LFP Analysis Pipeline

Documentation for Neural Signal Analysis and Anxiety State Detection

This document provides comprehensive guidance for using the LFP (Local Field Potential) analysis pipeline developed for the Provenza Lab. The system is designed to analyze neural signals from DBS (Deep Brain Stimulation) patients with OCD and investigate relationships between frequency band power and anxiety levels measured through SUDS (Subjective Units of Distress Scale).

Table of Contents

1. Project Structure

The project is organized into modular components for maintainability and reusability:

File	Description
`classes/__init__.py`	Package initialization, exports LFPData and BandAnalysis classes
`classes/LFPData.py`	Core class for loading, preprocessing, and managing LFP data
`classes/BandAnalysis.py`	Frequency band power analysis (Theta, Alpha, Beta)
`plot_functions.py`	Visualization functions for LFP signals and analysis results
`analysis.ipynb`	Main Jupyter notebook demonstrating the complete analysis workflow

2. Dependencies and Setup

Required Libraries

- **numpy** - Numerical computing
- **pandas** - Data manipulation and analysis
- **scipy** - Signal processing (scipy.signal)
- **matplotlib** - Static visualizations
- **seaborn** - Statistical visualizations
- **plotly** - Interactive visualizations
- **scikit-learn** - Machine learning models

Installation

```
pip install numpy pandas scipy matplotlib seaborn plotly scikit-learn
```

3. Core Classes

3.1 LFPData Class

The LFPData class handles all data loading, preprocessing, and phase management operations for LFP signals.

Initialization

```
patient = LFPData(patient_id="AA002", session_date="2025-04-10", fs=250)
```

Parameters:

- `patient_id` (str): Unique patient identifier
- `session_date` (str): Recording session date
- `fs` (int): Sampling frequency in Hz (default: 250)

Key Methods

Method	Description
`load_data(lfp_path, task_path)`	Load LFP CSV and task data files. Automatically detects channel naming convention.
`interpolate_nans(plot=False)`	Detect and interpolate NaN values in LFP signals using linear interpolation.
`demean()`	Remove DC offset by subtracting the mean from each channel.
`preprocess(...)`	Complete preprocessing pipeline: NaN interpolation, demeaning, artifact removal, phase definition.
`define_phases()`	Extract experimental phases (Baseline, Exposure, Compulsions, Relief) from task markers.

3.2 BandAnalysis Class

The BandAnalysis class performs frequency band power calculations using Welch's method for spectral estimation.

Initialization

```
analysis = BandAnalysis(lfp_data=patient, freq_bands=None, window_sec=5.0)
```

Default Frequency Bands:

- **Theta:** 4-8 Hz
- **Alpha:** 8-13 Hz
- **Beta:** 13-30 Hz

Key Methods

Method	Description
`calculate_all_bands()`	Calculate power for all frequency bands continuously across the signal.
`get_band_continuous(band, hemi)`	Retrieve continuous power values and timestamps for a specific band and hemisphere.
`get_band_by_phase(band, hemi)`	Split band power data by experimental phases.
`align_with_suds(band, hemi)`	Align band power with SUDS scores using nearest neighbor matching.

4. Analysis Workflow

The complete analysis follows a structured workflow from data loading to machine learning classification.

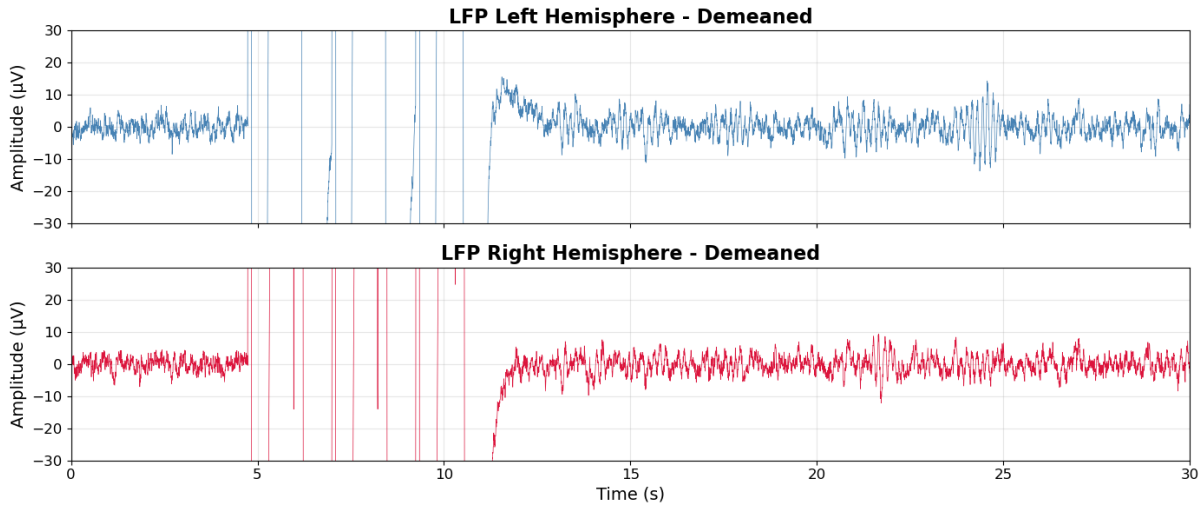
Step 1: Load Data

```
from classes import LFPData, BandAnalysis
patient = LFPData("AA002", "2025-04-10_Session1")
patient.load_data(lfp_csv_path, task_csv_path)
```

Step 2: Inspect Raw Signals

Before preprocessing, visualize the raw signals to identify artifacts at the beginning and end of the recording:

```
plot_raw_lfp(patient, time_start=0, time_end=30)  # First 30 seconds
plot_raw_lfp(patient, time_start=1230, time_end=1260)  # Last 30 seconds
```



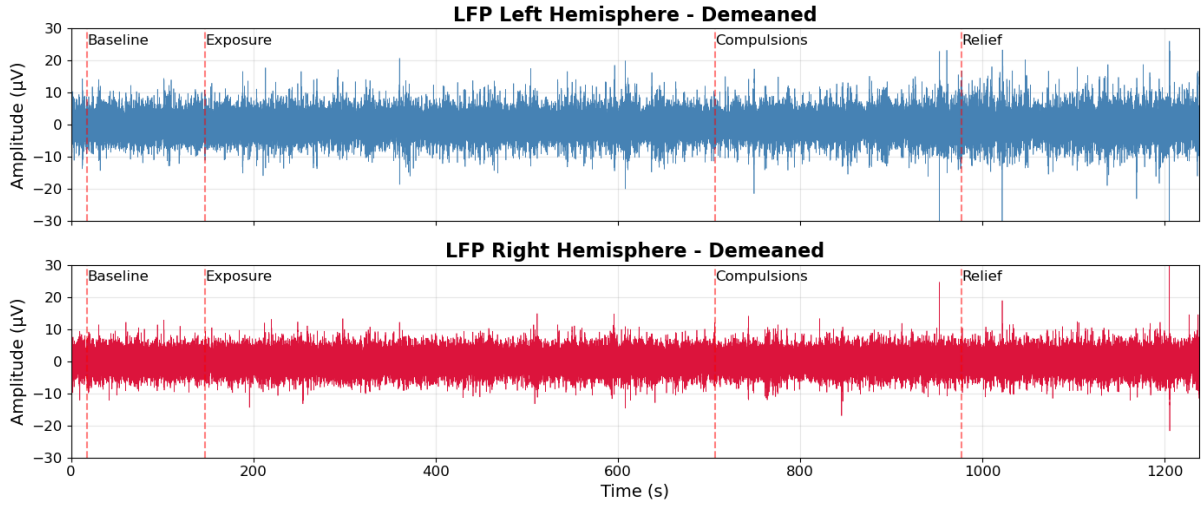
Step 3: Preprocess

Apply preprocessing with artifact exclusion based on visual inspection:

```
patient.preprocess(artifact_start_sec=10, artifact_end_sec=5,
plot_nans=True)
```

This performs:

1. NaN interpolation (linear)
2. Mean removal (demeaning)
3. Task data cleaning and SUDS extraction
4. Artifact removal from signal edges
5. Phase definition (Baseline, Exposure, Compulsions, Relief)



Step 4: Band Power Analysis

```
analysis = BandAnalysis(patient, window_sec=2.0)
analysis.calculate_all_bands()

# align with suds
powers, suds_aligned = analysis.align_with_suds('Theta', 'left')
# timeseries plots
plot_band_suds_timeseries(analysis, 'Theta')
```

Step 5: Create Analysis DataFrame

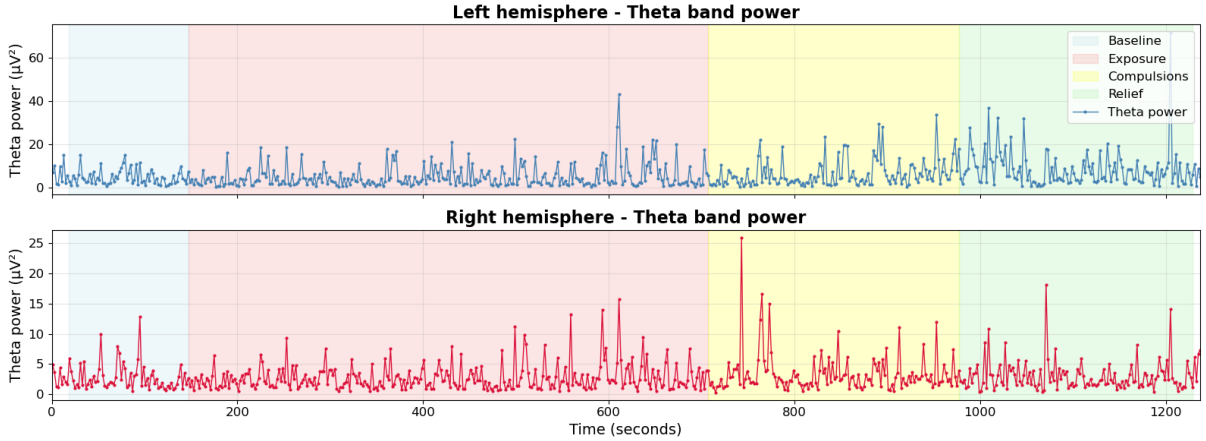
```
powers, suds_aligned = analysis.align_with_suds('Theta', 'left')
df = pd.DataFrame({
    'time': analysis.results['Theta']['times'],
    'Theta_left': analysis.results['Theta']['left'],
    # ... add other bands
```

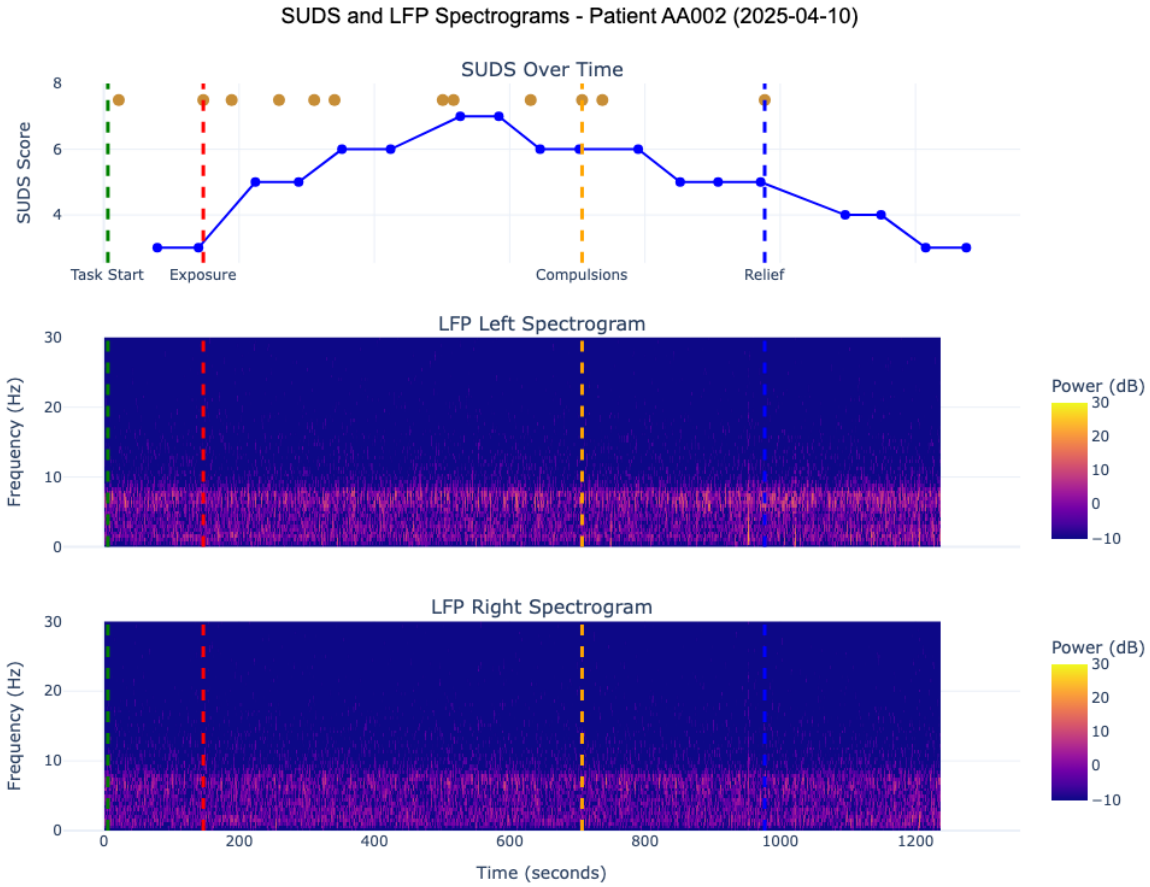
```
})
```

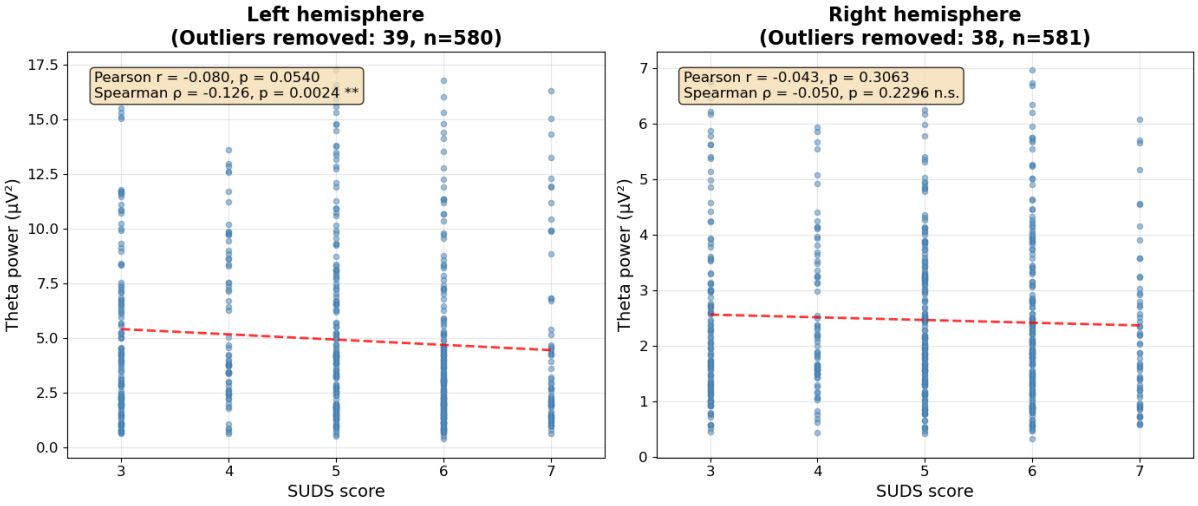
5. Plotting Functions

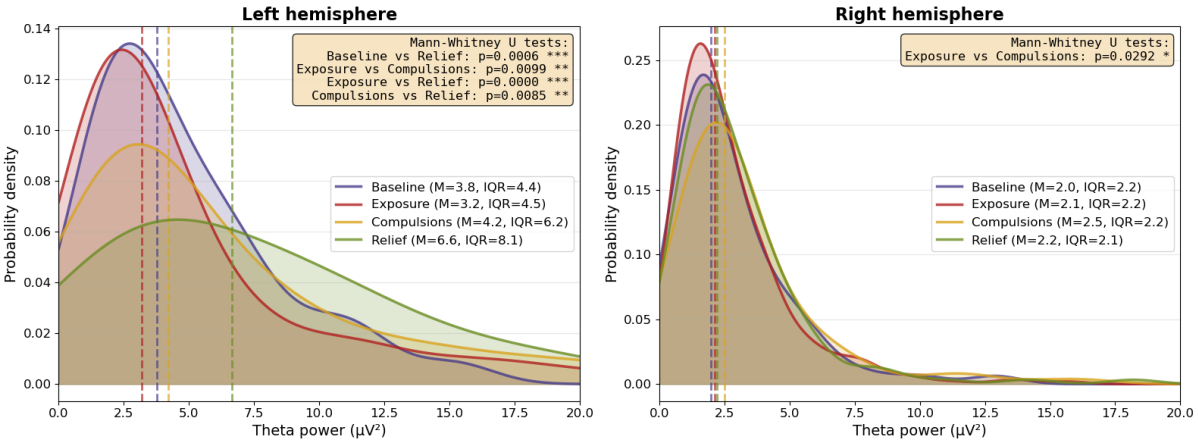
The plot_functions.py module provides comprehensive visualization tools for LFP analysis.

Function	Description
`plot_raw_lfp(patient, ...)`	Plot raw LFP signals with optional time window and phase markers.
`plot_spectrograms(patient, ...)`	Generate time-frequency spectrograms for both hemispheres.
`plot_band_continuous(...)`	Visualize band power over time with experimental phase highlighting.
`plot_suds_correlation(...)`	Scatter plots with Pearson/Spearman correlations between band power and SUDS.
`plot_band_distributions(...)`	Compare band power distributions across experimental phases.
`plot_suds_and_spectrograms(...)`	Interactive Plotly visualization combining SUDS timeline with spectrograms.




SUDS and LFP Spectrograms - Patient AA002 (2025-04-10)
SUDS Over Time
SUDS Score
8
6
4
Task Start
Exposure
Compulsions
Relief
LFP Left Spectrogram
Frequency (Hz)
30
20
10
0
Power (dB)
30
20
10
0
−10
LFP Right Spectrogram
Frequency (Hz)
30
20
10
0
Power (dB)
30
20
10
0
−10
0
200
400
600
800
1000
1200
Time (seconds)


Left hemisphere
(Outliers removed: 39, n=580)
Pearson r = -0.080, p = 0.0540
Spearman ρ = -0.126, p = 0.0024 **
Theta power (μV²)
SUDS score
Right hemisphere
(Outliers removed: 38, n=581)
Pearson r = -0.043, p = 0.3063
Spearman ρ = -0.050, p = 0.2296 n.s.
Theta power (μV²)
SUDS score


Left hemisphere
Mann-Whitney U tests:
Baseline vs Relief: p=0.0006 ***
Exposure vs Compulsions: p=0.0099 **
Exposure vs Relief: p=0.0000 ***
Compulsions vs Relief: p=0.0085 **
Baseline (M=3.8, IQR=4.4)
Exposure (M=3.2, IQR=4.5)
Compulsions (M=4.2, IQR=6.2)
Relief (M=6.6, IQR=8.1)
Probability density
Theta power (μV²)
Right hemisphere
Mann-Whitney U tests:
Exposure vs Compulsions: p=0.0292 *
Baseline (M=2.0, IQR=2.2)
Exposure (M=2.1, IQR=2.2)
Compulsions (M=2.5, IQR=2.2)
Relief (M=2.2, IQR=2.1)
Probability density
Theta power (μV²)

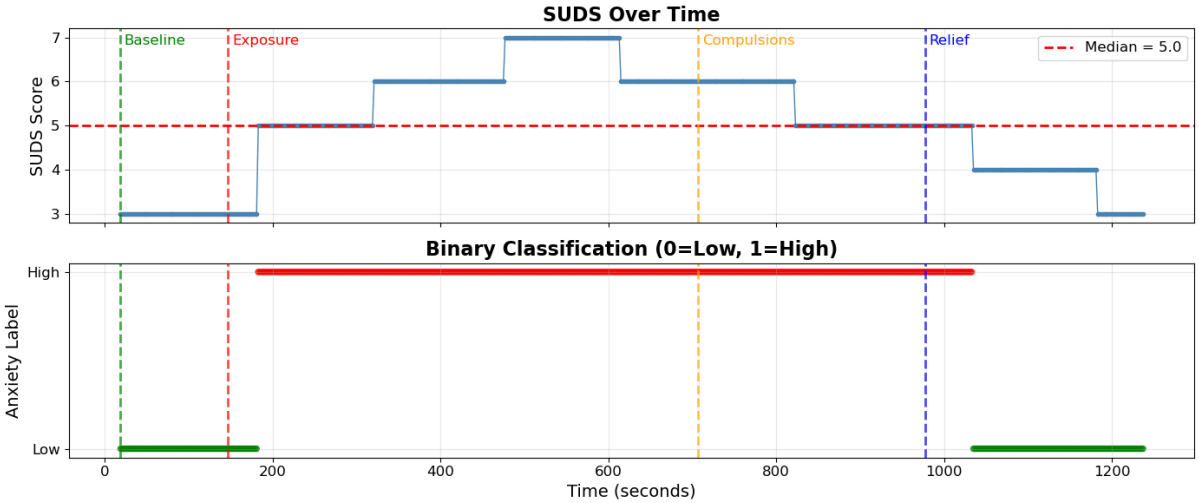
6. Machine Learning Pipeline

The notebook implements two classification approaches to detect stress/anxiety states from neural features.

6.1 Model 1: Anxiety Detection (SUDS-based)

Binary classification of High vs Low anxiety based on median SUDS score.

```
median_suds = df['suds'].median()
df['anxiety_binary'] = (df['suds'] >= median_suds).astype(int)
```



6.2 Model 2: Stress Detection (Phase-based)

Binary classification of Stress (Exposure + Compulsions) vs No Stress (Baseline + Relief) based on experimental phases.

```
stress_phases = ['Exposure', 'Compulsions']
df['stress_binary'] = df['Phase'].isin(stress_phases).astype(int)
```

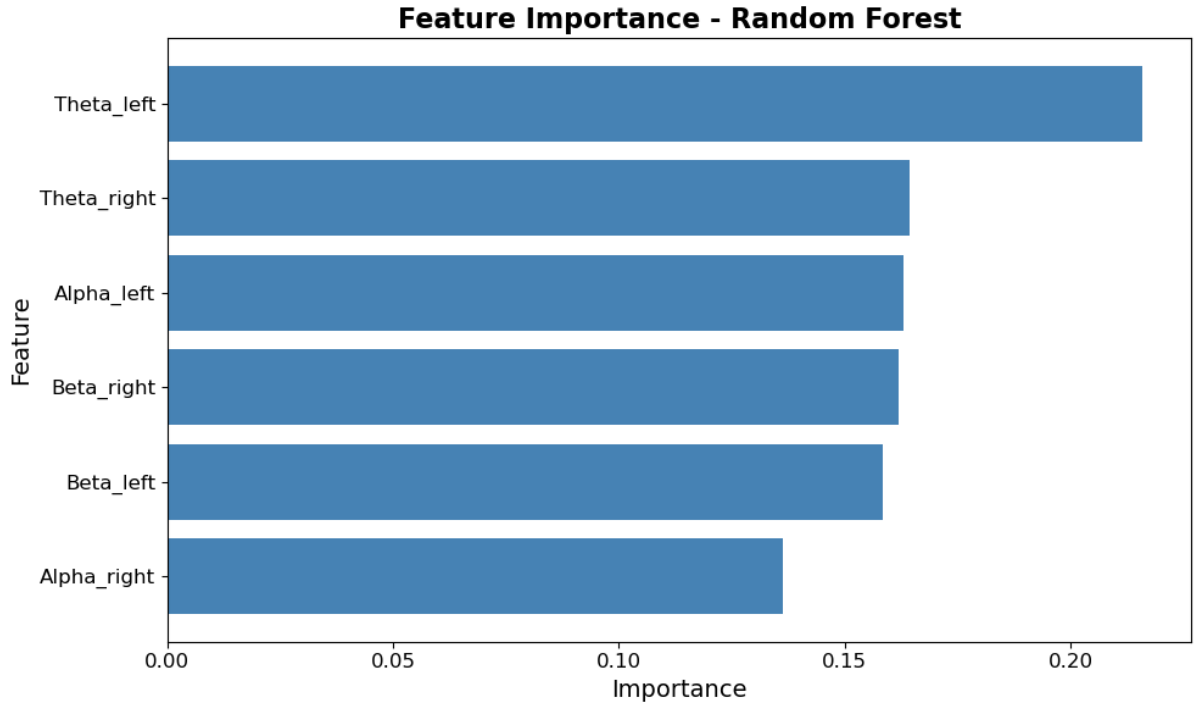
6.3 Key Results

Model	Target	Accuracy	Key Finding
Logistic Regression	Anxiety (SUDS)	48.4%	Below chance level
Random Forest	Anxiety (SUDS)	66.7%	Biased toward majority class
Random Forest (balanced)	Stress (Phase)	61.5%	Above chance, honest performance

6.4 Feature Importance

Analysis revealed that **Theta_left** is the most predictive feature (importance = 0.22), followed by Theta_right (0.17). This supports left-lateralized effects observed in exploratory

analysis.



7. Key Conclusions

1. SUDS-based anxiety detection fails with band power features alone, likely due to weak correlations and limited SUDS range.
2. Phase-based stress detection shows modest improvement (61.5% vs 50% chance), indicating clearer neural signatures between experimental conditions.
3. Left-hemisphere Theta activity is the most reliable neural marker for distinguishing stress states.
4. Future work could explore band ratios, asymmetry indices, or temporal dynamics for improved classification.

8. Data Requirements

LFP Data CSV

The system automatically detects one of the following channel naming patterns:

- `TD_Aic_ONE_THREE_LEFT / TD_Aic_ONE_THREE_RIGHT`
- `TD_Aic_ZERO_TWO_LEFT / TD_Aic_ZERO_TWO_RIGHT`
- `TD_Other_ZERO_TWO_LEFT / TD_Other_ZERO_TWO_RIGHT`
- `TD_Other_ONE_THREE_LEFT / TD_Other_ONE_THREE_RIGHT`

Task Data CSV

Required columns and entries:

- **Timer** column with format MM:SS:ms
- **Event** column containing 'SUDS' and 'Note' entries
- **Entry** column with phase markers: 'Task Start', 'exposure', 'compulsions', 'relief'