الجمهورية الجزائرية الديمقراطية الشعبية
Peoples Democratic Republic of Algeria
وزارة التعليم العالي و البحث العلمي
Ministry of Higher Education and scientific research
المدرسة الوطنية العليا للتكنولوجيات المتقدمة
National High School of Advanced Technology
Department of Electrical Engineering and Industrial Computer Science

Final Year Project to Obtain the Diploma of Engineering

Field:

Electronics

Specialty:

Embedded Systems

Subject:

A Smart Glove Empowered by AI embedded on a Raspberry Pi

Realized by

Benbrahim Tarek

Examination Committee:

Chair	Zellat Khadidja	ENSTA
Supervisor	Bougherirah Hamida	USD Blida
Co-Supervisor	Habani Lamia	ENSTA
Examiner	Bouchama Samira	ENSTA
Examiner	Kheira Lakhdari	ENSTA

Algiers, The 26/06/2024

Academic year 2023 - 2024

Abstract

This thesis presents the development of a smart glove system designed to translate ASL gestures into text and speech. The system uses a combination of flex sensors for finger movement detection, a module that combines accelerometer and a gyroscope for hand gesture capture, and a microcontroller for data processing. Data from all the sensors modules is acquired using an esp32, while the Raspberry Pi 4 is used for executing a machine learning model based on Random Forest algorithms. This model is trained with a dataset to recognize ASL gestures and convert them into text which is then synthesized into speech using a TTS module. That glove aims to help the deaf community to interact easily with the other categories of the society.Initial testings showed successful results,future work will focus on improving the device's capabilities with other sign languages.

Keywords: machine learning ,microcontroller , Random forest , wireless communication , smart glove , sign language.

Résumé:

Cette thèse présente un gant intelligent traduisant les gestes ASL en texte et parole. Il utilise des capteurs de fléxions , un accéléromètre, un gyroscope pour détecter les mouvements, et un microcontrôleur pour le traitement des données. Un modèle d'apprentissage automatique sur Raspberry Pi 4 reconnaît et convertit les gestes en texte, puis en parole avec un module TTS. Le gant aide la communauté sourde à interagir plus facilement, avec des résultats positifs lors des premiers tests.

Mots-clés: apprentissage automatique, microcontrôleur, forêt aléatoire, communication sans fil, gant intelligent, langue des signes.

ملخص:

تقدم هذه الأطروحة تطوير نظام قفاز ذكي مصمم لترجمة لغة الإشارة الأمريكية إلى نص وكلام. يستخدم النظام مزيجًا من أجهزة الاستشعار المرنة للكشف عن حركة الأصابع، ووحدة تجمع بين مقياس التسارع والجيروسكوب لالتقاط إيماءات اليد، ووحدة تحكم دقيقة لمعالجة البيانات. يتم الحصول على البيانات من جميع وحدات الاستشعار باستخدام وحدة esp32 ، بينما يتم استخدام جهاز Raspberry Pi 4 لتنفيذ نموذج تعلّم آلي يعتمد على خوارزميات الغابة العشوائية. يتم تدريب هذا النموذج باستخدام مجموعة بيانات للتعرف على إيماءات لغة الإشارة الأمريكية وتحويلها إلى نص يتم توليفه بعد ذلك إلى كلام باستخدام وحدة تحويل النص إلى كلام. يهدف هذا القفاز إلى مساعدة مجتمع الصم على التفاعل بسهولة مع الفئات الأخرى من المجتمع، وقد أظهرت الاختبارات الأولية نتائج ناجحة، وسيركز العمل المستقبلي على تحسين قدرات الجهاز مع لغات الإشارة الأخرى.

الكلمات المفتاحية: تعلم الآلة، المتحكم الدقيق، خوارزميات الغابة العشوائية، الاتصالات اللاسلكية، القفاز الذكي، لغة الإشارة.

Acknowledgments

I would like to extend our heartfelt gratitude to the Almighty God, the All-Powerful and Merciful, who gave us the strength and dedication to complete this work.

*I deeply appreciate the guidance and dedication of my co-supervisor **Mme.Habani** and my supervisor **Mme.Bougherira**, who generously shared their expertise, time, and effort towards achieving this great work.*

Many thanks to the jury members for their interest in our work and their willingness to evaluate it.

I am also grateful to my family members, whose unwavering support and assistance have been instrumental in my journey towards the completion of this endeavor.

Dedication

These first words are dedicated to my beloved parents , my mother , my father. These words are full of deep feelings that i wanna write for you , you both gave me support , energy and the zeal to be what I am now.

My mother , your priceless contribution to me is what I can't forget , your words , your sacrifices to create the current version of me are unforgettable , May Allah bless you and keep you safe for our family , your absence is what i can't think about.

The father , the power of your words tell the amount of wisdom you have , your strength , your confidence taught me a lot to face this life tall and to never think about giving up. May Allah bless you too.

Other feelings expressed to my sisters , my little brother , your company is and still a priceless journey.

To my dear friends , getting to know you wasn't a coincidence , it was a lesson , a historical event that i can't lose off my memories , May Allah keep us together and forever

Table of contents

List of Figures

List of Tables

List of Abbreviations

1. ADC: Analog to Digital Converter
2. ANN: Artificial Neural Network
3. AUC: Area under the ROC Curve
4. ASL: American Sign Language
5. IDE: Integrated Development Environment
6. I2C: Inter-Integrated Circuit
7. OOB: Out Of Bag
8. PCB: Printed Circuit Board
9. RF: Random Forest
10. ROC: Receiver Operating Characteristic
11. SVM: Support Vector Machine
12. TN: True Negative
13. TP: True Positive
14. VNC: Virtual Network Computing

General Introduction

The ability to communicate effectively is fundamental to human interaction. However, for individuals who are deaf, spoken language and written text may not be accessible.ASL is a visual language used by many in the deaf community, but its understanding and interpretation by those who are unfamiliar with it can be a challenge. This issue shows the importance of finding new solutions.Motivated by this need, our project consists of the development of a smart glove system designed to translate ASL gestures into text and speech. The smart glove integrates a combination of flex sensors and accelerometers to detect and interpret hand movements, along with a microcontroller for data processing. A machine learning model based on Random Forest algorithms is trained with a dataset to recognize ASL gestures and convert them into text. The text is then synthesized into speech using a text-to-speech (TTS) module, providing a comprehensive communication solution.

The smart glove is designed to improve communication between deaf and hearing communities by providing a more accessible and inclusive way to communicate.

The thesis consists of three chapters. The first chapter covers the current state of the art, smart gloves development, proposed approach, required material, machine learning algorithms, and why random forest was chosen. It also mentions briefly the internship journey. The second chapter explains how the glove was embedded, including components wiring , machine learning algorithm details including its hyper-parameters, and dataset pre-processing. The last chapter discusses the tests and results of the model training.

Chapter 1

State of the Art

1.1 Introduction

This chapter provides an overview of the current state-of-the-art in smart glove technology for communication with deaf or hard-of-hearing individuals. The following sections will explore the technical details of existing systems, including hardware and software components

1.2 Literature Survey

.

The Table 1.1 below shows the existing literature on smart gloves and their applications in various fields.

Authors	System Description	Technology	Accuracy	Sign Language
Jagannath N., Manasa V., T. V. Prasad	Designed a smart glove using IoT technology for capturing and translating hand gestures [1] .	Flex sensors, accelerometer, gyroscope, Wi-Fi or GSM for data transmission, haptic feedback, speech output	85%	Indian Sign Language
Muhammad Rifki Kurniawan, Ahmad Hasibul Hadi, Siska Novitasari	Developed a smart glove to translate sign language into text or speech, processed by an Arduino board and transmitted via Bluetooth [2].	Flex sensors, accelerometer, gyroscope, Bluetooth	91.7%	Indonesian Sign Language
S. Sruthi, R. Sivaranjani, S. Sathya	Presented a smart glove using sensor fusion techniques to improve gesture recognition accuracy, including a speech recognition module [3].	Flex sensors, accelerometer, gyroscope, magnetometer, Arduino board, fusion algorithm	97.5%	Not specified
Neven Saleh, Mostafa Farghaly, Eslam Elshaaer	Developed a low-cost smart glove system to recognize hand gestures in Arabic sign language [4].	Flex sensors, triaxial MEMS module (MPU-6050), Android application	90%	Arabic Sign Language
Sawant Pramada, Archana Suhas Vaidya	iscussed developing an algorithm to accurately detect the number of fingers opened in a gesture representing an alphabet in Binary Sign Language [5].	Algorithm development for gesture recognition	Not specified	Binary Sign Language

Table 1.1: Literrature survey

One of The most advanced smart gloves in the world was created by Hadeel Ayoub, a 37-year-old Saudi Arabian woman. She is a London-based designer, programmer, and researcher in human-computer interaction. This glove, the first of its kind, assists deaf individuals without speech ability to communicate using any sign language they prefer. It offers the flexibility to select from 40 different spoken languages.[6].

Figure 1.1: Hadeel Ayoub

Figure 1.2: Brightsign smart glove

1.3 Machine learning

Our project is mainly based on machine learning with AI , hence let's have a brief talk about 3 algorithms : NN (Neural Network) , RF (Random Forest) and SVM (Support Vector Machine) , the three of them are related to supervised learning which is a type of machine learning where the model is trained on a labeled dataset[7]. In supervised learning, each training example in the dataset consists of inputs and a corresponding output. The goal of supervised learning is to learn a mapping from inputs to outputs, so that the model can make predictions or decisions when given new, unseen data. These details are illustrated in **Figure 1.3**

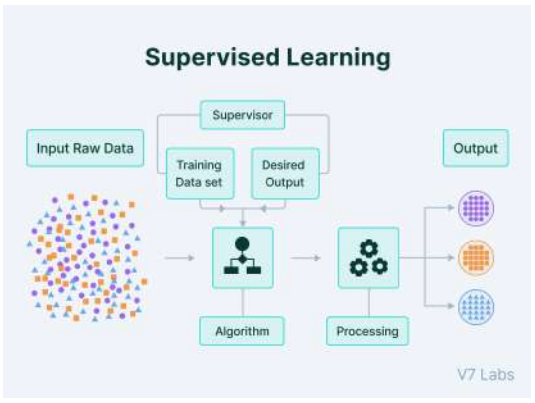
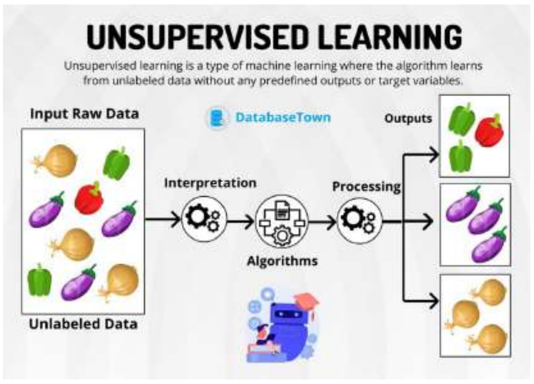


Figure 1.3: supervised learning


UNSUPERVISED LEARNING
Unsupervised learning is a type of machine learning where the algorithm learns from unlabeled data without any predefined outputs or target variables.
Input Raw Data
DatabaseTown
Outputs
Interpretation
Processing
Algorithms
Unlabeled Data
Figure 1.4: unsupervised learning

On the other side and as shown in **Figure 1.4** , we have unsupervised learning, which is also a type of machine learning that does not need any target labeled to be specified, it just regroups patterns, and all similar dataset, it is really helpful when dealing with quite large datasets.We can find many algorithms developed for this type of learning like K-means , DBSCAN and Hierechial clustering.[8].

1. ***ANN:*** A neural network is a type of machine learning model that emulates the decision-making process of the human brain. This is achieved by simulating the interconnected behavior of biological neurons, enabling them to recognize patterns, evaluate choices, and make decisions. Each neural network comprises layers of nodes, or artificial neurons, including an input layer, one or more hidden layers,(See **Figure 1.5**) and an output layer. Nodes are interconnected, each with its own weight and threshold. When the output of a node exceeds its threshold, it activates and transmits data to the next layer. Otherwise, the data is not forwarded to the subsequent layer.[9].

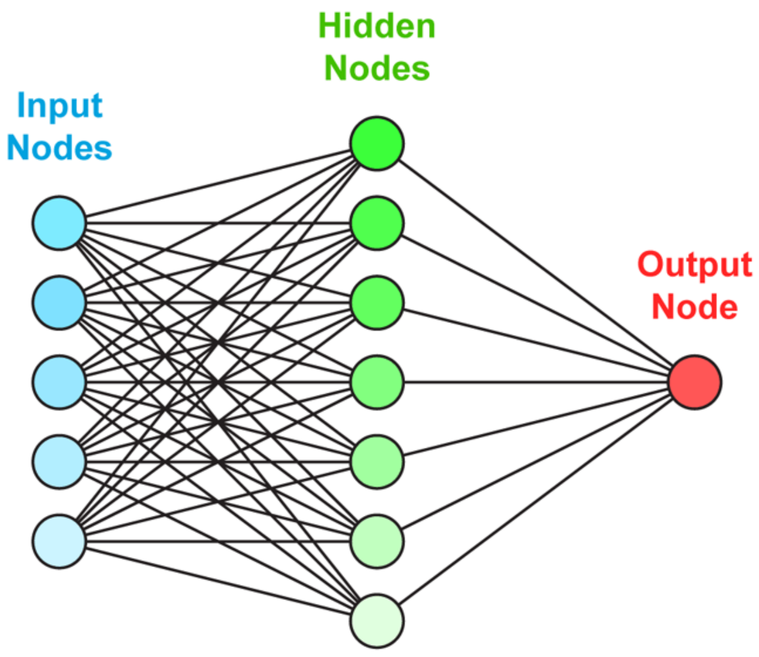


Figure 1.5: Neural Network

2. ***SVM (Support Vector Machine):*** The Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression tasks, although it is particularly well-suited for classification. Its primary goal is to identify the optimal hyperplane within an N-dimensional space that effectively separates data points into different classes in the feature space[10]. The algorithm

aims to maximize the margin between the closest points of different classes as see in **Figure 1.6**, with the dimension of the hyperplane determined by the number of features. For instance, in the case of two input features, the hyperplane is a line, while with three input features, it becomes a 2-D plane. However, visualizing the hyperplane becomes challenging as the number of features exceeds three.

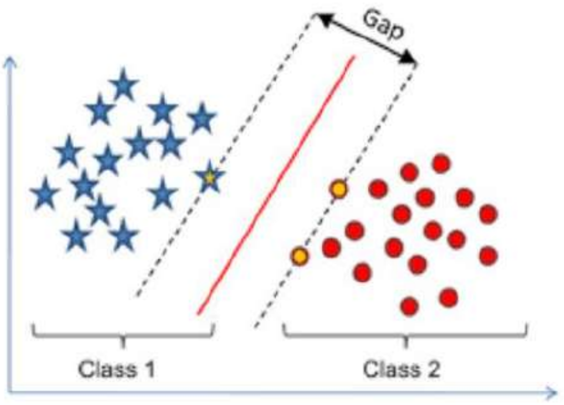


Figure 1.6: Support Vector Machine

3. ***Random Forests classification:*** Random Forest is an ensemble of Decision Trees whereby the final/leaf node will be either the majority class for classification problems or the average for regression problems. A random forest will grow many classification trees and for each output from that tree, we say the tree 'votes' for that class as shown in **Figure 1.7**.[11].

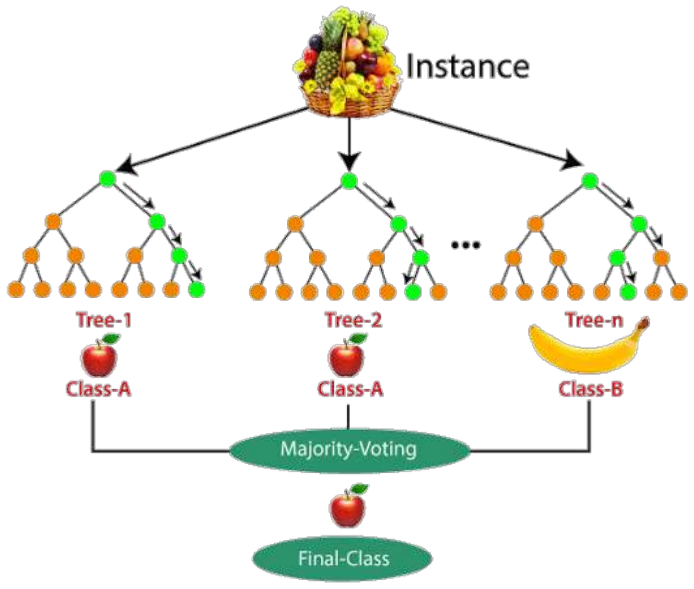


Figure 1.7: Random Forest

1.4 Embedded platforms

1.4.1 Propsed methodology

This study aims to develop a smart glove system that translates American Sign Language (ASL) gestures into text and speech. The system will utilize five flex sensors to detect finger movements and a module integrating an accelerometer and gyroscope (MPU6050 module) to capture hand gestures, all integrated into the glove. An ESP32 will be used for data acquisition from the flex sensors and MPU6050, transmitting the collected data to a Raspberry Pi 4 via Wi-Fi. The Raspberry Pi 4 will handle more advanced data processing tasks. An AI model based on Random Forest algorithms will be trained using the sensors data to recognize ASL gestures and convert them into text, which will then be synthesized into speech. The **Figure 1.8** and **Figure 1.9** show different American sign language expressions and alphabet.

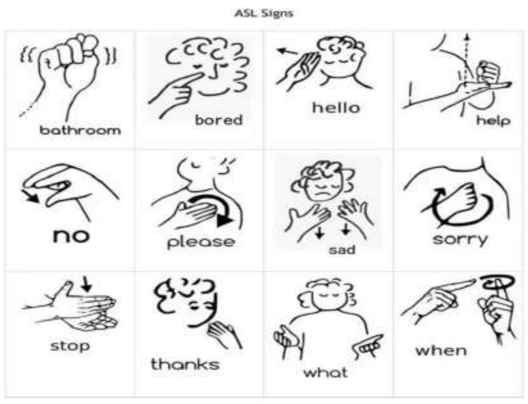


Figure 1.8: Sign language common words

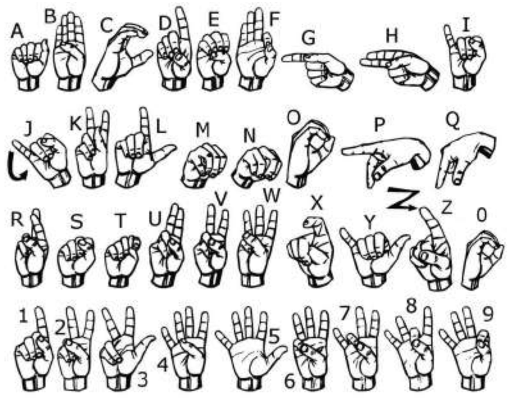


Figure 1.9: American sign language alphabet

1.4.2 Raspberry pi 4 over Arduino

According to literature survey, smart gloves have been developed using Raspberry Pi and Arduino. The focus of our approach is primarily on Raspberry Pi, specifically the Raspberry Pi 4 Model B. We chose it because it is well-suited for training machine learning models,Additionally, Raspberry Pi provides advantages in terms of processing power and connectivity when compared to Arduino. The table 1.2 provided below summarizes the key reasons for choosing Raspberry Pi over Arduino for our project.

Figure 1.10: Raspberry Pi & Arduino Uno

Feature	Raspberry Pi 4 Model B (4GB RAM)	Arduino
Processor	Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz	Varies by model (e.g., ATmega328P on Uno)
RAM	4GB LPDDR4	None (memory is onboard the microcontroller)
Storage	MicroSD slot for loading OS and data storage	No onboard storage, uses external EEPROM
Operating System	Runs full Linux distributions (e.g., Raspbian)	No OS, runs simple firmware
Connectivity	Gigabit Ethernet, Wi-Fi 802.11ac, Bluetooth 5.0	Limited (usually requires external modules)
USB Ports	2 x USB 3.0, 2 x USB 2.0	Typically none or limited
GPIO Pins	40-pin header	Varies by model (e.g., 14 digital, 6 analog on Uno)
Video Output	2 x micro-HDMI ports, supports dual 4K monitors	None
Power Supply	5V via USB-C or GPIO header	Varies (e.g., USB or barrel jack)
Development Environment	Supports various IDEs, programming languages	Arduino IDE, C/C++
Typical Use Cases	Full-fledged computer, media center, server, IoT	Embedded systems, simple robotics, IoT
Cost	Generally higher than Arduino	Generally lower than Raspberry Pi

Table 1.2: Raspberry pi Vs Arduino Uno

Now, let us see the list of materials needed to realize the project:

Raspberry pi 4 model B : The Raspberry Pi 4 is a popular single-board computer (SBC) developed by the Raspberry Pi Foundation as shown in **Figure 1.11**. It is the fourth generation of the Raspberry Pi series and offers improved performance and features compared to its predecessors[12]. The model B has:

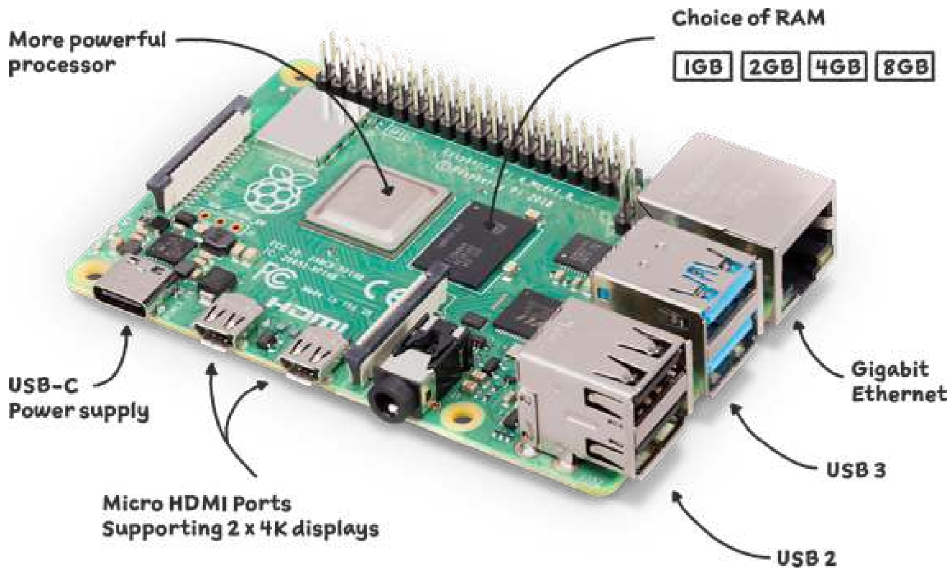


Figure 1.11: Raspberry pi 4 model B

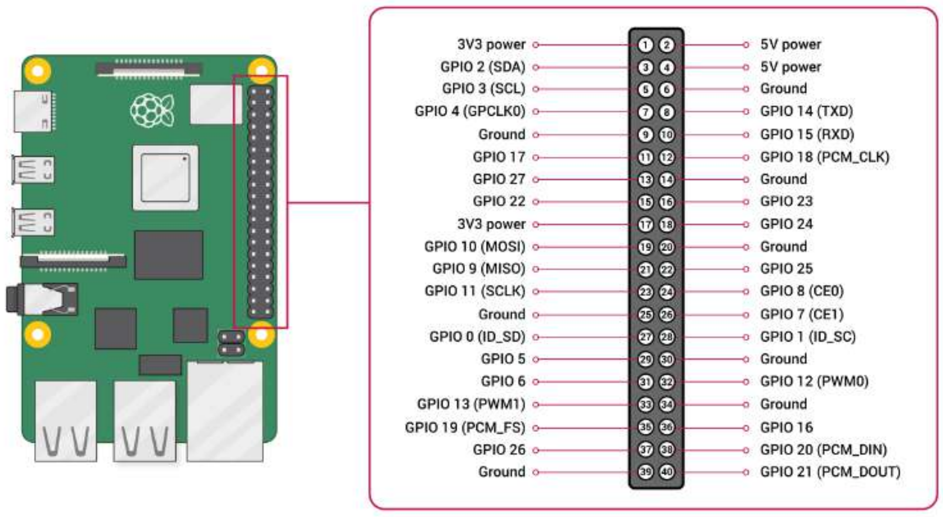

3V3 power 1 2 5V power
GPIO 2 (SDA) 3 4 5V power
GPIO 3 (SCL) 5 6 Ground
GPIO 4 (GPCLK0) 7 8 GPIO 14 (TXD)
Ground 9 10 GPIO 15 (RXD)
GPIO 17 11 12 GPIO 18 (PCM_CLK)
GPIO 27 13 14 Ground
GPIO 22 15 16 GPIO 23
3V3 power 17 18 GPIO 24
GPIO 10 (MOSI) 19 20 Ground
GPIO 9 (MISO) 21 22 GPIO 25
GPIO 11 (SCLK) 23 24 GPIO 8 (CE0)
Ground 25 26 GPIO 7 (CE1)
GPIO 0 (ID_SD) 27 28 GPIO 1 (ID_SC)
GPIO 5 29 30 Ground
GPIO 6 31 32 GPIO 12 (PWM0)
GPIO 13 (PWM1) 33 34 Ground
GPIO 19 (PCM_FS) 35 36 GPIO 16
GPIO 26 37 38 GPIO 20 (PCM_DIN)
Ground 39 40 GPIO 21 (PCM_DOUT)

Figure 1.12: Raspberry pi 4 model B Pinout

1. Processor: Quad-core ARM Cortex-A72 (64-bit) processor running at 1.5GHz.

2. Memory: Options for 2GB, 4GB, or 8GB of LPDDR4 RAM.

3. Connectivity: Dual-band 802.11ac wireless networking (Wi-Fi), Bluetooth 5.0, Gigabit Ethernet, and USB 3.0 ports.
4. Video Output: Dual micro-HDMI ports supporting up to 4K resolution.
5. GPIO: 40-pin GPIO header for connecting external devices and components.
6. Storage: MicroSD card slot for operating system and data storage.
7. Power: USB-C power port for powering the device.

Esp32 Wroom S3 : The **Figure 1.13** shows an ESP32 which is a microcontroller manufactured by Espressif Systems, featuring the ESP32 chip with integrated WiFi and Bluetooth. The microcontroller boasts a dual-core processor utilizing the Xtensa LX7 architecture, operating at a clock speed of 240 MHz and equipped with 4Mb of memory[13]. Its sixteen 12 bits adc channel and ability to communicate with other boards wirelessly make it the better choice for our project as the board has:

1. Processor: Dual-core Xtensa® LX7 microcontroller, operating at up to 240 MHz.
2. RAM: 512 KB internal SRAM.
3. ROM: 384 KB of ROM for bootloader, security functions, and library functions.
4. Flash: Typically 4 MB (external), used for program storage.
5. Cache: 16 KB of instruction cache and 8 KB of data cache.

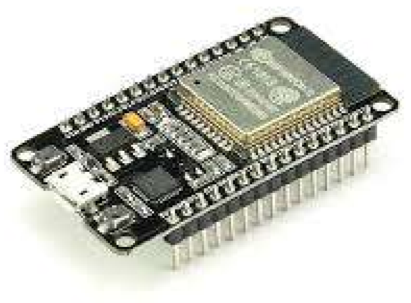


Figure 1.13: ESP32 Wroom S3

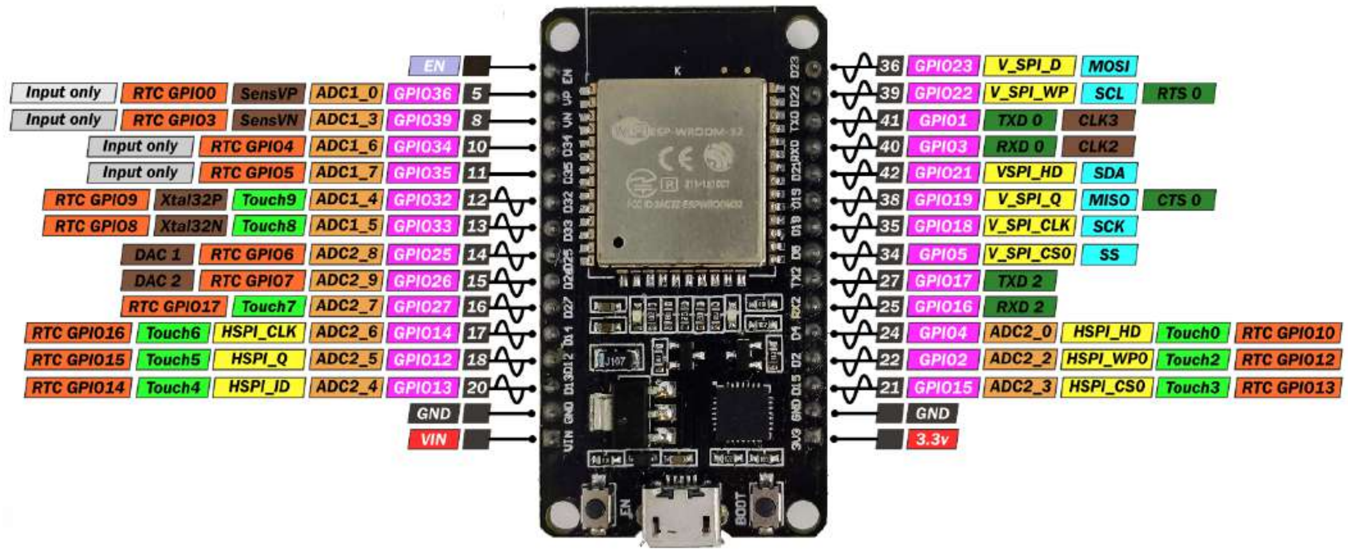

EN
Input only RTC GPIO0 SensVP ADC1_0 GPIO36 5
Input only RTC GPIO3 SensVN ADC1_3 GPIO39 8
Input only RTC GPIO4 ADC1_6 GPIO34 10
Input only RTC GPIO5 ADC1_7 GPIO35 11
RTC GPIO9 Xtal32P Touch9 ADC1_4 GPIO32 12
RTC GPIO8 Xtal32N Touch8 ADC1_5 GPIO33 13
DAC 1 RTC GPIO6 ADC2_8 GPIO25 14
DAC 2 RTC GPIO7 ADC2_9 GPIO26 15
RTC GPIO17 Touch7 ADC2_7 GPIO27 16
RTC GPIO16 Touch6 HSPI_CLK ADC2_6 GPIO14 17
RTC GPIO15 Touch5 HSPI_Q ADC2_5 GPIO12 18
RTC GPIO14 Touch4 HSPI_ID ADC2_4 GPIO13 20
GND
VIN
36 GPIO23 V_SPI_D MOSI
39 GPIO22 V_SPI_WP SCL RTS 0
41 GPIO1 TXD 0 CLK3
40 GPIO3 RXD 0 CLK2
42 GPIO21 VSPI_HD SDA
38 GPIO19 V_SPI_Q MISO CTS 0
35 GPIO18 V_SPI_CLK SCK
34 GPIO5 V_SPI_CS0 SS
27 GPIO17 TXD 2
25 GPIO16 RXD 2
24 GPIO4 ADC2_0 HSPI_HD Touch0 RTC GPIO10
22 GPIO2 ADC2_2 HSPI_WP0 Touch2 RTC GPIO12
21 GPIO15 ADC2_3 HSPI_CS0 Touch3 RTC GPIO13
GND
3.3v
ESP-WROOM-32
EN
BOOT

Figure 1.14: Esp32 Wroom S3 pinout

flex sensor 2.2 sen-10264: Flex sensor 2.2" (SEN-10264) is a type of bend sensor that changes its resistance (25K Ohms ±30%)based on the degree of bending. It is 2.2 inches (56mm) in length and is typically used to detect the bending of fingers or other objects.[14].

Figure 1.15: flex sensor 2.2 inch

MPU6050 Module: the MPU6050 sensor module is a complete 6-axis motion tracking device. It combines 3-axis Gyroscope, 3-axis accelerometer and Digital Motion Processor all in small package. Also, it has additional feature of an on-chip Temperature sensor. It has I2C bus interface to communicate with the microcontrollers.[15]. This module is used to detect hand gesture movement and orientation.

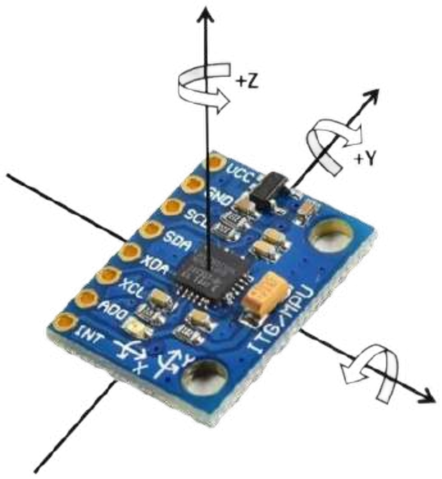


Figure 1.16: MPU6050 (accelerometer & gyroscope)

As mentioned in section 1.4.1 that in our thesis , Random Forest algorithms were chosen and that's for the following reasons:

1. ***Over-fitting :*** comparing to NN and SVM , Random forests deals with the problem of overfitting by creating multiple trees, with each tree trained slightly differently so it overfits differently.This makes random forest very robust to overfitting and able to handle complex relationships between the features and the target variable.[16].

2. ***High and stable accuracy:*** Random Forest's high accuracy stems from its ability to handle both classification and regression tasks effectively. It can handle large datasets with high dimensionality and is less prone to overfitting compared to individual decision trees.[17].

3. ***Less training time :*** Random Forests can be parallelized, allowing the individual decision trees to be trained simultaneously on different subsets of the data. This parallelization can significantly reduce the overall training time, especially on multi-core or distributed computing systems.[18].

4. ***Ability to estimate missing data:*** Random Forests have the ability to estimate missing data, making them a useful choice for datasets with incomplete information. When training a Random Forest model, missing values in the dataset do not need to be imputed or filled in beforehand, as the algorithm can handle missing data internally during the training process.[19].

Random Forest algorithms often provide excellent results when working with tabular datasets. Tabular datasets are structured data organized into rows and columns, where each column represents a feature or attribute, and each row represents an individual data point or observation[20].They are well-suited for tabular data because they can handle a mix of categorical and numerical features which is the case of the used dataset.

1.5 Internship experience at STREAM

In addition to the theoretical exploration , the internship has been for a month in STREAM which is an algerian company that manufactures electronic devices such as smart phones and televisions , it has the leadership in the country with exportation to different european countries including Germany and France. I have been in Bomare company which is a affiliate. There are three departments :

Figure 1.17: STREAM System

BOMARE COMPANY
SEOWOO
Go Digital Go Interactive
من أجل السلام والتطوير بين بلدينا وشركتينا
[illegible]
For the peace and the development between our two countries and companies
2011.7.29

Figure 1.18: Algeria-Korea partnership certificate

1.5.1 Assembling Unit

in this unit , the product is ready , it just needs to be covered up and sold to the clients.

1.5.2 PCB Production unit

the PCBs are imported from China , all they do is soldering electronic components into the board with specific machines and that includes soldering , and then physical inspection of the produced boards.

1.5.3 Repairing Unit

they do repair all clients electronic devices , there is also another department that takes in charge the industrial project development

Figure 1.19: PCB Production uni

Figure 1.20: PCB Production uni

Figure 1.21: PCB Production uni

Figure 1.22: PCB Production uni

1.6 Conclusion

In conclusion, the first chapter of our thesis has provided a comprehensive overview of our research topic, focusing on the development of a smart glove for American Sign Language (ASL) translation.Additionally, we talked briefly about the proposed methodology, including the algorithms and components that will be used. The next chapter would give bright details concerning the methodology and and sign language to speech synthesizer design.

Chapter 2

Embedded sign language to speech synthesizer design

2.1 Introduction

In this chapter, we will provide further details about the approach used in developing this project, including specifics about the software and hardware used.

2.2 System block diagram and flowchart

The **Figure 2.1** shows a "Smart Glove System" created to convert sign language into text and speech, improving communication for people with hearing impairments. The glove has five flex sensors to detect finger movements, and an MPU6050 sensor records the hand's position and movements. These sensors transfer information to the ESP32 Wroom S3 microcontroller and then , the microcontroller processes the data and sends it wirelessly to a Raspberry Pi 4 Model B that contains a machine learning model which interprets the sign language gestures , the system changes the sign language into text and then into speech, which are both displayed and heard audibly. The **Figure 2.2** represents the functional flowchart of the system.

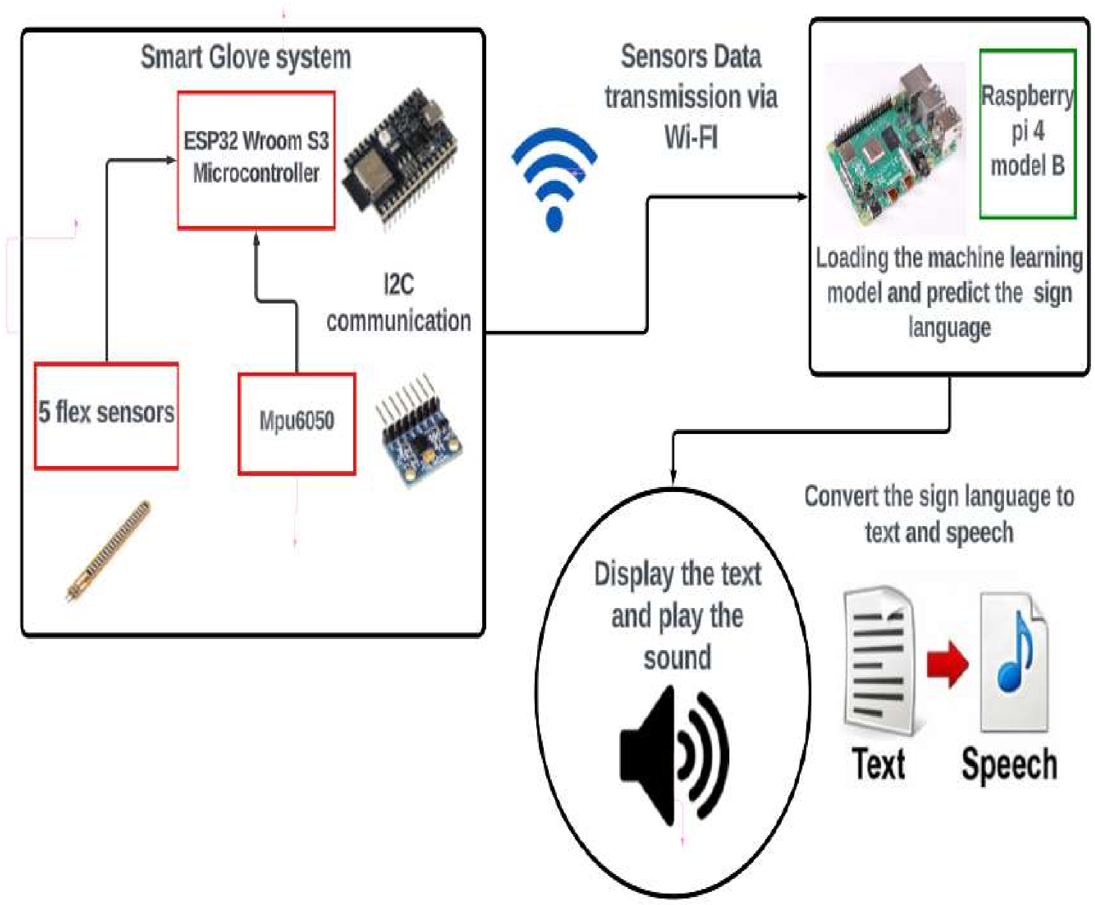

Smart Glove system
ESP32 Wroom S3 Microcontroller
I2C communication
5 flex sensors
Mpu6050
Sensors Data transmission via Wi-FI
Raspberry pi 4 model B
Loading the machine learning model and predict the sign language
Convert the sign language to text and speech
Display the text and play the sound
Text
Speech

Figure 2.1: System block diagram

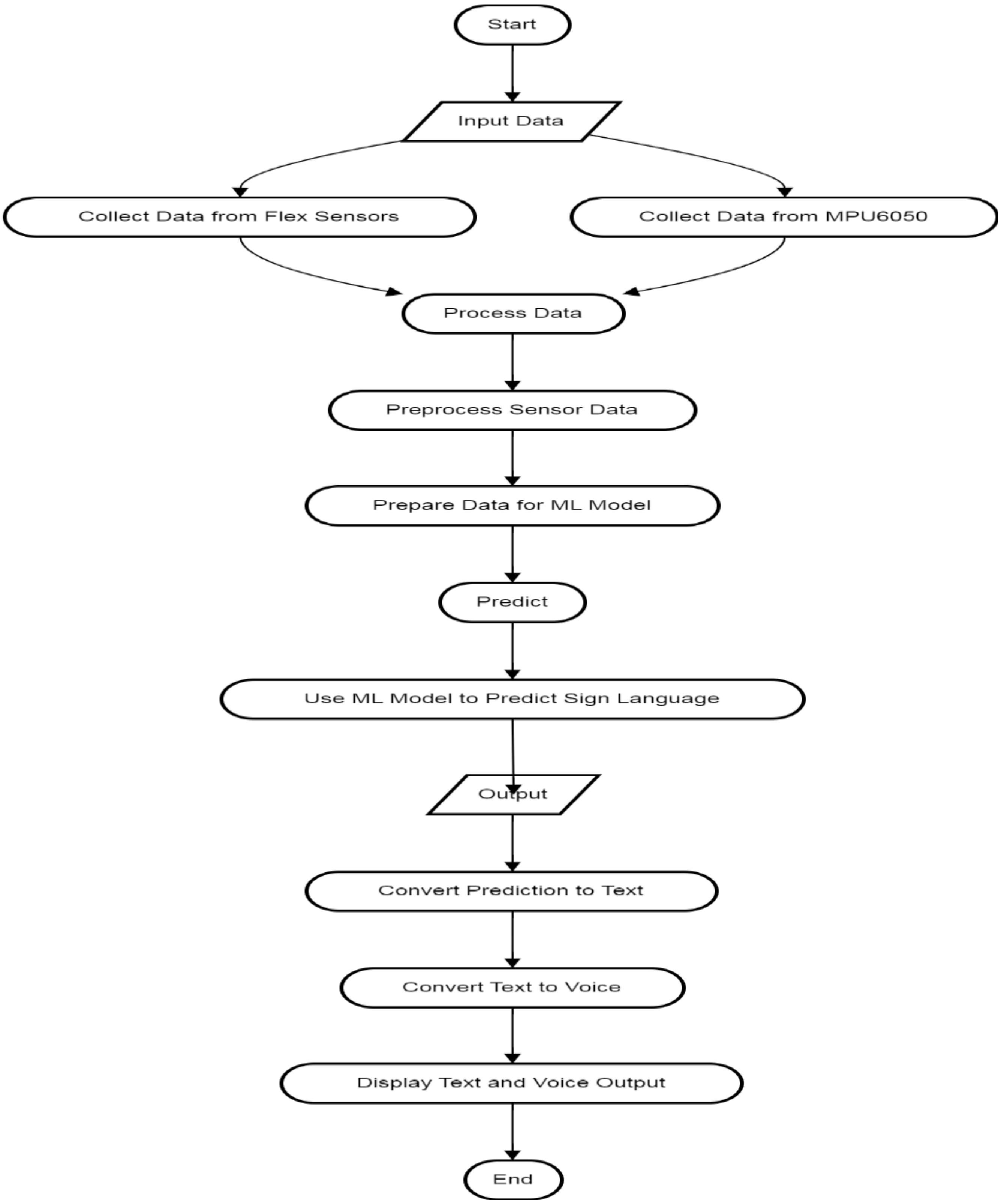

Start
Input Data
Collect Data from Flex Sensors
Collect Data from MPU6050
Process Data
Preprocess Sensor Data
Prepare Data for ML Model
Predict
Use ML Model to Predict Sign Language
Output
Convert Prediction to Text
Convert Text to Voice
Display Text and Voice Output
End

Figure 2.2: Functional flowchart

2.3 Hardware assembling

Our project is based on flex sensors and the MPU6050 module. The first ones are connected to the esp32's ADC channels since the board has 16 ADC channels. This setup was opted for because the Raspberry Pi 4 lacks ADC channels, whereas the MPU6050 is connected to the I2C pins of the microcontroller.

2.3.1 Flex sensors

we talked briefly about the flex sensors that they are variable resistors that change when bending , the value of its resistance when it's flat (0°) is approximately 25KΩ and about 100kΩ when it's 90°.[21].

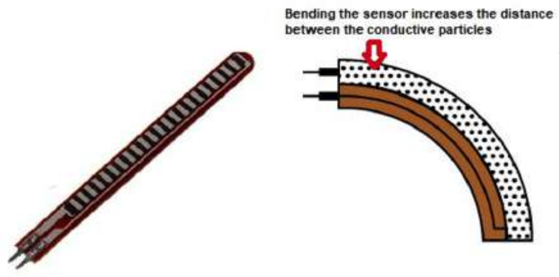

Bending the sensor increases the distance between the conductive particles

Figure 2.3: Flex sensor functionality

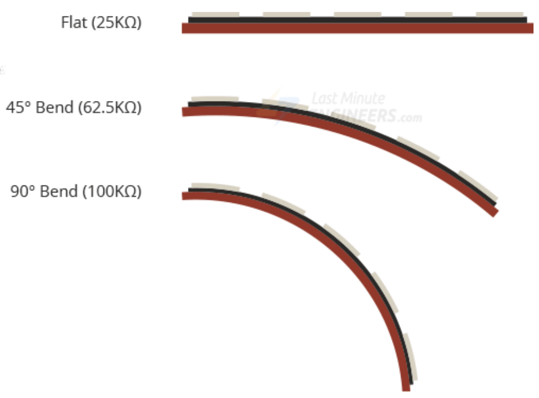


Figure 2.4: Different bending angles of flex sensor

Flex sensors Wiring to ESP32 : The flex sensor has two inputs named p1 and p2, where p2 is connected to 5v and p1 is connected to a resistor that is then connected to ground (pull down resistor). The resistor is added to create a voltage divider, allowing the voltage to change based on the flex bending. Another pin between the resistor and p2 is connected to one of the Arduino's ADC channels so it would be possible to read the analog value. In our case , we use five flex sensors , and the value of the resistors are all the same (10KΩ) Theoretically, the voltage on the resistor can be calculated using the voltage divider theorem. Considering the added resistor as R2, the flex sensor as R1, and the target voltage as V:

$$V = \left(\frac{R1}{R2 + R1}\right) \times 5 \tag{2.1}$$

when the sensor is flat : R1 = 10KΩ , R2 = 32,5KΩ

$$V = \left(\frac{R1}{R2 + R1}\right) \times 5 = 1.17V \tag{2.2}$$

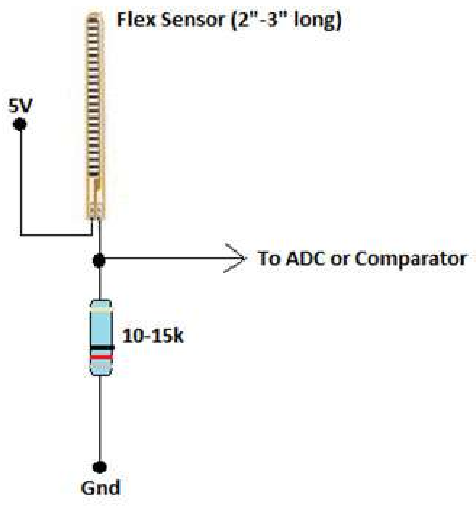


Figure 2.5: Flex sensor wiring system

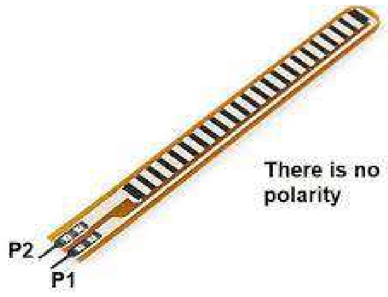


Figure 2.6: Flex sensor's Polarities

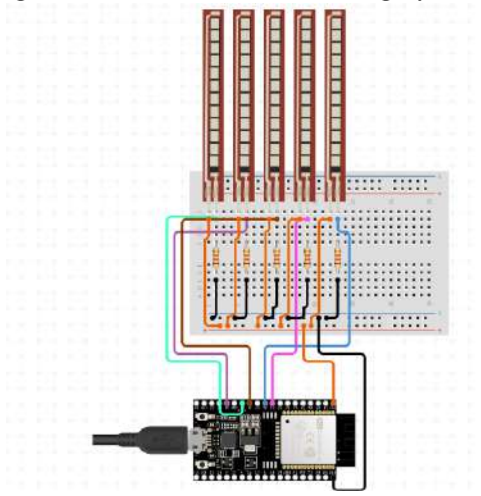


Figure 2.7: Wiring Diagram of Flex Sensors to ESP32

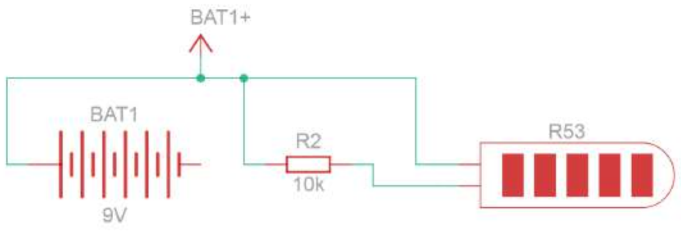


Figure 2.8: Flex sensors wiring schematic

2.3.2 *MPU6050*

The MPU6050 module simultaneously calculates acceleration and rotation for all three axes (x, y, and z) with a three-axis accelerometer and a three-axis gyroscope. Acceleration is determined by sensing the deflection of a small mass within the sensor when subjected to acceleration, utilizing capacitance variation principles. Electrodes inside the sensor measure capacitance changes due to mass deflection, enabling the sensor to ascertain acceleration along each axis.[22].

Regarding rotation, the gyroscope operates based on the Coriolis effect. A small vibrating element inside the gyroscope deflects perpendicular to the rotation direction due to the Coriolis force when the sensor rotates. This deflection is directly linked to the angular rate of rotation. By measuring the vibrating element's deflection, the sensor can determine the rate of rotation around each axis.[22].

The reason for choosing MPU6050 is due to the sign language expressions and the two alphabet(J & Z) that require both movement and rotation of the hand.

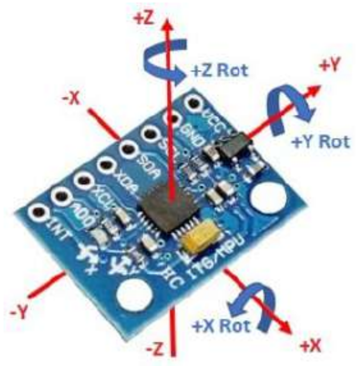


Figure 2.9: MPU6050 Axis

MPU6060 wiring to ESP32 : The module is connected directly to the esp32 via I2C communication , the two pins SDA and SCL are connected to the I2C pins (GPIO21 for SDA and GPIO22 for SCL).

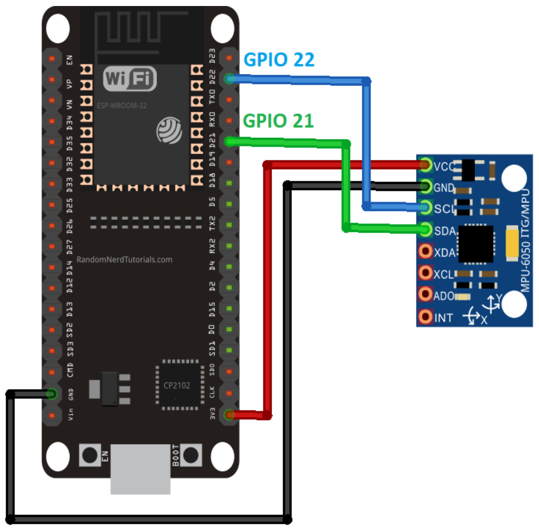


Figure 2.10: Wiring Diagram of MPU6050 to ESP32

2.4 Esp32 wireless communication with raspberry pi

The esp32 sends the sensors data to the raspberry pi 4 via Wi-Fi with TCP/IP protocol.The Raspberry Pi 4 running a TCP/IP server listens for incoming sensor data packets from the ESP32 over the Wi-Fi network[23]. Once the data is received, the Raspberry Pi processes and analyzes it.

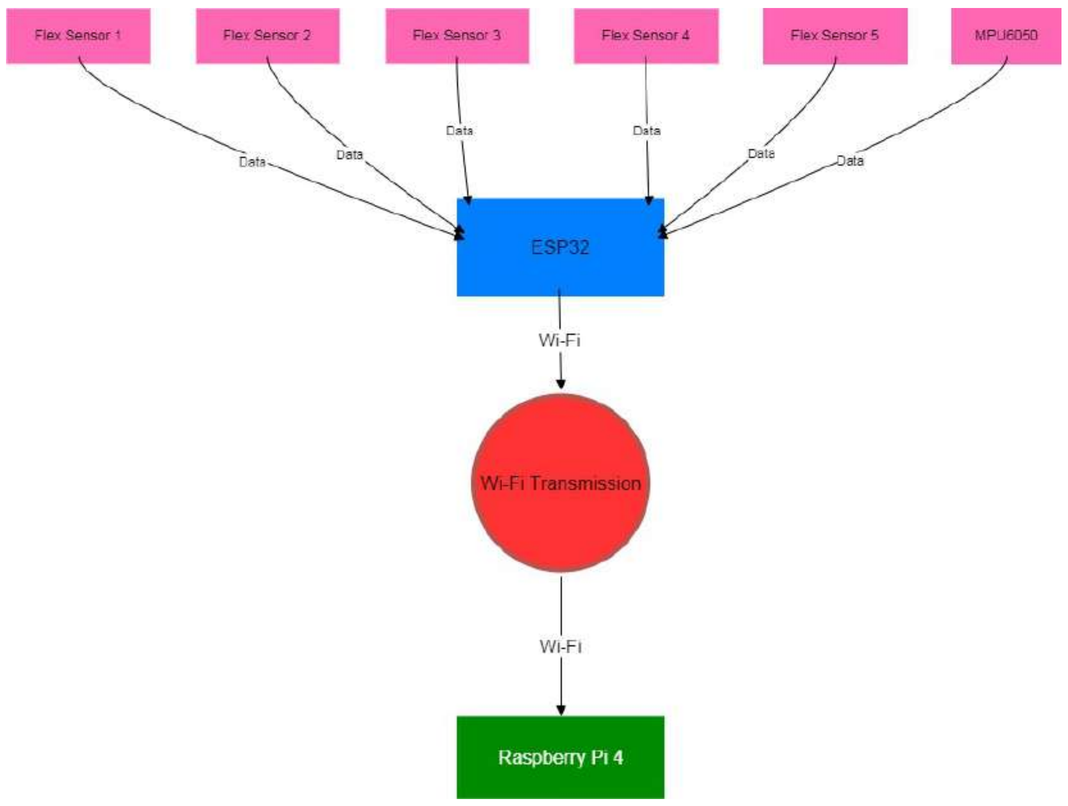


Figure 2.11: Esp32-Raspberry pi 4 wireless communication diagram

2.5 Random forest classifier

As already mentioned that the data processing in the project is based on random forest algorithms , there are two types : the classifier and the regressor , in our case , we intend to work with the classifier since we want to predict an alphabet on the output as each alphabet or expression represents a class.

2.5.1 Description of the classifier

Given a dataset with N samples , random forest creates many subsets called bootstrap samples ,trains a decision tree on each bootstrap sample , after growing trees and for classification, each tree ”votes” for a class, and the class with the most votes is predicted by the random forest algorithm. the final step is aggregation which is the process of combining multiple individual predictions or results into a single, final prediction or result.[18]. The description is more illustrated in the **Figures 2.12 , 2.13 , 2.14 , 2.15**

This classifier has different parameters that have a crucial imapct on the model's performaces:

1. Number of estimators : the estimators are simply the trees , many trees make a forest , the number of trees can play a huge impact , the more they are , the good the model is.[24].

2. max depth: this parameter represents the maximum height the trees can reach within the forest,[24].

3. minimum samples split: it controls the number of samples needed to split internal tree nodes by limiting its growth to avoid overfitting.[24].

4. minimum samples leaf:this is the minimum number of samples that a node must hold after getting split.[24].

5. maxleaf nodes: it sets the maximum number of leaf nodes in the tree , this also can help preventing overfitting as the model will stop growing nodes whenever it reaches the maximum number of leaf nodes.[24].

6. Random state : The random state is used to shuffle the data before splitting it into a training set.[24].

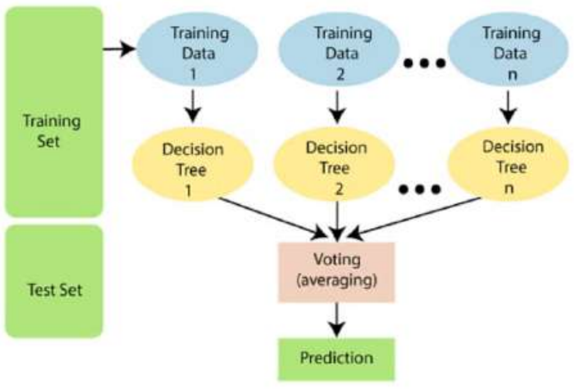


Figure 2.12: Voting Mechanism in Ensemble Learning for Prediction

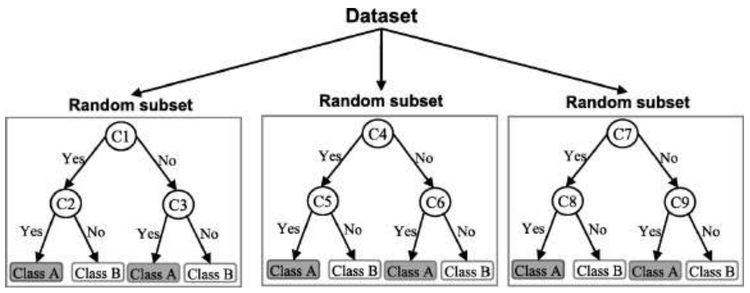
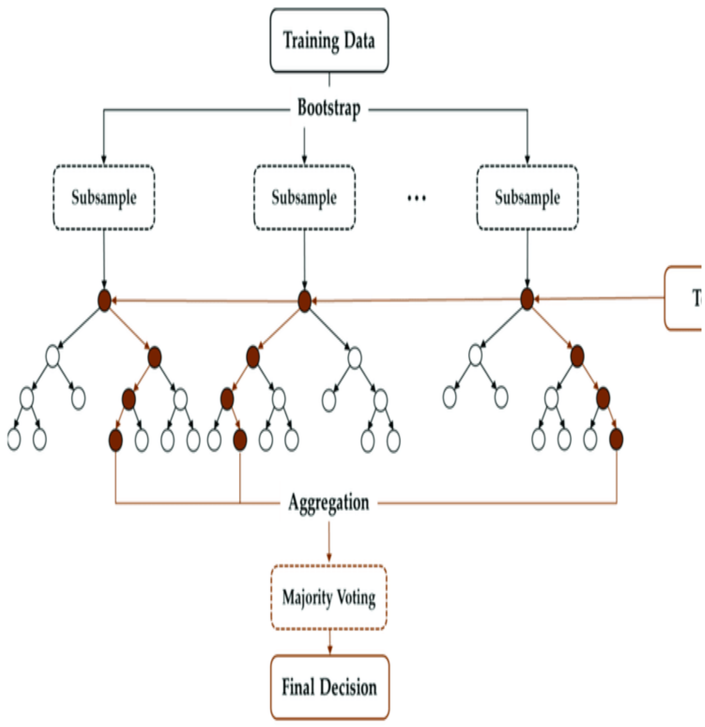
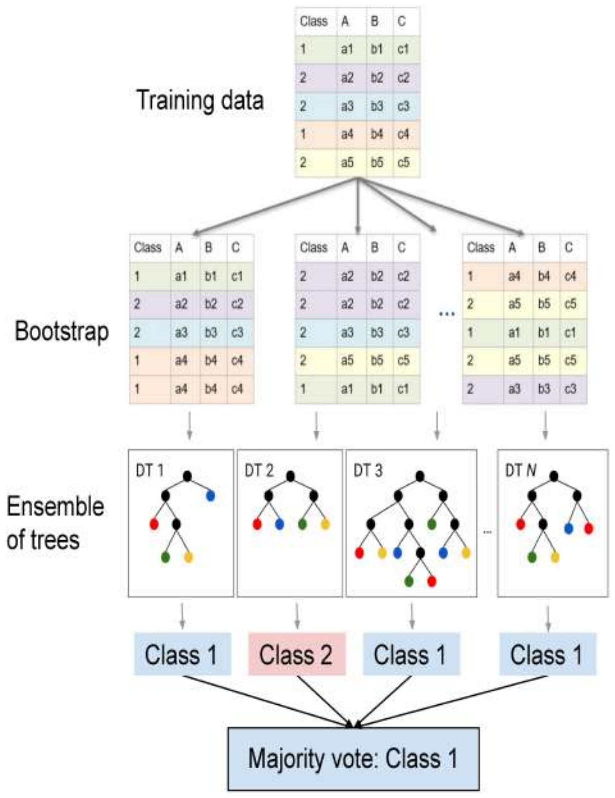


Figure 2.13: Random Subset Selection and Classification with Decision Trees





Class	A	B	C
1	a1	b1	c1
2	a2	b2	c2
2	a3	b3	c3
1	a4	b4	c4
2	a5	b5	c5

Class	A	B	C
1	a1	b1	c1
2	a2	b2	c2
2	a3	b3	c3
1	a4	b4	c4
1	a4	b4	c4

Class	A	B	C
2	a2	b2	c2
2	a2	b2	c2
2	a3	b3	c3
2	a5	b5	c5
1	a1	b1	c1

Class	A	B	C
1	a4	b4	c4
2	a5	b5	c5
1	a1	b1	c1
2	a5	b5	c5
2	a3	b3	c3

Figure 2.14: Bootstrapping and Aggregation in Random Forests

Figure 2.15: Random forests data bootstrapping

2.6 Validation and evaluation methods

2.6.1 OOB score

Out of bag is a method of validating a random forests model's prediction error as it consists of the dataset entries that were neglected and not trained by the machine learning algorithm. This method provides a good idea about the model's performances on unseen data.[25].

$$\text{OOB Score} = \frac{1}{n}\sum_{i=1}^{n} I(y_i = \hat{y}_i^{\text{OOB}}) \tag{2.3}$$

2.6.2 Accuracy

The accuracy of a machine learning model represents the number of correct predictions among the total of them.[26].

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{2.4}$$

2.6.3 Precision

Tt's the number of true positive predictions among the the number of correct ones , it becomes very crucial if the model is trained to predict both negative and positive values.[27].

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.5}$$

2.6.4 ROC-AUC

Tt gives information about how the model can distinguish between classes.The ROC-AUC score ranges from 0 to 1.[28].

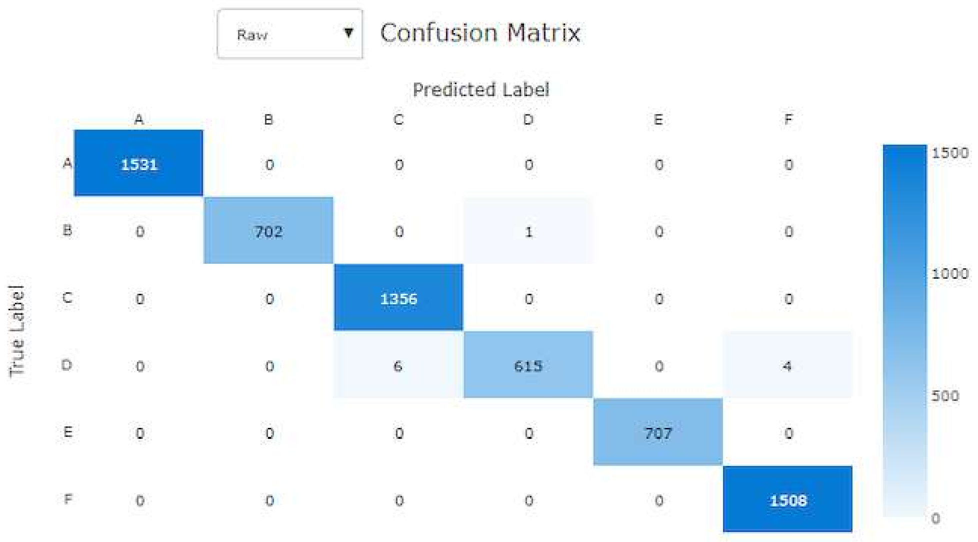
$$\text{ROC-AUC} = \int_0^1 \text{TPR}(FPR)\, d(\text{FPR}) \tag{2.6}$$

2.6.5 Features importance

The importance of features gives us a better understanding of the most influential features that were used during the training of the model.[11].

2.6.6 Confusion matrix

it's a table that shows both true and predicted different classes, it can help visualize the model performances and how each classes is getting trained[29]. , here is an example :



True Label \ Predicted Label	A	B	C	D	E	F
A	1531	0	0	0	0	0
B	0	702	0	1	0	0
C	0	0	1356	0	0	0
D	0	0	6	615	0	4
E	0	0	0	0	707	0
F	0	0	0	0	0	1508

Figure 2.16: Enter Caption

2.6.7 F1-score

The F1 score can be interpreted as a harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal[30]. The formula for the F1 score is:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.7}$$

2.6.8 Correlation matrix

The correlation matrix is a matrix that shows the correlation between variables. It gives the correlation between all the possible pairs of values in a matrix format.[31]. The correlation coefficient is denoted by “r”, and it ranges from -1 to 1.

1. If r = -1, it means that there is a perfect negative correlation.
2. If r = 0, it means that there is no correlation between the two variables.

3. If r = 1, it means that there is a perfect positive correlation.[32].

2.7 Training dataset

2.7.1 Description

The dataset used in this project isn't ours, but we could find one that contains 40 ASL including 26 alphabet and 14 expressions collected from 24 volunteers as each one of them trained the 40 ASL[33] , 1500 entries for each sign language. Here's an sample from the dataset of the letter A as seen on the Table 2.1

timestamp	user_id	flex_1	flex_2	flex_3	flex_4	flex_5	Q_w	Q_x	Q_y	Q_z	GYR_x	GYR_y	GYR_z	ACC_x	ACC_y	ACC_z	$ACCx\_body$	$ACCy\_body$	$ACCz\_body$
1616133208.5124247	001	20.0	58.0	72.0	77.0	58.0	0.773499	0.087097	-0.627747	0.002014	-0.068702	-0.015267	0.015267	9.314306	1.379321	1.770508	-0.205762	0.082544	-0.155518
1616133208.5224576	001	20.0	58.0	70.0	76.0	58.0	0.773438	0.08606	-0.627991	0.001587	-0.076336	-0.015267	0.022901	9.29038	1.372144	1.762134	-0.23208	0.086133	-0.162695
1616133208.5324895	001	20.0	58.0	73.0	75.0	60.0	0.773376	0.0849	-0.628235	0.001099	-0.076336	-0.022901	0.022901	9.301147	1.34104	1.754956	-0.22251	0.066992	-0.16748
1616133208.5422645	001	19.0	58.0	73.0	79.0	58.0	0.773254	0.083679	-0.62854	0.000732	-0.076336	-0.022901	0.022901	9.331055	1.271655	1.739404	-0.194995	0.011963	-0.179443

Table 2.1: Dataset sample

2.7.2 Dataset pre-processing

This dataset sample has different "features". In terms of important ones , there are a total of 11: five from the flex sensors data and six derived from the MPU6050 (3-axis accelerometer, 3-axis gyroscope). for the flex sensors , the data are the angles of bending not the analog values as a consequence , these analog values need to be converted to angles. Our aim is to train the model to predict all 26 alphabets and 14 expressions: you, good, no, yes, hungry, deaf, please, sorry, thank you, hello, me, bad, fine, goodbye , but before adding a step to the training process ,The dataset should be cleaned as it may contain noise or missing data, which includes:

1. Deleting duplicates.

2. Filling the missing values.

3. Eliminating the unnecessary features.

2.8 Conclusion

In this chapter, we outlined the methodology used for creating the smart glove for sign language recognition, covering both software and hardware. The following chapter will delve into the tests and results.

Chapter 3

Results and implementation of the smart glove

3.1 Introduction

In this last chapter, we highlight the final steps that led us to implment the smart glove design in reality and elaborate the crucuial resutls we reached.

3.2 Working environment

Some software were used to assist in completing the project. The software used include:

1. Google Colab : it's a hosted notebook that gives the access to computing ressources[34] , it's used for training machine and deep learning models.

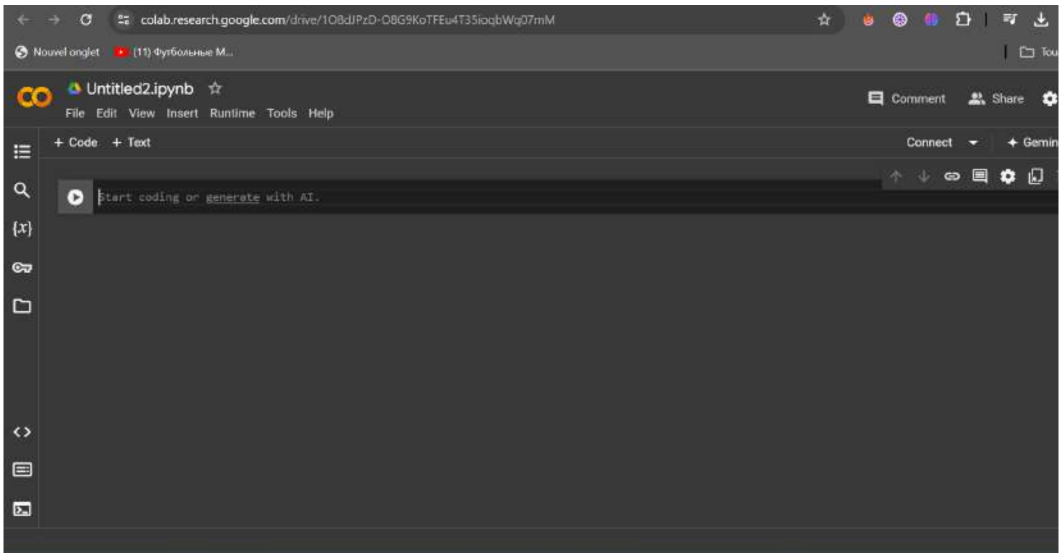

colab.research.google.com/drive/1O8dJPzD-O8G9KoTFEu4T35ioqbWq07mM
Nouvel onglet
(11) Футбольные М...
Untitled2.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
Connect
Gemini
Start coding or generate with AI.

Figure 3.1: Google Colab

2. EasyEda: it's a software for designing and simulating pcb circuits , it has some interesting features including downloading components libraries easily by signing up , and there are a lot of contributors to this platform. We used it to design the PCB.

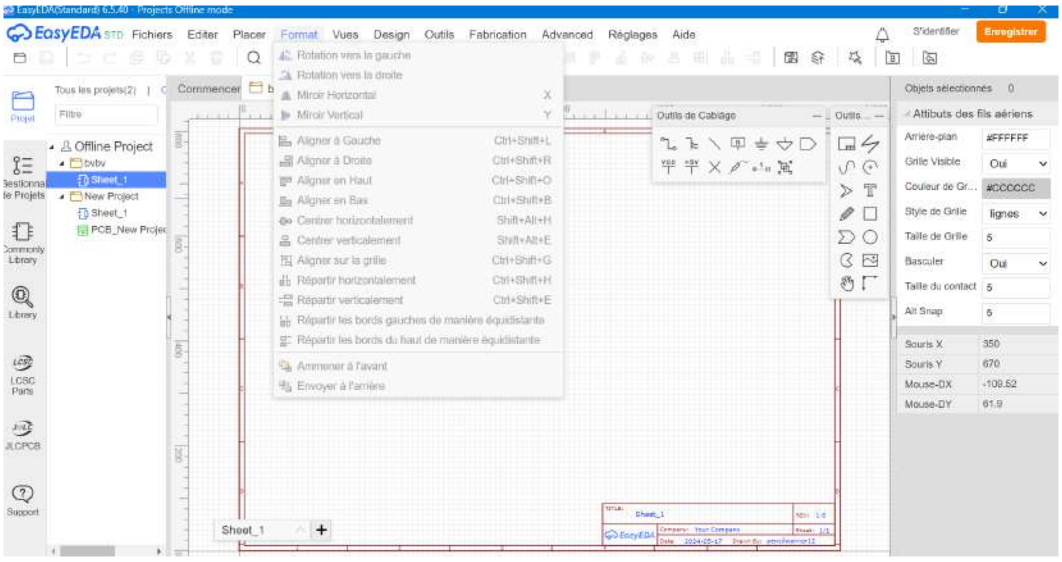


Figure 3.2: EasyEda

3. Raspian OS : the Raspian os is debian-based linux distribution that runs on raspberry pi 4 , it's crucial as it will run the machine model with all installed libraries in.

   Setting up the raspberry pi 4: An 8GB or larger SD card and a software tool called Raspberry Imager are required.[35].

   (a) Download raspbian OS iso file and raspberry pi imager from the official website.

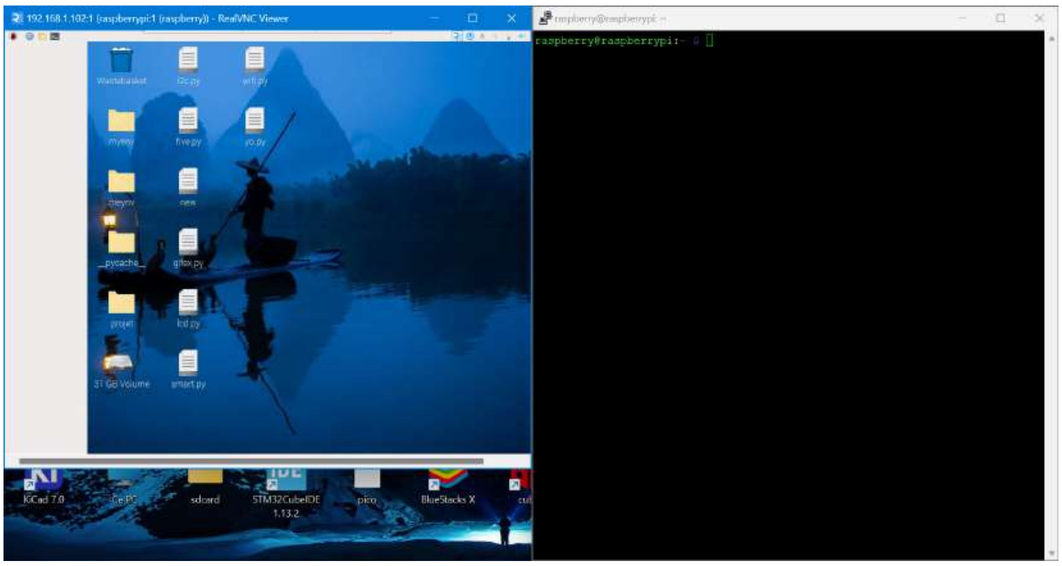
   (b) Insert the sd card into the computer.

   (c) Run raspberry pi imager and select the board model and the iso file , modify wifi's setings , login's information (username and password) and click run.

   (d) Once the process's finished , insert the sd card into the raspberry pi board and plug it into the power supply. If you have a screen with an HDMI input, you can connect your Raspberry Pi to it using an HDMI cable. This will allow you to interact with your Raspberry Pi directly if not , connect it to a network , install a vnc viewer software on the computer , (both devices should be connected to the same network).

   (e) To connect the raspberry pi to the vncviewer , another program called Putty is used to establish a ssh connection between the board and the laptop, once it's done , type the command **$sudo raspi-config** , enable the vnc feature , and reboot the board.

(f) Everything is ready , open the vncviewer and type the ip address of the raspberry pi followed by :1



```
raspberry@raspberrypi:~ $ 
```

Figure 3.3: Raspberry pi 4 Linux Desktop

4. Real vnc viewer : VNC Viewer is used for local computers and mobile devices you want to control from[36]. it's used to control remotely the raspberry board.

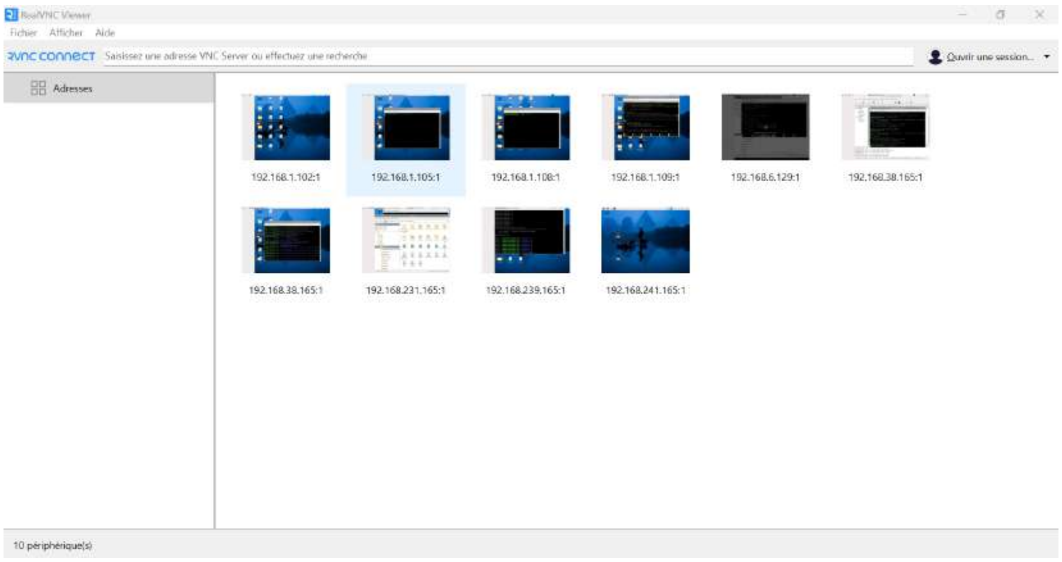


Figure 3.4: RealVnc Viewer

3.3 PCB circuit design

1. The first thing to do is to design the circuit of the smart glove which is displayed in the **Figure 3.5**.

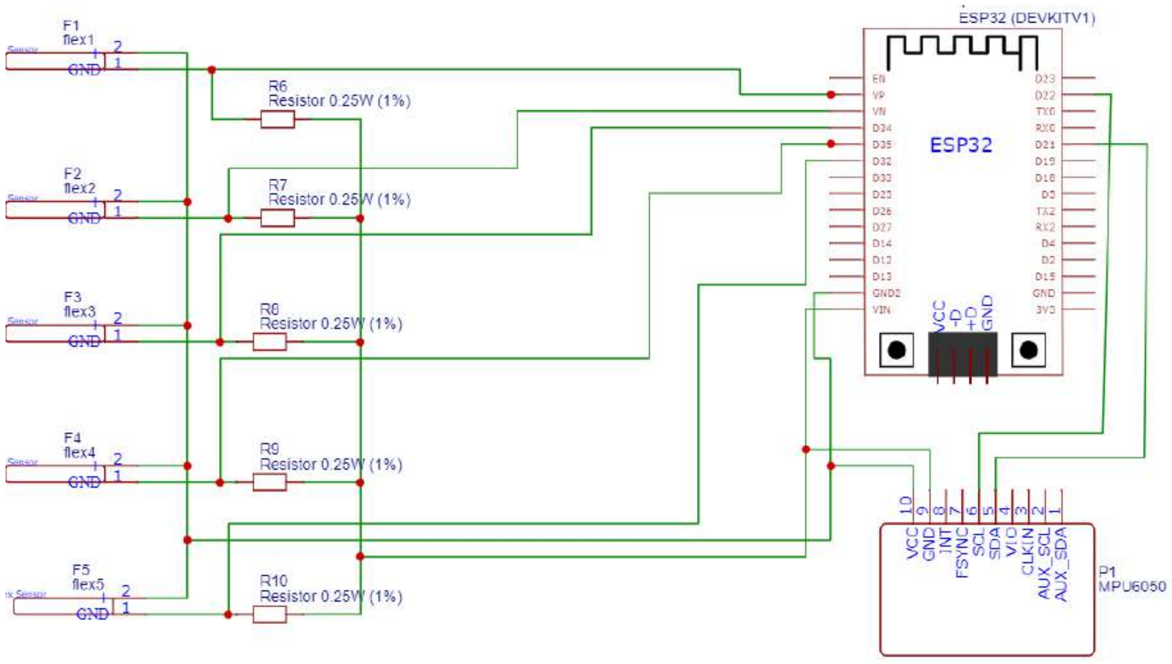
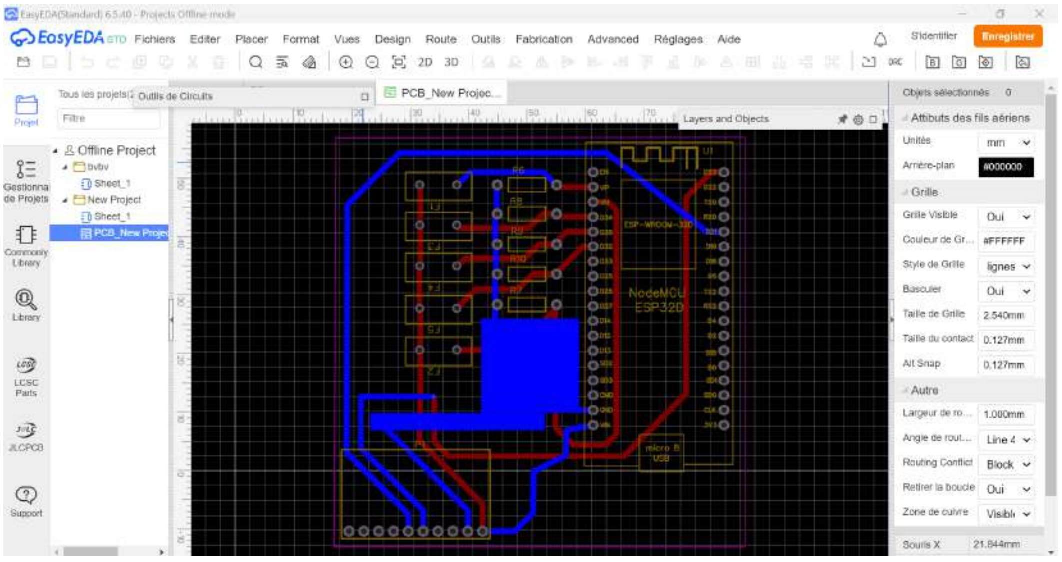


Figure 3.5: Smart glove's Schematic

2. After designing the circuit , The schematic was converted to a PCB, as shown in the **Figures 3.6,3.7**. The PCB consists of two layers: a copper layer and a components layer as seen in **Figures 3.8 and 3.9**.



Objets sélectionnés	0
Attibuts des fils aériens	
Unités	mm
Arrière-plan	#000000
Grille	
Grille Visible	Oui
Couleur de Gr...	#FFFFFF
Style de Grille	lignes
Basculer	Oui
Taille de Grille	2.540mm
Taille du contact	0.127mm
Alt Snap	0.127mm
Autre	
Largeur de ro...	1.000mm
Angle de rout...	Line 4
Routing Conflict	Block
Retirer la boucle	Oui
Zone de cuivre	Visibl
Souris X	21.844mm

Figure 3.6: EasyEda PCB Design

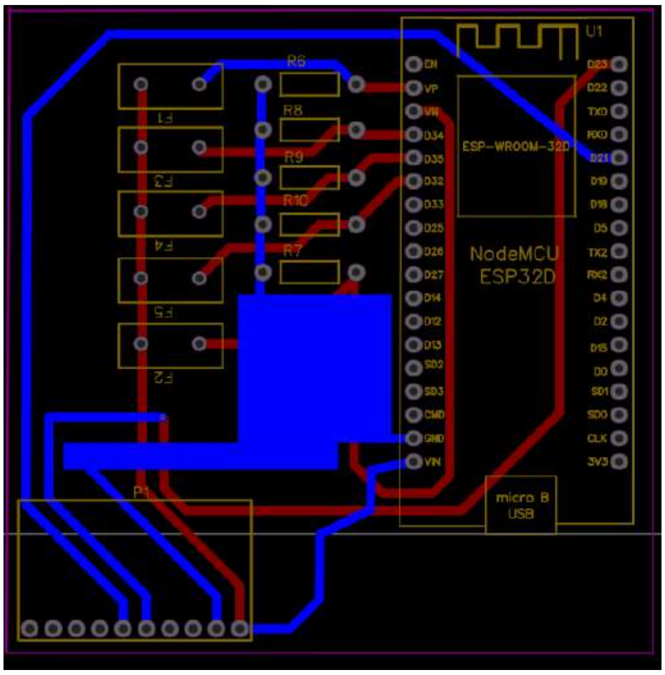


Figure 3.7: PCB Design result

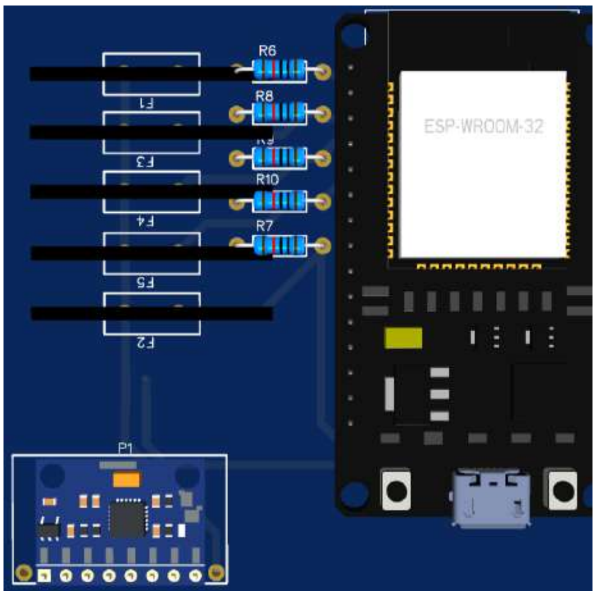


Figure 3.8: 3D Top view of Double-Sided PCB Prototype

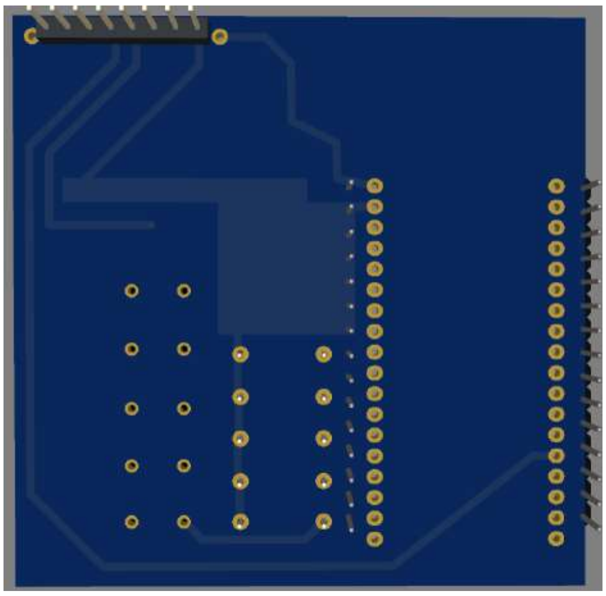


Figure 3.9: 3D bottom View of Double-Sided PCB Prototype

3. The circuit design is finished , now , it's time for soldering components , we couldn't print a pcb thus , we used a double-sided pcb prototype which are shown the **Figures 3.11 , 3.12 , 3.10**. The solder mask is used as an insulator for additional protection and to prevent oxidation [37]. The mask is treated with UV light.

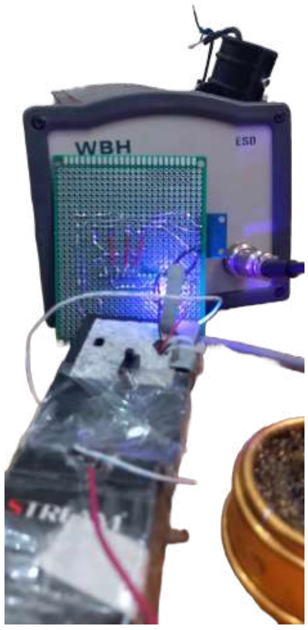


Figure 3.10: UV Exposure After Applying Solder Mask

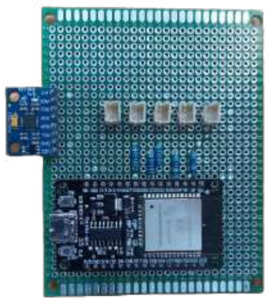


Figure 3.11: Top View of Double-Sided PCB Prototype

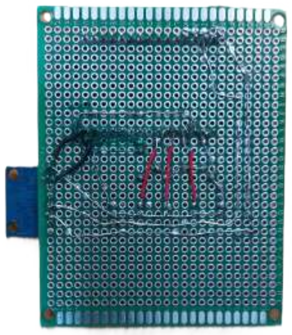


Figure 3.12: Bottom View of Double-Sided PCB Prototype

4. The flex sensors are attached directly onto the glove using silicone including the board too , as a consequence we get a fully smart glove with all its necessary components as illustrated in **Figure 3.13**

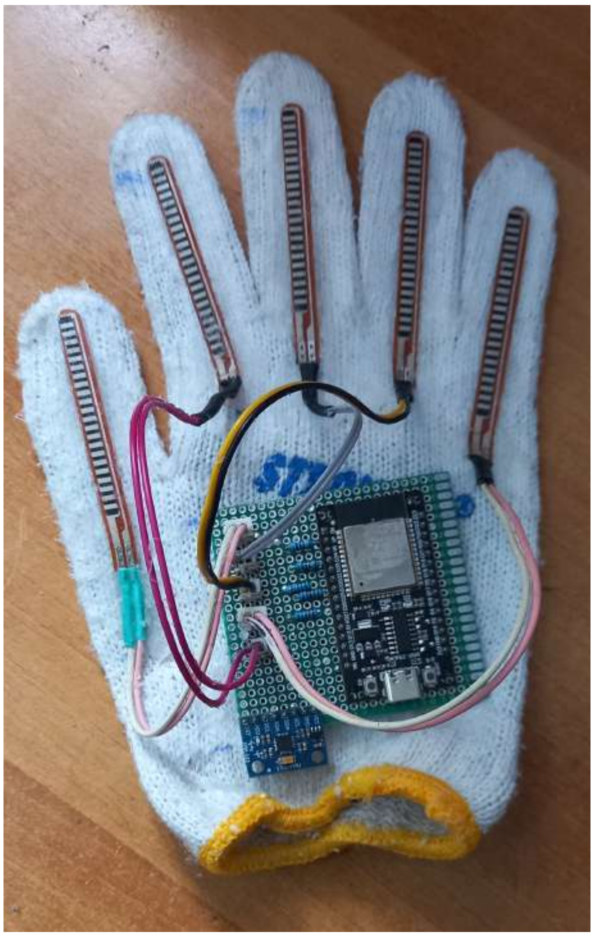


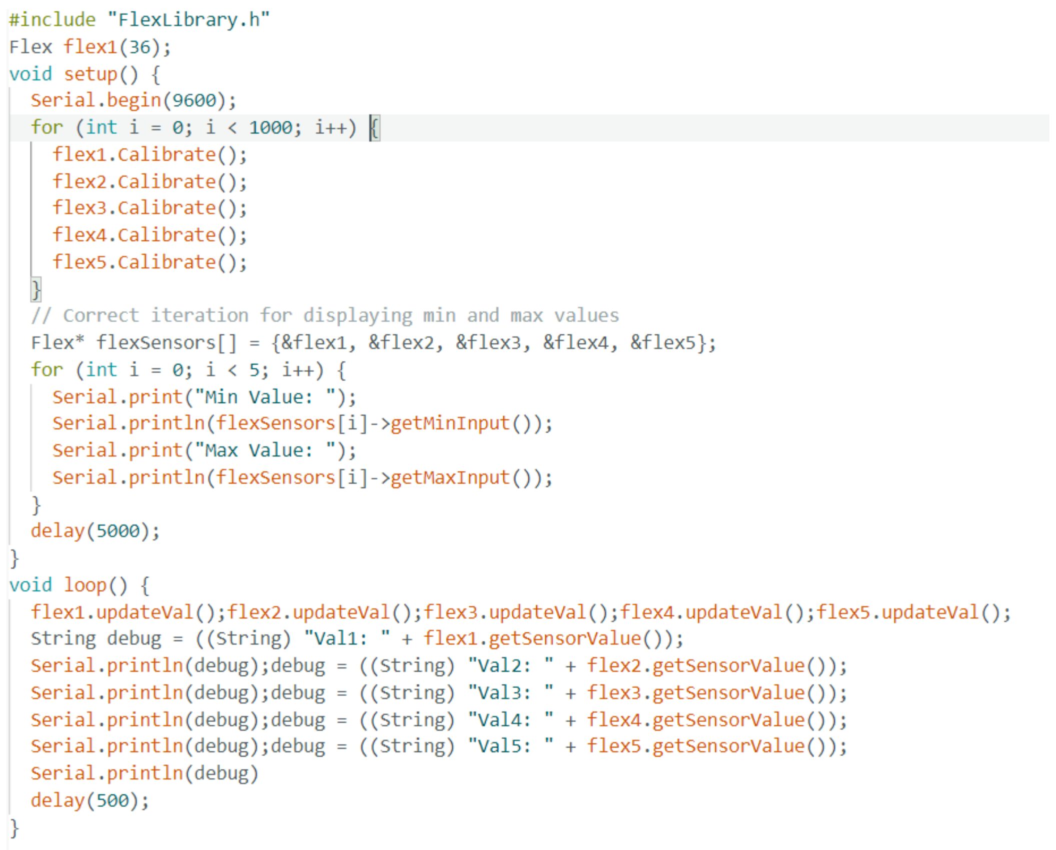
Figure 3.13: Smart Glove

3.4 Sensors Calibration

Calibrating the sensors is an essential step before using them as it ensures intact measurements and high precision.

3.4.1 Flex sensors calibration

To calibrate the flex sensors ,The specific library available on the Arduino IDE is installed, and here is the complete code for the calibration, shown in **Figure 3.14**:



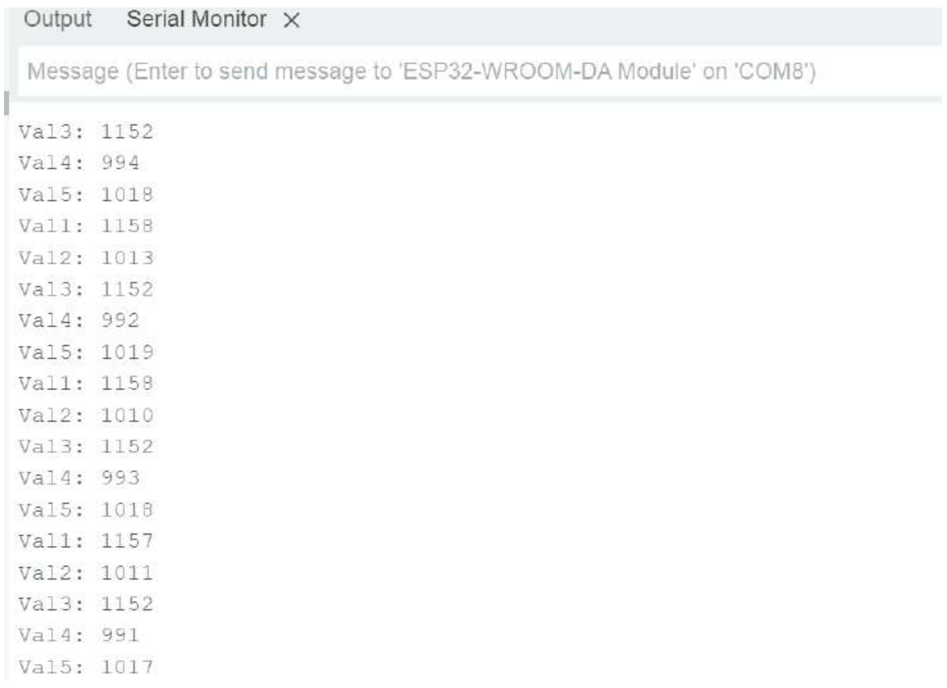
```
#include "FlexLibrary.h"
Flex flex1(36);
void setup() {
  Serial.begin(9600);
  for (int i = 0; i < 1000; i++) {
    flex1.Calibrate();
    flex2.Calibrate();
    flex3.Calibrate();
    flex4.Calibrate();
    flex5.Calibrate();
  }
  // Correct iteration for displaying min and max values
  Flex* flexSensors[] = {&flex1, &flex2, &flex3, &flex4, &flex5};
  for (int i = 0; i < 5; i++) {
    Serial.print("Min Value: ");
    Serial.println(flexSensors[i]->getMinInput());
    Serial.print("Max Value: ");
    Serial.println(flexSensors[i]->getMaxInput());
  }
  delay(5000);
}
void loop() {
  flex1.updateVal();flex2.updateVal();flex3.updateVal();flex4.updateVal();flex5.updateVal();
  String debug = ((String) "Val1: " + flex1.getSensorValue());
  Serial.println(debug);debug = ((String) "Val2: " + flex2.getSensorValue());
  Serial.println(debug);debug = ((String) "Val3: " + flex3.getSensorValue());
  Serial.println(debug);debug = ((String) "Val4: " + flex4.getSensorValue());
  Serial.println(debug);debug = ((String) "Val5: " + flex5.getSensorValue());
  Serial.println(debug)
  delay(500);
}
```

Figure 3.14: Flex sensors Calibration code

The analog values of the five flex sensors range between 1073 and 1125 (See **Figure 3.15**) . Since the ESP32's ADC channels are 12 bits, the voltage between the flex sensors can be calculated as follows:

$$V = \frac{1073 \times 5}{2^{12} - 1} = 1.31V \tag{3.1}$$

Theoretically , the result was 1.17V.

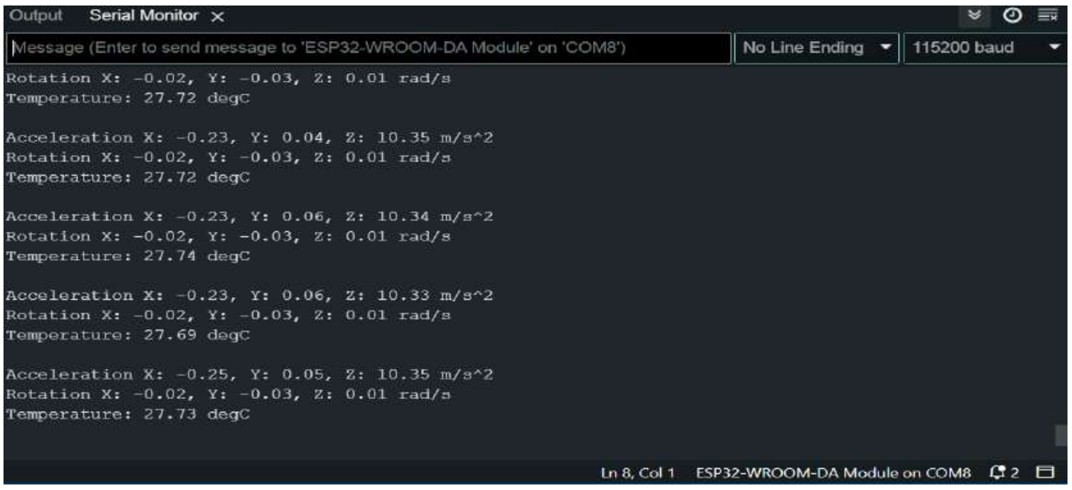

Output
Serial Monitor
Message (Enter to send message to 'ESP32-WROOM-DA Module' on 'COM8')

```
Val3: 1152
Val4: 994
Val5: 1018
Val1: 1158
Val2: 1013
Val3: 1152
Val4: 992
Val5: 1019
Val1: 1158
Val2: 1010
Val3: 1152
Val4: 993
Val5: 1018
Val1: 1157
Val2: 1011
Val3: 1152
Val4: 991
Val5: 1017
```

Figure 3.15: Flex sensors calibration results

3.4.2 MPU6050 Calibration

In the case of mpu6050 ,Its accelerometer and gyroscope should be calibrated. , but before that , we make sure that the module is working and to know that ,The specific library that provides the privilege to read both accelerometer and gyroscope data is installed. the code for that is shown in **Figure 3.16**


Output
Serial Monitor
Message (Enter to send message to 'ESP32-WROOM-DA Module' on 'COM8')
No Line Ending
115200 baud
Ln 8, Col 1
ESP32-WROOM-DA Module on COM8
2

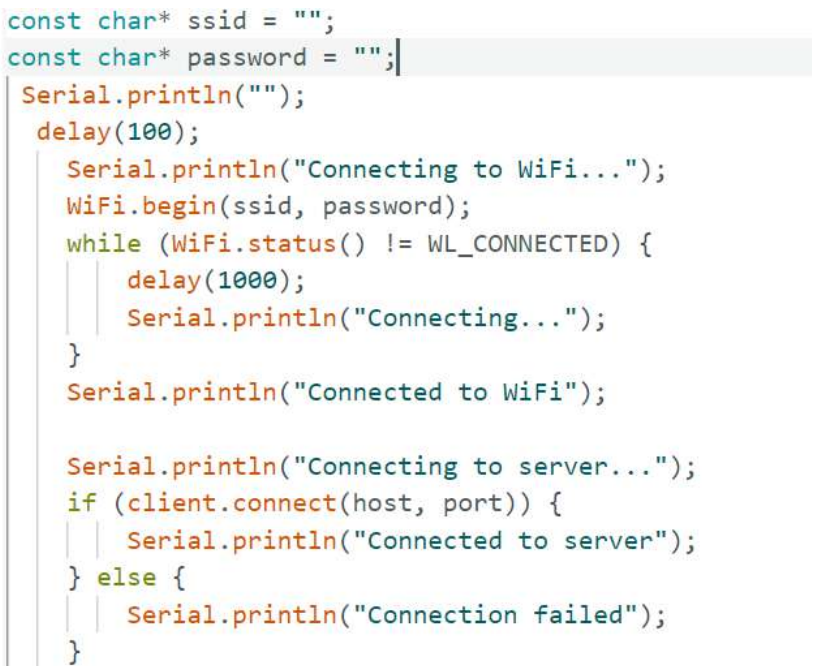
```
Rotation X: -0.02, Y: -0.03, Z: 0.01 rad/s
Temperature: 27.72 degC

Acceleration X: -0.23, Y: 0.04, Z: 10.35 m/s^2
Rotation X: -0.02, Y: -0.03, Z: 0.01 rad/s
Temperature: 27.72 degC

Acceleration X: -0.23, Y: 0.06, Z: 10.34 m/s^2
Rotation X: -0.02, Y: -0.03, Z: 0.01 rad/s
Temperature: 27.74 degC

Acceleration X: -0.23, Y: 0.06, Z: 10.33 m/s^2
Rotation X: -0.02, Y: -0.03, Z: 0.01 rad/s
Temperature: 27.69 degC

Acceleration X: -0.25, Y: 0.05, Z: 10.35 m/s^2
Rotation X: -0.02, Y: -0.03, Z: 0.01 rad/s
Temperature: 27.73 degC
```

Figure 3.16: MPU6050 Calibration process

3.5 Establishing wireless communication

As mentioned before , the communication between esp32 and the raspberry pi 4 is via TCP/IP protocol , there are the following steps to follow to achieve it :

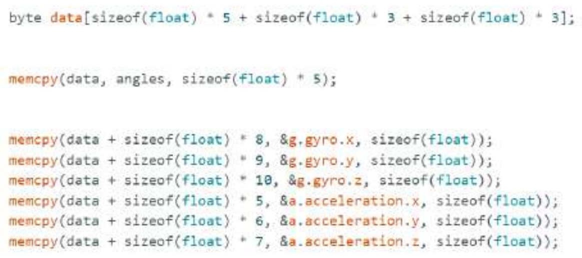
1. Both devices should be connected to a WI-FI network , to connect the esp32 to a WI-FI network , The code shown in **Figure 3.17** is used:



```
const char* ssid = "";
const char* password = "";
 Serial.println("");
  delay(100);
    Serial.println("Connecting to WiFi...");
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting...");
    }
    Serial.println("Connected to WiFi");

    Serial.println("Connecting to server...");
    if (client.connect(host, port)) {
        Serial.println("Connected to server");
    } else {
        Serial.println("Connection failed");
    }
```

Figure 3.17: ESP32 WI-Fi connection code

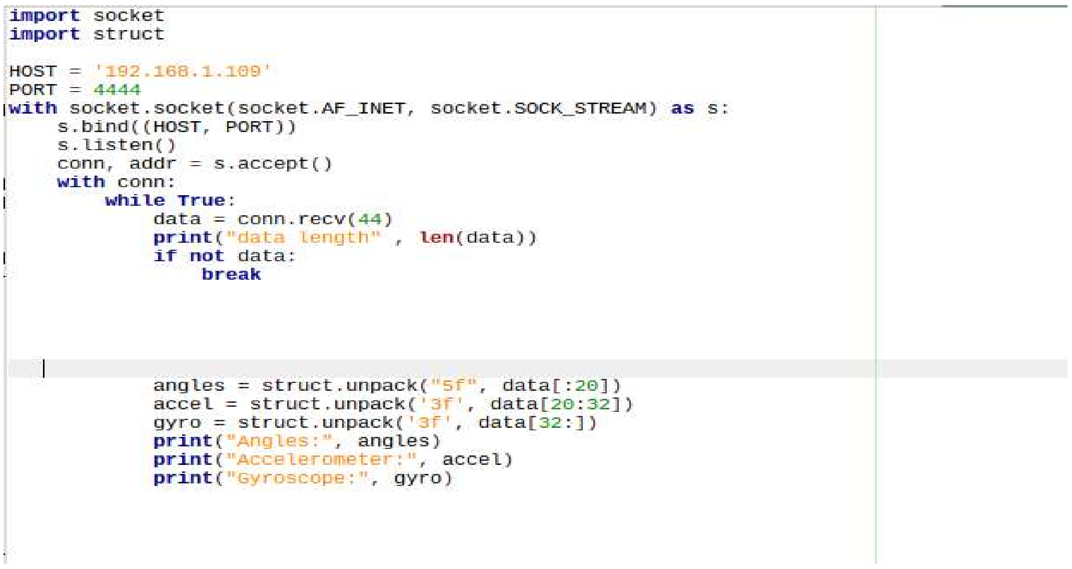
2. The sensors data are sent in a binary format , and to do that ,A buffer is created and filled with the data, as shown in **Figure 3.18**:



```
byte data[sizeof(float) * 5 + sizeof(float) * 3 + sizeof(float) * 3];


memcpy(data, angles, sizeof(float) * 5);


memcpy(data + sizeof(float) * 8, &g.gyro.x, sizeof(float));
memcpy(data + sizeof(float) * 9, &g.gyro.y, sizeof(float));
memcpy(data + sizeof(float) * 10, &g.gyro.z, sizeof(float));
memcpy(data + sizeof(float) * 5, &a.acceleration.x, sizeof(float));
memcpy(data + sizeof(float) * 6, &a.acceleration.y, sizeof(float));
memcpy(data + sizeof(float) * 7, &a.acceleration.z, sizeof(float));
```

Figure 3.18: Sensors data processing code

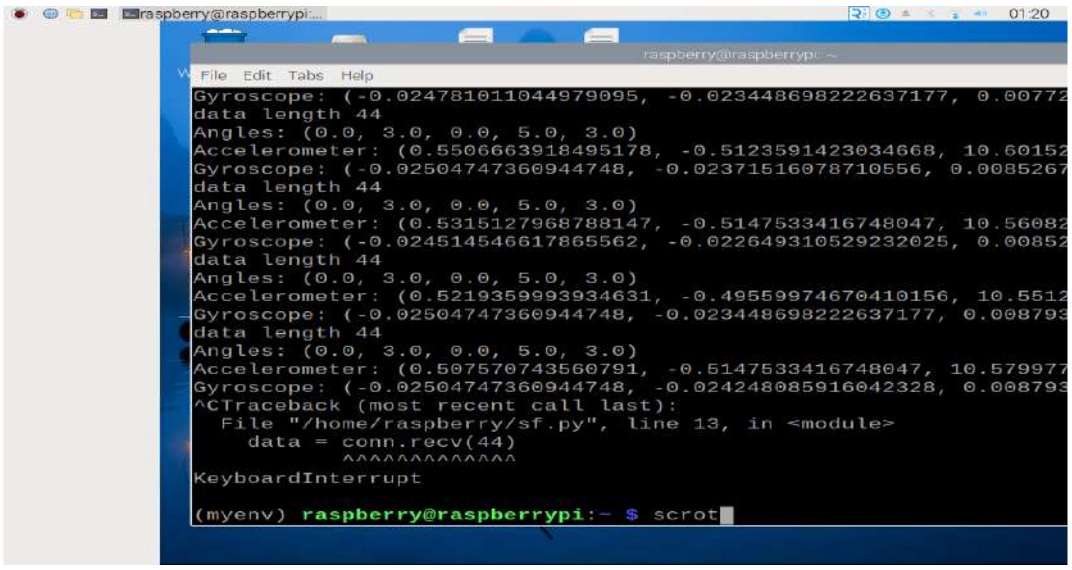
3. Since the raspberry pi 4 is already connected to WI-FI ,a Python script is created to set up the TCP/IP connection and receive the sensor data. The code is shown below in **Figure 3.19**:



```python
import socket
import struct

HOST = '192.168.1.109'
PORT = 4444
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        while True:
            data = conn.recv(44)
            print("data length" , len(data))
            if not data:
                break



            angles = struct.unpack("5f", data[:20])
            accel = struct.unpack('3f', data[20:32])
            gyro = struct.unpack('3f', data[32:])
            print("Angles:", angles)
            print("Accelerometer:", accel)
            print("Gyroscope:", gyro)
```

Figure 3.19: Python Code for data reception

4. Everything is set up , now , we run both codes , the response is represented as seen in **Figure 3.20**



```
Gyroscope: (-0.024781011044979095, -0.023448698222637177, 0.00772
data length 44
Angles: (0.0, 3.0, 0.0, 5.0, 3.0)
Accelerometer: (0.5506663918495178, -0.5123591423034668, 10.60152
Gyroscope: (-0.025047473609447 48, -0.023715160787105 56, 0.0085267
data length 44
Angles: (0.0, 3.0, 0.0, 5.0, 3.0)
Accelerometer: (0.5315127968788147, -0.5147533416748047, 10.56082
Gyroscope: (-0.024514546617865562, -0.022649310529232025, 0.00852
data length 44
Angles: (0.0, 3.0, 0.0, 5.0, 3.0)
Accelerometer: (0.5219359993934631, -0.49559974670410156, 10.5512
Gyroscope: (-0.025047473609447 48, -0.023448698222637177, 0.008793
data length 44
Angles: (0.0, 3.0, 0.0, 5.0, 3.0)
Accelerometer: (0.507570743560791, -0.5147533416748047, 10.579977
Gyroscope: (-0.025047473609447 48, -0.024248085916042328, 0.008793
^CTraceback (most recent call last):
  File "/home/raspberry/sf.py", line 13, in <module>
    data = conn.recv(44)
           ^^^^^^^^^^^^^^
KeyboardInterrupt

(myenv) raspberry@raspberrypi:~ $ scrot
```

Figure 3.20: Sensors data reception results

3.6 Random forests Model training

3.6.1 Training steps

Our machine learning has been trained for different cases :

1. Training the model for only alphabet.
2. Training the model for only expressions.

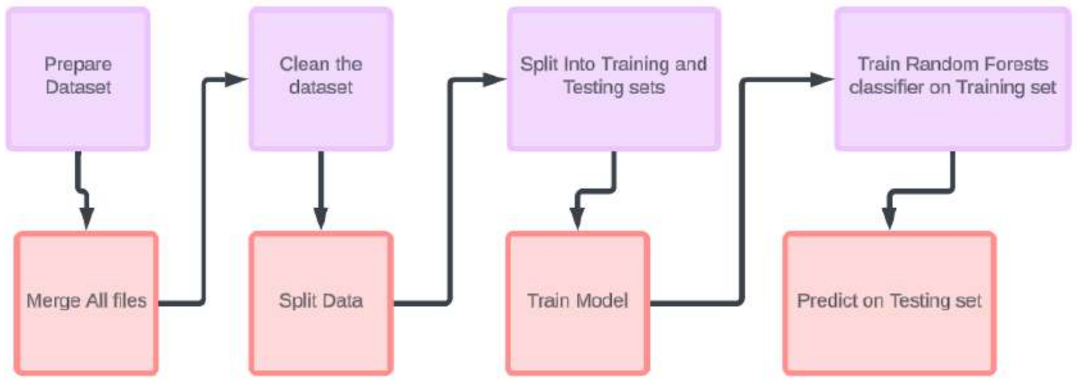
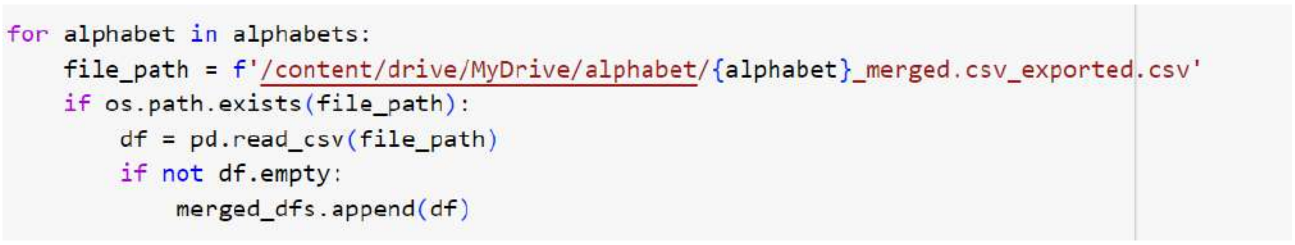


Figure 3.21: Machine learning Training Process

3.6.2 Dataset cleaning process

As mentioned before , before using the dataset , Data cleaning is necessary. , in our case ,Negative values were eliminated from the flex sensor data. The number of important features is 11. The cleaning is done in three steps:

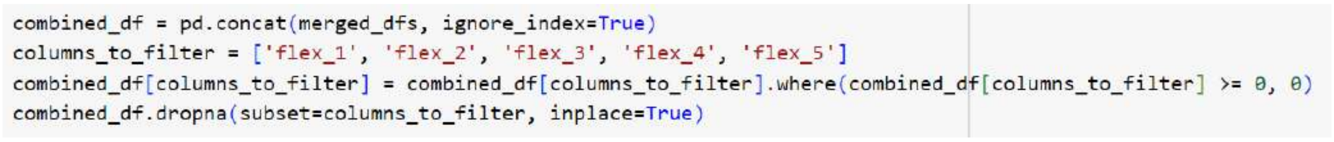
1. Merging the csv files : All the datasets should be gathered in one file before the training process, as explained in the next code shown in **Figure 3.22**:



```
for alphabet in alphabets:
    file_path = f'/content/drive/MyDrive/alphabet/{alphabet}_merged.csv_exported.csv'
    if os.path.exists(file_path):
        df = pd.read_csv(file_path)
        if not df.empty:
            merged_dfs.append(df)
```

Figure 3.22: Python code for merging dataset

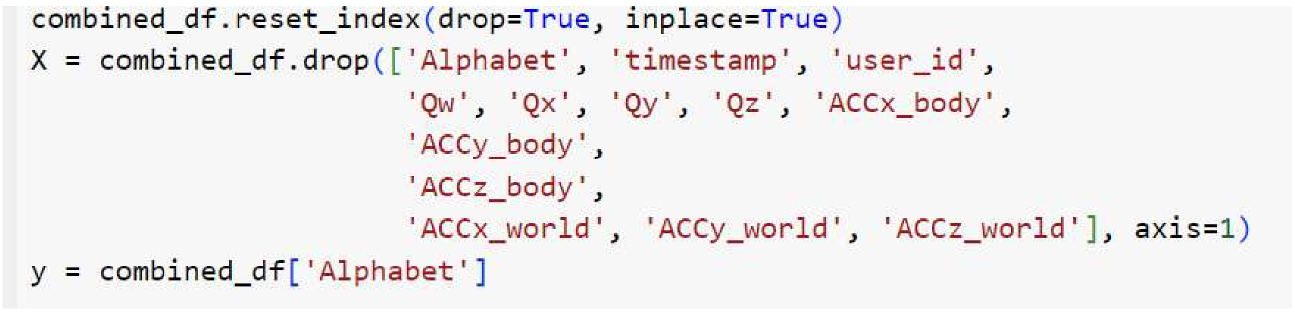
2. Cleaning the the used dataset: and that includes filling the missing values and removing the negative angle values , the process is shown in the code above illustrated in **Figure 3.23**



```
combined_df = pd.concat(merged_dfs, ignore_index=True)
columns_to_filter = ['flex_1', 'flex_2', 'flex_3', 'flex_4', 'flex_5']
combined_df[columns_to_filter] = combined_df[columns_to_filter].where(combined_df[columns_to_filter] >= 0, 0)
combined_df.dropna(subset=columns_to_filter, inplace=True)
```

Figure 3.23: Python code for cleaning dataset

3. After that , The unnecessary features are eliminated, leaving 11 important ones: , and to do that , The **Figure 3.23** shows the code for that.



```
combined_df.reset_index(drop=True, inplace=True)
X = combined_df.drop(['Alphabet', 'timestamp', 'user_id',
                      'Qw', 'Qx', 'Qy', 'Qz', 'ACCx_body',
                      'ACCy_body',
                      'ACCz_body',
                      'ACCx_world', 'ACCy_world', 'ACCz_world'], axis=1)
y = combined_df['Alphabet']
```

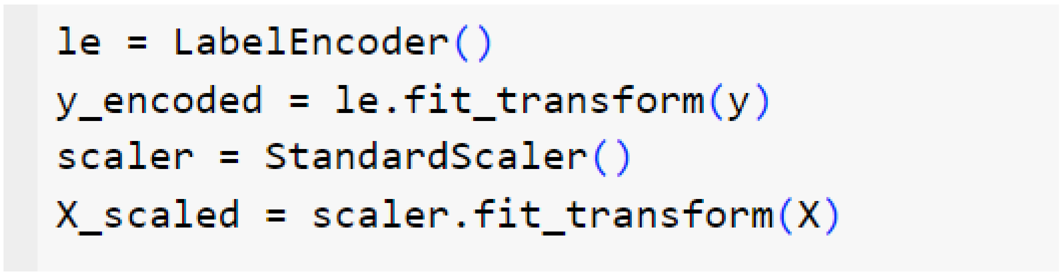
Figure 3.24: Python code for eliminating unnecessary dataset

The cleaning process is finished , the results are shown in the table 3.1 :

flex_1	**flex_2**	**flex_3**	**flex_4**	**flex_5**	**GYRx**	**GYRy**	**GYRz**	**ACCx**	**ACCy**	**ACCz**
0	46	47	40	37	-0.022901	-0.015267	-0.015267	9.380102	2.204761	-0.616089
0	47	44	38	35	-0.022901	-0.015267	-0.015267	9.347803	2.209546	-0.618481
0	47	48	46	33	-0.015267	-0.015267	-0.007634	9.317896	2.183228	-0.634033
0	48	50	40	33	-0.015267	-0.007634	-0.007634	9.326269	2.154517	-0.662744
0	47	47	41	36	-0.007634	-0.007634	-0.007634	9.34541	2.18562	-0.679492
0	45	50	44	36	-0.007634	-0.007634	0.0	9.348999	2.228687	-0.697437
0	49	46	43	37	0.0	-0.007634	0.0	9.353785	2.263379	-0.709399
0	51	48	42	38	-0.007634	-0.007634	0.007634	9.363355	2.321997	-0.704614
0	48	48	38	35	-0.015267	-0.015267	0.015267	9.369336	2.42727	-0.680689
0	52	44	40	37	-0.022901	-0.015267	0.022901	9.374121	2.534936	-0.655566
0	49	48	42	37	-0.022901	-0.022901	0.022901	9.386084	2.612695	-0.620874
0	49	48	40	35	-0.022901	-0.022901	0.015267	9.392065	2.647388	-0.588574
0	48	46	38	33	-0.015267	-0.022901	0.007634	9.414795	2.639014	-0.551489
0	49	47	41	35	-0.015267	-0.015267	0.0	9.473413	2.604321	-0.508423
0	47	47	40	35	-0.007634	-0.015267	-0.007634	9.527246	2.558862	-0.446216
0	48	47	40	35	-0.015267	-0.022901	-0.015267	9.563135	2.511011	-0.385205

Table 3.1: Dataset pre-processing results

4. Scaling the data : and that means fitting the data into a specific scale , it's shown in the **Figure 3.25**:

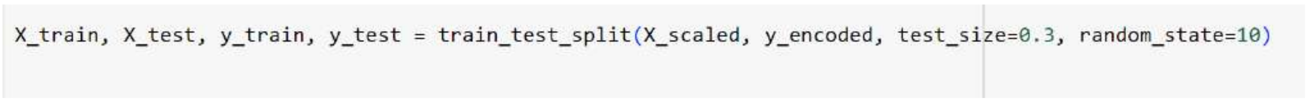


```
le = LabelEncoder()
y_encoded = le.fit_transform(y)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Figure 3.25: Python code for data normalisation

3.7 Tests and results

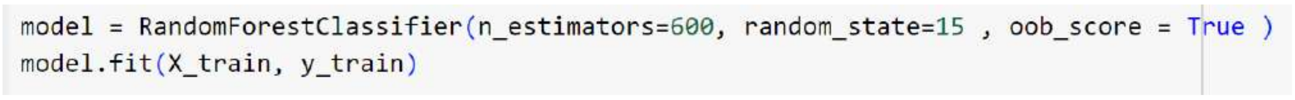
After preparing and normalising the dataset , the next step is training the model thus splitting the data into training and testing sets (30% for testing , 70% for training) , it's explained on the **Figure 3.26** :



```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded, test_size=0.3, random_state=10)
```

Figure 3.26: Python Code for Splitting Dataset into Training and Testing Sets

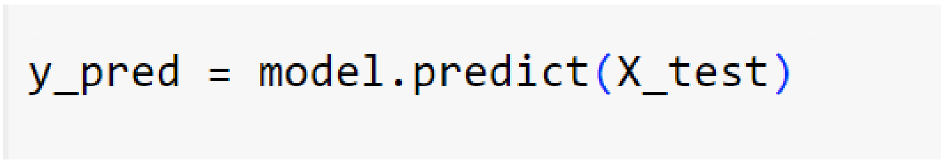
Next, the random forest function is declared without changing its default parameters: , The number of trees was randomly set to 600 as shown in **Figure 3.27**



```
model = RandomForestClassifier(n_estimators=600, random_state=15 , oob_score = True )
model.fit(X_train, y_train)
```

Figure 3.27: Random forests classifier python code

The last step is to to train the model on the unseen which is the testing set as shown in **Figure 3.28**



```
y_pred = model.predict(X_test)
```

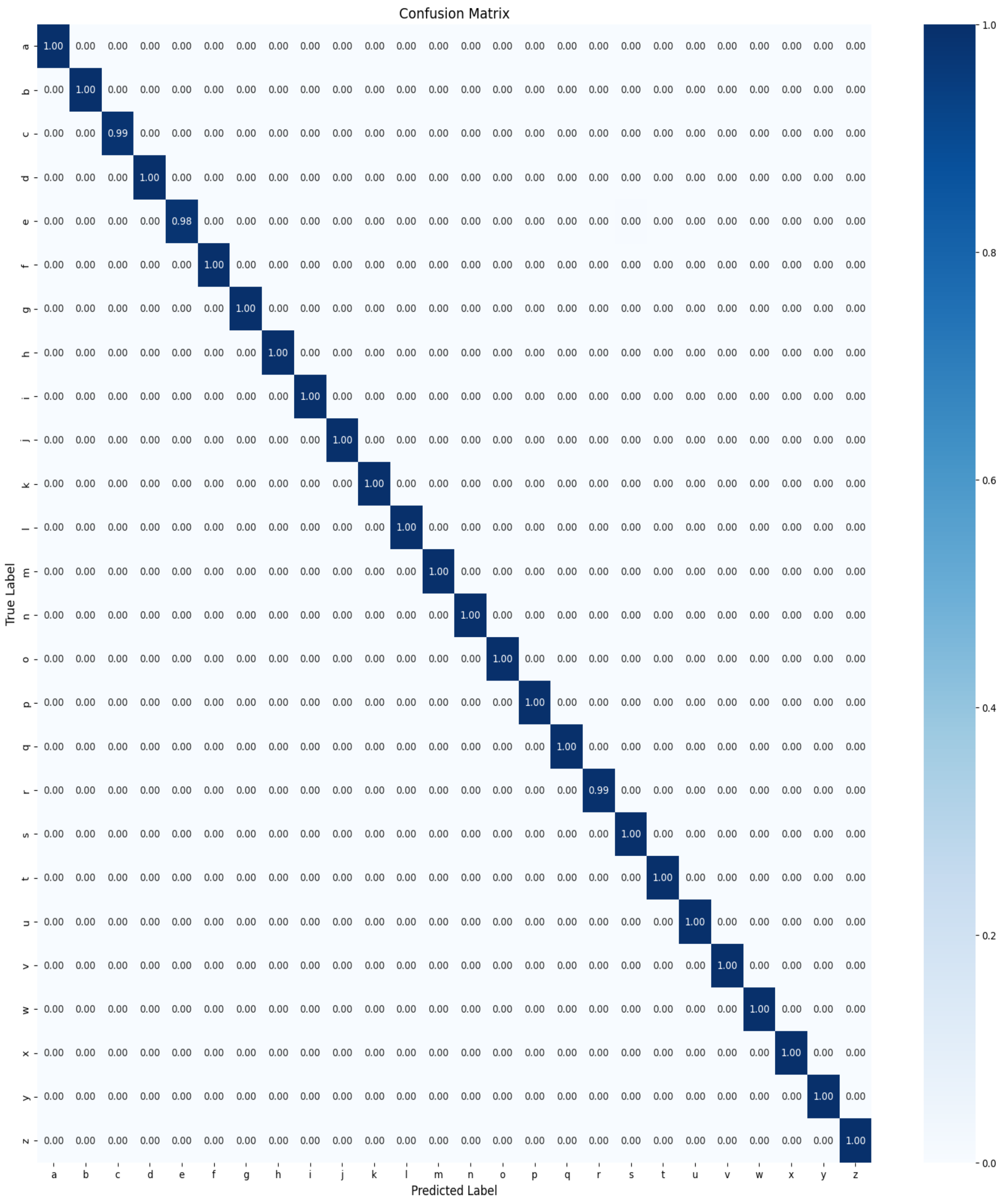
Figure 3.28: Python code for generating predictions on Test data

3.7.1 Training the model for predicting alphabet

First, the model is trained without modifying the classifier's parameters:

1. max_depth=None
2. min_samples_split=2
3. min_samples_leaf=1
4. max_samples=None
5. max_leaf_nodes=None

As results , an accuracy of 99.7% the figure below shows the confusion matrix of the trained machine learning model:

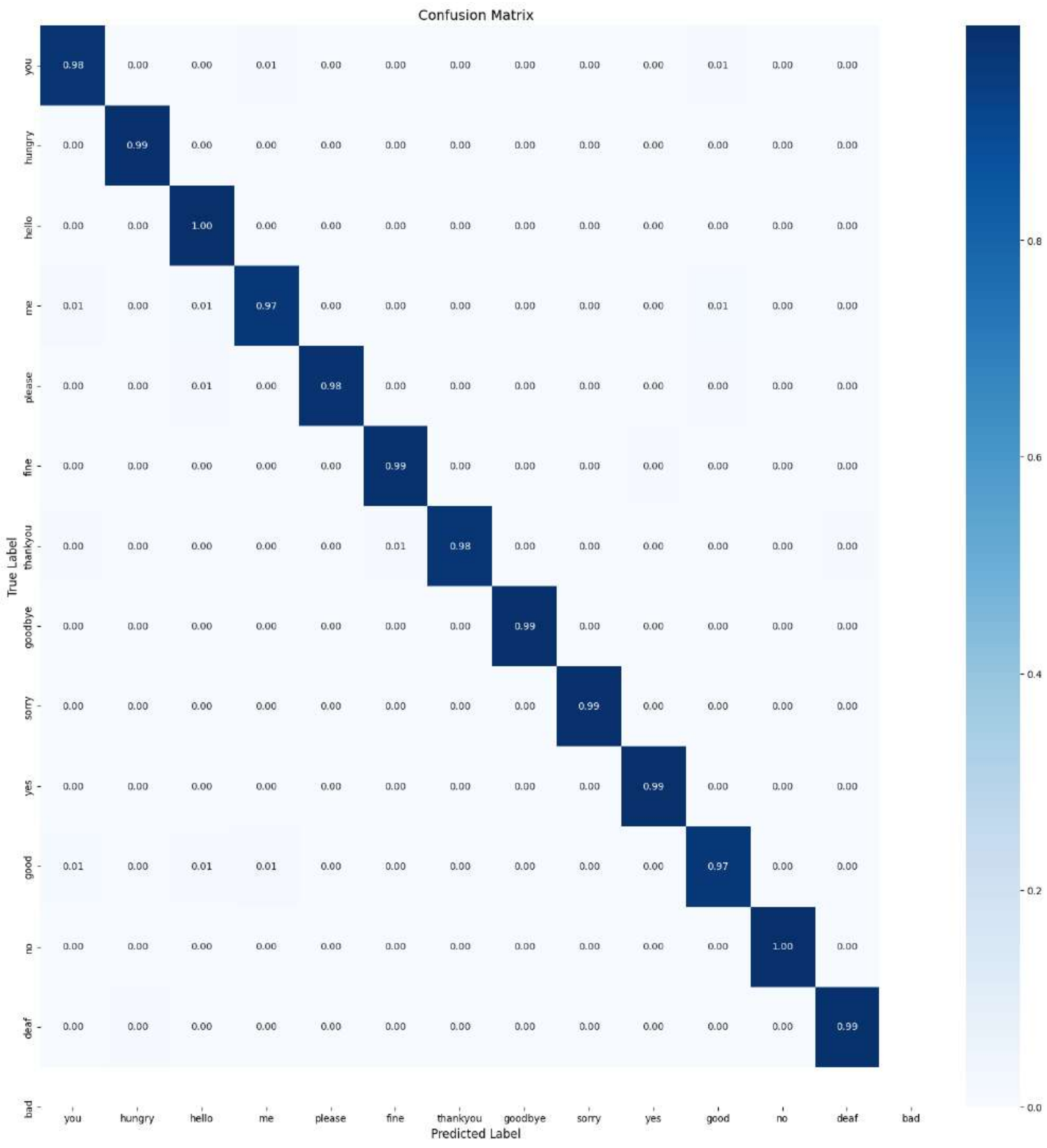


	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
c	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
d	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
e	0.00	0.00	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
g	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
h	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
i	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
j	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
k	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
l	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
m	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
n	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
o	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
p	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
q	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
s	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
t	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
v	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
w	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
x	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
y	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Figure 3.29: Alphabet Sign language Confusion Matrix (99% model accuracy)

3.7.2 Training the model for predicting expressions

With the same parameters , The accuracy this time was 98% as it's shown in the figure below :



	you	hungry	hello	me	please	fine	thankyou	goodbye	sorry	yes	good	no	deaf	bad
you	0.98	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	
hungry	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
hello	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
me	0.01	0.00	0.01	0.97	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	
please	0.00	0.00	0.01	0.00	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
fine	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00	
thankyou	0.00	0.00	0.00	0.00	0.00	0.01	0.98	0.00	0.00	0.00	0.00	0.00	0.00	
goodbye	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	
sorry	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	0.00	
yes	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	0.00	0.00	0.00	
good	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.97	0.00	0.00	
no	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	
deaf	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.99	
bad														

Figure 3.30: Sign language expressions Confusion Matrix(98% model accuracy)

Also, the oob score, ROC curve, and accuracy for different numbers of trees were calculated, as shown in **Figure 3.31** and **Figure 3.32**:

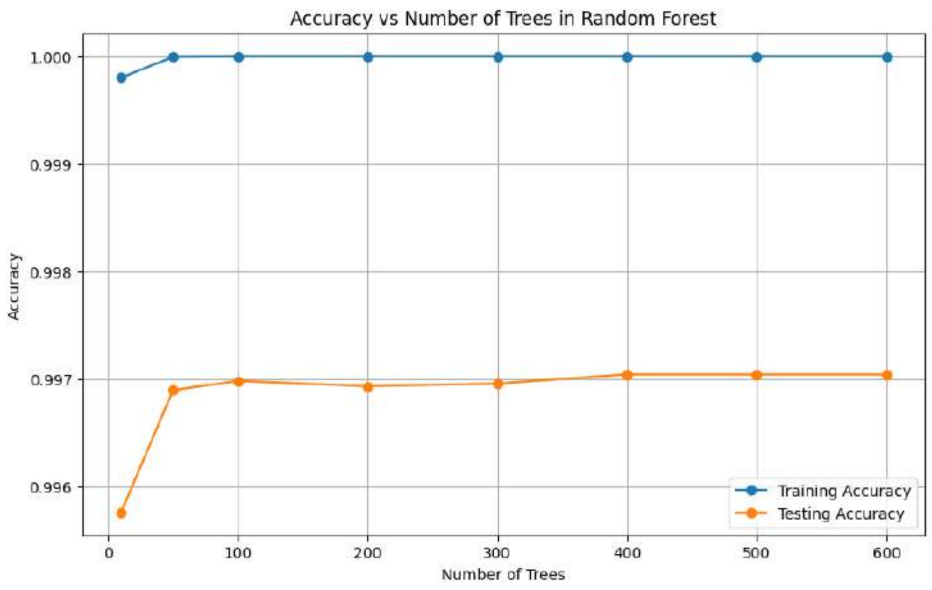


Figure 3.31: Testing and training accuracy for different number of trees (99% model accuracy)

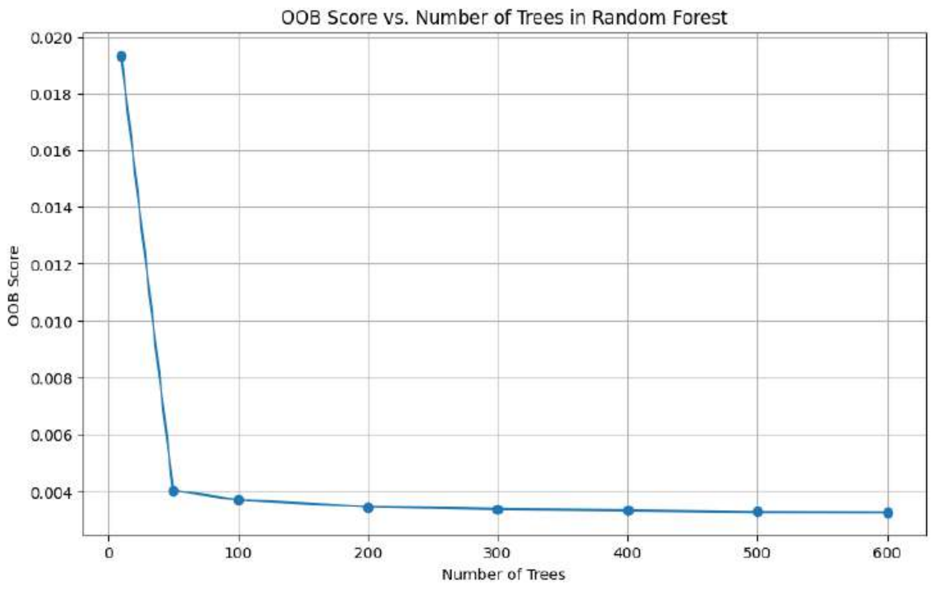


Figure 3.32: OOB Score for different number of trees(99% model accuracy)

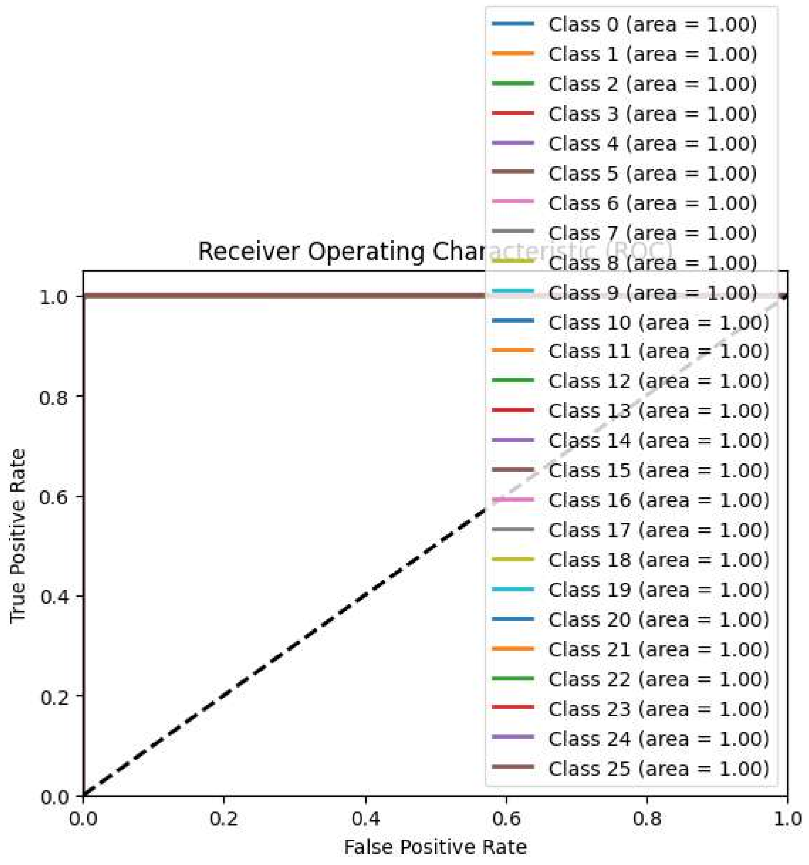


Figure 3.33: Alphabet sign language ROC curve(99% model accuracy)

Interpretation: both confusion matrix show a perfect classification for mostly all classes , as the majority of the values lie on the diagonal, indicating that the models predictions match perfectly the actual labels. Since there are no misclassifications , it's considered that both models overfit as they memorised prefectly the data and are unable to generalize well on unseen data.

The **Figure 3.31** shows both training and testing accuracy for different number of trees. As seen , the model has perfectly learned the training dataset as the accuracy becomes 100% changing of oob values for different number of trees. The max value is 0.025 and then it decreased to 0.004 while the number of trees becomes 600. That value is very small. A value of 0.004 means the model misclassified 0.4% of the training data which can explain the perfect results got. In addition to that , the changing of number of trees didn't have an impact on model's accuracy.

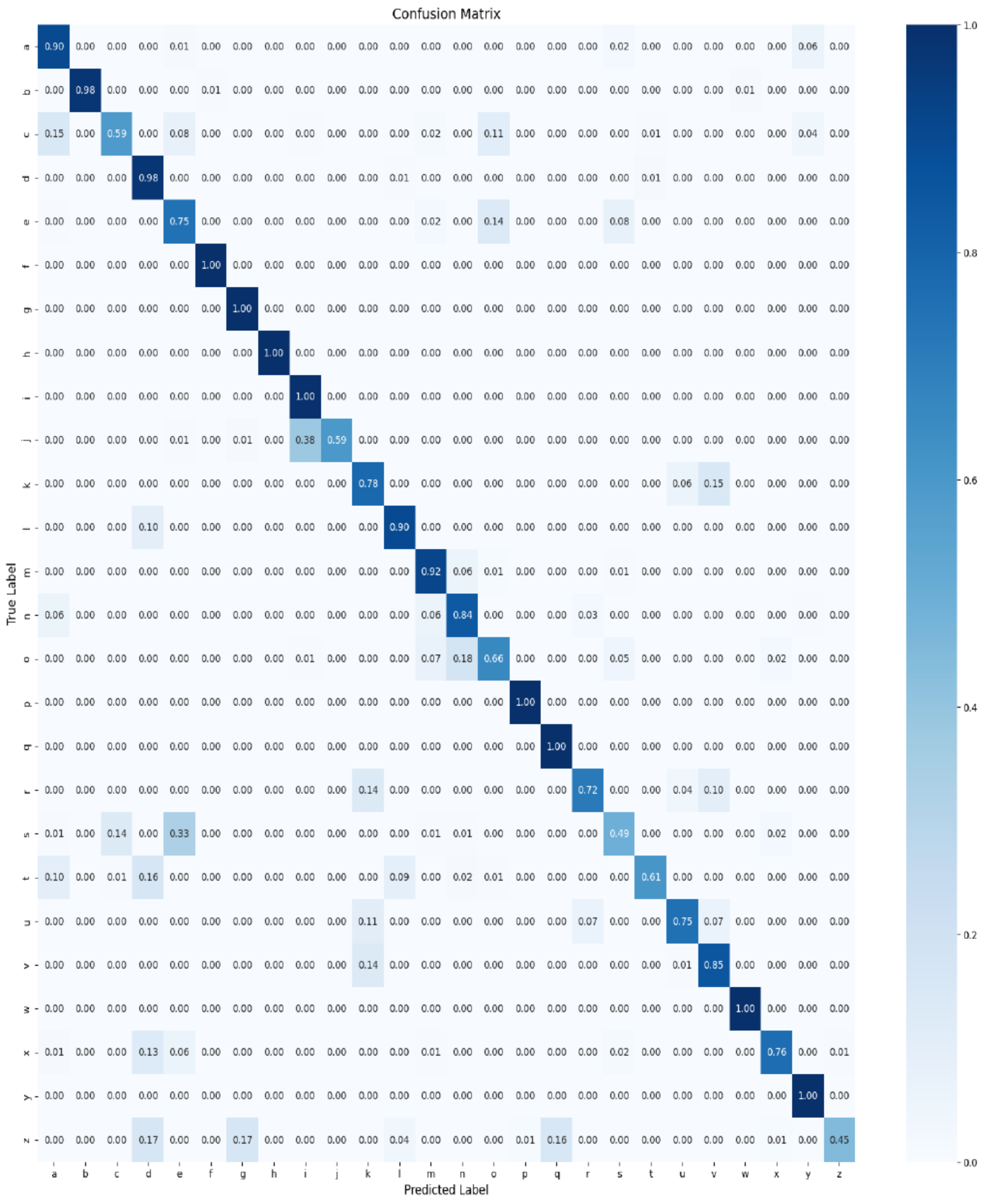
In conclusion , it's clear this model is overfitting thus hypertuning of the classifier's parameters needs to be looked forward to.

3.7.3 Fine-tuning classifier's hyperparameters

the model is trained again with different parameters but with the same number of trees.

1. min_samples_leaf=5
2. max_samples = 50
3. max_depth = 5
4. min_samples_split=4
5. max_leaf_nodes = 150

The model is trained again for predicting alphabet , and we got an accuracy of 82% and an oob score of 0.18 , the **Figure 3.35** and **Figure 3.36** give more details :



	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0.90	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.06	0.00
b	0.00	0.98	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
c	0.15	0.00	0.59	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.11	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.04	0.00
d	0.00	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
e	0.00	0.00	0.00	0.00	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.14	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00
f	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
g	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
h	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
i	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
j	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.38	0.59	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
k	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.78	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.15	0.00	0.00	0.00	0.00
l	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
m	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.06	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
n	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.84	0.00	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
o	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.07	0.18	0.66	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.02	0.00	0.00
p	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
q	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
r	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.72	0.00	0.00	0.04	0.10	0.00	0.00	0.00	0.00
s	0.01	0.00	0.14	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.49	0.00	0.00	0.00	0.00	0.02	0.00	0.00
t	0.10	0.00	0.01	0.16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.61	0.00	0.00	0.00	0.00	0.00	0.00
u	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.75	0.07	0.00	0.00	0.00	0.00
v	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.85	0.00	0.00	0.00	0.00
w	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00
x	0.01	0.00	0.00	0.13	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.76	0.00	0.01
y	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00
z	0.00	0.00	0.00	0.17	0.00	0.00	0.17	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.00	0.01	0.16	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.45

Figure 3.34: Alphabet Sign language Confusion Matrix (82% accuracy)

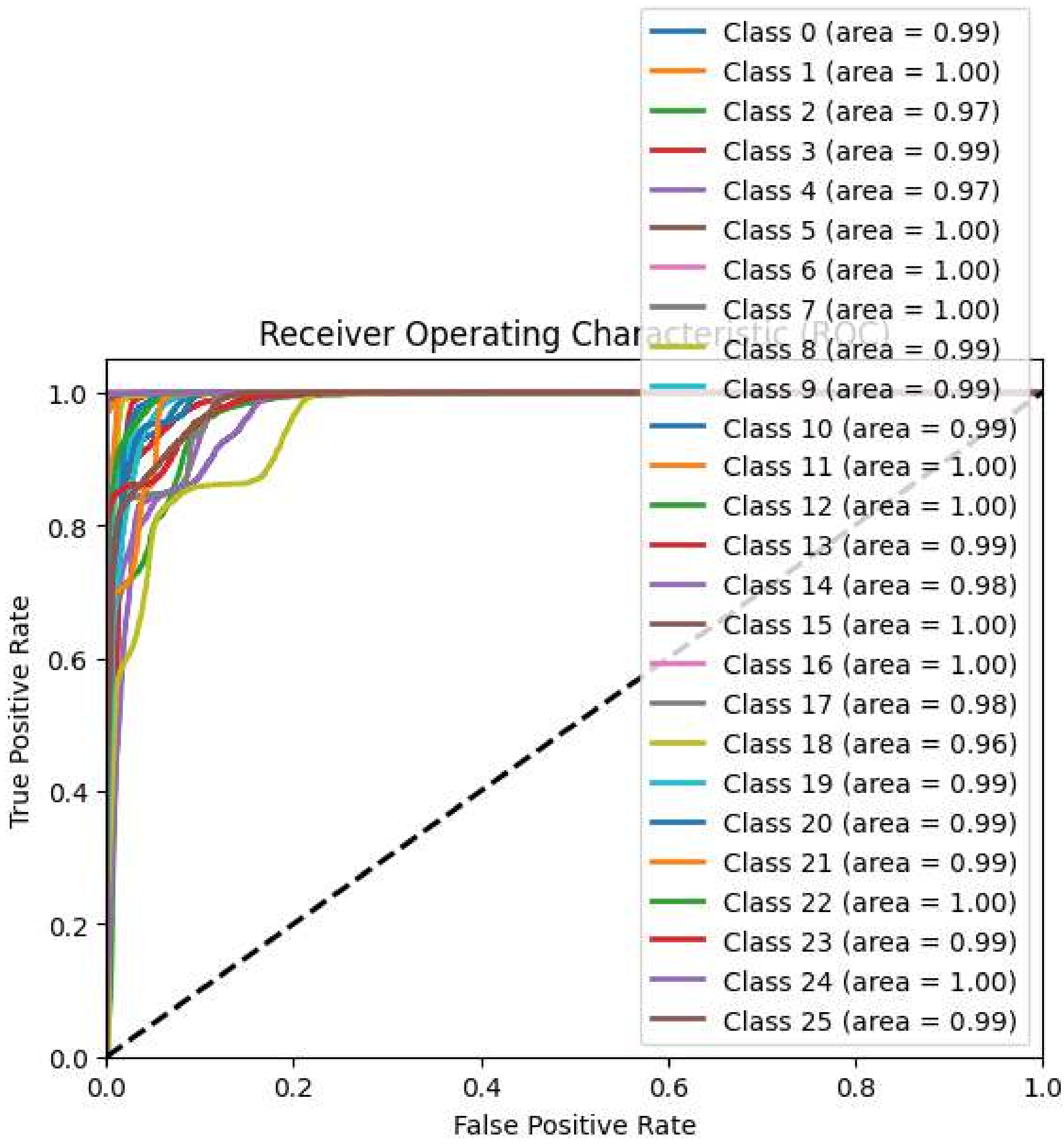


Figure 3.35: Alphabet sign language ROC curve(82% accuracy)

The out-of-bag (OOB) score and accuracy for different numbers of trees were also recalculated, as shown in **Figure 3.36** and **Figure 3.37**.

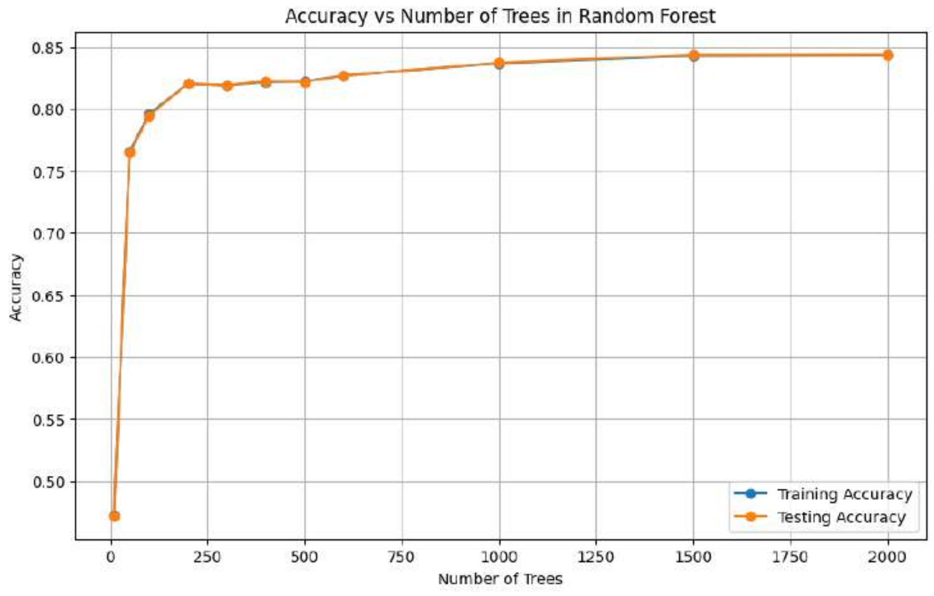


Figure 3.36: Testing and training accuracy for different number of trees(82% accuracy)

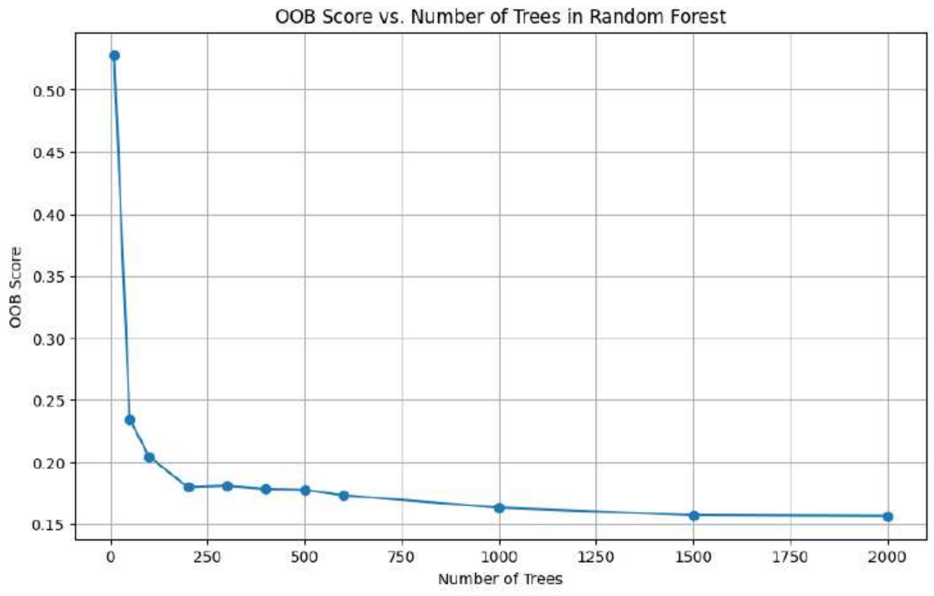
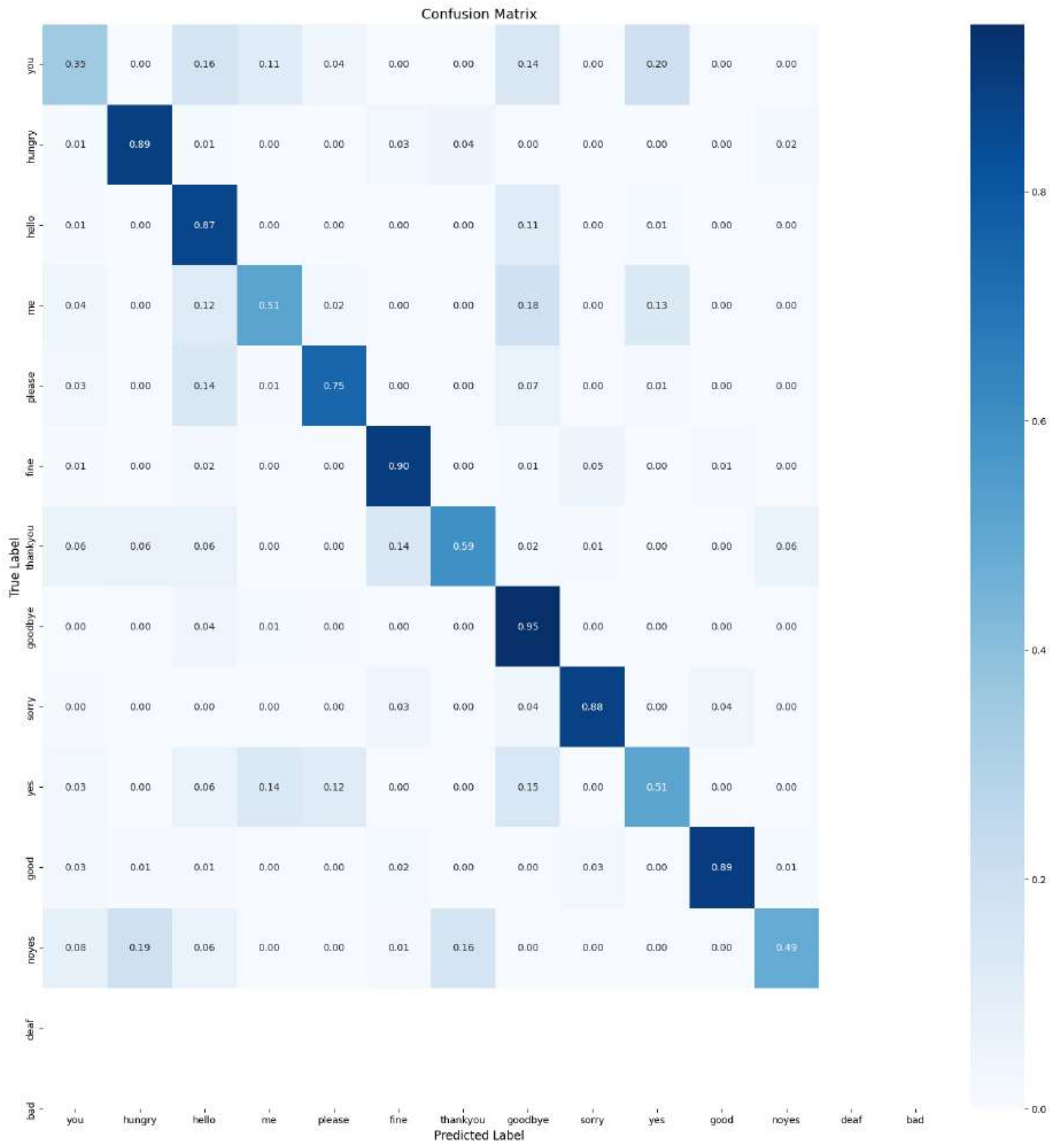


Figure 3.37: OOB Score for different number of trees(82% model accuracy)

On the other side, only 71% accuracy was achieved when the model was trained to predict sign language expressions. **Figure 3.38** below elaborates on the model's performance.



	you	hungry	hello	me	please	fine	thankyou	goodbye	sorry	yes	good	noyes	deaf	bad
you	0.35	0.00	0.16	0.11	0.04	0.00	0.00	0.14	0.00	0.20	0.00	0.00		
hungry	0.01	0.89	0.01	0.00	0.00	0.03	0.04	0.00	0.00	0.00	0.00	0.02		
hello	0.01	0.00	0.87	0.00	0.00	0.00	0.00	0.11	0.00	0.01	0.00	0.00		
me	0.04	0.00	0.12	0.51	0.02	0.00	0.00	0.18	0.00	0.13	0.00	0.00		
please	0.03	0.00	0.14	0.01	0.75	0.00	0.00	0.07	0.00	0.01	0.00	0.00		
fine	0.01	0.00	0.02	0.00	0.00	0.90	0.00	0.01	0.05	0.00	0.01	0.00		
thankyou	0.06	0.06	0.06	0.00	0.00	0.14	0.59	0.02	0.01	0.00	0.00	0.06		
goodbye	0.00	0.00	0.04	0.01	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00		
sorry	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.04	0.88	0.00	0.04	0.00		
yes	0.03	0.00	0.06	0.14	0.12	0.00	0.00	0.15	0.00	0.51	0.00	0.00		
good	0.03	0.01	0.01	0.00	0.00	0.02	0.00	0.00	0.03	0.00	0.89	0.01		
noyes	0.08	0.19	0.06	0.00	0.00	0.01	0.16	0.00	0.00	0.00	0.00	0.49		
deaf														
bad														

Figure 3.38: Enter Caption

Interpretation : This time , both confusion matrix shown in **Figure 3.34** and **Figure 3.38** were affected by the hyperparameters changing , as seen, some misclassifications occurred, such as the class ”a” being sometimes misclassified. predicted as ”d” (0.15), and ”b” is sometimes predicted as ”m” (0.09) , also classes such as g (0.75), ”q” (0.72), ”w” (0.76), ”s” (0.49) and ”z” (0.45) show lower diagonal values, indicating poorer model performance for them. The same case for the expressions like the class ”me” was predicted sometimes like ”hello”.

Some changes in the training and testing accuracy can also be spotted, as it's shown in **Figure 3.36** , there's a slight difference between both accuracies but when the number of trees reaches 250, the testing accuracy becomes a bit greater than the training accuracy and that means the model has well generalized predictions on unseen data.

Interpreting the **Figure 3.37** , The oob score was 0.50 and it decreased to 0.15 when the number of trees became 2000 which means the model misclassified 15% of the training data , there were an increasing this time comparing to the first try as it was only 0.4%.

it can observed that this time , the increasing of number of trees had a big influence on model's accuracy , **The Figure 3.39 and Figure 3.40** both show how the impact of these estimators when we maintained the classifier's parameters and when they are changed afterwards , The more trees there are, the greater the accuracy.

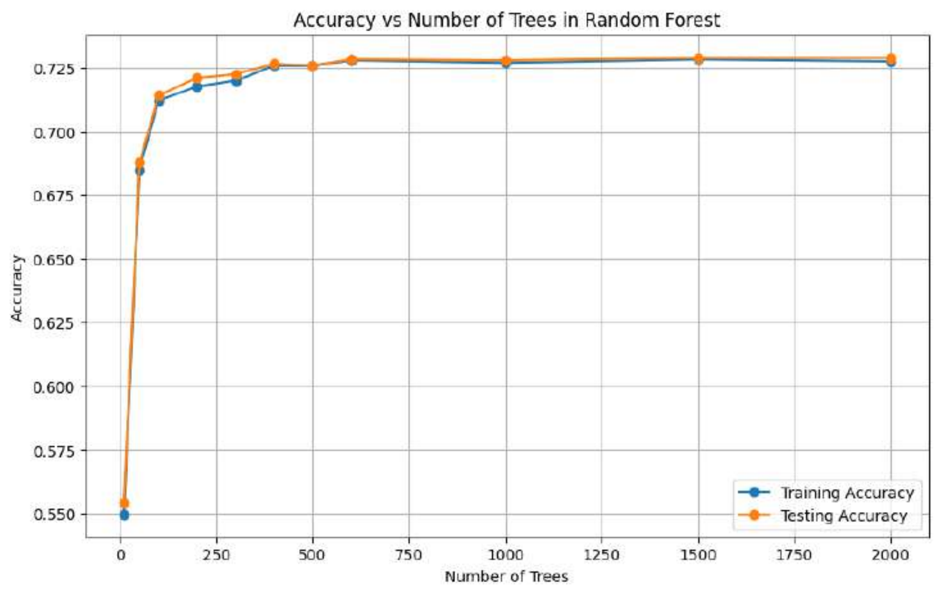


Figure 3.39: Testing and training accuracy with classifier's changed parameters

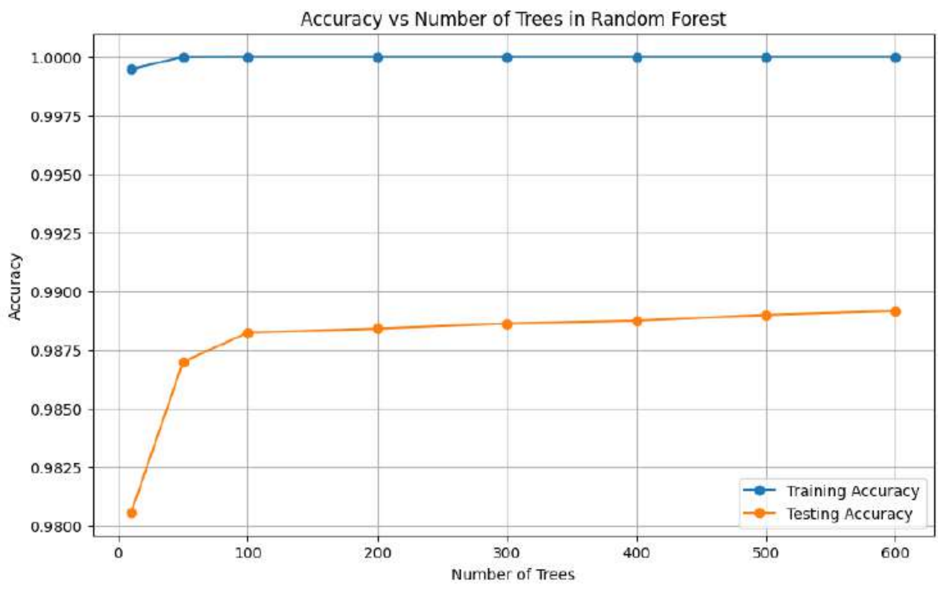


Figure 3.40: Testing and training accuracy with classifier's default parameters

To identify why some classes are misclassified, the similarity of their sign language gestures was examined. Specifically, the correlation matrix for classes 'A' and 'S' was plotted, as shown in **Figure 3.42** and **Figure 3.44**, since these letters have gestures that are somewhat similar in sign language.

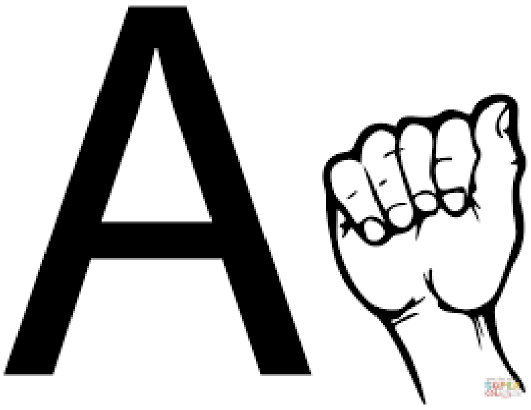


Figure 3.41: letter A in sign language

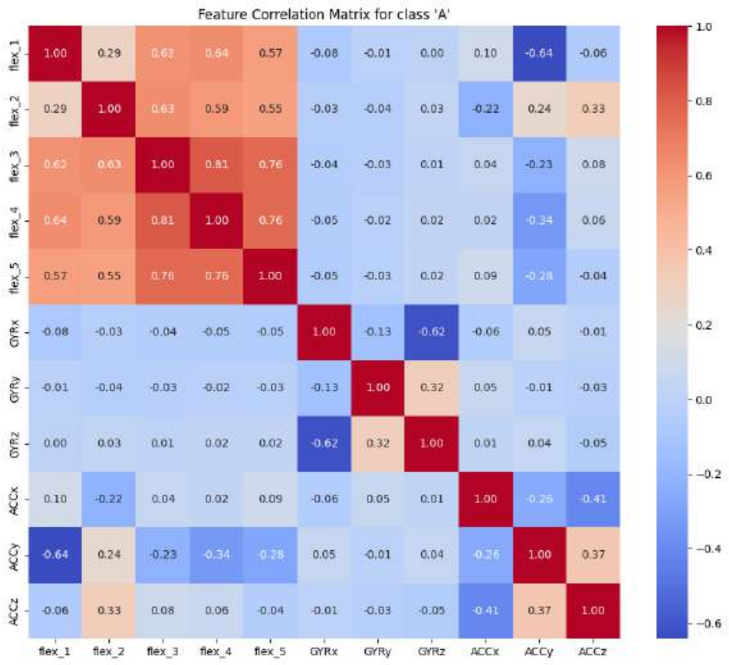


Figure 3.42: correlation plot of the letter A

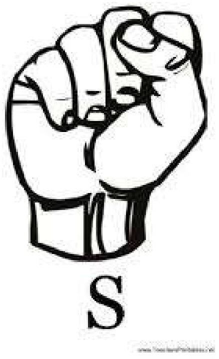


Figure 3.43: letter S in sign language

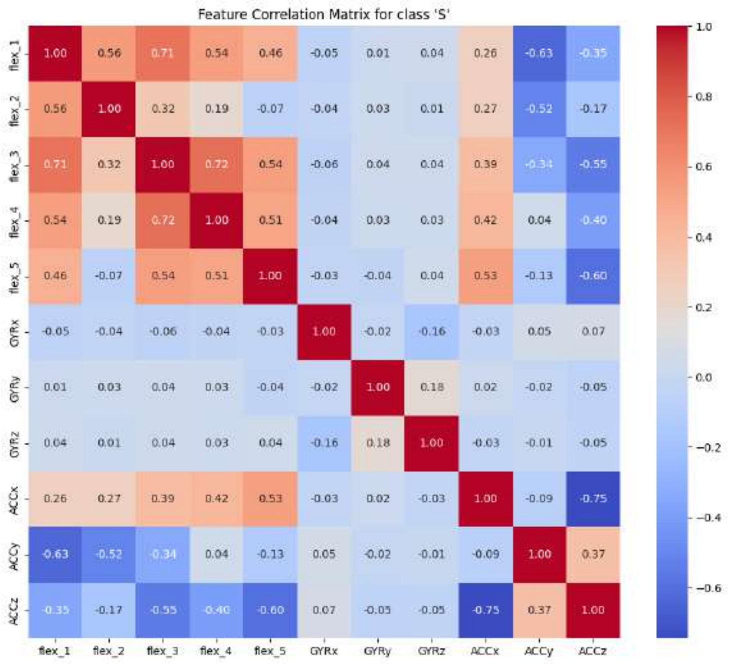


Figure 3.44: correlation plot of the letter S

Interpretation:

The correlation coefficient in class ’A’ between ’flex 1’ and ’flex 3’ is 0.62, suggesting a moderately positive linear association. This indicates that ’flex 3’ tends to increase in value in tandem with ’flex 1”s increasing value. The correlation coefficient in class ’S’ between ’flex 1’ and ’flex 3’ is 0.72, suggesting a more robust positive linear link in contrast to class ’A’. This indicates that in the sign language gesture for the letter ’S’ as opposed to the letter ’A,’ the values of ’flex 1’ and ’flex 3’ are either more closely related to each other or have a bigger influence on each other.

In conclusion , it can be concluded that the changing of the classifier’s hyperparameters had a positive influence on the model’s performances whereas the model struggles to differentiate between the classes that has somewhat the same gestures.

To get deeper into the details of how the algorithm work , we intended to plot The decisions trees for each model, we choose to plot one tree from the first model with 99% accuracy and the second model with 82% accuracy as shown in The **Figure 3.45** and **Figure 3.46**

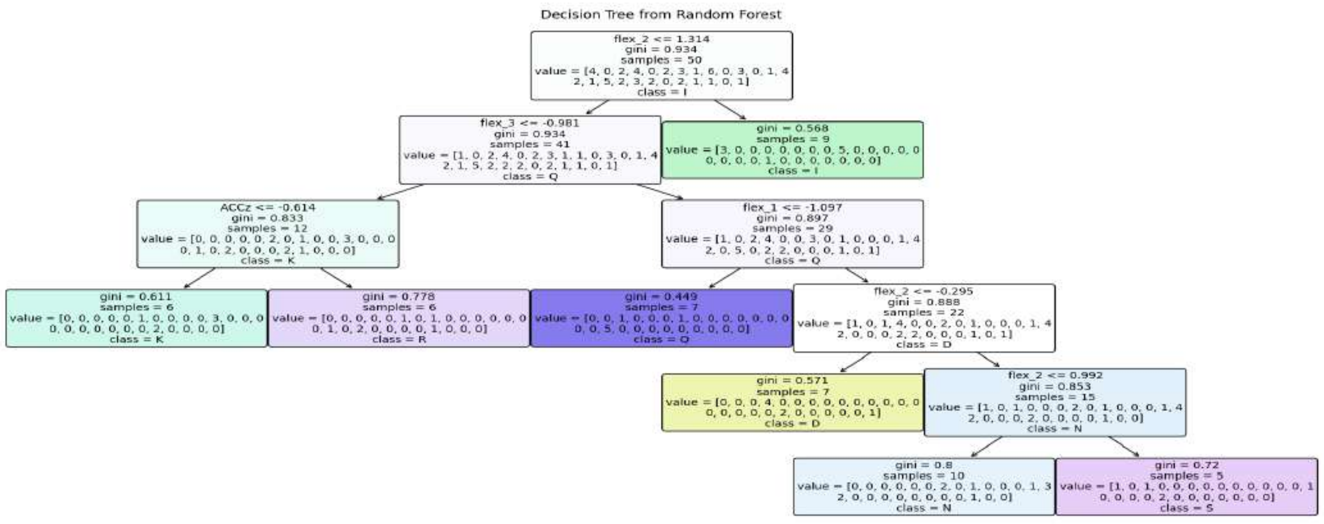


Figure 3.45: Decision Tree plot (82% accuracy)

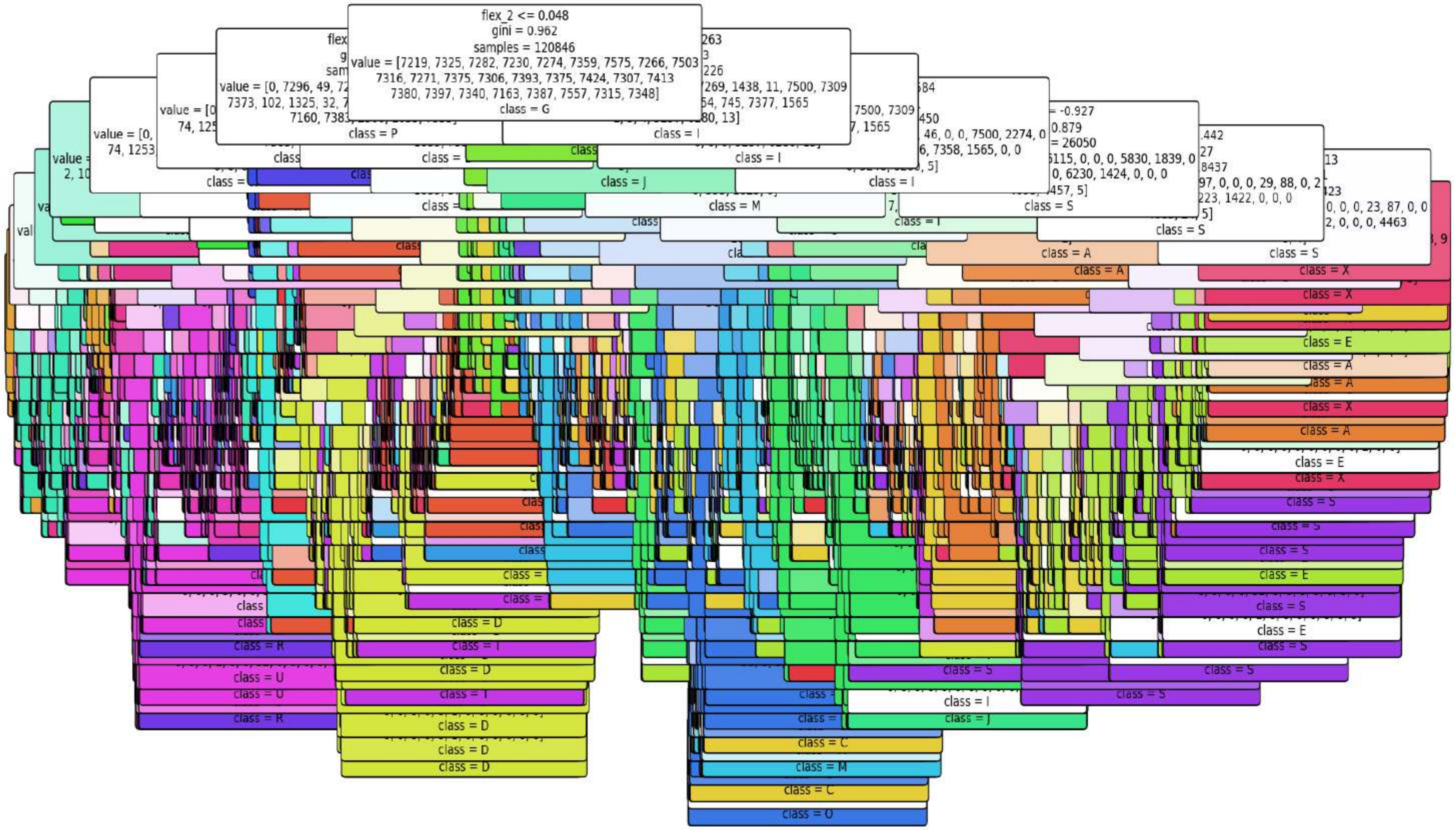


Figure 3.46: Decision Tree plot (99% accuracy

For the **Figure 3.45** , it represents a decision tree from the the second model , As seen , the algorithm starts growing the tree with its nodes and leaves depending on the parameter ”max_depth” which we set it as 5 and that means the tree splits 5 times from the root node to the leaf node.

The root node starts with a decision based on the feature flex_2 , the value 1.314 is the sensor’s threshold. Each node contains:

1. The feature and threshold for the split.
2. The Gini impurity of the node.
3. The number of samples at the node.
4. The distribution of classes at the node (value).
5. The predicted class (class).

After splitting , the algorithm calculates the number of classes samples in each leaf and make preidictions based on the class that has the maxiumum number of samples. It everytime tries the to decrease the impurity until the leaf becomes pure , in our case , it stops when it reaches the number of trees depth set before.

For the **Figure 3.46** , it’s the same thing but this time , it’s remarkable that the tree has many leaves and that’s because the value of ”max_depth” was set to ”None” which means the tree will keep growing leaves until they contain a minimum number of samples which lead most of the time to overfitting[24] and that explains the high accuracy of the model (99%).

3.7.4 Classification report

	precision	recall	f1-score	support
you	0.56	0.29	0.39	3189.00
hungry	0.78	0.91	0.84	3190.00
hello	0.49	0.89	0.63	3112.00
me	0.73	0.47	0.57	3207.00
please	0.76	0.73	0.75	3179.00
fine	0.80	0.90	0.85	3144.00
thankyou	0.75	0.57	0.65	3178.00
goodbye	0.98	0.98	0.98	3102.00
sorry	0.60	0.92	0.73	3152.00
yes	0.92	0.88	0.90	3067.00
good	0.57	0.57	0.57	3205.00
no	0.95	0.91	0.93	3207.00
deaf	0.82	0.48	0.60	3018.00
accuracy	0.73	0.73	0.73	0.73
macro avg	0.75	0.73	0.72	40950.00
weighted avg	0.75	0.73	0.72	40950.00

Table 3.2: sign language expressions Classification report

	precision	recall	f1-score	support
A	0.74	0.89	0.81	3204.0
B	0.99	0.98	0.99	3155.0
C	0.82	0.70	0.75	3227.0
D	0.66	0.97	0.78	3170.0
E	0.63	0.83	0.71	3166.0
F	0.99	1.00	0.99	3122.0
G	0.86	1.00	0.93	3102.0
H	1.00	1.00	1.00	3163.0
I	0.76	0.99	0.86	3068.0
J	1.00	0.67	0.80	3183.0
K	0.73	0.79	0.76	3174.0
L	0.83	0.98	0.90	3072.0
M	0.87	0.93	0.90	3141.0
N	0.86	0.82	0.84	3013.0
O	0.79	0.74	0.76	3182.0
P	0.99	1.00	0.99	3172.0
Q	0.85	1.00	0.92	3176.0
R	0.86	0.72	0.78	3164.0
S	0.84	0.56	0.67	3198.0
T	0.96	0.57	0.71	3140.0
U	0.75	0.86	0.80	3161.0
V	0.81	0.79	0.80	3234.0
W	0.98	1.00	0.99	3157.0
X	0.87	0.76	0.81	3070.0
Y	0.92	1.00	0.96	3142.0
Z	0.99	0.43	0.60	3144.0
accuracy	0.84	0.84	0.84	0.84
macro avg	0.86	0.84	0.83	81900.0
weighted avg	0.86	0.84	0.84	81900.0

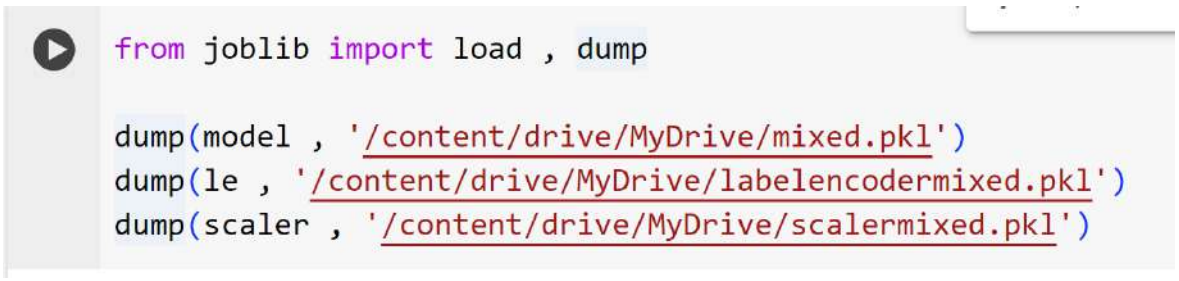
Table 3.3: sign language alphabet Classification report

Interpretation : The support values in the tables 3.2 and 3.3 represent the number of samples for each class, and the slight equality among all classes can be determined due to the same number of entries trained for each class ([38]).

3.8 Model Deployment

The step that comes after the model's training is deployment which is the process of integrating the model into a production environment and that's done in two steps :

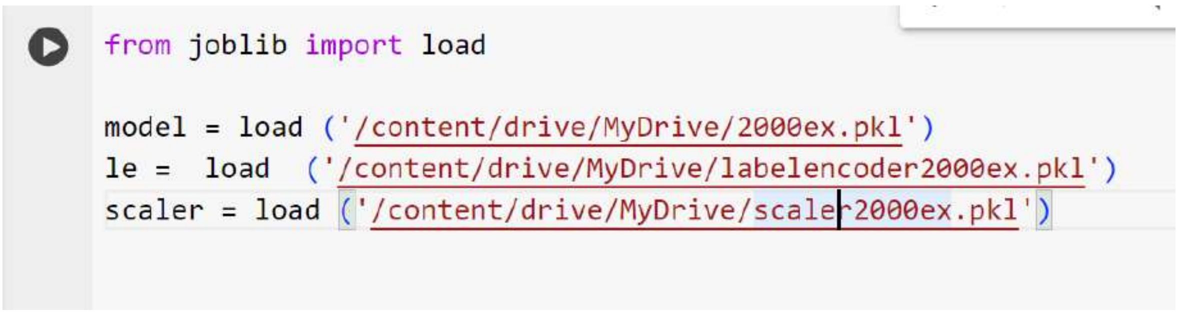
1. Dumping the model which is saving it in a specific extension , in our case , the files are saved as .pkl files that are shown in the **Figure 3.47** below :



```
from joblib import load , dump

dump(model , '/content/drive/MyDrive/mixed.pkl')
dump(le , '/content/drive/MyDrive/labelencodermixed.pkl')
dump(scaler , '/content/drive/MyDrive/scalermixed.pkl')
```

Figure 3.47: Python Code for Dumping machine learning model

2. Loading the model which consists of installing all the necessary libraries in the working environment to predict all the classes and to load the model , all the files dumped previously need to be loaded as shown in **Figure 3.48**



```
from joblib import load

model = load ('/content/drive/MyDrive/2000ex.pkl')
le =  load  ('/content/drive/MyDrive/labelencoder2000ex.pkl')
scaler = load ('/content/drive/MyDrive/scaler2000ex.pkl')
```

Figure 3.48: Python Code for loading machine learning model

3.9 Conclusion

In that chapter, we gave details about the results of training the machine learning model and calibrating the sensors. We also talked about the software used for that as it is crucial to adjusting them to accurately record hand movements and gestures. This precision was important too for the machine learning model to interpret the data correctly and give immediate feedback to the user.

General Conclusion

This report has elaborated a project that consists of making a smart glove that translates ASL to text and speech , that device is a bridge that connects the deaf community with the whole society , as it makes it easier for them to communicate freely and easier too for the other categories to communicate without the need of learning the sign language.Despite this , certain limitations exist, one of them is the difficulty to differentiate between alphabet or expressions that have somewhat the same gesture.

The first attempt to realise the project was trying I2C communication between the smart glove and the raspberry pi 4 as the first one is equipped with arduino nano , breadboard and five flex sensors , and the mpu6050 is connected directly to the raspberry pi 4. As a part of improvements , we attempted to optimise it and making the connection wiressly using esp32 instead of arduino nano.

In addition, we faced some challenges when choosing the electronic components , as some of the parts weren't available hence , it was difficult to get them. **Future improvements** The smart glove development won't end up here, as we are dedicated to look up further improvements such as :

1. Implementing another machine or deep learning models for algerian sign language classification trained with a dataset that should be collected by us.

2. Improving the real time accuracy.

3. Developing a mobile application to make it easier for the deaf community to interact within the society.

The dataset used for training and testing the model can be accessed at `https://figshare.com/articles/dataset/ASL-Sensor-Dataglove-Dataset_zip/20031017`.

Bibliography

[1] R Jayanthi et al. “Empowering Independence: A Smart Glove for Gesture-Based Navigation of Physically Challenged Individuals with Arduino Technology”. en. In: 05.05 (2023).

[2] Muhammad Rifki Kurniawan, Ahmad Hasibul Hadi, and Siska Novitasari. “A Smart Glove for Sign Language Translation”. In: *Proceedings of the 2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. IEEE. 2017, pp. 303–308.

[3] K. P. Sruthi et al. “Smart Glove with Sensor Integration for the Deaf and Dumb”. In: *Journal of Emerging Technologies and Innovative Research* 6.4 (2019), pp. 589–594.

[4] Neven Saleh et al. “Smart glove-based gestures recognition system for Arabic sign language”. en. In: *2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE)*. Aswan, Egypt: IEEE, Feb. 2020, pp. 303–307. ISBN: 978-1-72814-801-4. DOI: `10.1109/ITCE48509.2020.9047820`. URL: `https://ieeexplore.ieee.org/document/9047820/`.

[5] Sawant Pramada. “Intelligent Sign Language Recognition Using Image Processing”. en. In: *IOSR Journal of Engineering* 03.02 (Feb. 2013), pp. 45–51. ISSN: 22788719, 22503021. DOI: `10.9790/3021-03224551`.

[6] About Her. *8 Things To Know About Hadeel Ayoub, Saudi Arabian Inventor of The Smart Glove*. Accessed: 2024-06-13. 2019. URL: `https://www.abouther.com/node/21376/people/leading-ladies/8-things-know-about-hadeel-ayoub-saudi-arabian-inventor-smart-glove`.

[7] V7 Labs. “Supervised vs. Unsupervised Learning [Differences & Examples]”. In: (2023). URL: `https://www.v7labs.com/blog/supervised-vs-unsupervised-learning`.

[8] DatabaseTown. “Unsupervised Learning: Types, Applications & Advantages”. In: (2023). URL: `https://databasetown.com/unsupervised-learning-types-applications/`.

[9] IBM. *What is a Neural Network?* `https://www.ibm.com/topics/neural-networks`. 2023.

[10] GeeksforGeeks. "Support Vector Machine Algorithm". In: (2023). URL: https://www.geeksforgeeks.org/support-vector-machine-algorithm/.

[11] Leo Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[12] *Raspberry Pi 4 Model B Specifications.* https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. Accessed: Insert date accessed.

[13] Espressif Systems. *ESP32 Microcontroller Description.* Internal Documentation. Accessed: Insert date accessed. Year of Publication.

[14] SparkFun Electronics. *Flex Sensor Special Edition Datasheet.* https://cdn.sparkfun.com/assets/9/5/b/f/7/FLEX_SENSOR_-_SPECIAL_EDITION_DATA_SHEET_v2019__Rev_A_.pdf. Accessed: Insert date accessed.

[15] *MPU-6050.* Accessed: June 19, 2024. TDK InvenSense. URL: https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/%5C#:%5C~:text=The%5C%20MPU-D6050%5C%20devices%5C%20combine.

[16] James Montantes. *3 reasons to use random forest over a neural network–comparing machine learning versus deep...* Apr. 2024. URL: https://towardsdatascience.com/3-reasons-to-use-random-forest-over-a-neural-network-comparing-machine-learning-versus-deep-f9d65a154d89.

[17] E R Sruthi. *Random Forest — Introduction to Random Forest Algorithm.* June 2021. URL: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/.

[18] Sruthi E. R. *Understand Random Forest Algorithm With Examples (Updated 2024).* June 2021. URL: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#:~:text=Random%20forest%20is%20highly%20complex.

[19] Adam Pantanowitz and Tshilidzi Marwala. "Missing Data Imputation Through the Use of the Random Forest Algorithm". en. In: *Advances in Computational Intelligence.* Ed. by Wen Yu and Edgar N. Sanchez. Vol. 116. Advances in Intelligent and Soft Computing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 53–62. ISBN: 978-3-642-03155-7. DOI: 10.1007/978-3-642-03156-4_6. URL: http://link.springer.com/10.1007/978-3-642-03156-4_6.

[20] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. "Why do tree-based models still outperform deep learning on typical tabular data?" en. In: ().

[21] Last Minute Engineers. *In-Depth: Interfacing Flex Sensor with Arduino.* https://lastminuteengineers.com/flex-sensor-arduino-tutorial/#google_vignette. Apr. 2020.

[22] Jobit Joseph. *How Does the MPU6050 Accelerometer & Gyroscope Sensor Work and Interfacing It With Arduino.* May 2022. URL: https://circuitdigest.com/microcontroller-projects/interfacing-mpu6050-module-with-arduino.

[23] *TCP/IP server on an ESP32.* 2023. URL: https://forum.arduino.cc/t/tc-ip-server-on-an-esp32/1140930/4.

[24] GeeksforGeeks. *Hyperparameters of Random Forest Classifier.* https://www.geeksforgeeks.org/hyperparameters-of-random-forest-classifier/. Jan. 2021.

[25] Navnina Bhatia. *What is Out of Bag (OOB) score in Random Forest?* June 2019. URL: https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710.

[26] Noam Bressler. *How to Check the Accuracy of Your Machine Learning Model.* Nov. 2021. URL: https://deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model/.

[27] Google Developers. *Classification: Precision and Recall — Machine Learning Crash Course — Google Developers.* https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall. Mar. 2019.

[28] Google. *Classification: ROC Curve and AUC — Machine Learning Crash Course.* 2019. URL: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc.

[29] Rohit Kundu. *Confusion Matrix: How To Use It & Interpret Results [Examples].* Sept. 2022. URL: https://www.v7labs.com/blog/confusion-matrix-guide.

[30] scikit-learn. $sklearn.metrics.f1_score|scikit-learn0.21.2documentation$. 2019. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html.

[31] Sanskar Wagavkar. *Introduction to The Correlation Matrix.* Mar. 2023. URL: https://builtin.com/data-science/correlation-matrix#:~:text=Shutterstock%20%2F%20Built%20In-.

[32] GeeksforGeeks. *Create a correlation matrix using Python.* https://www.geeksforgeeks.org/create-a-correlation-matrix-using-python/. Nov. 2020.

[33] Md. Ahasan Atick Faisal. "DU-ASL-DATA-GLOVE-DB". In: (June 2022). DOI: 10.6084/m9.figshare.20031017.v2. URL: https://figshare.com/articles/dataset/ASL-Sensor-Dataglove-Dataset_zip/20031017.

[34] URL: https://colab.google/.

[35] *Raspberry Pi Documentation - Getting Started.* https://www.raspberrypi.com/documentation/computers/getting-started.html.

[36] RealVNC® Ltd. *All you need to know about VNC remote access technology.* URL: https://discover.realvnc.com/what-is-vnc-remote-access-technology#:~:text=VNC%20Viewer%20is%20used%20for.

[37] Bittele Electronics. *How Important are Solder Masks and Silkscreens.* https://www.7pcb.com/blog/how-important-are-solder-masks-silkscreens.

[38] The scikit-developers. *Classification Report — Yellowbrick v1.5 documentation.* URL: https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html#:~:text=Support%20is%20the%20number%20of.