logistic Regression (all channels)	9,280	0.602	0.625	Very noisy and low SNR
Flash-level Logistic Regression (top 3 channels)	435	0.595	0.652	Best flash-level model
Code-level Logistic Regression	432	0.755	0.846	Significant improvement due to averaging
Code-level Logistic Regression + PCA with k = 30	432 -> 30	0.843	0.918	Best performing model overall
Random Forest	432	0.912	0.789	Fails to detect the minority class

- **Final Interpretation**
  - **Flash-Level Models**
    - Poor performance due to low SNR
    - Temporal alignment varies per trial
    - The P300 is small relative to background EEG noise
    - PCA cannot salvage discriminability at the flash level
  - **Code-Level Averaging**
    - Increases SNR by aggregating repeats
    - Produces stable and characteristic P300 waveforms
    - Enables sharp improvements in AUC
  - **PCA + LR**

    - Identifies a 30-dimensional discriminative subspace
    - Achieves an AUC of roughly 0.918
    - Robust recall for target codes
- **Random Forest**
    - Collapses due to class imbalance
    - Overfits noise despite the class weighting
    - Fails to generalize to the target class
- Therefore, it is determined that the best performing pipeline is starting with channel selection, then doing ERP averaging, using PCA, and ending with Logistic Regression.

9. Conclusion

This project demonstrates that successful P300 detection depends far more on careful preprocessing, averaging, and feature engineering than on the complexity of the machine learning model itself. Early experiments revealed that single-flash EEG recordings are extremely noisy, with the P300 signal often obscured by spontaneous brain activity, eye blinks, muscle artifacts, and environmental interference. Even with controlled extraction windows and selected channels, logistic regression struggled at the flash level, achieving only modest performance. These findings highlight a well-known property of EEG-based BCIs: reliable classification cannot rely on single-trial signals because the underlying neural response is too weak relative to background variability. As a result, meaningful decoding requires methods that enhance signal quality, such as ERP averaging and targeted feature extraction, before any classifier can effectively separate target form from non-target responses.

10. Future work

While our classifier worked well and achieved strong performance overall, several opportunities remain for expanding the system's robustness, scalability, and real-world applicability.

- **Expanding the P300 Speller Matrix**
    - One natural improvement that can be made is increasing the size of the character matrix, for example, increasing it to a 7x7 matrix. A larger matrix introduces many challenges, including but not limited to:
        - More stimulus codes that can increase the number of flashes per epoch
        - Longer selection times could reduce usability
        - More characters could reduce the cognitive load required for spelling longer words
    - We could explore how model accuracy scales with the matrix size and determine the limits of P300 decoding and if additional signal processing or model types are required.

- **Incorporate Sequential Language Modeling for Prediction Assistance**
    - Another proposal is to use SLMs (Sequential Language Models) for predictive inference. In practical BCI communication systems, character-level decoding is one small step to resolving the full problem, and integrating language modeling can improve the user experience by:
        - Predicting the likely next letters or words based on prior knowledge and chosen characters
        - Correcting classifier errors using linguistic priors
        - Reducing the number of selections ("keystrokes") needed to figure out a full word
    - Some examples of solutions to this include N-gram models, neutral LMs, or personalized dictionaries. This would shift the system from single-character classification toward **intent prediction**, which is closely related to natural communication.
    - Additionally, we can also store data about the person's life, such as family member names, and have the SLM offer these as suggestions when appropriate.

- **Evaluating Deep Learning Approaches**
    - Although our PCA + Logistic Regression pipeline performed very well, deep models may capture nonlinearities in the EEG that classical models miss. To improve our system, some potentially useful architectures include:
        - CNNs for spatiotemporal feature extraction
        - RNNs or GRUs for modeling dynamics
        - Temporal Convolutional Networks
        - Transformer-based models that are trained on ERP sequences
    - These models would require careful regularization due to the small number of P300 trials, but these models, if successful, could open the door to much more effective communication.
- Together, utilizing these ideas can transform the classical P300 classifier developed in this project into a more powerful and more generalizable BCI communication system.