TECHNISCHE UNIVERSITÄT
CHEMNITZ

Department of Electrical Engineering and Information Technology

Chair of Measurement and Sensor Technology

Project Documentation

„Project Lab Embedded Systems"

Group:	**17**
Members:	**Mr. Swapnil Subhash Pawar**
	Mr. Ruchikkumar Tejani
	Mr. Bavly Mikhail
Project:	**Hand Gesture Recognition Using Machine Learning**
Date:	**16th June 2023**

Table of Contents

1 General Introduction

Hand gesture recognition is a field of study that utilizes machine learning techniques to interpret and understand human hand movements and gestures. It has extensive practical applications in areas such as human-computer interaction, sign language recognition, and virtual reality control.

A hand gesture recognition system based on machine learning typically involves components such as an EMG (electromyography) sensor and an Arduino Nano 33 BLE. The objective of such a project is to predict hand gestures by capturing EMG data from hand movements and transmitting it to a laptop using serial communication. The EMG sensor, such as the Myoware muscle sensor, detects and records the electrical signals produced by muscle activity during hand movements. The Arduino Nano 33 BLE acts as the central controller, collecting EMG data from the sensor and establishing a connection with the laptop.

To recognize and interpret hand gestures, machine learning algorithms like decision trees are employed in the project. The decision trees algorithm is trained using a labeled dataset that associates EMG data with different hand gestures. By learning the unique patterns and characteristics of each gesture, the algorithm becomes capable of classifying and identifying the gestures based on the EMG data received from the Arduino.

By integrating the EMG sensor, Arduino Nano 33 BLE, serial communication, and the decision trees machine learning algorithm, a hand gesture recognition system can be developed. This system has potential applications in various fields, including human-computer interaction, virtual reality control, and prosthetics, where hand gestures can serve as an intuitive and natural input method.

1.1 Member Responsibilities

Mr. Swapnil Subhash Pawar	Hardware development, Data acquisition and source code development
Mr. Ruchikkumar Tejani	Feature extraction and feature selection
Mr. Bavly Mikhail	Machine learning model development

2 Theoretical background and state of the art

2.1 Hand gesture recognition

Hand gesture recognition refers to the automated identification and interpretation of gestures made by human hands or fingers, with the aim of comprehending and responding to their intended meaning or command. This process entails the capture and analysis of hand and finger motion, shape, and position, utilizing diverse sensing technologies like cameras, depth sensors, or electromyography (EMG) sensor.

The primary objective of hand gesture recognition is to precisely and effectively interpret the gestures made by the user and convert them into meaningful commands or actions. This facilitates smooth and natural interaction between humans and machines, resulting in an improved user experience and the emergence of novel forms of human-computer interfaces.

2.2 State of the art

Table 2.1: State of the art of Hand gesture recognition

Reference	Application	Sensors	placement	Signal Processing	Classification Algorithm	Performance Evaluation & accuracy
[1] 2022	Artificial limb	EMG sensor	Forearm	Low-pass Butterworth filter	DNN	81.65%
[2] 2022	3D printed robotic hand	EMG + IMU	Forearm	Convolution, then feeding it to a fully connected layer	ANN	N/A
[3] 2019	Wearable arm band	3 Myoware sensor	Forearm, and wrist	Root-Mean-Square (RMS) based rectification	CNN+ bidirectional-LSTM	93.7%
[4] 2018	Robotics, gaming	EMG sensor	Arm cortex, Forearm	Rectify the signal, low-pass Butterworth filter	Feed-forward Neural Network	90.1%
[5] 2023	Robotic prosthesis	Myo armband	Forearm	Feature extraction, Frequency features	KNN	N/A
[6] 2023	Prosthetic hand	Myoware sensor	Forearm	Windowing, Wavelet transform	ELM, SVM, Fine trees	95%
[7] 2022	Wearable Sensors for Digital Healthcare	Myo armband	Forearm	Comparison of hand gesture recognition system.	SVC with RBF Kernel, RF	92%
[8] 2019	Sign language	visible RGB	N/A	Spatial temporal Semantic Alignment.	3D-CNNs	88.4%

In the field of hand gesture recognition, the state of the art refers to the most advanced approaches and systems that achieve accurate and efficient recognition and interpretation of hand gestures. Through extensive research, we identified the best hardware and software components for our project. The Myoarm band and Myoware muscle sensor emerged as precise and accurate tools for detecting EMG signals by capturing changes in muscle contraction. The Arduino Nano 33 BLE stood out among other controller boards due to its low energy consumption and cost-effectiveness.

For software, we selected the Arduino IDE for coding and programming Arduino microcontrollers. Python was chosen for establishing serial communication with the Arduino board and receiving data on the laptop, thanks to its approachable syntax, extensive library support, and active developer community. MATLAB was selected for signal processing and implementing machine learning algorithms, leveraging its robust visualization capabilities and tools like the classification learner.

2.3 Hardware

The hardware component involves the utilization of the Arduino Nano 33 BLE board and two EMG Myoware sensors. The Arduino Nano 33 BLE functions as the central controller and data collection unit. Equipped with Bluetooth Low Energy (BLE) capabilities, it facilitates wireless communication. The EMG Myoware sensors are connected to the Arduino board to capture the electrical signals generated by muscle activity during hand movements. These sensors are designed specifically for EMG measurement and offer reliable interfaces for detecting muscle activity in targeted hand muscle groups.

Controller:

The Arduino Nano 33 BLE features the Nordic Semiconductor nRF52840 microcontroller with a 64 MHz ARM Cortex-M4 processor. It consists of 14 digital I/O pins ,8 analog pins, 6 PWM pins with 3.3V operating voltage. This powerful microcontroller offers substantial computational capabilities for various applications. The board includes built-in 2.4 GHz BLE connectivity, allowing seamless wireless communication with BLE-compatible devices. Serial communication is currently utilized for data transmission, but future involve leveraging the BLE functionality for wireless data transmission. The Arduino Nano 33 BLE is preferred for its low power consumption, small form factor, powerful microcontroller, cost-effectiveness, and compatibility with the Arduino ecosystem [9].

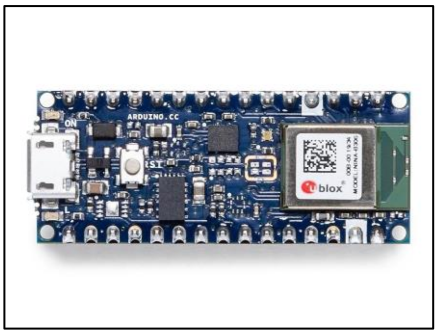

Figure 2.1. Arduino nano 33 BLE

EMG (electromyography) sensor:

An EMG (electromyography) sensor detects and measures the electrical signals produced by muscles. It uses electrodes to capture muscle contractions, amplifying and filtering the signals for accurate measurement. EMG sensors find applications in research, healthcare, and human-computer interaction. In hand gesture recognition systems, they play a vital role in capturing muscle signals for gesture interpretation. Different types of EMG sensors are available, including surface, intramuscular, wireless, dry, wet, and integrated sensors. For our project, we selected the Myoware muscle sensor, a surface EMG sensor, based on careful evaluation and alignment with our project requirements and objectives.

MyoWare sensor:

The MyoWare sensor consists of a sensor board and electrode pads. The sensor board amplifies and processes EMG signals, while the electrode pads detect muscle activity. These pads are designed for easy attachment to the skin and ensure optimal electrical contact using conductive gel. The sensor offers a user-friendly and compact solution for capturing EMG signals, making it suitable for various applications such as research, and human-computer interaction. It is commonly used in hand gesture recognition systems and projects requiring muscle activity measurement and analysis [10].

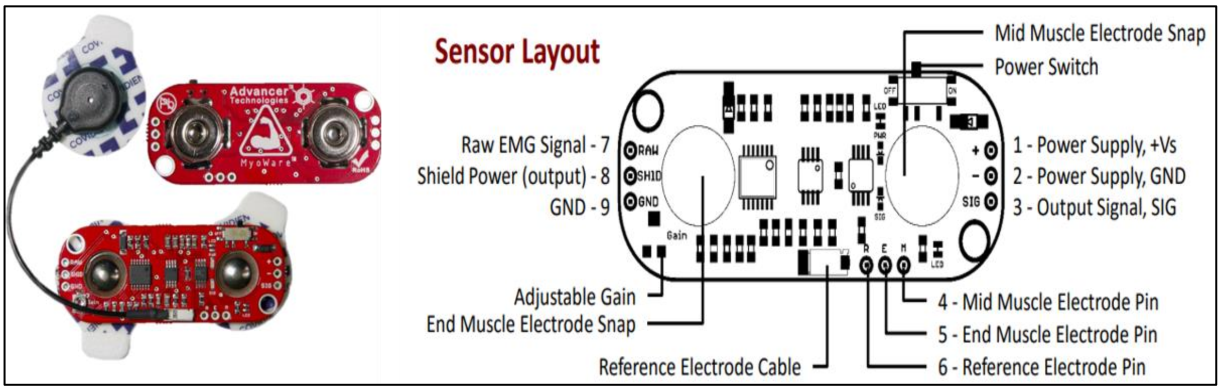
Sensor Layout
Mid Muscle Electrode Snap
Power Switch
Raw EMG Signal - 7
Shield Power (output) - 8
GND - 9
1 - Power Supply, +Vs
2 - Power Supply, GND
3 - Output Signal, SIG
Adjustable Gain
End Muscle Electrode Snap
Reference Electrode Cable
4 - Mid Muscle Electrode Pin
5 - End Muscle Electrode Pin
6 - Reference Electrode Pin

Figure 2.2. Myoware sensor

2.4 Software

The acquired data from the EMG MyoWare sensors contains valuable information about muscle activity during hand movements. The Arduino board reads the analog signals from the sensors and converts them into digital data. This data, which includes amplitude, frequency, and timing information further processed, analysed, and utilized for the feature extraction to recognise the gesture.

Arduino IDE:

The Arduino IDE is a software application designed for programming Arduino boards. It provides a user-friendly interface for tasks like writing code, compiling, and uploading to Arduino microcontrollers. The IDE includes features such as a serial monitor for real-time data communication and a code editor with advanced functionalities. To receive EMG data from two sensors on an Arduino Nano 33 BLE, analog input pins are configured, and appropriate programming techniques are used to capture the analog values. The data can then be processed and utilized for hand gesture recognition.

Python:

Python is a widely used programming language that is well-suited for receiver part of our project. It provides simplicity, platform independence, integration capabilities, debugging tools, and an extensive library ecosystem. We developed a Python script to establishing the serial communication with an Arduino nano 33 BLE for data reception and storage in an Excel file.

MATLAB:

MATLAB is a comprehensive programming platform that caters specifically to the needs of engineers and scientists. It provides a powerful environment for system analysis and design, with a strong emphasis on computational mathematics. MATLAB offers extensive support for machine learning, empowering users to leverage algorithms and techniques for data analysis, prediction, and decision-making. It provides a wide range of built-in functions, libraries, and tools for tasks such as data pre-processing, data visualisation, feature selection, model training, evaluation, and deployment. Additionally, MATLAB excels in creating graphical user interfaces (GUIs), facilitating the development of interactive applications with customizable elements. With its rich set of features and capabilities, MATLAB is an indispensable tool for engineers and scientists seeking to solve complex problems and drive innovation in various domains.

3 Data processing

3.1 Data acquisition

In hand gesture recognition using MyoWare sensors, precise sensor placement is crucial for accurate and reliable gesture detection. Strategic placement on specific muscle groups captures the electrical signals generated during hand movements. Optimal sensor positioning on the forearm targets relevant muscles for the intended gestures. To minimize noise interference, we ensured hair removal from the application area, promoting optimal electrode contact.

Hand gestures can be categorized as static or dynamic. Static gestures involve stationary hand poses, while dynamic gestures incorporate continuous or repeated hand movements for communication and expression.

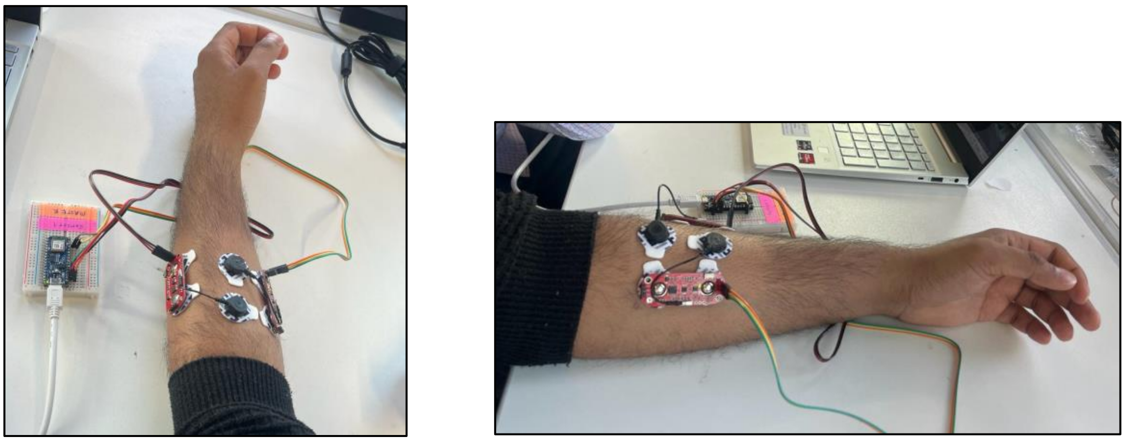

Figure 3.1 EMG sensor placement

In our project, we carefully selected three distinct static hand gestures that exhibit unique muscle activation patterns in the forearm. By capturing and analysing the specific muscle activity associated with each gesture, we aim to achieve accurate recognition and classification in our hand gesture recognition system.

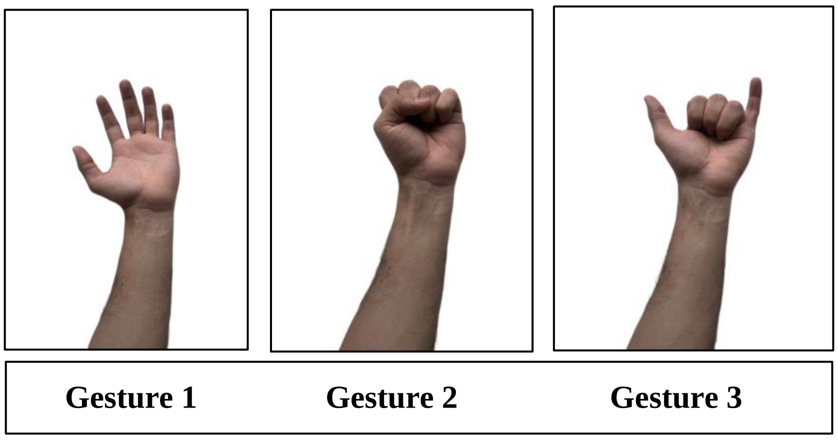

Figure 3.2. Types of gestures

We used a sampling rate of 200 samples per second to accurately capture rapid changes and finer details in the EMG signals. To receive and store the data, we developed a Python script that establishes serial communication with an Arduino Nano 33 BLE device. The script reads data at a 5-second interval, consisting of elapsed time, EMG1, and EMG2 values. The acquired data is displayed in the console and written to an Excel workbook. This implementation provides an efficient solution for EMG data acquisition and management in our project.

Here are the gesture patterns obtained from the data collected using two EMG sensors.

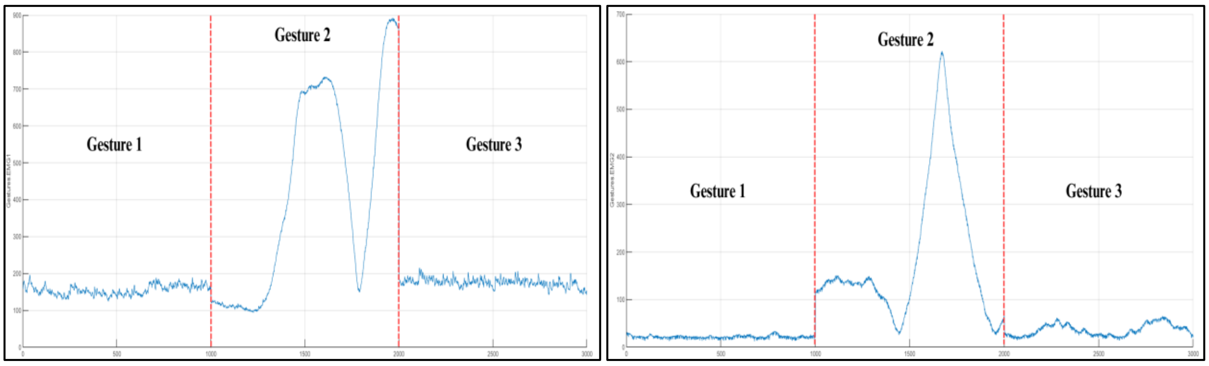

Figure 3.3 Gesture waveforms of EMG1 and EMG2 sensors

3.2 Feature extraction

Feature extraction is a crucial step in data analysis, where raw data is transformed into a concise and meaningful set of features. In our project, we emphasize time domain feature extraction techniques for signal analysis. We specifically focus on three important time domain features: Root Mean Square (RMS), Variance, and Standard Deviation. These features capture the statistical properties of the signals, enabling us to gain insights and improve prediction accuracy.

Root Mean Square (RMS)

RMS represents the square root of the mean of the squared values of the signal samples. It indicates the overall magnitude or energy of the signal. The RMS feature can be calculated using the equation:

$$\mathbf{RMS} = \sqrt{\frac{1}{N} * \sum x^2}$$

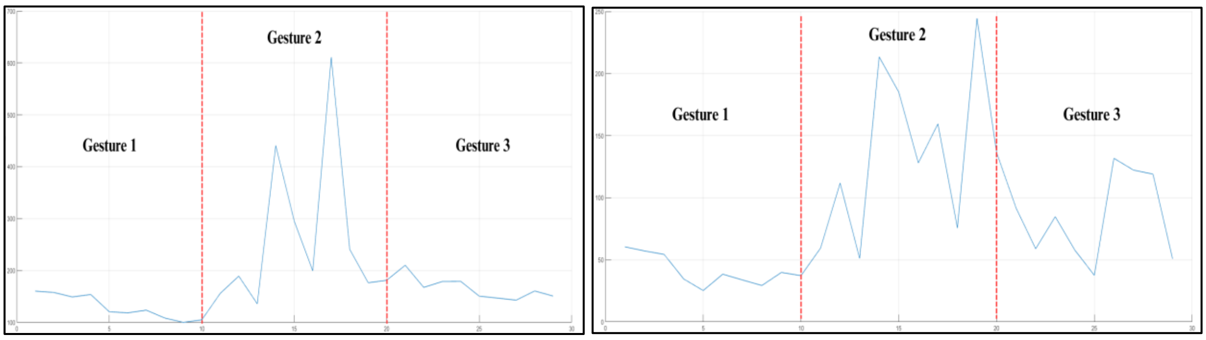

Figure 3.4 RMS waveforms of EMG1 and EMG2

Variance

Variance is a statistical measure that quantifies the dispersion or variability of a signal. It reflects how individual sample values deviate from the mean of the signal. To calculate variance, we find the average of the squared differences between each sample value and the mean of the signal. This provides valuable information about the spread and distribution of the data, allowing us to understand the level of variability present in the signal.

$$\textbf{Variance} = \left(\frac{1}{N}\right) * \sum (x - mean)^2$$

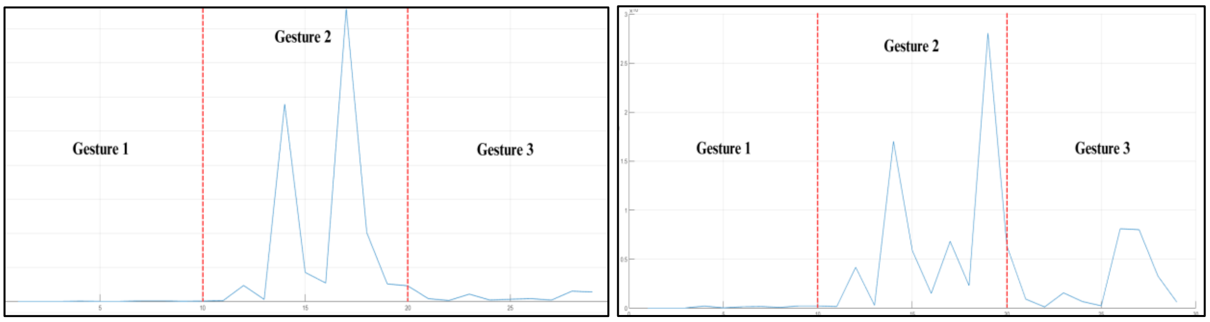

Figure 3.5 Variance waveforms of EMG1 and EMG2

Standard Deviation

Standard Deviation is closely related to variance and provides a measure of the dispersion or spread of the signal. It is the square root of the variance:

$$\textbf{Standard Deviation} = \sqrt{\textbf{Variance}}$$

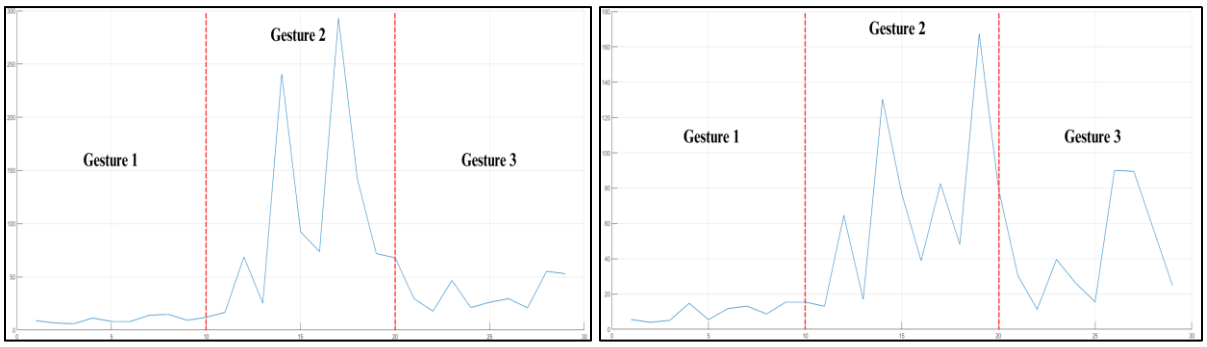

Figure 3.6 Standard deviation waveforms for EMG1 and EMG2

In our project, we utilized the windowing method with a window size of 1500 samples for calculating the time domain features (RMS, Variance, and Standard Deviation). The windowing method involves dividing the signal into smaller segments or windows to analyze localized portions of the data. This approach enables us to extract key characteristics and gain insights into the signal's energy, variability, and statistical properties.

4 Machine learning classification algorithm

Machine learning is a branch of artificial intelligence that uses statistical techniques to analyze data and extract patterns for making predictions or decisions. It encompasses supervised, unsupervised, and reinforcement learning approaches. Machine learning finds applications in computer vision, natural language processing, recommendation systems, and predictive analytics, among others. It automates tasks, provides insights, and improves decision-making processes in diverse industries.

4.1 Model Development

The Fine Tree model in the Classifier Learner app of MATLAB is a user-friendly and effective tool for developing machine learning models. It is well-suited for classification tasks and offers simplicity and interpretability. The model can handle various types of data, including numeric and categorical variables, making it versatile. Additionally, the app provides features like automated hyperparameter optimization and integrates seamlessly with the MATLAB environment, allowing for efficient model development and analysis. Overall, the Fine Tree model is an asset for creating accurate and interpretable machine learning models in a straightforward manner.

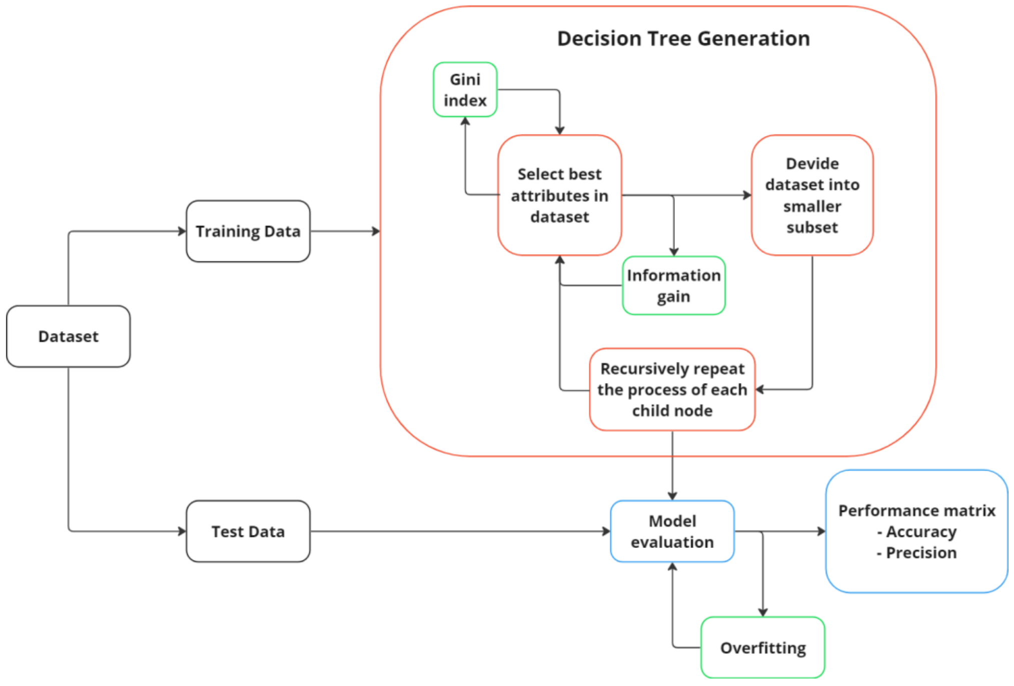

Figure 4.1 Working of the Fine tree classifier

The Fine Tree algorithm in MATLAB is a classification method that constructs decision trees for predictive modeling. It follows a recursive binary partitioning process to build the tree structure. Here is a concise explanation of how the Fine Tree algorithm works:

- **Tree Construction:** The Fine Tree algorithm begins with the entire training dataset and recursively selects the best features to split the data based on criteria such as information gain or Gini impurity. This process continues until a stopping criterion is met, such as reaching a maximum tree depth or having a minimum number of samples for splitting.

- **Leaf Node Assignment:** After constructing the tree structure, the algorithm assigns class labels to the leaf nodes. The label for each leaf node is determined based on the majority class of the training samples falling into that specific node.
- **Prediction:** To classify new, unseen data, the Fine Tree algorithm navigates the tree from the root node to a leaf node, considering the feature values of the input. The class label assigned to the leaf node is then used as the predicted class for the input data.

The Fine Tree algorithm incorporates techniques such as pruning and node purity considerations to improve model performance and prevent overfitting. Pruning involves simplifying the tree by removing unnecessary nodes, enhancing the generalization capability of the model.

4.2 Feature selection

Feature selection is the process of choosing relevant features from a set of available features to enhance model performance and interpretability by eliminating irrelevant or redundant features. By carefully selecting a smaller set of features, it becomes easier to interpret and understand how the model behaves and makes decisions. It improves computational efficiency by reducing the amount of data that needs to be processed. It helps in achieving shorter model training times, as fewer features require less computation during the learning process.

In our project we initially extracted six features from our dataset and achieved a high accuracy of 93.1%. However, by using feature selection techniques, we identified that a single feature, specifically the standard deviation of the first EMG sensor, provided the same level of accuracy.

4.3 Model evaluation

In our machine learning model, we employed the Fine Tree algorithm to classify a gesture dataset. To ensure reliable performance evaluation, we utilized the 5-fold cross-validation technique, which involves dividing the dataset into five subsets and performing training and testing on different combinations. After training and evaluating the model, we achieved an accuracy of 93.1%. This accuracy metric indicates the percentage of correctly classified instances in the dataset. By utilizing the Fine Tree algorithm and applying cross-validation, we aimed to build a robust and accurate model for gesture classification. The achieved accuracy demonstrates the effectiveness of our approach in accurately predicting and classifying gestures in the dataset.

Classification model is evaluated using various metrics to assess their performance. We used below metrics to assess the performance of our model.

1. **Accuracy:** Accuracy is a performance metric in machine learning that quantifies the proportion of correctly classified instances out of the total number of instances in a dataset. It is calculated using the following equation:

$$\textbf{Accuracy} = \frac{\textbf{Number of Correct Predictions}}{\textbf{Total Number of Predictions}}$$

   Our model achieved an accuracy of **91.3%**, indicating its strong performance in correctly classifying instances. The results highlight the effectiveness of our machine learning approach in accurately recognizing and categorizing hand gestures. This high accuracy demonstrates the model's ability to reliably interpret and respond to the selected 3 gestures.

2. **Confusion matrix:** It summarizes the model's predictions by counting the occurrences of true positive, true negative, false positive, and false negative. Each cell in the confusion matrix represents specific outcomes:
    - True Positive (TP): Correctly predicted positive class.
    - True Negative (TN): Correctly predicted negative class.
    - False Positive (FP): Incorrectly predicted positive class when the actual class was negative.
    - False Negative (FN): Incorrectly predicted negative class when the actual class was positive.

    Analyzing the values in the confusion matrix helps assess the accuracy of the classification algorithm and gain insights into the types of errors it makes.

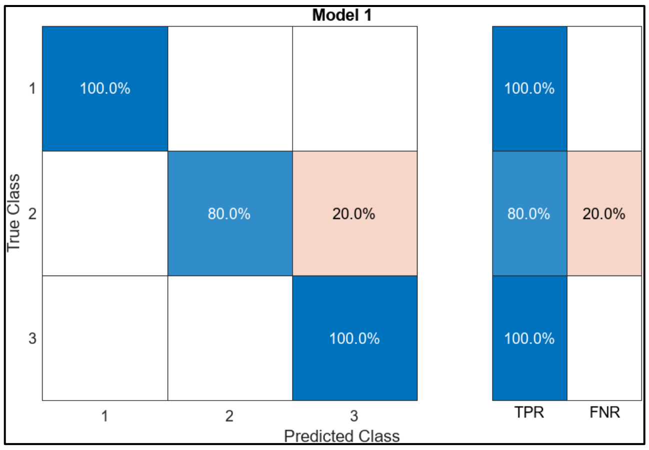

Figure 4.2 Confusion matrix

3. **True Positive Rate (TPR):** The true positive rate (TPR), also known as sensitivity or recall, is a performance metric that quantifies the proportion of positive instances correctly identified by a classification model out of the total actual positive instances in a dataset. It is calculated using the formula:

$$\mathbf{TPR} = \frac{\mathbf{TP}}{\mathbf{TP + FN}}$$

4. **The False Negative Rate (FNR):** The False Negative Rate (FNR) is a performance metric that measures the proportion of negative instances incorrectly classified as positive by a classification model. It is calculated using the formula:

$$\mathbf{FNR} = \frac{\mathbf{FN}}{\mathbf{TP + FN}}$$

Figure 4.3 illustrates the TPR (True Positive Rate) and FNR (False Negative Rate) values associated with each gesture in the dataset. The TPR indicates the proportion of correctly identified positive instances, while the FNR represents the rate of incorrectly classified negative instances. These values provide valuable insights into the model's performance and its effectiveness in accurately detecting and classifying gestures.

5 Conclusion and future work

The project titled "Hand Gesture Recognition using Machine Learning" aimed to classify hand gestures based on EMG data collected from two sensors. Feature extraction involved computing RMS, variance, and standard deviation for each sensor data using a windowing method with a window size of 1500 samples. This yielded a total of six features per gesture. By training a Fine Tree machine learning model with 5-fold cross-validation, we achieved an impressive accuracy of 93.1%. Surprisingly, through feature selection, we found that the same accuracy could be attained by using just one feature: the standard deviation of the first EMG sensor. This reduction in features streamlined the classification process without sacrificing accuracy.

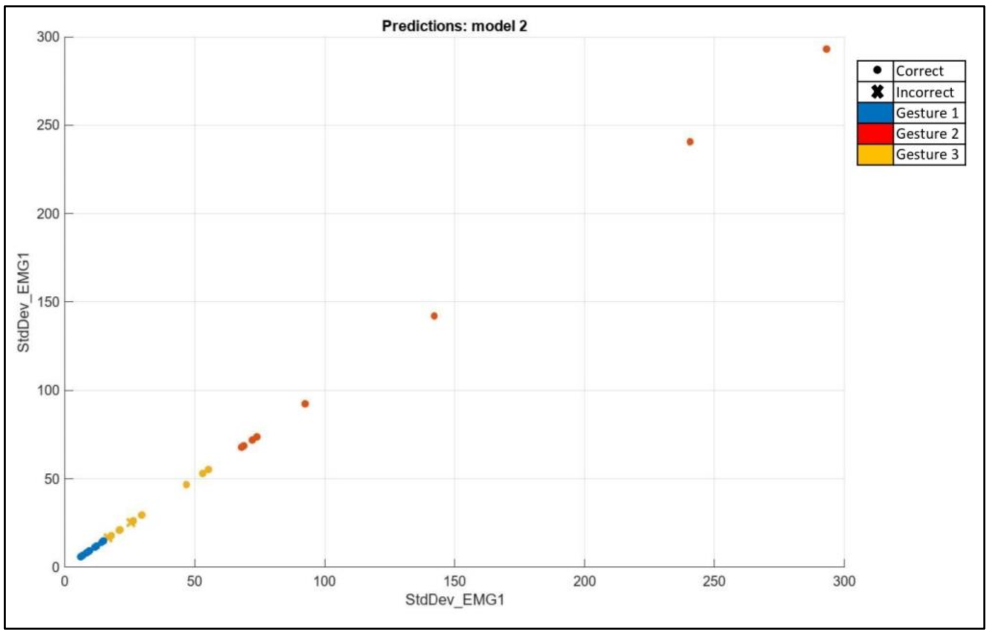

Figure 5.1 Scatter plot of predicted classification

The scatter plot in **Figure 5.1** demonstrates the successful classification of the selected distinctive gestures. However, when confronted with closely related gestures (as shown in **Figure 5.2**), the model struggled to provide accurate classifications.

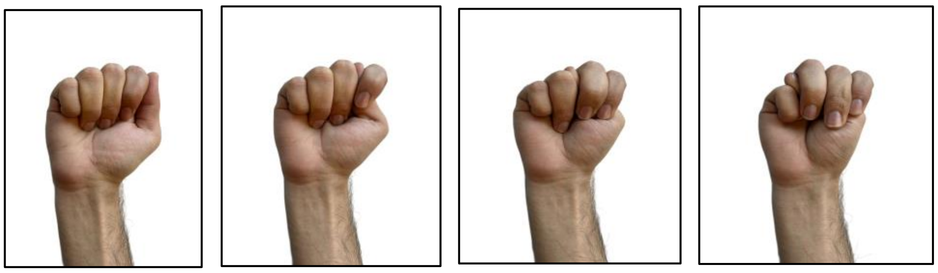

Figure 5.2 Closely related gestures

In conclusion, our model exhibits accurate classification of the three chosen distinctive gestures with a commendable accuracy of 93.1%. However, it falls short when confronted with closely related gestures. Future work could focus on improving sensor placement, increasing the number of sensors, and exploring combinations of different sensors to enhance the model's ability to accurately classify both distinctive and closely related gestures.

6 References

[1] A. G. J. J. Marco E. Benalcázar und A. P. V. H. A. A. Zea, „Hand gesture recognition using machine learning and the Myo armband," *25th European Signal Processing Conference (EUSIPCO)*, 2017.

[2] K. P. A. B. A. R. M. A. Afsaneh Kashizadeh, „Myoelectric Control of a Biomimetic Robotic Hand Using Deep Learning Artificial Neural Network for Gesture Classification," *IEEE Sensors Journal*, 2022.

[3] D. W. R. Z. Y. Y. Qian Zhang, „MyoSign: enabling end-to-end sign language recognition with wearables," *24th international conference on intelligent user interfaces*, 2019.

[4] C. E. A. J. A. Z. P. Z. A. G. J. M. S. Marco E. Benalcázar, „Real-time hand gesture recognition based on artificial feed-forward neural networks and EMG," *26th European Signal Processing Conference (EUSIPCO)*, 2018.

[5] O. A. K. H. F. a. B. H. Kerdjidj, „Implementing Hand Gesture Recognition using EMG on the Zynq Circuit," *IEEE Sensors Journal*, p. 8, 2023.

[6] T. P. A. a. S. K. Tyagi, „EMG-based Gesture Recognition using Extreme Learning Machine," *3rd International conference on Artificial Intelligence and Signal Processing (AISP) IEEE*, p. 6, 2023.

[7] S. M. V. H. M. H.-J. I. A.-K. S. a. M. R. Aarthy, „Recognition of Hand Gesture Using Electromyography Signal: Human-Robot Interaction," *Journal of Sensors*, 2022.

[8] H. R. V. J. V. M. P. Mahdi Abavisani, „Improving the performance of unimodal dynamic hand-gesture recognition with multimodal training," *IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[9] „Arduino nano 33 BLE," *Product reference manual SKU:ABX00030*, p. 12, 2023.

[10] „3-lead Muscle / Electromyography Sensor for Microcontroller Applications," *Myoware Muscle Sensor Datasheet*, p. 8.