Exploring the dataset: NeuroEmo: "An fMRI Dataset for Emotion Recognition"

Irene Silvestro

July 2025

Introduction

Dataset

For this project I searched for a fMRI dataset about emotion elicitation. I found this dataset at the link:

OpenNeuro ds005700 (v1.2.0)

I report the most important details about the description of the dataset:

- **Description**: The purpose of the study is to enable emotion recognition analysis using culturally relevant stimuli, such as Indian Bollywood movie clips, in fMRI data collected from Indian participants. This dataset explores the neural basis of emotions in a contextually meaningful way, focusing on functional and dynamic connectivity.
This dataset contains raw fMRI data collected from 40 healthy participants who watched emotional Indian movie video clips.
Task-febold was collected for matrix size 128x128x36, voxel size =-1.8x1.8x4 mm, slice thickness 4, space between slices 4, echo time 0.035, repetition time 3, flip angle 90, slice 36.

- **Task** : Participants viewed movie clips designed to evoke different emotions task details:

Onset (s)	Duration (s)	Emotion
0	30	Calm
30	30	White noise
60	30	Afraid
90	30	White noise
120	30	Delighted
150	30	White noise
180	30	Depressed
210	30	White noise
240	30	Excited
270	30	White noise
300	30	Delighted
330	30	White noise
360	30	Depressed
390	30	White noise
420	30	Calm
450	30	White noise
480	30	Excited
510	30	White noise
540	30	Afraid
570	30	White noise

Preprocessing

From this article related to the dataset, I took note of how the data were preprocessed:
NeuroEmo: A neuroimaging-based fMRI dataset to extract temporal affective brain dynamics for Indian movie video clips stimuli using dynamic functional connectivity approach with graph convolution neural network (DFC-GCNN).

Preprocessing steps: Motion correction, Slice time correction, Coregistration, Spatial registration/Normalization Spatial smoothening.

Figure 1: Task sequence followed during data collection process for eliciting emotions.

Next, I had to resize the dataset shape $(128, 128, 36, 200)$ to match the shape of our atlas (Glasser and Schaefer), which was $(52, 62, 52)$, using a Nilearn function: *resample_to_img*.

Then, I splitted temporally the dataset which now had the size $(52, 62, 52, 200)$. So I created a *.tsv* file with the structural division of the time and the labels of the task and using that file I dividede the file for each subject obtaining the slitted dataset:

Then, I temporally split the dataset, which now had the shape $(52, 62, 52, 200)$. I created a *.tsv* file containing the temporal structure and task labels. Using this file, I divided the dataset for each subject, obtaining the split data as follows:

After that, I concatenated the identical task data together, reducing them to 5 splits for the emotions, each with a data shape of $(52, 62, 52, 20)$.
In a second step, I concatenated all the emotion tasks together and all the white-noise tasks together, obtaining a data shape of $(52, 62, 52, 100)$.
The data that I've used for the next parts are :

Note: The duration of both, the video and the fMRI acquisition, is 600 seconds. The repetition time (TR) is 3 seconds, so the total number of volumes is 200. After splitting, each emotion task fMRI lasts for 20 time points $(52, 62, 52, 20)$ (excluding the white noise data). In contrast, the data for all emotions and white noise are correctly sized as $(52, 62, 52, 100)$ each.

Seed-Based Functional Connectivity

As an initial approach to exploring the dataset, I began with the simplest method: seed-based functional connectivity. In this notebook, I used the Glasser Atlas and investigated different seed regions.

In the article *NeuroEmo: A neuroimaging-based fMRI dataset*, the authors show that their results, in line with the literature on the topic, highlight the amygdala and insula as the most relevant brain areas involved in emotional tasks. I therefore decided to focus on these regions to investigate the emotional task.

In my code I used the Glasser Atlas. After defining the mask and the atlas I defined different seed regions, and for each case of study I selected only one of them. I decided to explore: Amygdala, Insula and Primary Visual Cortex.
I used the first two seed regions with data from five different emotions, while the last seed region was used to compare the visual task (["allemotions"]) with the auditory task (["whitenoise"]).

After defining the seed region, I computed the average functional connectivity (FC) map across five different emotional conditions. For each emotion, I followed the steps below:

- Imported the data paths of ten subjects.
- Chose Pearson correlation as the functional connectivity measure.
- Initialized two empty arrays: one for correlation values and one for the associated p-values.
- For each subject:
  - Loaded the preprocessed fMRI data.
  - Applied a temporal filter to each voxel's time series within the brain mask.
  - Extracted the mean time series of the seed region.
  - For each voxel within the mask, computed the correlation between the voxel's signal and the seed region signal, along with the corresponding p-value.
- Concatenated the resulting subject-level FC maps into a group-level array.
- Computed the average FC map across subjects for each emotion and visualized the results.

Afterward, I defined a code to compare two different conditions using a **t-test** comparing calm to all the other emotions.
I chose an Independent (Two-Sample) t-Test: Compares the means of two independent groups.

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{1}$$

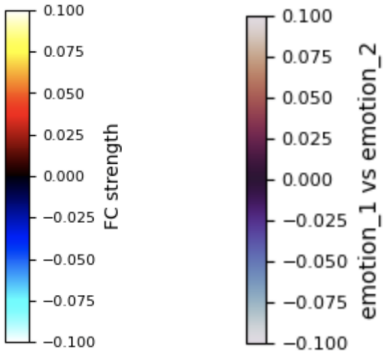
Where $\bar{X}_1, \bar{X}_2$ are the sample means, s_1^2, s_2^2 are the sample variances, n_1, n_2 are the sample sizes.
The **p-value** is computed based on the t-distribution with degrees of freedom calculated using the **Welch–Satterthwaite equation** for unequal variances:

$$df = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{\left(\frac{s_1^2}{n_1}\right)^2}{n_1 - 1} + \frac{\left(\frac{s_2^2}{n_2}\right)^2}{n_2 - 1}} \tag{2}$$

So following this steps I calculated the t-test:

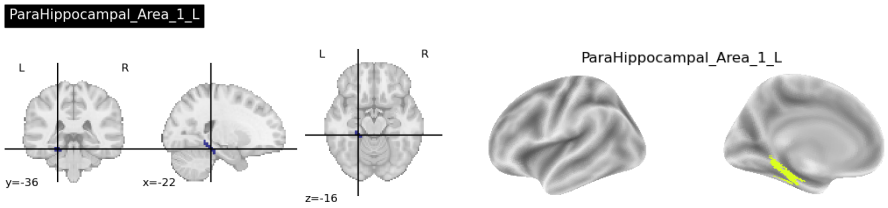
- Loop through the voxels, extracting the correlation values for the same voxel from the two conditions in the previously calculated FC maps.
- Perform the t-test on the extracted values.
- Apply a multiple comparison correction on the p-values using the FDR (False Discovery Rate) method.
- Insert the corrected p-values into the map.
- Plot the most significant results on the NIfTI map.

In the FC maps, I use the color gradient shown on the left, while for the t-test results, I apply the gradient on the right. In the FC maps, yellow-red indicates a positive correlation with the signal from the seed region. In the t-test maps, regions that are statistically different between the two conditions are highlighted: orange indicates areas where the first condition shows higher activity, and purple indicates areas where the second condition is more active.



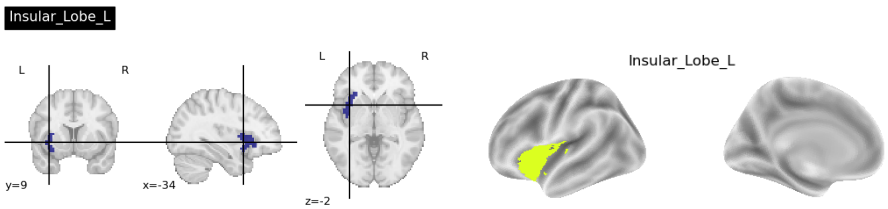
Amygdala

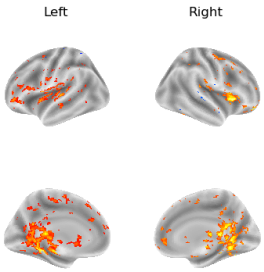
As we have see in the neuroanatomy lessons the amygdala is a nucleus near the hippocampus. While exploring the Glasser dataset, the region that appeared most similar to the location of the amygdala was 'ParaHippocampal_Area_1_L'.



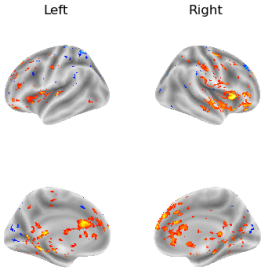
Insula

To extract the signal from the insula, I selected all the regions labeled in the Glasser atlas as belonging to the 'ins' lobe.

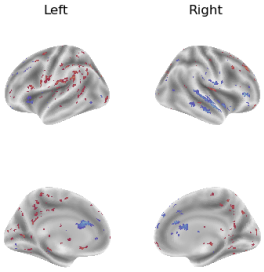




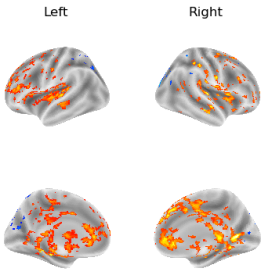
(a) Calm



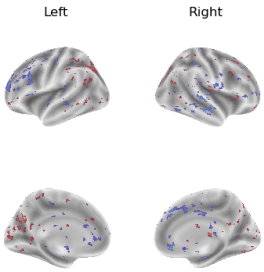
(b) Afraid



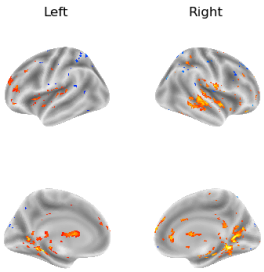
(c) Calm vs Afraid



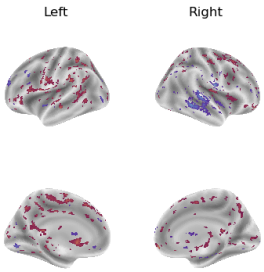
(d) Delighted



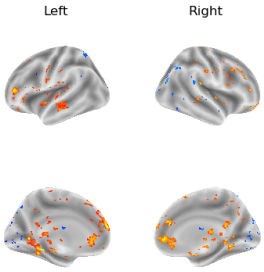
(e) Calm vs Delighted



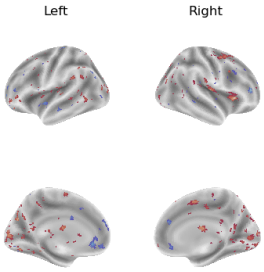
(f) Excited



(g) Calm vs Excited

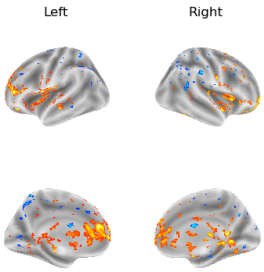


(h) Depressed

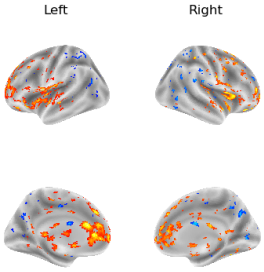


(i) Calm vs Depressed

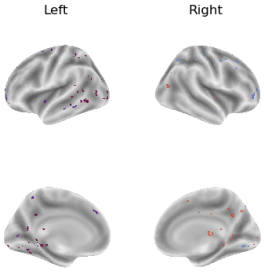
Figure 2: FC maps with amygdala as the seed region (left) and the t-test results comparing Calm to the other emotions (right).



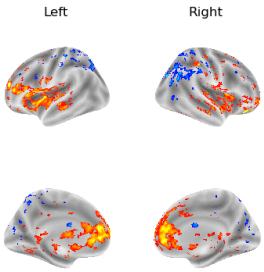
(a) Calm



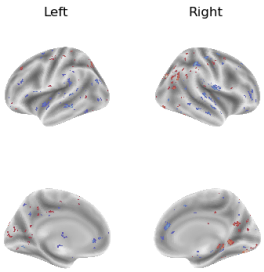
(b) Afraid



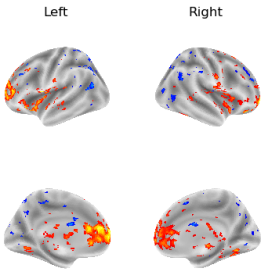
(c) Calm vs Afraid



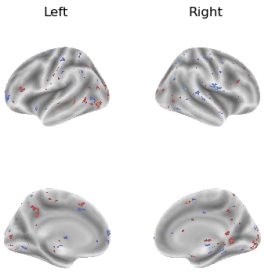
(d) Delighted



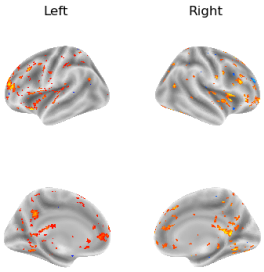
(e) Calm vs Delighted



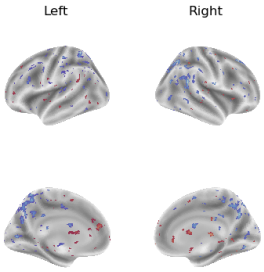
(f) Excited



(g) Calm vs Excited



(h) Depressed

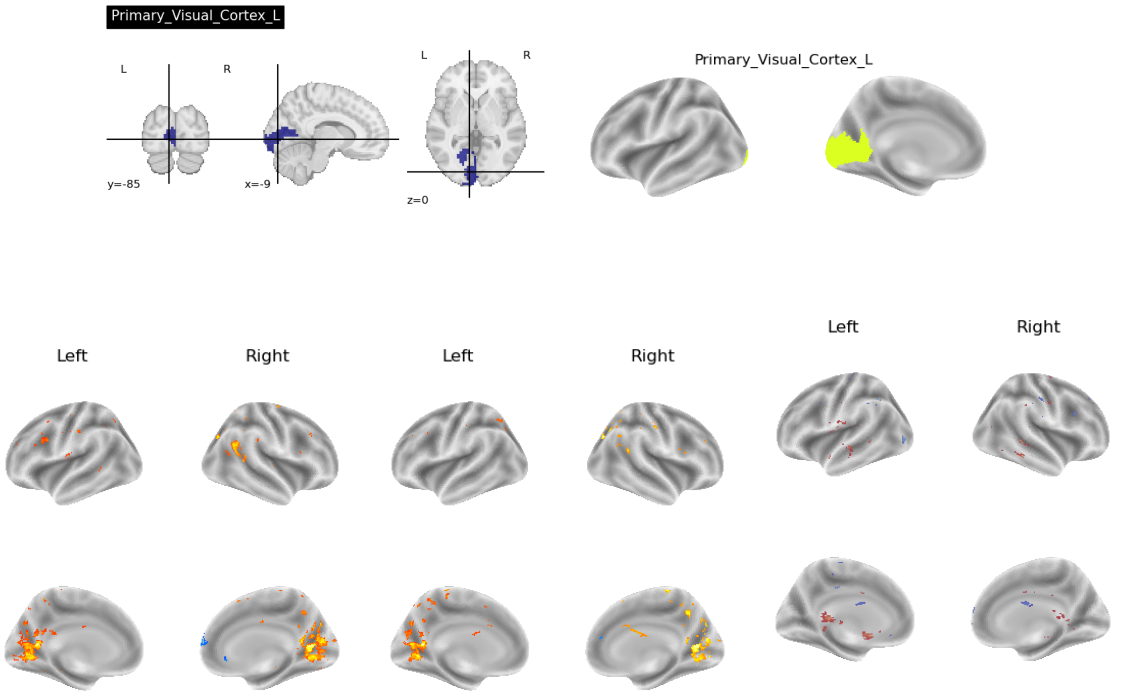


(i) Calm vs Depressed

Figure 3: FC maps with insula as the seed region (left) and the t-test results comparing Calm to the other emotions (right).

Primary Visual Cortex

In this section I decided to compare, in a significant seed region, the fcmaps resulting from the data obtained from the concatenation of all emotions together for each subject and all the white noise together for each subject:

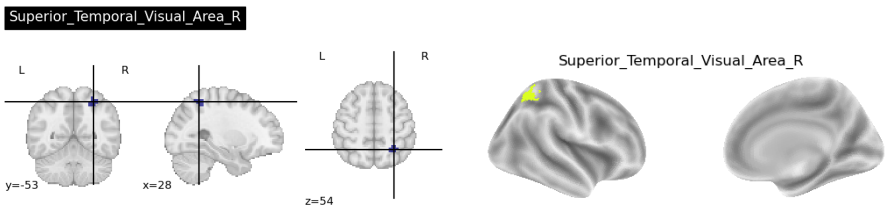


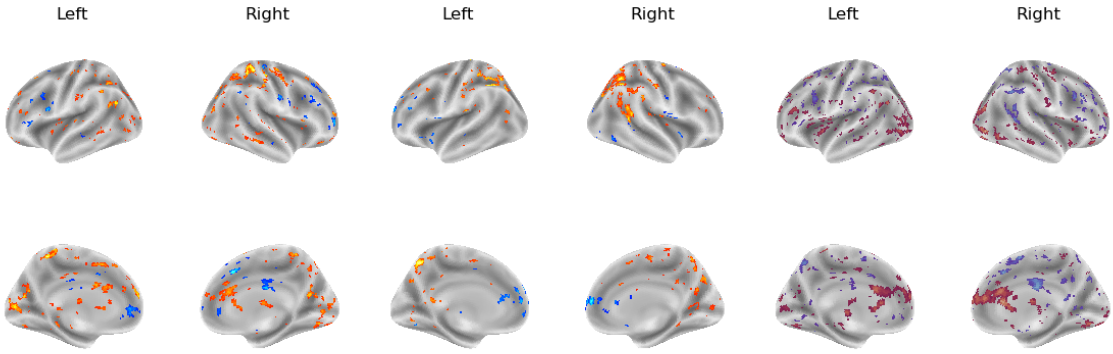
(a) All Emotions (b) White Noise (c) T-Test All Emotions vs. White Noise

Figure 4: FC maps with Primary Visual Cortex as the seed region (left), White Noise (middle), and the T-test results comparing All Emotions to White Noise (right).

Superior Temporal Visual Area

To conclude, referring to the aforementioned article, which analyzes the differences in DFC maps across different emotions, they report a significant difference in the functional activity of the region 'Temporal_Sup_R' between the Calm and Excited emotions. Therefore, I decided to compare them in the static FC map:





(a) Calm (b) Excited (c) T-Test calm vs. excited

Figure 5: FC maps with Right Temporal Superior Cortex as the seed region (left), White Noise (middle), and the T-test results comparing All Emotions to White Noise (right).

Dynamic Functional Connectivity

As a second analysis, I performed a Dynamic Functional Connectivity (DFC) analysis. In fact, in the article cited above, the authors emphasize the importance of incorporating temporal dynamics when studying emotional tasks.

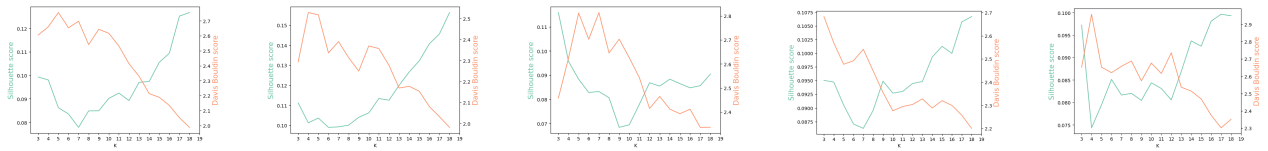
In the first step, I computed functional connectivity using a sliding-window approach and concatenated the data across all subjects. For this part, I had to choose two key parameters:
- **win_size**: the number of time points included in each window used to compute ROI-wise correlations.
- **stride**: the number of time points by which the window moves forward to compute the next correlation matrix.

In the second step of DFC, I clustered each instantaneous connectivity matrix in two different ways:

- 1. **Clustering the full connectivity matrices**: In this case, we directly cluster the connectivity matrices obtained previously by fixing a number k of clusters. The result of this method is a fixed number of matrices representing the centroids, whose number depends on the number of clusters specified for the K-means algorithm. These centroids represent the connectivity states identified by K-means. However, this approach may produce results that are difficult to interpret due to the high dimensionality of the data. For this reason, it is often preferable to reduce the dimensionality first, as done in the second method.
- 2. **Clusterizing the leading eigenvector of each instantaneous connectivity matrix**: Before proceeding with k-means, we extract the eigenvector corresponding to the largest eigenvalue for each matrix, and this will be the input for the k-means algorithm. Initially, we set k, and later, we attempt to find the optimal k using the Silhouette score and the Davies-Bouldin index. The first measures the intra-class distance (which we would like to be minimal), and the second measures the ratio between intra-cluster and inter-cluster distances (which we aim to be as low as possible). dopo aver trovato il numero ottimale per k procediamo a rifare il kmeans e calcoliamo per ogni stato il PO = percentage of occupancy e il LF = life time . After that, we calculate an average transition matrix: first, we compute the transition probability matrix between states for each subject, and then we plot the average of these matrices. For this reason, the matrix will not be normalized by row.

DFC results for all emotions with `win_size` = 8 and `stride` = 2

I chose a reasonable set of parameters to balance the limited number of time points in the dataset with the need to compare a sufficient number of correlation matrices. Specifically, I set `win_size` = 8 and `stride` = 2. In a later section, I will show how the results for the emotion *delighted* vary when changing the values of these parameters.



Calm Afraid Delighted Excited Depressed

Figure 6: Plots of Silhouette and Davis Bouldin score to define a global k.

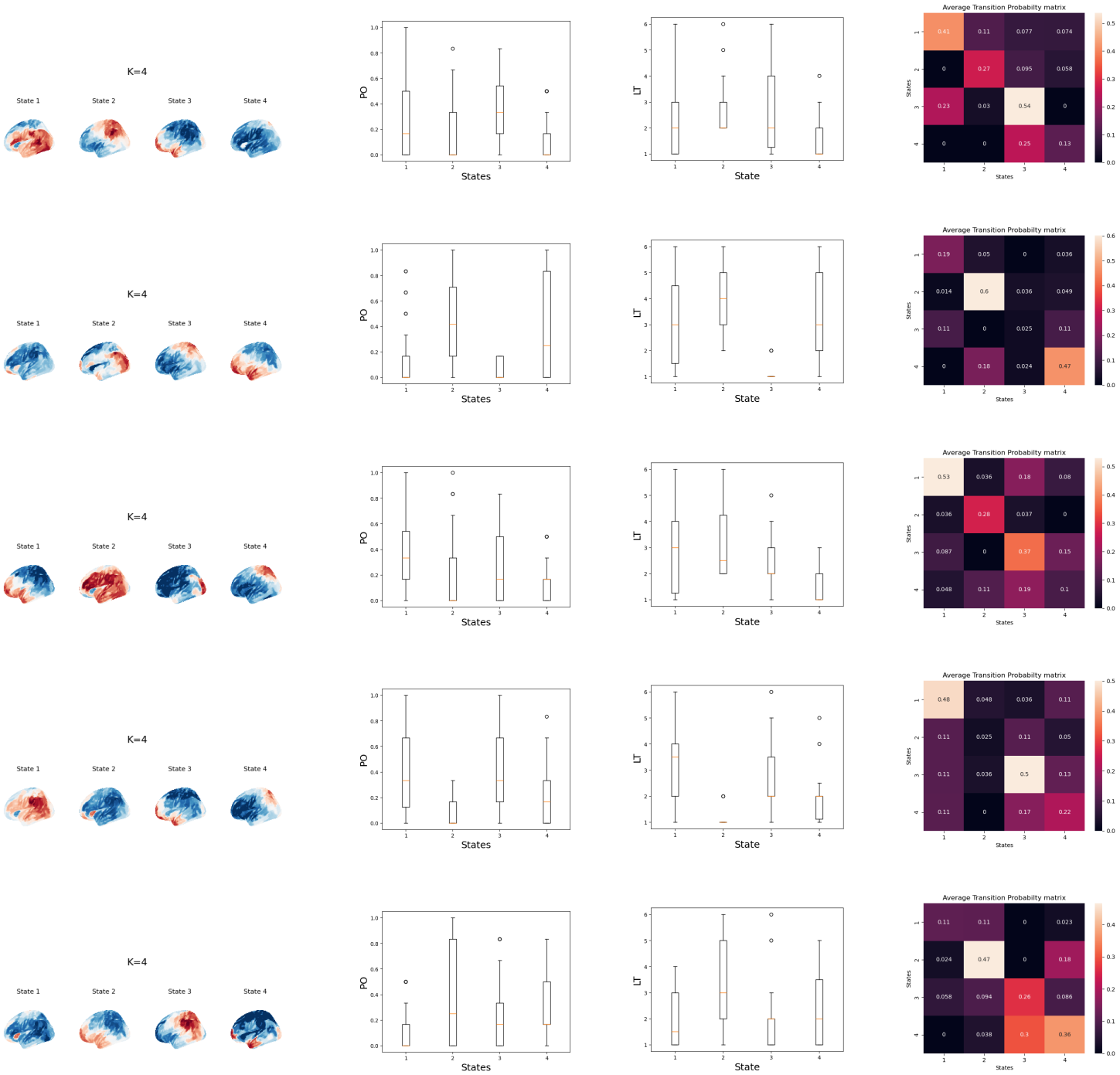
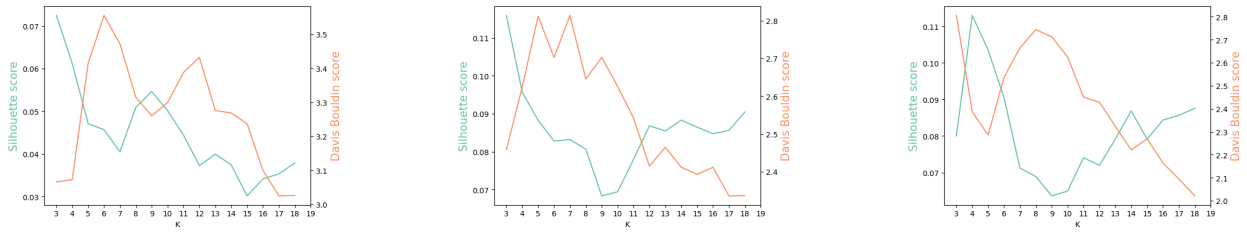


Figure 7: Results in order of: Calm, Afraid, Delighted, Excited, Depressed.

DFC results for delighted with different values of `win_size` and `stride`



`win_size` = 4 `stride` = 2 `win_size` = 8 `stride` = 2 `win_size` = 10 `stride` = 4

Figure 8: Plots of Silhouette and Davis Bouldin score to compare in different parameters' values.

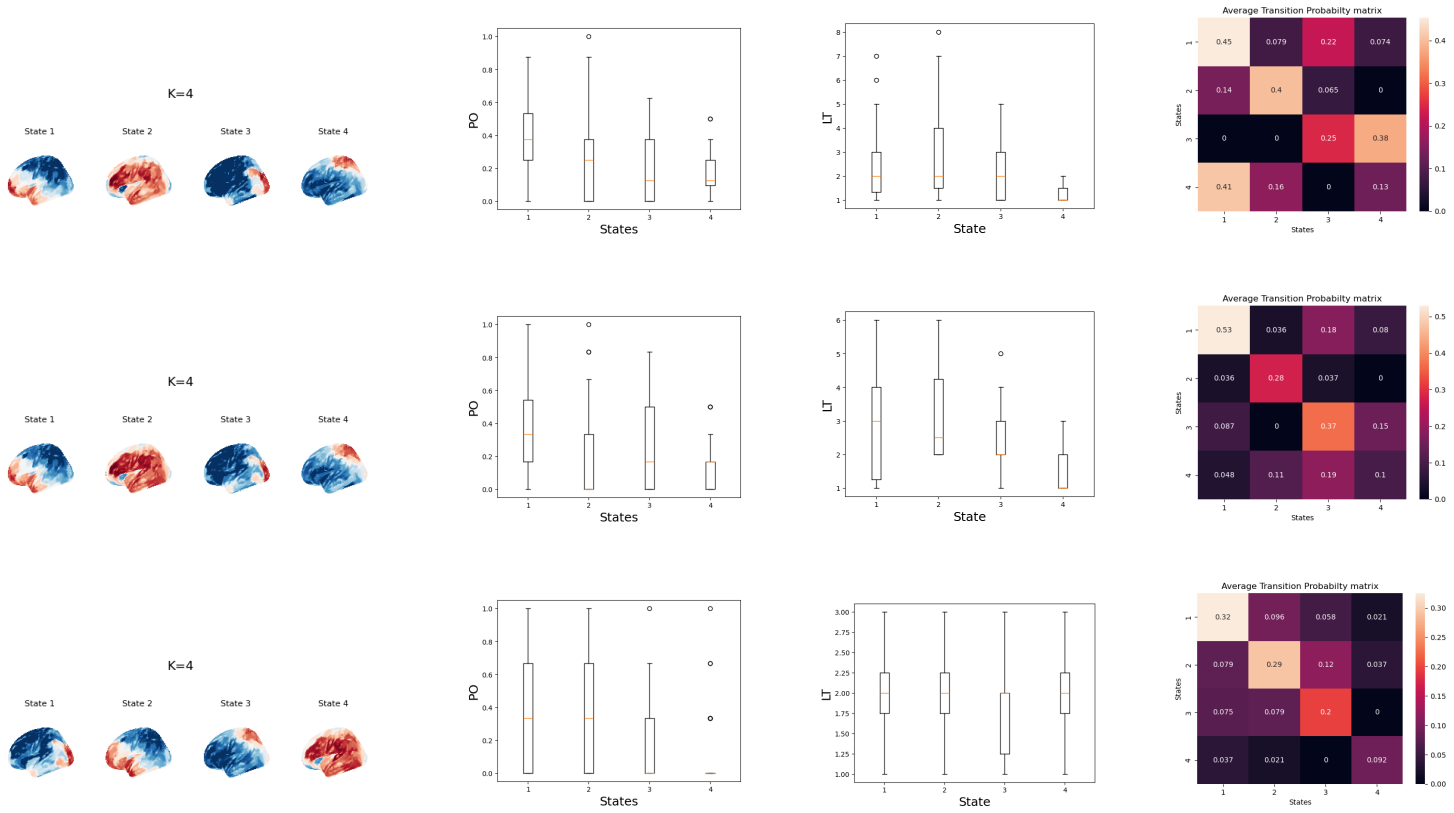


Figure 9: Results for Delighted in different parameters' values, in order: `win_size` = 4 `stride` = 2, `win_size` = 8 `stride` = 2, `win_size` = 10 `stride` = 4.

Machine Learning

During the previous part, I saved the eigenvectors under different conditions, which I used to train a learning model.
In the first case, I wanted to see if the information contained in the eigenvectors from the connectivity matrix in FC was informative enough for a learning model. These eigenvectors were obtained from the previous DFC code, with the parameters `win_size=10` and `stride=10`.
In the second and third cases, I trained and tested the model with two different eigenvectors: one with the parameters `win_size=4` and `stride=2`, and the other with `win_size=8` and `stride=2`.

Preprocessing and Splitting the Dataset

First, I loaded the data, assigning labels, since they were already conveniently separated.
Before proceeding with the learning phase, I preprocessed the data. This step allowed, in all cases, to increase the accuracy of the models by a few percentage points.

Standardize features by removing the mean and scaling to unit variance.
The standard score of a sample x is calculated as:

$$z = \frac{x - \mu}{\sigma}$$

where μ is the mean of the training samples and σ is the standard deviation of the training samples.

Note: The test set is standardized without recalculating the mean and standard deviation, in order to avoid introducing biases during the testing phase.

I split the dataset into 70% for training and 30% for testing, choosing randomly from the original dataset, using the *train_test_split* function from *sklearn*.

RFC and MLP with eigenvectors from parameters :`win_size=10` and `stride=10`

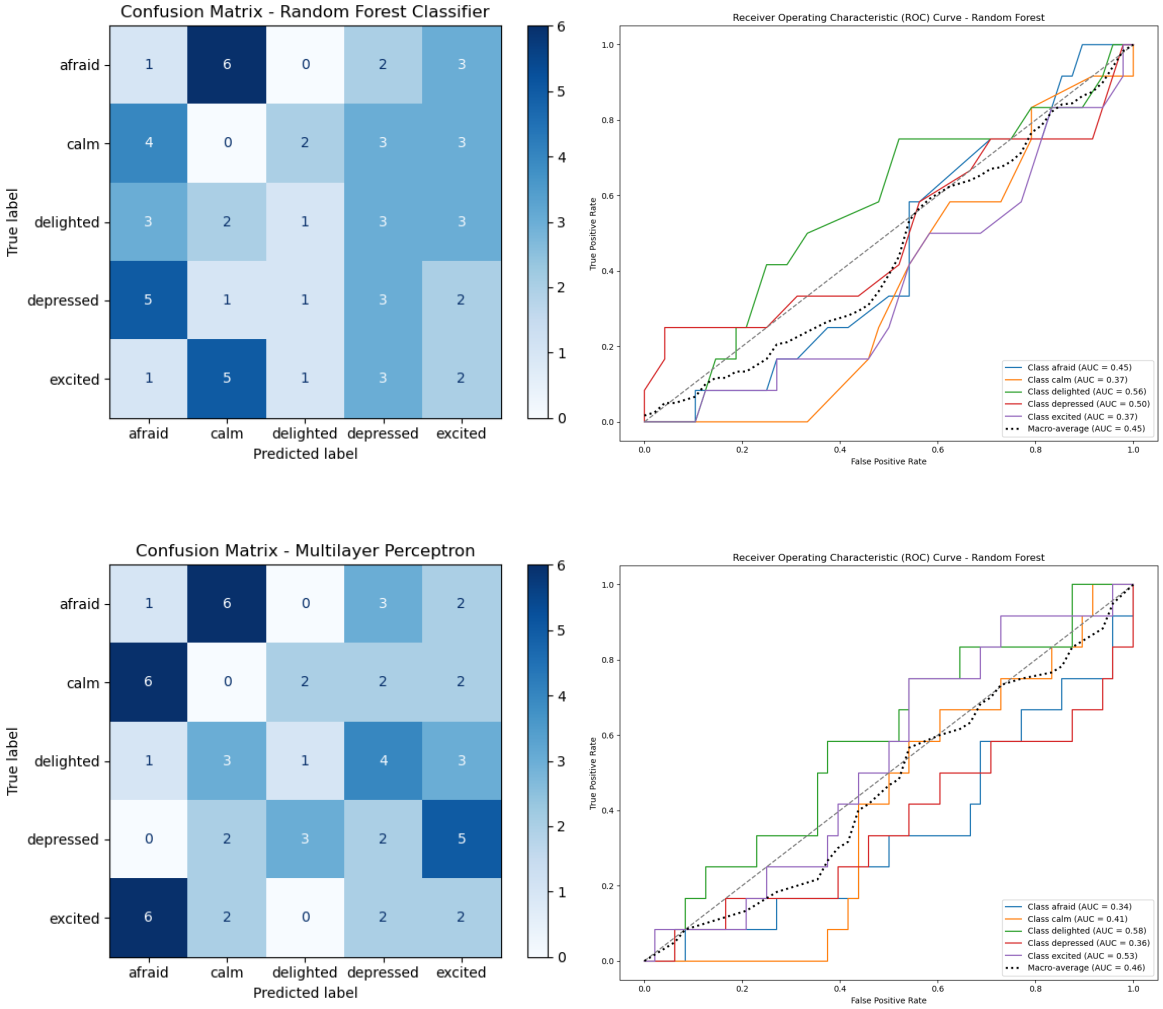


Figure 10: Random Forest Classifier and Multilayer Perceptron with eigenvectors from parameters: `win_size=10` and `stride=10`

RFC and MLP with eigenvectors from parameters :`win_size=4` and `stride=2`

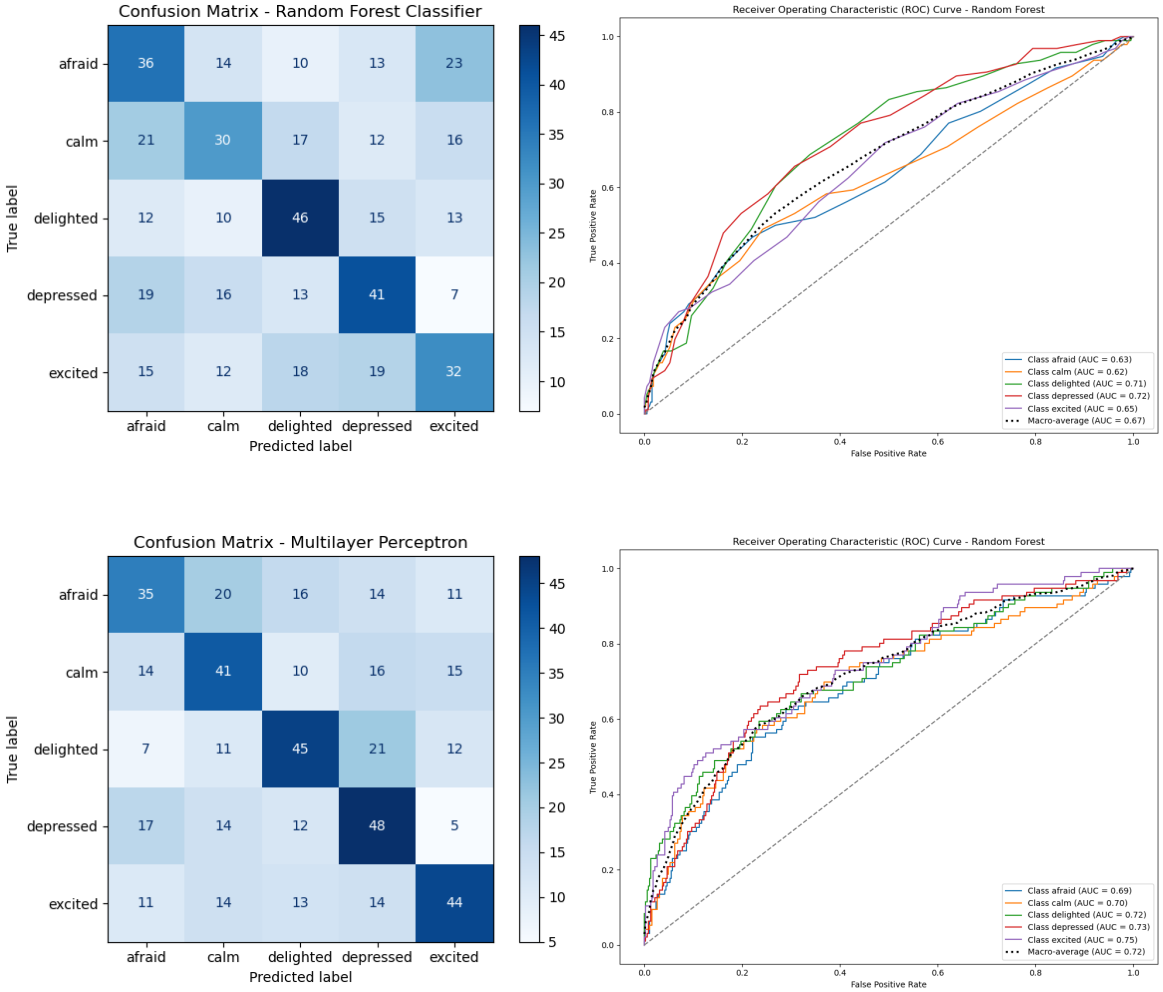


Figure 11: Random Forest Classifier and Multilayer Perceptron with eigenvectors from parameters :`win_size=4` and `stride=2`

RFC and MLP with eigenvectors from parameters :`win_size=8` and `stride=2`

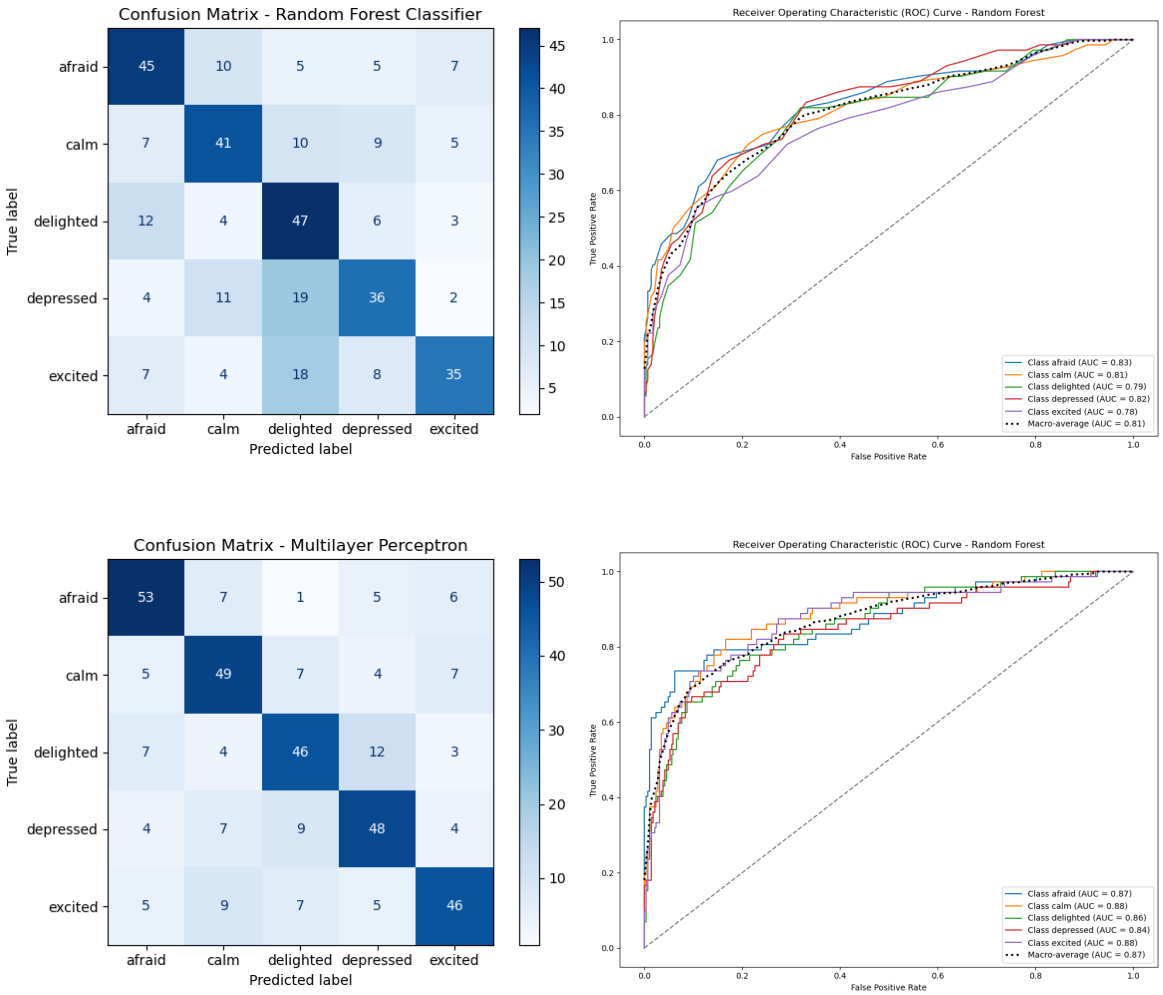


Figure 12: Random Forest Classifier and Multilayer Perceptron with eigenvectors from parameters :`win_size=8` and `stride=2`