Comparative Analysis of Machine Learning Algorithms for EEG Signal Classification and Clustering

Mehdi Jamalkhah
(810100111)

Mohammad Javad Pesarakloo
(810100103)

Abstract

One of the most powerful cognitive processes that involves mental simulation of movement without physical execution is motor imagery. In this study, our focus is on extracting and preprocessing EEG signals and feeding them to machine-learning models for motor imagery classification and clustering.

Electroencephalogram (EEG) signals play an important role in understanding brain activities and thinking processes. Today, processing and classifying these signals using machine learning methods have many uses in brain-computer interfaces, clinical diagnosis, and neuroscience.

In this project, we will first get to know EEG signal data and explore different ways to prepare the data, clean it, and remove any unwanted noise, which is common with real-world signals. Then, using techniques for extracting features from these signals that can be useful.

Additionally, we try to properly classify the data using the features extracted earlier with different machine learning algorithms. And finally, the results will be compared and analysed.

1 Electroencephalography (EEG) Signals

EEG is a non-invasive method to record macroscopic electrical activity by placing electrodes on the scalp. EEG recordings represent the activity of the surface of the brain. The neurons in the brain fire, which generate little pulses of electricity. If we place electrodes on the scalp, we can get very weak measurements of the voltage of the electrical activity. The recorded waveforms reflect this cortical electrical activity, and they are quite small, measured in microvolts (mV). Also, it detects a population-level neural activity. It does not depict one single neurons activity.

2 Introduction to Motor Imagery

In this study, we are exploring the EEG-based brain-computer interface(BCI), using motor imagery data. Motor imagery is a mental rehearsal technique in which an individual mentally simulates a specific movement or action without physically performing it; thus, a good dataset is required to first, capture the related-to-movement signals from EEG records. This process can be done through a pipeline like the following [9]:

1. Ask volunteers to move a limb in a specific direction randomly
2. record EEG signal during each trial
3. label the signals, distinctively

But the signals driven from the above pipeline are raw and need some preprocessing and feature extraction:

1. removing outliers using bandpass filters
2. keeping relevant EEG channels
3. reduce the dimensionality of data using feature extraction or dimensionality reduction

each of the above items has several algorithms and methods which will be described in other sections of the text. The following figure abstracts this pipeline:

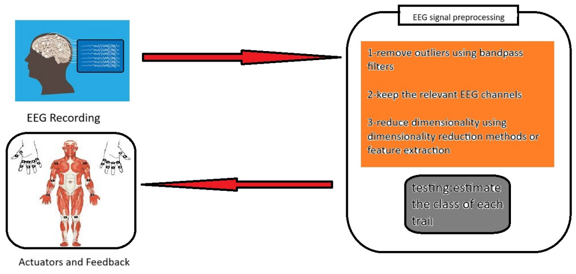
EEG signal preprocessing
1-remove outliers using bandpass filters
2-keep the relevant EEG channels
3-reduce dimensionality using dimensionality reduction methods or feature extraction
testing:estimate the class of each trail
EEG Recording
Actuators and Feedback

Figure 1: An abstraction of the training pipeline

Several challenges in this pipeline can have a great influence on the models which will be trained.Let's explore some of them:

3 Challenges in Motor Imagery

3.1 Selection of electrodes

These signals are driven from different electrodes. A challenging task is identifying the electrodes showing relevant changes in movement. It is of high importance to determine electrodes that carry significant information and also the ones that do not improve classification rate and their feature vectors introduce noise. Also, the number of selected electrodes matters a lot. Let's explore some solutions to this challenge:

Four-electrode approach We can use physiological information. The μ waves are located in the somatosensory cortex; thus we can easily use the signals driven from the electrodes that were located over the sensorimotor cortex for analysis. Based on MRI scans, the usage of C3, C4, CP3, and CP4 proved to be more accurate [8].

PCA This method is described mathematically later in this text. Considering it as a general dimensionality reduction method, we can calculate the principal components of the EEG signal and use the electrodes that form the eigenvector with the largest corresponding eigenvalue [8].

3.2 Performance of MI-BCI

In MI-BCI systems, responsiveness, and instantaneity are critical and lack of it can sometimes lead to life-or-death issues. Thus its performance should be enhanced. Let's explore some solutions for this issue [10]:

Channel Selection In channel selection, we remove non-relevant channels which reduces power consumption and the search space.

Dimensionality Reduction and Feature Selection The performance is increased by finding the most optimal feature.

3.3 Individual Differences

Each person's brain activity is unique and this variability can make it challenging to develop a generalized EEG model. Let's explore some methods:

Normalization Techniques The amplitude of the EEG signal differs highly from person to person; thus by normalizing the amplitude, the EEG model can be robust to amplitude differences between persons.

4 Preprocessing of EEG signals

The EEG signals captured from the brain are raw and often noisy and consequently need preprocessing. To understand the significance of this phase, we first should explore different types of waves of the brain:

1. Delta Waves(0.5 to 4 Hz): Delta waves are the slowest brain waves, typically associated with deep sleep and restorative processes.
2. Theta Waves (4 to 8 Hz): Theta waves are associated with light sleep, relaxation, and meditative states. They can also be present during creative and imaginative activities.
3. Alpha Waves (8 to 12 Hz): Alpha waves are prominent during relaxed, wakeful states with closed eyes, often associated with calmness and relaxation.
4. Mu Waves (8 to 13 Hz): Mu waves are similar in frequency to alpha waves but are specifically related to the motor cortex and sensory-motor rhythms.
5. Beta Waves (13 to 30 Hz): Beta waves are associated with active thinking, concentration, problem-solving, and active focus. They are present during alert, attentive states.
6. Gamma Waves (30 to 100 Hz): Gamma waves are the fastest brain waves and are associated with high-level cognitive functions, including perception, problem-solving, and consciousness.

The brain waves most related to movement are **Mu waves** and **Beta waves**. Due to this explanation, there is a demand for a band-pass filter that gives us frequencies between 8 to 30 Hz.

The next crucial preprocessing is applying spatial filters.EEG signals from different electrodes represent a mixture of activities from various brain regions. Spatial filtering aims to isolate the patterns of activity relevant to the task at hand, such as movement. Also, EEG recordings often contain noise from various sources, including muscle activity, eye blinks, and electrical interference. Spatial filters can help attenuate these artifacts by focusing on the spatial characteristics of the signals that are most likely to originate from the brain region of interest.

Let's explore some types of spatial filtering:

4.1 Common Average Reference (CAR)

In this method, the average of all EEG signals (common activity in the brain) is subtracted from each electrode's signal. The idea under the CAR is to remove the average brain activity, which can be seen as EEG noise. The formula used to compute the CAR is as follows [4]:

$$U_i^{CAR} = U_i - \frac{1}{n}\sum_{j=1}^{n} U_j \tag{1}$$

4.2 Principal Component Analysis (PCA)

In this method, the goal is to project data into new coordinates which maximizes the variance of data. Given n data points in d dimensions:

$$X = \left\langle \begin{matrix} | & | & & | \\ x_1 & x_2 & \dots & x_n \\ | & | & & | \end{matrix} \right\rangle$$

we are reducing dimensionality from d to l:

$$W = \left\langle \begin{matrix} | & | & & | \\ w_1 & w_2 & \dots & w_l \\ | & | & & | \end{matrix} \right\rangle$$

projecting X down to:

$$Y = X^T W \tag{2}$$

such that the projected variance is maximized. Note that we first shift the data to be centered at zero ($\hat{E}[X] = 0$):

$$\begin{aligned}\hat{E}[Y] &= \frac{1}{n}\sum_{i=1}^{n} W^T x_i \\ &= W^T(\frac{1}{n}\sum_{i=1}^{n} x_i) \\ &= W^T \hat{E}[X] = 0\end{aligned}$$

$$\begin{aligned}V\hat{A}R[Y] &= \hat{E}[Y^2] - \underbrace{\hat{E}^2[Y]}_{0} \\ &= \hat{E}[(W^T X)(X^T W)] \\ &= W^T \hat{E}[XX^T]W \\ &= W^T \underbrace{S}_{\text{sample covariance matrix}} W \qquad (3)\end{aligned}$$

thus the objective can be written as:

$$\begin{aligned}&\text{maximize} \quad W^T S W \\ &\text{subject to} \quad \|W\|_2 = 1\end{aligned}$$

$$\begin{aligned}\mathcal{L}(W, \lambda) &= W^T S W - \lambda(\|W\|_2 - 1) \\ \Rightarrow \frac{\partial \mathcal{L}}{\partial W} &= 2SW - 2\lambda W = 0 \\ &\Rightarrow SW = \lambda W \qquad (4)\end{aligned}$$

from the above equation, we get that W is eigen vector of S and λ is eigen value.If we replace the value of SW with λW in the objective, we have:

$$\text{objective} = W^T \lambda W = \lambda W^T W = \lambda \underbrace{\|W\|_2}_{1} = \lambda \qquad (5)$$

thus, W is the corresponding eigen vector of the highest eigenvalue of S. This idea can be generalized and the second principal component is the corresponding eigen vector of the second greatest eigen value of S, and so on.

As we know, the signal collection methods are unstable and the signals driven from them are a mixture of interfering signals and have high dimensions. In this case, PCA can be applied.

4.3 Independent Component Analysis (ICA)

The theory of this algorithm will be described in the following section . Applying this algorithm, we can identify and isolate components related to brain activities, in this case, movement.

The brain does a lot of tasks at the same time. One part of the brain helps regulate the heartbeat, while another part is responsible for blinking the eyes intermittently. Additionally, the brain ensures the body maintains a steady breathing pattern.

The brain carries out these different tasks in parallel, so different parts of the brain generate different electrical impulses. The electrodes placed on the scalp's surface measure an overlapping combination of all these electrical activities, and a single point on the brain's surface reflects a sum or mixture of these disparate electrical impulses.

Turns out a pretty good way to clean up this data, is to use the ICA algorithm and separate the signal into its independent components.

As shown in Figure 2, the raw EEG signals have a high correlation. This is because the general activity of the brain is much stronger than the specific activities we want to focus on. Using an ICA filter can help with this. ICA can split the signal into separate parts. This lets us separate the main brain activity from the specific things we are interested in measuring. By isolating these different parts, we can study the particular neural activities we want.

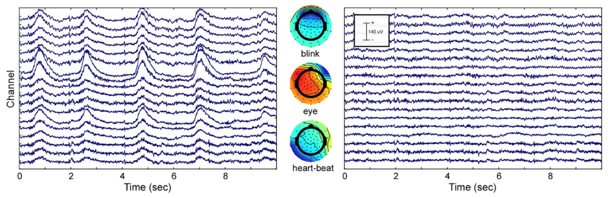

Figure 2: ICA-based artifact attenuation. Left) Original EEG time course, shown for a subset of 18 electrodes and 10 seconds. Center) ICA topographies represent eye blinks (Top), lateral eye movements (Middle), and heartbeat (Bottom). Right) EEG data after ICA-based artifact attenuation. The EEG time courses were reconstructed excluding the identified artifact components. This figure is adapted from [10]

4.4 Minimum Norm Estimation (MNE)

Minimum Norm Estimation is an advanced signal processing technique to estimate the source of neural activity within the brain from the electrical potentials recorded on the scalp.EEG signals alone, do not carry sufficient information to determine the precise spatial distribution of the underlying neuronal source because it can be from anywhere. Two strategies are often distinguished:

1. focusing on those solution parameters that can be estimated reliably from the data alone
2. including a priori knowledge from other sources than the data under analysis, thus reducing the number of parameters to be estimated to a tractable number

Mathematically as described in [5], the electric potential observed at discrete locations above the scalp surface has a linear relationship with the source distribution within the head. This can be formulated in matrix notation as

$$d = Ls \qquad (6)$$

where d is the observed vector, s is the source vector, and L is the lead field matrix.

Minimum norm estimation can be achieved by solving a following optimization problem

$$(\hat{s} - \hat{s}_0)^T C_s(\hat{s} - \hat{s}_0) = \min \quad (7)$$

$$(L\hat{s} - d)^T (L\hat{s} - d) = \min \quad (8)$$

where s_j is the estimated solution, $\hat{s}_0$ is an a priori approximation of the solution, and C_s is a weighting matrix, representing the metric associated with the source space (e.g., a prior knowledge about the approximate locations or covariances of sources)

The solution to this problem is

$$\hat{s} = \hat{s}_0 + C_s^{-1} LT(LC_s^{-1}L^T)^{-1}(d - L\hat{s}_0) \quad (9)$$

If no prior model $\hat{s}_0$ is included, the equation reduces to

$$\hat{s} = C_s^{-1} LT(LC_s^{-1}L^T)^{-1} d \quad (10)$$

4.5 Laplacian Filter

Equation 11 calculates the Laplacian filter, which approximates the second derivative by subtracting the mean activity at surrounding electrodes from the channel of interest.

$$V_i^{LAP} = V_i - \sum_{j \in S_i} g_{ij} V_j \quad (11)$$

where

$$g_{ij} = \frac{\frac{1}{d_{ij}}}{\sum_{j \in S_i} \frac{1}{d_{ij}}}$$

S_i is the set of electrodes surrounding the ith electrode, and d_{ij} is the distance between electrodes i and j. For the large Laplacian, it was the set of next-nearest-neighbor electrodes[7].

One of the main characteristics of the Laplacian filter is that it is reference-free. This may seem puzzling, as the surface Laplacian is typically computed from the scalp-recorded potentials, which are themselves reference-dependent quantities. The Laplacian is reference-free because it captures the spatial gradient of the potential field, rather than the absolute potential values. By focusing on these local spatial patterns, the Laplacian transform effectively removes the influence of the reference electrode from the signal. being reference-free is a major advantage because it avoids the ambiguity and potential distortions introduced by the choice of reference [6].

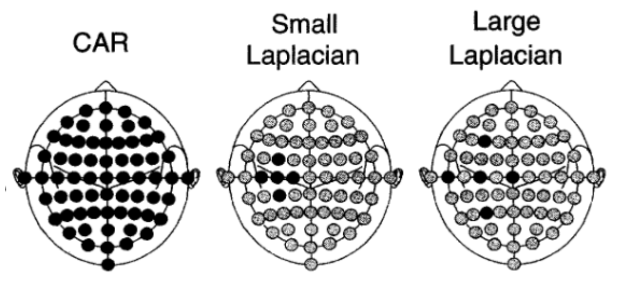

Figure 3: Elecrode locations used in each spatial filter method. You can see the difference in the localized feature in each method. This figure is adapted from [7].

5 Feature Extraction

5.1 Independent Component Analysis (ICA)

ICA is a technique used to separate mixed signals into their independent sources. The goal of ICA is to find a linear transformation of the data such that the transformed data is as close to being statistically independent as possible.

There are two assumptions in ICA:

1. Source signals are statistically independent.
2. Source signals exhibit non-Gaussian distributions.

We want to figure out what is the density of sources (S), from given signals X:

$$S \in \mathbb{R} \qquad S_j^{(t)} : \text{source j at time t}$$

$$X^{(i)} = AS^{(i)} \quad (12)$$

Goal: find $W = A^{-1}$

$$S^{(i)} = WX^{(i)} \quad (13)$$

By knowing $P_S(s)$ and Equation 13 we can drive:

$$P_X(x) = P_S(WX)|W| \quad (14)$$

where $|W|$ is the determinant of W, and normalize the pdf.

Finally, we must choose a non-Gaussian distribution for random variable S. Let's pick the sigmoid function as the CDF of S.

$$F_S(s) = P(S \le s) = \frac{1}{1 + e^{-s}} \quad (15)$$

and it turns out this will work well. Many choices work fine.

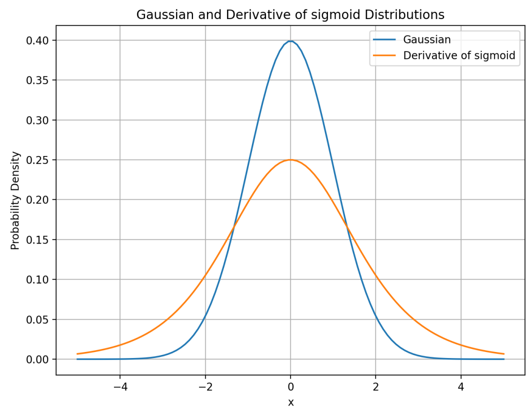

Figure 4: Gaussian and derivative of sigmoid distribution: Gaussian density goes to zero very quickly, but the other distribution, taken by the computed derivative of sigmoid, goes to 0 more slowly and this captures many natural phenomena better than a Gaussian density because there are a larger number of extreme outliers, which are more than one or two standard deviations away.

As it said multiple distributions work, for example, Laplacian distribution:

$$P_S(s) = \frac{1}{2b} e^{-\frac{|s-\mu|}{b}} \quad \text{(Laplacian distribution)} \tag{16}$$

By the first assumption we have:

$$P_S(s) = \prod_{i=1}^{n} P_S(s_i) \tag{17}$$

Now by using the above equation and equation 14, we can drive the following formula:

$$\begin{aligned} P_X(x) &= P_S(WX)|W| \\ &= (\prod_{i=1}^{n} P_S(W_i^T X))|W| \end{aligned} \tag{18}$$

we use maximum likelihood estimation to find W:

$$\mathcal{L}(W) = \sum_{i=1}^{m} log \left((\prod_{j=1}^{n} P_S(w_j^T X^{(i)})|W| \right) \tag{19}$$

Stochastic gradient ascent:

$$\nabla_W \mathcal{L}(W) = \begin{bmatrix} 1 - 2g(w_1^T x^(i)) \\ \vdots \\ 1 - 2g(w_n^T x^(i)) \end{bmatrix} (x^{(i)})^T + (W^T)^{-1} \tag{20}$$

where g is the sigmoid function.

By this updating rule, ICA can find a pretty good matrix W, for unmixing the sources.

5.2 Common Spatial Patterns (CSP)

Common Spatial Pattern is a technique to analyze multi-channel data, recorded from two classes. CSP provides a data-driven supervised approach to decompose the signal. This decomposition is parameterized by a matrix $W \in \mathbb{R}^{C\times C}$ (C being the number of channels) that projects the signal $x(t) \in \mathbb{R}^C$ in the original sensor space (e.g. EEG signals) to $x_{CSP}(t) \in \mathbb{R}^C$, which lives in source space, as follows:

$$x_{CSP}(t) = W^T x(t) \tag{21}$$

Which is the same as Equation 17.

Let $\Sigma^{(+)} \in \mathbb{R}^{C\times C}$ and $\Sigma^{(+)} \in \mathbb{R}^{C\times C}$ be the estimates of the covariance matrices of the EEG signal in two classes (e.g. left-hand imagination and right-hand imagination):

$$\Sigma^{(c)} = \frac{1}{|\Phi_c|} \sum_{i\in\Phi_c} X_i X_i^T \quad (c \in \{+,-\}) \tag{22}$$

where Φ_c is the set of indices corresponding to trials belonging to class C. Then CSP analysis is given by the simultaneous diagonalization of the two covariance matrices

$$\begin{aligned} W^T\Sigma^{(+)}W &= \Lambda^{(+)}, \\ W^T\Sigma^{(-)}W &= \Lambda^{(-)} \end{aligned} \tag{23}$$

where $\Lambda^{(c)}$ is diagonal. W is usually determined s.t. $\Lambda^{(+)} + \Lambda^{(-)} = \mathcal{I}$. Mathematically this can achieved by solving the generalized eigenvalue problem

$$\Sigma^{(+)}w = \lambda\Sigma^{(-)}w \tag{24}$$

Then Equation 23 is satisfied when W composed of generalized eigenvectors $w_j (j = 1, ..., C)$ from Equation 24. Theses eigenvectors are used as the column vectors in W, and $\lambda_j^{(c)} = w_j^T \Sigma^{(c)} w_j$ being the corresponding diagonal elements of $\Lambda^{(c)}$, while λ in Equation 24 equals $\lambda_j^{(+)}/\lambda_j^{(-)}$. Note that $\lambda_j^{(c)} \geq 0$ is the variance in condition c in the corresponding surrogate channel and $\lambda_j^{(+)} + \lambda_j^{(-)} = 1$. Hence a large value of $\lambda_j^{(+)}(\lambda_j^{(-)})$ close to one indicates that the associated spatial filter w_j produces high variance in the positive(negative) condition and low variance in the negative(positive) condition, respectively. This contrast between the two classes is useful in the discrimination [1].

5.3 Limits of standard CSP

The CSP method computes spatial filters in a naive, data-driven manner. This approach may produce suboptimal results, potentially failing to extract the true motor imagery-related neural activity, in certain situations.

A major source of errors comes from the difficulty in correctly estimating the class covariance matrices. Since poorly estimated covariance matrices do not properly represent the underlying brain processes, this will directly affect the spatial filter calculation. Also, the increasing number of electrodes used in experiments further complicates the estimation problem. Because we need more data to reliably estimate the high-dimensional covariance matrices, Otherwise we should have prior information or regularization.

Furthermore, the covariance matrix estimation may be negatively affected by EEG artifacts, such as eye blinks. These artifacts often have much greater signal strength than the target activity. If not properly removed, they may dominate the covariance matrix estimation and lead to overfitted CSP solutions, because we cannot extract the main feature of the signal, which increases the generalization of model [2].

5.4 Robust Estimation

Average estimates of covariance matrix can affected by outliers. In this subsection, we propose robust estimates of the class-wise based on [3].

Suppose that we have a set of covariance matrices $\{\Sigma_1, \ldots, \Sigma_n\}$. Their average can obtained from:

$$\bar{\Sigma} = \frac{1}{n}\sum_{i=1}^{n} \Sigma_i = min_{\bar{\Sigma}\in PD(d)} \sum_{i=1}^{n} ||\Sigma_i - \bar{\Sigma}||^2 \tag{25}$$

where $PD(d)$ denotes the set of d by d positive definite matrices. To make the average covariance $\bar{\Sigma}$ more robust, we can reduce the contributions of outlier co-variances in Equation 25. Let ρ be an increasing function, slower than linear, and consider rewriting the optimization problem

$$\bar{\Sigma} = min_{\bar{\Sigma} \in PD(d)} \sum_{i=1}^{n} \rho(||\Sigma_i - \bar{\Sigma}||^2) \tag{26}$$

It solution can be obtained by an iterative procedure:

$$\bar{\Sigma}_{(t+1)} = \sum_{i=1}^{n} \frac{\rho'(||\Sigma_i - \bar{\Sigma}_{(t)}||^2)}{\sum_{j=1}^{n} \rho'(||\Sigma_j - \bar{\Sigma}_{(t)}||^2)} \tag{27}$$

Example of the function ρ are $\rho(s) = \sqrt{s}$ and $\rho(s) = log(1 + \alpha s)$ with an appropriate $\alpha > 0$.

6 Classification

6.1 Dataset

These data sets were recorded from healthy subjects. In the whole session, motor imagery was performed without feedback. For each subject two classes of motor imagery were selected from the three classes left hand, right hand, and foot (side chosen by the subject; optionally also both feet).

You can download the dataset from **here**.

6.2 Preprocessing

We can apply different CAR, Laplacian, and band-pass filters. We tried some of them and saw that the CAR filter hurt the final scores of classification. So we used the Laplacian and band-pass filters. After applying each filter, we visualized the signals and the scatter plot of trials.

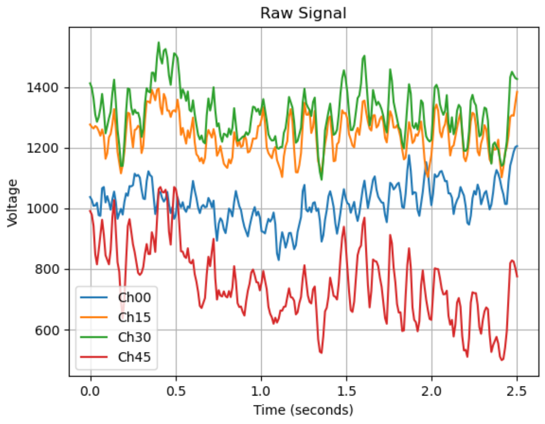

Figure 5: A sample of raw EEG signal

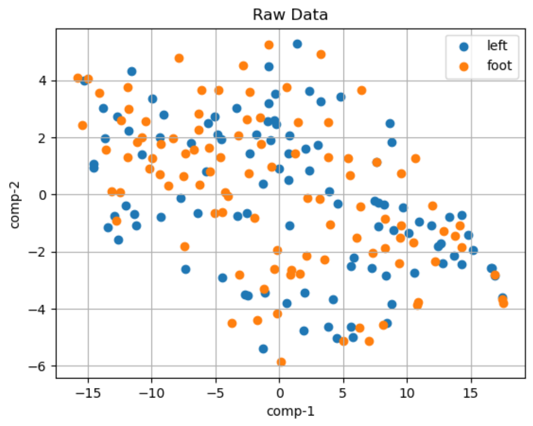

Figure 6: Scatter plot of Raw data

After applying a large laplacian filter:

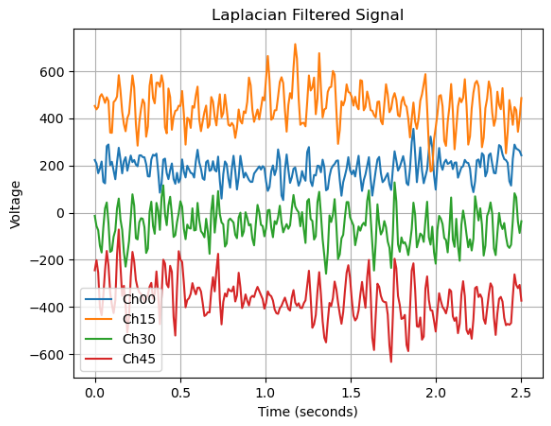

Figure 7: A sample of laplacian filtered EEG signal

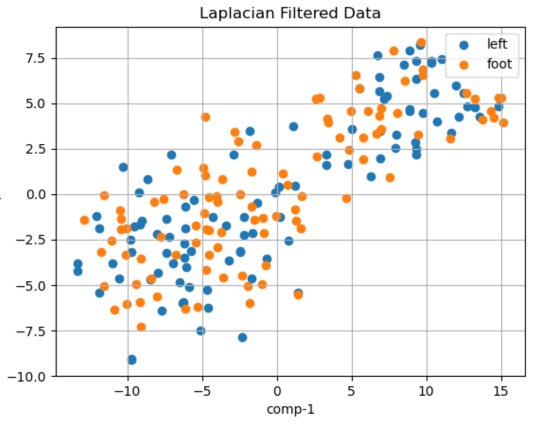

Figure 8: Scatter plot of laplacian filtered data

As you can see in the above figures, the voltage of the signals has reduced, and the data has gotten closer to each other. This is because the Laplacian filter replaces the distances with their differences and shifts the data to a better scale, which makes small changes more significant.

Now let's apply a band-pass filter to extract frequencies between 8 and 40 Hz:

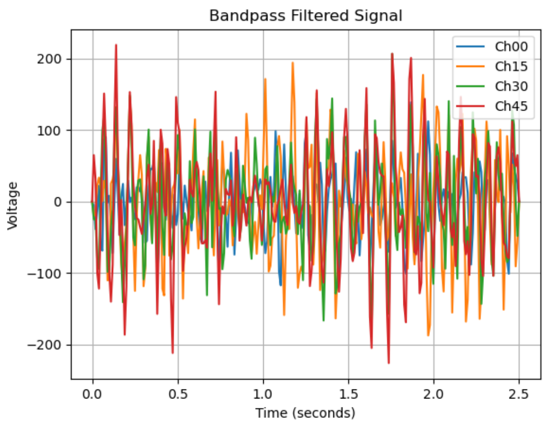

Figure 9: A sample of bandpass filtered EEG signal

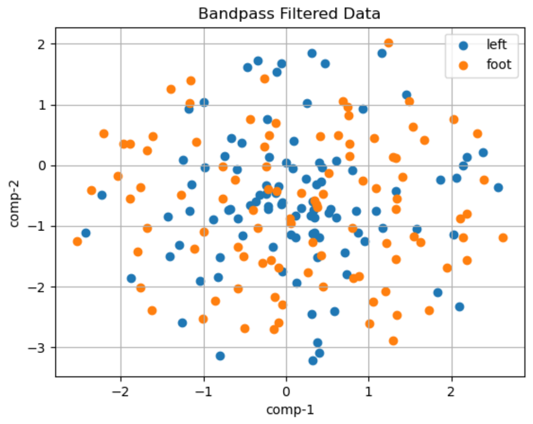

Figure 10: Scatter plot of bandpass filtered data

Not yet completely separable data, but as you can see, it is better than the previous one. The pure region has increased.

Another option for preprocessing is PCA, the results of which will be shown at the end after training all classifiers.

6.3 Feature Extraction

We need to extract proper features from the data. In this project, we want to explore two ways, named ICA and CSP.

In this section, we will just look at the scatter of the data after applying each algorithm, and more analyses in classification will come in the following sections. Notice that we should find the proper W matrix (in CSP or ICA) only with the training data; otherwise, data leakage will happen. So we plot both the training and test data to see how generalized these algorithms are.

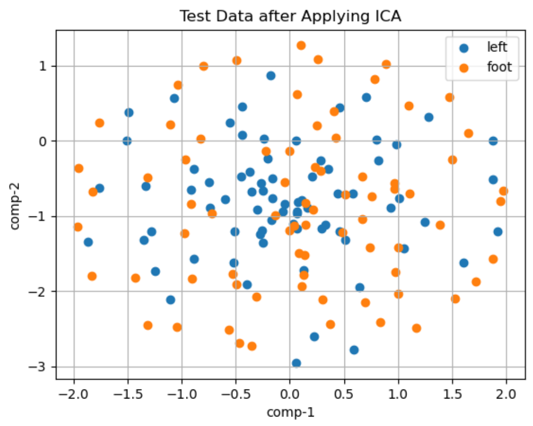

Figure 11: Scatter plot of train data after applying ICA

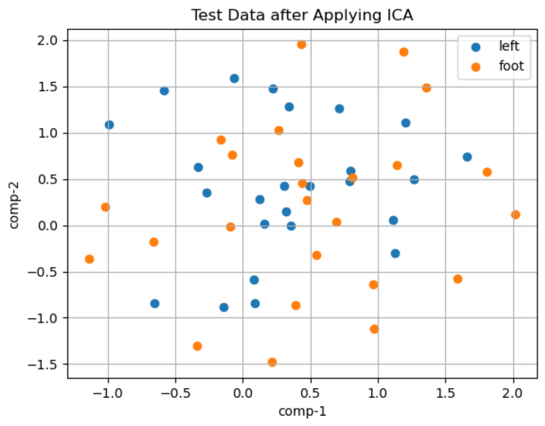

Figure 12: Scatter plot of test data after applying ICA

The data has not separated well, and it is not linearly separable. It will likely require a more complex classifier, such as neural networks, to classify it properly.

However, it is better than the data shown in Figure 10, which means this algorithm has extracted some patterns in the data. Also, there is no significant difference between the training and test data, which suggests the algorithm has not overfitted.

Now let's look at CSP:

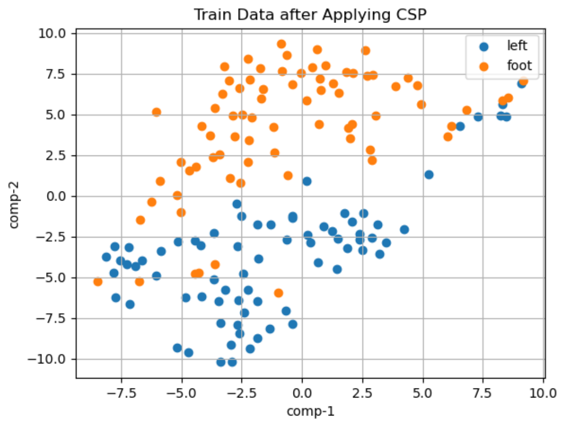

Figure 13: Scatter plot of train data after applying CSP

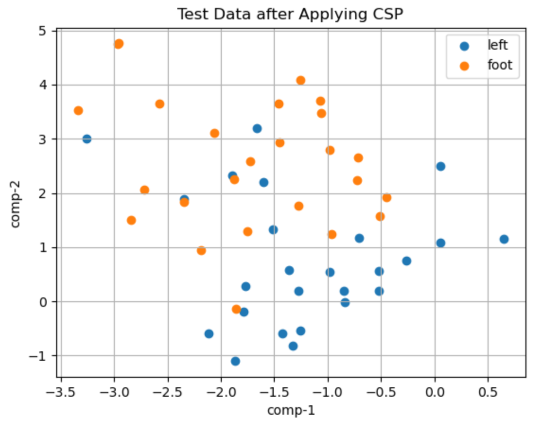

Figure 14: Scatter plot of test data after applying CSP

Great! As you can see in the above figures, both the training and test data are almost linearly separable. This algorithm could capture very meaningful patterns in the data.

Referring to Section 5, both ICA and CSP were almost the same, at least in terms of the idea behind them, which assumes some independent sources and tries to find them. However, there was a huge difference between them: ICA is unsupervised, while CSP is supervised. More accurately, CSP considers different sources for each class, but in ICA, the sources for all classes are the same. This difference in the approaches leads to the observed result.

6.4 Different Classifiers

Now our data is clean enough to classify it. For each classifier, we will report the accuracy, precision, recall, confusion matrix, and ROC curve.

Notice that in this section, we will provide the results of the best-preprocessed data, and the compared results for the different methods of preprocessing will be described later in the paper.

Logistic Regression Since our data is almost linear separable, we expect to get a good result out of this linear classification.

	precision	recall	f1-score
left	0.89	0.96	0.92
foot	0.96	0.88	0.92
accuracy			0.92
macro avg	0.92	0.92	0.92
weighted avg	0.92	0.92	0.92

Table 1: Classification Report of Logistic Regression

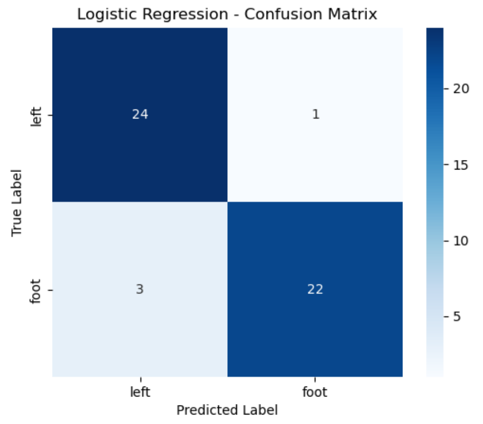

Figure 15: Confusion matrix of logistic regression model

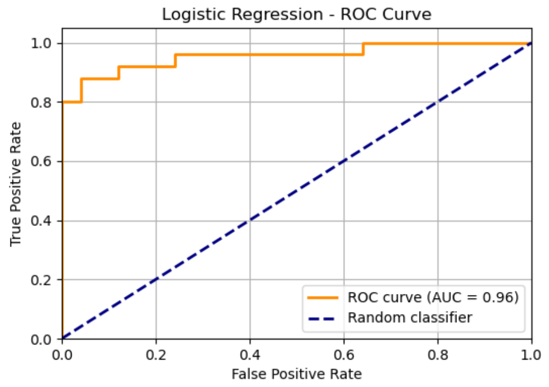

Figure 16: ROC curve of logistic regression model

The model has a very good result in foot class but in the other class is not as well as that.

SVM We train an SVM with RBF kernel and the results are as follows:

	precision	recall	f1-score
left	0.89	0.96	0.92
foot	0.96	0.88	0.92
accuracy			0.92
macro avg	0.92	0.92	0.92
weighted avg	0.92	0.92	0.92

Table 2: Classification Report of SVM

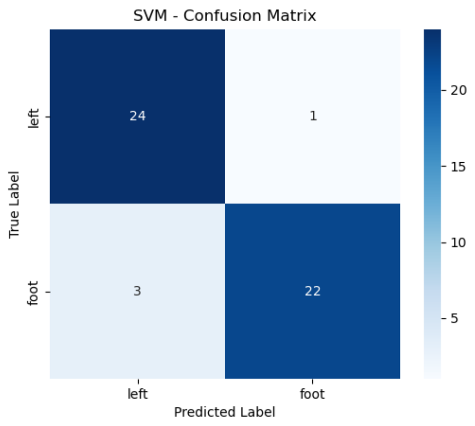

Figure 17: Confusion matrix of SVM model

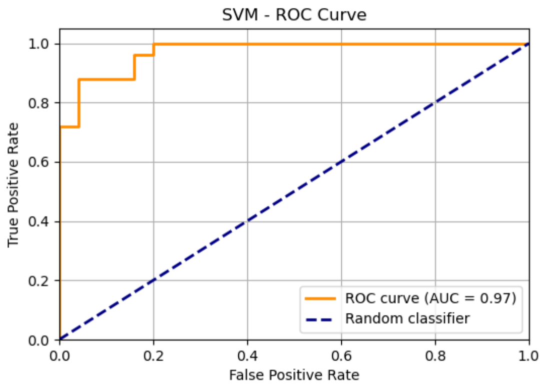

Figure 18: ROC curve of SVM model

KNN We trained the KNN classifier with different values of k and finally found out that 3 is the optimal value of k.

	precision	recall	f1-score
left	0.86	0.96	0.91
foot	0.95	0.84	0.89
accuracy			0.90
macro avg	0.91	0.90	0.90
weighted avg	0.91	0.90	0.90

Table 3: Classification Report of KNN

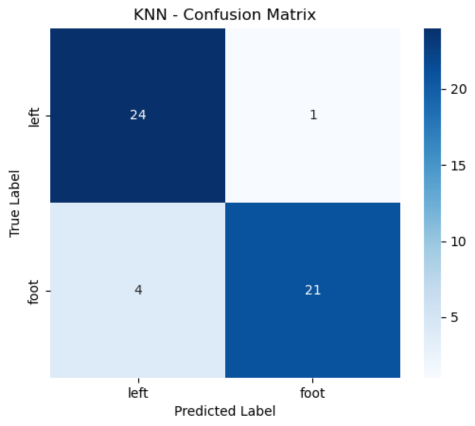

Figure 19: Confusion matrix of KNN model

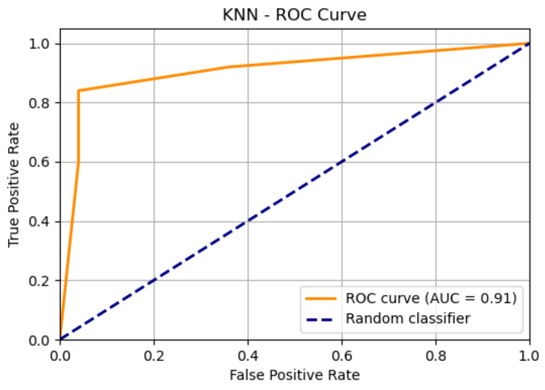

Figure 20: ROC curve of KNN model

MLP MLP (Multi-Layer Perceptron) is a strong classifier, but it requires a large amount of data to train the huge number of parameters. However, in this situation, we do not have enough data.

Therefore, we designed a simple MLP architecture with only one hidden layer. Despite the limited data, the results were still somewhat good.

	precision	recall	f1-score
left	0.85	0.88	0.86
foot	0.88	0.84	0.86
accuracy			0.86
macro avg	0.86	0.86	0.86
weighted avg	0.86	0.86	0.86

Table 4: Classification Report of MLP

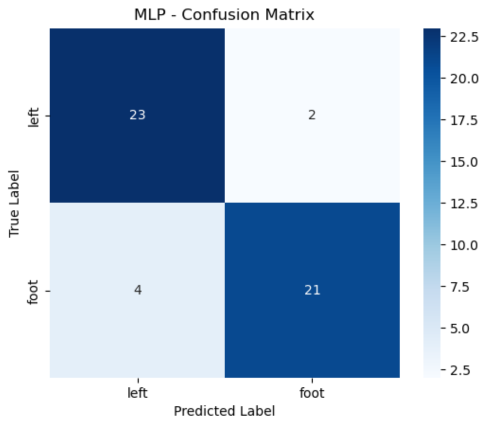

Figure 21: Confusion matrix of MLP model

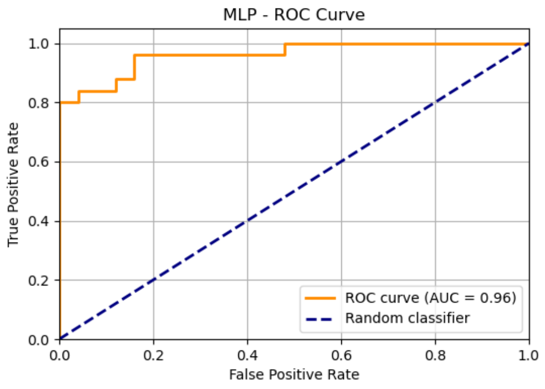

Figure 22: ROC curve of MLP model

AdaBoost For AdaBoost, we use logistic regression as the base estimator. However, there is a slight difference between this and the logistic regression we saw in the previous section.

XGBoost is likely to overfit. To prevent this, we can increase the regularization parameter of the base estimator. This change leads to a less accurate individual classifier, but adding a lot of classifiers in the boosting process makes the overall classifier powerful and generalized.

	precision	recall	f1-score
left	0.89	0.96	0.92
foot	0.96	0.88	0.92
accuracy			0.92
macro avg	0.92	0.92	0.92
weighted avg	0.92	0.92	0.92

Table 5: Classification Report of AdaBoost

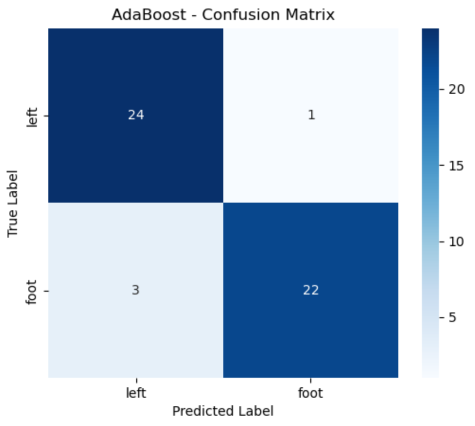

Figure 23: Confusion matrix of AdaBosst model

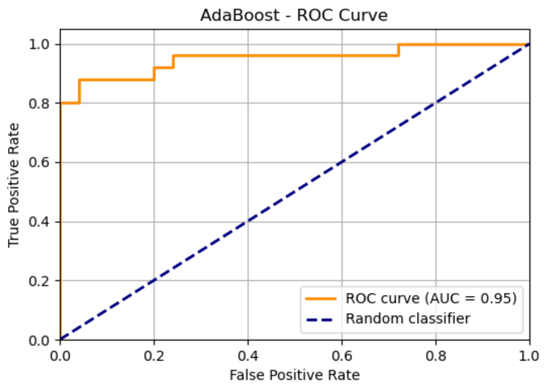

Figure 24: ROC curve of AdaBoost model

XGBoost XGBoost, like AdaBoost, is likely to overfit, so we increased its regularization parameter until we got good accuracy on the test data. As you can see, this model had the best accuracy among all the models we tried.

	precision	recall	f1-score
left	0.89	1.00	0.94
foot	1.00	0.88	0.94
accuracy			0.94
macro avg	0.95	0.94	0.94
weighted avg	0.95	0.94	0.94

Table 6: Classification Report of XGBoost

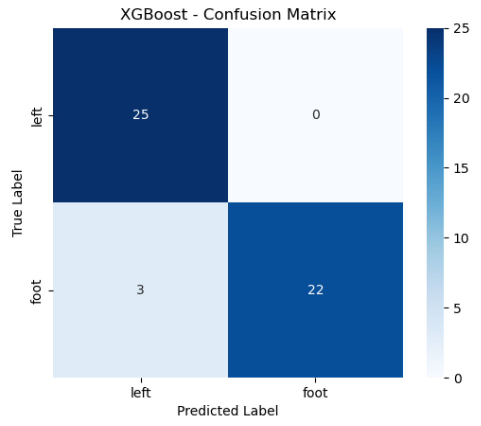

Figure 25: Confusion matrix of XGBoost model

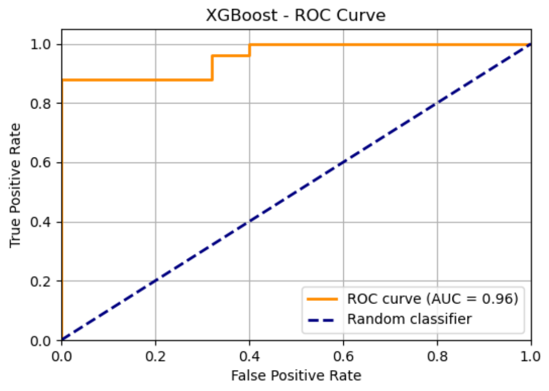

Figure 26: ROC curve of XGBoost model

6.5 Comparing Results

Now we want to see which classifier had better results. For this reason, we use two files from our dataset and apply different preprocessing methods as described in the previous sections. The reason for using more than one file is to see the generalization power of our model since we adjust hyperparameters to get a better result for a file and then test another new file too. You can see the results in Table 7.

In general, CSP without applying PCA and normalization has the best accuracy. By applying PCA, the accuracy drops more than when applying normalization. This result may be due to the feature extraction methods, ICA, or CSP. These methods may need all the data to capture the real pattern of the data, while PCA reduces the dimensions.

Normalization hurts the accuracy too, except in one situation. The reason behind this can be that the absolute voltage of the EEG signal implies some information, and normalization deletes this.

Finally, CSP is better than ICA, as we expected and explained in the previous section when we plotted the scatter plot of the data after applying each method. Referring to Section 5, both ICA and CSP were almost the same, at least in terms of the idea behind them, which assumes some independent sources and tries to find them. However, there was a huge difference between them: ICA is unsupervised, while CSP is supervised. More accurately, CSP considers different sources for each class, but in ICA, the sources for all classes are the same. This difference in the approaches leads to the observed result.

7 Clustering

After applying CSP with 59 components, we have 150 samples, each of which owns a vector of length 59 as their feature space. As a well-known unsupervised method, we now want to cluster our data points, regardless of their labels to see what patterns they hold. As the initial feature space is very sparse due to high dimensionality, we use **Principal Component Analysis** to reduce the dimension to 2 to be able to visualize data points. The algorithms we are applying here are KMeans and DB-scan.In each machine learning model, the first step is to select the proper hyperparameters. Let's explore these algorithms and find their proper hyperparameters. Before that, we should note that using **Common Spatial Pattern**, we are extracting patterns from different channels and as these channels might have some common patterns, in our new feature space, they will lay in the same clusters and some of them which do not share common patterns, appear in different clusters.

7.1 DB-scan

As we know, two hyperparameters have to be selected for this algorithm, **min_pts** and **eps**. One well-known way to find the optimal min_pts and eps is to visualize the k-th nearest neighbor distance of all data points where k is equal to min_pts. The place where this distance starts to rise early, we are in the area of points that are not directly density reachable. Therefore with this method, we can find the optimal eps for each min_pts. Now let's plot this figure for a range of 3 to 12 min_pts:

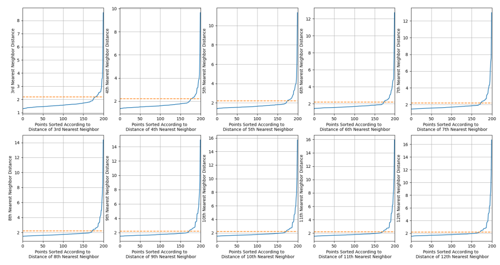

Figure 27: optimal eps for different min_pts

As we see in the above figure, 2.2 is a good eps for most of the min_pts. Now we can select a min_pts.To compare min_pts, we can use the homogeneity and silhouette score to evaluate the quality of our clustering. Doing so and comparing results, we observed that with the following hyperparameters, the best DB-scan clustering is achieved:

Normalization	PCA	Feature	Logistic Regression	SVM	KNN	MLP	AdaBoost	XGBoost
		CSP	**0.92**	**0.92**	**0.90**	**0.86**	**0.92**	**0.94**
✓		CSP	0.86	0.84	0.80	0.84	0.88	0.88
	✓	CSP	0.66	0.66	0.60	0.58	0.62	0.60
✓	✓	CSP	0.62	0.60	0.60	0.60	0.62	0.58
		ICA	0.56	0.50	0.56	0.56	0.56	0.52
✓		ICA	0.64	0.60	0.56	0.50	0.62	0.54
	✓	ICA	0.56	0.52	0.50	0.62	0.56	0.60
✓	✓	ICA	0.64	0.58	0.54	0.52	0.60	0.56
		CSP	**0.84**	**0.84**	**0.86**	**0.84**	**0.84**	0.88
✓		CSP	0.80	0.78	0.74	0.74	0.78	**0.94**
	✓	CSP	0.54	0.52	0.52	0.52	0.52	0.54
		ICA	0.54	0.50	0.50	0.54	0.54	0.56

Table 7: Compare different classification methods and different preprocessing methods: The top and bottom of the table are related to the files BCICIV_calib_ds1a.mat and BCICIV_calib_ds1f.mat, respectively. The numbers represent the accuracy of the model which is between 0 and 1.

- min_pts = 3
- eps = 2.2

which gives the following results:

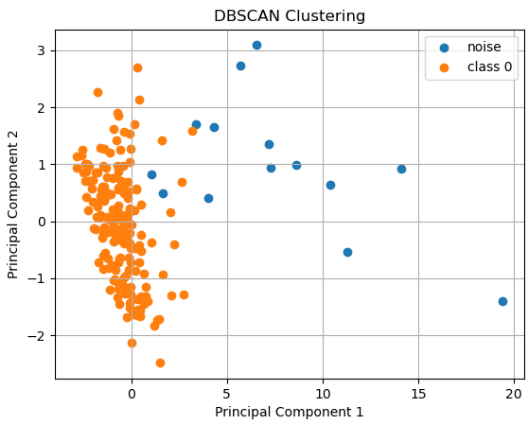

Figure 28: DB-scan visualization

and clustering scores are as follows:

- Homogeneity_score: 0.0012
- Silhouette_score: 0.65

As we see in the Above figure, due to the compression of data samples in feature space, different class labels can not be separated and we get a very low homogeneity score. To get a higher homogeneity score, we need to use other clustering algorithms that are more robust to the data density.

7.2 K-means

In K-means clustering, we can use model inertia to find the optimal hyperparameter which is several clusters(K). Model inertia is the sum of squared differences of data points of each cluster from their centroids. Visualizing inertia per K for K in the range of 1 to 14, we get the following figure:

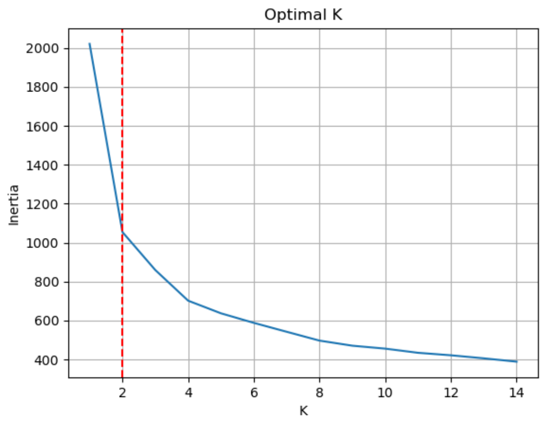

Figure 29: K-means elbow visualization

From the above figure, we observe that we have a knee in $K = 2$.Thus we apply K-means clustering with 2 clusters. The results are as follows:

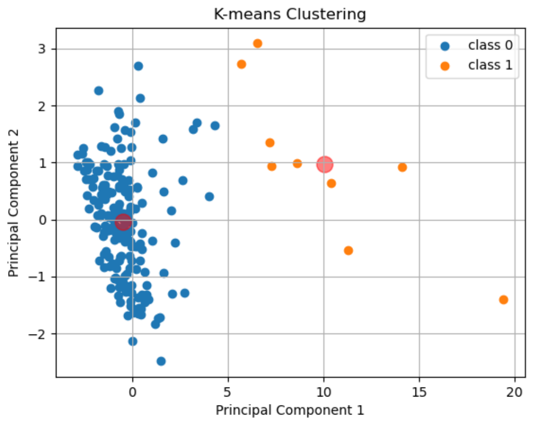

Figure 30: kmeans clustering

and clustering scores are as follows:

- Homogeneity_score: 0.00
- Silhouette_score: 0.72

We got a much better silhouette score with the K-means algorithm.

As suggested in the description of the project, let's visualize three more models with different K:

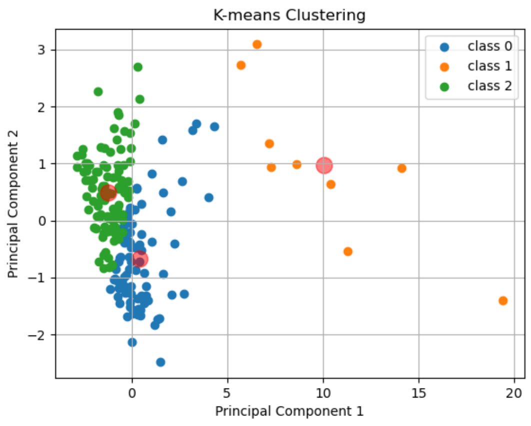

Figure 31: k-means, k=3

clustering scores for k=3:

- Homogeneity_score: 0.19
- Silhouette_score: 0.20

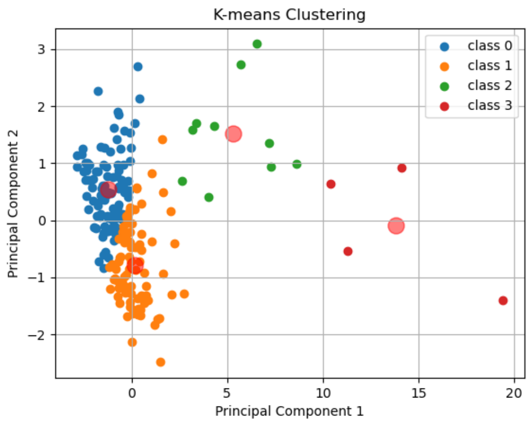

Figure 32: k-means, k=4

clustering scores for k=4:

- Homogeneity_score: 0.24
- Silhouette_score: 0.20

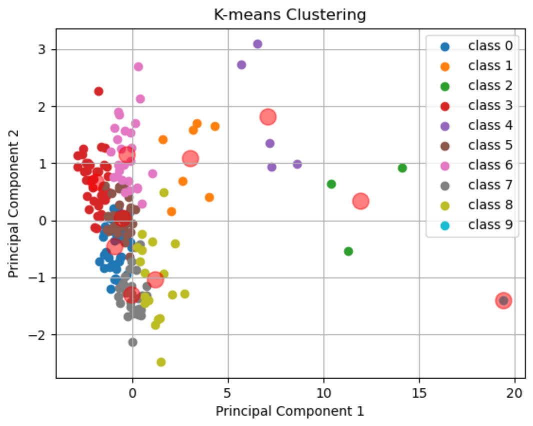

Figure 33: k-means, k=10

clustering scores for k=10:

- Homogeneity_score: 0.53
- Silhouette_score: 0.14

The reason we got better results in k=3 and k=4 despite observing knee at k=2, can be the way we are evaluating the model. First, we are sure that increasing the number of clusters will increase the amount of silhouette score but on the other hand, it might hurt homogeneity score unlike our case here where we saw an increase in the homogeneity score. The justification behind this rise is our feature space. This feature space might not be a decent one for the clustering task and increasing the number of clusters will lead to different label classes, appearing in different clusters with a higher chance and consequently increasing the homogeneity score.

7.3 Kernel K-means

To achieve a higher homogeneity score, one way is to transform data into another space where they can be separated better. Here we use Kernel Trick to get better homogeneity score. The results are as follows:

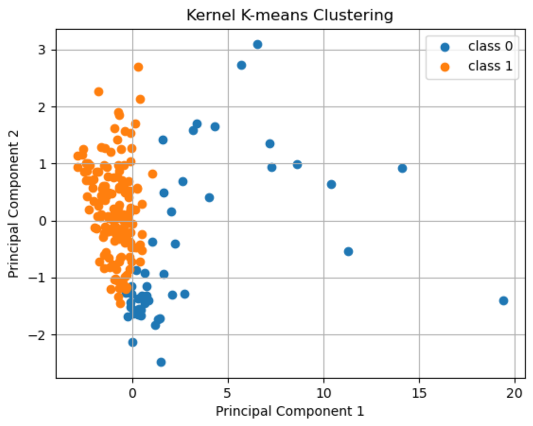

Figure 34: kernel clustering

and the clustering scores are:

- Homogeneity_score: 0.13
- Silhouette_score: 0.34

let's visualize two more models with different K:

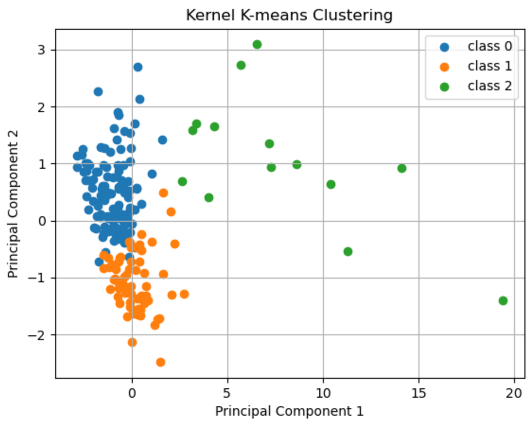

Figure 35: kernel clustering, k = 3

and the clustering scores are:

- Homogeneity_score: 0.42
- Silhouette_score: 0.20

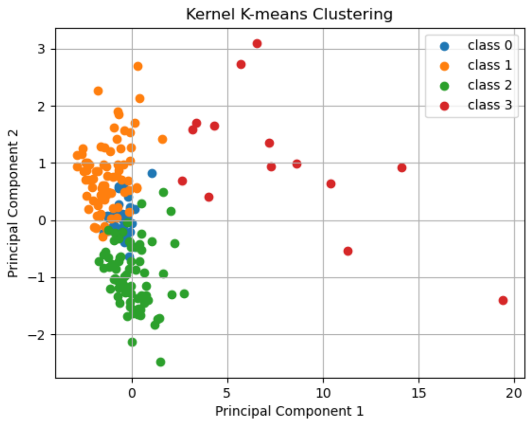

Figure 36: kernel clustering

and the clustering scores are:

- Homogeneity_score: 0.57
- Silhouette_score: 0.13

On average the results are better than k-means.

References

[1] B. Blankertz, R. Tomioka, S. Lemm, M. Kawanabe, and K.-R. Müller, "Optimizing Spatial filters for Robust EEG Single-Trial Analysis," IEEE Signal Proc. Magazine, 2008.

[2] W. Samek, M. Kawanabe and, K.-R Müller, "Divergence-based Framework for Common Spatial Patterns Algorithms", IEEE, 2013.

[3] M. Kawanabe and C. Vidaurre, "Improving BCI performance by modified common spatial patterns with robustly averaged covariance matrices," , 2009.

[4] J. Mouriño, J. Millán1, F. Cincotti2, S. Chiappa1, R. Jané, and F. Babiloni, "spatial filtering in the training process of a brain computer interface", 2021.

[5] O. Hauk, "Keep it simple: a case for using classical minimum norm estimation in the analysis of EEG and MEG data", 2003.

[6] C. Carvalhaesa, and J. Barrosb, "The Surface Laplacian Technique in EEG: Theory and Methods", 2015.

[7] D. McFarland, L. McCane, S. David, and J. Wolpaw, "Spatial filter selection for EEG-based communication", 1997.

[8] T. Müller, T. Ball, R. Kristeva-Feige, T. Mergner, and J. Timmer, "Selecting relevant electrode positions for classification tasks based on the electro-encephalogram", 2000.

[9] S. Gannouni, K. Belwafi, H. Aboalsamh, Z. AlSamhan, B. Alebdi, Y. Almassad, and H. Alobaedallah, "EEG-Based BCI System to Detect Fingers Movements, College of Computer and Information Sciences", King Saud University, 2020.

[10] M. Stropahl, A.-K. Bauer, S. Debener,and M.Bleichner, "Source-Modeling Auditory Processes of EEG Data Using EEGLAB and Brainstorm", 2018.

[11] A. Singh, A. Hussain, S. Lal, H. Guesgen, "A Comprehensive Review on Critical Issues and Possible Solutions of Motor Imagery Based Electroencephalography Brain-Computer Interface", School of Fundamental Sciences, Massey University, 2021.