Sharif University Of Technology

BME

EMG Processing

MohammadErfan Sheykh Jebeli (99101795)
Saeed Mansur Lakuraj (99102304)

Professor's Name : Dr. Jahed

July 3, 2024

1 Plot Before and After Filtering

1.1 Before Filtering

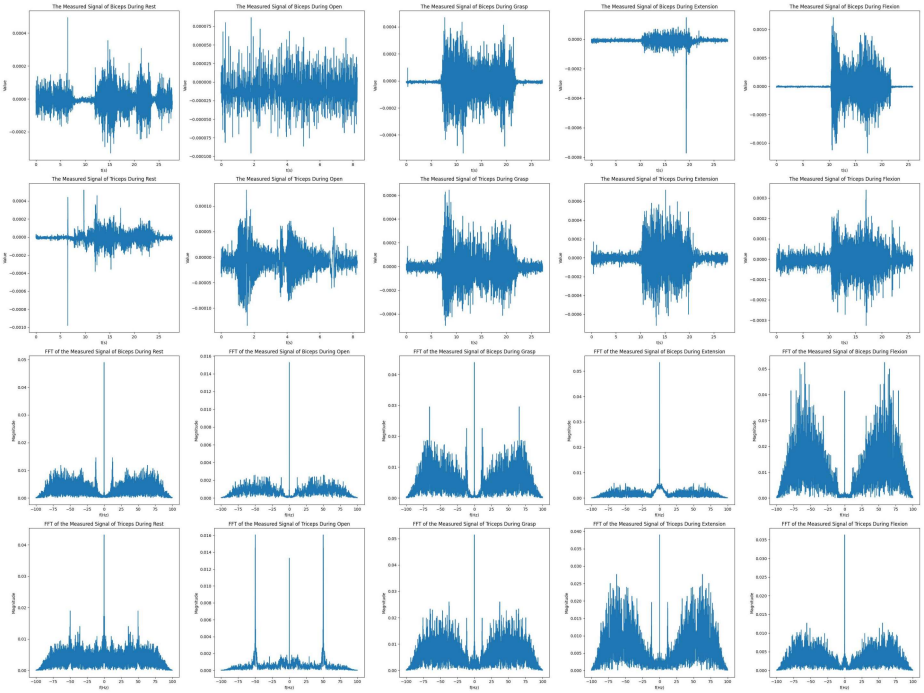

Figure 1: Specified Signals Before Filtering

1.2 After Filtering

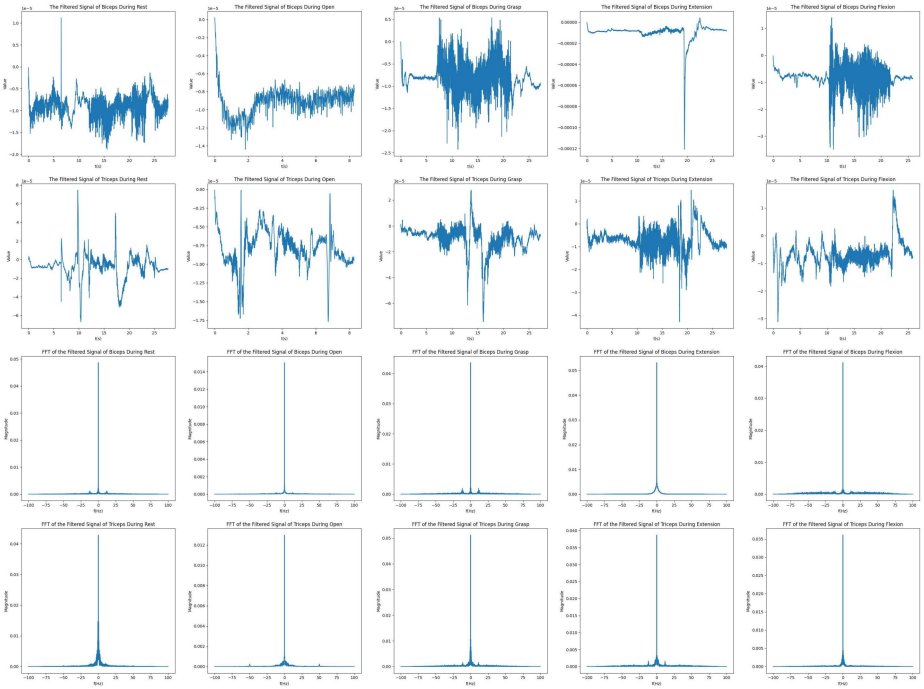

Figure 2: Specified Signals After Filtering

2 Processing, Feature Extraction, and Splitting

Question: What is Test-Train-Validation Split? and Why?
Answer: In machine learning, we typically divide our dataset into two or three subsets: the training set, the validation set (optional), and the test set. This process is known as **train-test split**. Here's a thorough explanation of why we do this:

1. Training Set: The training set is used to train the model. It is the largest subset, often comprising 60-80% of the total dataset. The model learns from this data and adjusts its internal parameters to minimize the error in its predictions.

2. Validation Set: The validation set is used to tune the model's hyperparameters and make decisions about the model's structure (like the number of hidden layers in a neural network). It's like a practice exam for the model before the final test. It's typically about 10-20% of the dataset.

3. Test Set: The test set is used to evaluate the final performance of the model. It serves as new, unseen data for the model. This is crucial because it gives us an unbiased estimate of how well our model is likely to perform on unseen real-world data. It's typically about 10-20% of the dataset.

The main reasons for doing a train-test split are:

- Avoid Overfitting: Overfitting occurs when a model learns the training data too well, including its noise and outliers, and performs poorly on new, unseen data. By using a separate test set, we can ensure that our model generalizes well to new data.

- Unbiased Evaluation: The test set provides an unbiased way to assess the performance of our model. Since the model hasn't seen this data during training, the performance on the test set is a good indicator of how the model will perform on unseen real-world data.

- Hyperparameter Tuning: The validation set allows us to tune the model's hyperparameters without touching the test set. This helps prevent "information leakage" from the test set into the model during training.

- Model Selection: By evaluating different models' performance on the validation set, we can choose the best model before testing it on the test set.

In summary, the train-test split is a crucial part of machine learning that helps us create models that generalize well to new, unseen data, avoid overfitting, and provide an unbiased evaluation of the model's performance. It's a fundamental practice in creating reliable and robust machine learning models.

Procedure: In this part, after filtering, we extracted all the mentioned features, from the specified windowing, used the standard scaling(mean=0, std=1), and performed test-train split of 80-20%.

3 Model Selection, and Model Training

Among the given classifiers, KNN and Random Forest showed the best test accuracy.(KNN: 93.88%, Random Forest: 99.35%) And, therefore, we have chosen them to report our accuracies! Confusion matrices for these two methods are as follows:

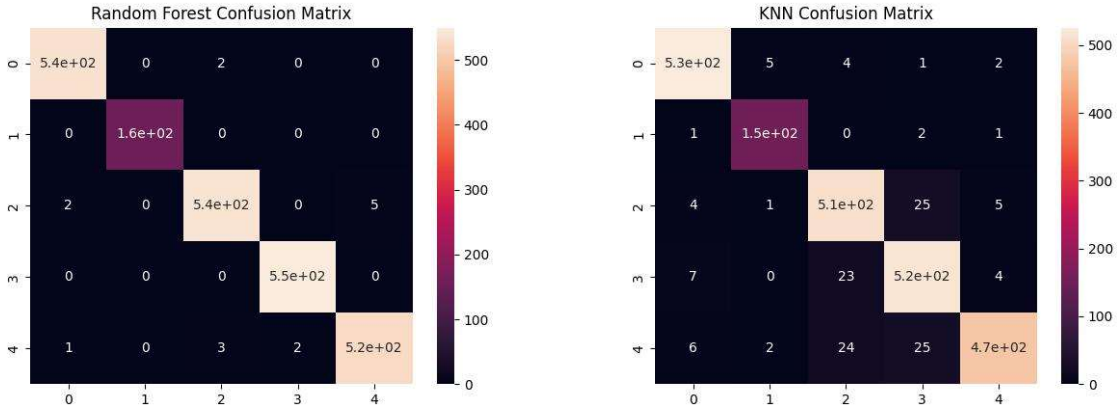

Figure 3: The Confusion Matrix on the Test Set for Random Forest and KNN

4 Issues Encoutered

The most challenging aspect of our project was the extraction of the original signal from the ADInstruments data file (‘.adicht‘). The initial step involved converting the ‘.adicht‘ file into a MATLAB file (‘.mat‘). This was not a straightforward task and required extensive online research to find a suitable MATLAB code that could perform this conversion.

Once we had the ‘.mat‘ file, the next hurdle was to extract the original signal from it. The ‘.mat‘ file was loaded as a dictionary in Google Colab, a cloud-based Python environment. Understanding the structure of this dictionary and locating the original signal within it proved to be a complex task. It was only after numerous attempts and a process of trial and error that we were finally able to successfully extract the original signal.

This experience is not unique to our project but is a common challenge in many machine learning problems. Data preprocessing, which includes tasks such as signal extraction, is often the most time-consuming part of a machine learning project. For instance, in image recognition tasks, raw images often need to be preprocessed to normalize lighting conditions, align the subject, and crop irrelevant parts of the image. Similarly, in natural language processing tasks, raw text data often needs to be preprocessed to remove stop words, perform stemming, and convert words to numerical vectors (a process known as word embedding).

In all these cases, the goal of preprocessing is to convert raw data into a format that the machine learning algorithm can use effectively. Despite its challenges, successful data preprocessing is crucial as the quality of the data and the manner in which it is prepared can significantly influence the performance of the final machine learning model. Therefore, while our project presented us with a steep learning curve, it also provided us with valuable experience in handling and manipulating real-world data, a skill that is essential in the field of machine learning.