**Amrita Vishwa Vidyapeetham**
**Amrita School Of Engineering, Bangaluru**
**Department of Electronics and Communication Engineering**
**19EAC386: Open Lab**

**Title of the Project**

# Study of Atrial Fibrillation signals in ECG and classifying them using Machine learning algorithms

*Submitted by*

| **Reg. Number** | **Name** |
|---|---|
| BL.EN.U4EAC20052 | Minal Mohrir |
| BL.EN.U4EAC20075 | Somiya Agrawal |
| BL.EN.U4EAC20084 | Vaishnavi P |

**Faculty in charge**

**Swaminadhan R.**

**Examiner 1** **Examiner 2**

# Table of Contents

# Chapter 1

# Introduction

## 1.1 Overall Background and System

**What is the application?**

ECG signals (electrocardiogram signals) are commonly used in the diagnosis and monitoring of various cardiac conditions, including atrial fibrillation (AF). Atrial fibrillation is an irregular and rapid heart rhythm that can increase the risk of stroke and other cardiovascular complications.

ECG signals play a crucial role in diagnosing atrial fibrillation. ECG recordings can detect the characteristic irregular and chaotic electrical activity in the atria, which is indicative of AF. Doctors often use ECG tracings to confirm the presence of AF and differentiate it from other arrhythmias.

ECG signals can also be used to identify individuals at risk of atrial fibrillation, especially those who may be asymptomatic. Screening programs often involve the use of portable ECG devices or wearable heart rate monitors that can detect irregular heart rhythms and alert individuals to seek further medical evaluation.

ECG signals are utilized to assess the effectiveness of treatment interventions for atrial fibrillation. By comparing ECG tracings before and after treatment, healthcare professionals can evaluate the impact of medications, cardioversion (restoring normal rhythm through electrical shocks), or catheter ablation procedures on the patient's heart rhythm.

**Why is it important?**

ECG signals play a crucial role in using machine learning (ML) algorithms for the classification of atrial fibrillation (AF).

ECG signals provide a wealth of information about the electrical activity of the heart. The ECG waveform consists of distinct components, such as P-waves, QRS complexes, and T-waves, which can reveal abnormalities associated with AF. ML algorithms can learn patterns and features from these signals to accurately classify AF.

ML algorithms can leverage ECG signals obtained from continuous monitoring devices, such as wearable devices or implantable cardiac monitors. Continuous monitoring provides a more comprehensive assessment of a patient's heart rhythm over an extended period, allowing ML models to capture transient or intermittent AF episodes that may be missed during short-term ECG recordings.

ECG is a non-invasive and widely accessible diagnostic tool. ECG recordings can be easily obtained using electrodes placed on the skin, making it a convenient and cost-effective method for AF classification. ML models trained on ECG data can be applied to large populations, facilitating screening and early detection of AF.

ML algorithms can extract a wide range of features from ECG signals, such as heart rate variability, waveform morphology, and spectral characteristics. These features provide valuable information for AF classification. ML models can learn to identify specific patterns or combinations of features that are indicative of AF, enabling accurate classification.

**Where is it necessary?**

ECG signals collected remotely through wearable devices or telemedicine platforms can be used for AF classification using ML. Patients can record their ECGs at home, and ML algorithms can analyze these signals to detect and classify AF episodes. Remote monitoring coupled with ML-based AF classification enables continuous tracking of patients' heart rhythms and timely intervention when necessary.

ML models trained on ECG signals can assist in personalized treatment decisions for individuals with AF. By analyzing ECG features and other relevant patient data, ML algorithms can predict the response to different treatment options and help guide therapeutic decisions. Additionally, ML-based risk stratification models can assess the risk of stroke or other cardiovascular events in patients with AF, aiding in personalized management plans.

It's worth noting that ML-based AF classification should always be used as an adjunct tool to support clinical decision-making, rather than as a replacement for medical expertise. Healthcare professionals should interpret the results provided by ML algorithms in the context of the patient's clinical history, symptoms, and other relevant information.

## 1.2 Objectives

The objective of this project is to study Electrocardiogram (ECG) signals and their relationship to Atrial Fibrillation (AF). The main focus is on applying machine learning algorithms such as SVM, CNN, KNN, and Random Forest to accurately classify ECG signals as AF or non-AF. The project aims to understand ECG signals, preprocess the data, extract relevant features, develop and evaluate the classification models, compare their performance, explore interpretability techniques, and provide practical implications and future recommendations for AF diagnosis.

## 1.3 Methodology

- AD8232 ECG Sensor
- ESP Development Board
- Arduino SD Card Module
- General Printed Circuit Board
- MATLAB
- Python

## 1.4 Outcomes

In this project, we embarked on a comprehensive study of ECG signals and aimed to classify atrial fibrillation (AF) and normal heartbeats using various machine learning algorithms. We explored the intricacies of ECG signals and leveraged the power of machine learning techniques to accurately classify these signals.

First, we delved into the fundamentals of ECG signals, understanding their characteristics, and gaining insights into the different components of a typical ECG waveform. We also examined the significance of ECG signals in diagnosing cardiac conditions, particularly AF.

To tackle the classification task, we employed a range of machine learning algorithms, including Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Random Forest, and Convolutional Neural Networks (CNN). Each algorithm brings its own strengths and capabilities to the table, offering distinct approaches to handle the complexities of ECG signal classification.

The SVM algorithm was chosen for its ability to find an optimal hyperplane to separate AF and normal heartbeats in a high-dimensional feature space. KNN, on the other hand, utilized the concept of similarity to assign labels based on the majority vote of neighboring instances. Random Forests are known for their robustness and ability to handle high-dimensional data. By aggregating the results from multiple trees, this algorithm can provide accurate and stable classification outcomes. Lastly, the CNN algorithm, known for its effectiveness in processing sequential data, employed convolutional layers to automatically extract relevant features from the ECG signals for classification.

In this we learned which algorithm gives highest accuracy in classification of AF and normal heartbeats.

# Chapter 2

# Summary of State of the Art and Description of Tools

## 2.1 Summary

| Topic | Description |
|---|---|
| System Level Overview | The objective is to apply machine learning algorithms, including Support Vector Machines (SVM), Convolutional Neural Networks (CNN), K-Nearest Neighbors (KNN), and Naïve Bayes, to accurately classify ECG signals as AF or non-AF. |
|  | Manual Diagnosis, Lack of Efficiency, Complexity of ECG Signals, Limited Scalability, Subjectivity and Variability, Potential for Missed Diagnoses |
| State of the Art |  |
| ECG Signal Processing and Feature Extraction | Previous works of authors on feature extraction from ECG signals. |
| Support Vector Machines (SVM) for AF Classification | Studies have shown that SVM can effectively utilize a combination of ECG features, including morphological, statistical, spectral, and heart rate variability features, to accurately distinguish between AF and normal sinus rhythm. |
| Convolutional Neural Networks (CNN) for AF Classification | The hierarchical feature extraction, translation invariance, and automatic feature learning capabilities of CNNs make them highly suitable for capturing the complex patterns and variations present in ECG signals. |
| K-Nearest Neighbors (KNN) for AF Classification | KNN is a simple and effective algorithm that can be used to capture patterns and relationships between different classes, making it suitable for AF classification tasks where identifying distinguishing features is crucial. |
| Random Forest for AF Classification | Random Forest has been shown to be effective in AF classification tasks, and it can handle high-dimensional feature spaces and effectively capture complex relationships between features and class labels. |
| Comparison between the normal ECG and Atrial-Fibrillation ECG | The ECG for AF is portable and can be used in a variety of settings, making it a valuable tool for early detection of AF and other heart conditions. |

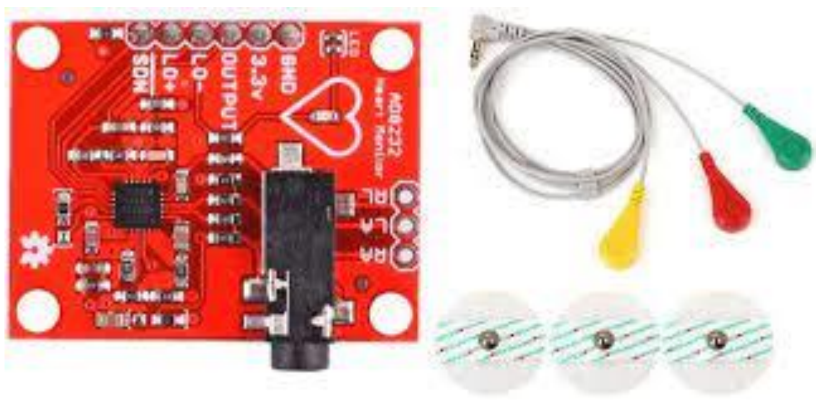
## 2.2 Description of Tools

### AD8232 ECG sensor

The AD8232 is an integrated ECG sensor module designed for measuring and processing electrocardiogram (ECG) signals. It is a single-lead ECG front-end module that provides amplification, filtering, and signal conditioning functions.

- Analog Front-End: The AD8232 includes an analog front-end circuitry responsible for capturing and amplifying the weak ECG signal from the electrodes. It employs a low-power, single-supply instrumentation amplifier with a high common-mode rejection ratio (CMRR) to reject common-mode interference and amplify the differential ECG signal. This helps in improving the signal quality and reducing noise and artifacts.
- Right Leg Drive (RLD): The RLD circuit is an important component of the AD8232. It provides a feedback mechanism to reduce common-mode noise, such as interference from mains power. The RLD electrode is typically connected to the right leg of the patient and helps to create a virtual ground for the ECG circuitry, improving the common-mode rejection performance.
- ECG Filtering: The AD8232 integrates a series of filters to process the ECG signal. This includes a high-pass filter to eliminate DC offsets and baseline wander, a low-pass filter to attenuate high-frequency noise, and a notch filter to remove interference caused by powerline frequencies (50 Hz or 60 Hz). These filters ensure that the ECG waveform is clean and free from unwanted noise and artifacts.
- Lead-off Detection: The AD8232 incorporates a lead-off detection feature to monitor the connection between the sensor and the electrode. It can detect if the electrode becomes disconnected or if the contact quality is poor, providing an indication of lead-off status. This feature is essential for ensuring data integrity and patient safety during ECG measurements.
- Output Data: The AD8232 provides both analog and digital output options. The analog output provides the amplified and filtered ECG signal in voltage form, suitable for direct connection to external data acquisition systems. The digital output is in the form of a serial peripheral interface (SPI) or an I2C interface, allowing easy integration with microcontrollers or digital signal processors (DSPs).
- Low Power Consumption: The AD8232 is designed to operate with low power consumption, making it suitable for portable and wearable applications. It has a shutdown mode that allows for further power savings when not actively acquiring ECG signals.
- Hardware and Software Integration: The AD8232 is typically used as part of a larger system. It can be connected to a microcontroller or other processing unit for further analysis and interpretation of the ECG signals. Analog Devices, the manufacturer of AD8232, provides software libraries, example code, and reference designs to facilitate the integration and development process.

  The AD8232 ECG sensor module offers a convenient and reliable solution for capturing and processing ECG signals. Its integration of amplification, filtering, lead-off detection, and output interfaces makes it a versatile component for a wide range of ECG monitoring applications, including wearable devices, patient monitoring systems, and healthcare research.



## ESP Development Board

An ESP development board is a versatile prototyping platform based on the ESP8266 or ESP32 microcontroller series developed by Espressif Systems. These boards provide a comprehensive environment for building and testing applications that leverage the Wi-Fi capabilities and other features of the ESP microcontrollers.

ESP development boards typically feature a compact design with various built-in components and interfaces. The core of the board is the ESP8266 or ESP32 microcontroller, which offers processing power and integrated Wi-Fi connectivity. The microcontroller's GPIO pins are exposed as headers, allowing for easy connection to external components and expansion of functionality. These pins can be used to interface with sensors, actuators, displays, and other modules, enabling a wide range of applications.

The boards often include a USB interface for power supply and programming purposes. This facilitates easy connectivity to a computer for programming the microcontroller and debugging applications. Additionally, the boards may feature onboard components such as LEDs, buttons, switches, and sensors for user interaction or basic system functionality.

ESP development boards are compatible with popular integrated development environments (IDEs) such as Arduino IDE, PlatformIO, and Espressif's own ESP-IDF and Arduino Core. These IDEs provide a familiar coding environment and offer libraries that simplify programming and accelerate development.

One of the key advantages of ESP development boards is their built-in Wi-Fi connectivity. The boards incorporate the necessary circuitry, antennas, and interfaces to enable wireless communication. This makes them ideal for developing applications in the Internet of Things (IoT) domain, allowing for seamless connectivity and remote control.

These boards often come with extensive documentation, tutorials, example projects, and a vibrant community of developers. The community support helps beginners get started quickly and provides resources for more advanced projects. Developers can seek assistance, share knowledge, and collaborate through forums and online communities dedicated to ESP development.
ESP development boards have gained popularity due to their low cost, versatility, and

robust ecosystem. They are widely used in hobbyist projects, educational settings, and even commercial product development. Their ease of use, rich features, and strong community support make them an attractive choice for those looking to prototype and deploy applications that require Wi-Fi connectivity and IoT capabilities.

## Arduino SD Card Module

The Arduino SD Card Module is an accessory module that allows Arduino microcontrollers to interface with Secure Digital (SD) or MultiMediaCard (MMC) memory cards. It provides a convenient way to store and retrieve data from external storage devices, enabling applications such as data logging, file storage, and data exchange.

The SD Card Module consists of a small PCB (printed circuit board) that typically incorporates an SD card slot, an SD card holder, and necessary supporting components. It connects to the Arduino board using digital input/output (I/O) pins and communicates with the microcontroller via the Serial Peripheral Interface (SPI) protocol.

The module's key component is the SD card slot, which accepts standard SD or MMC memory cards. The slot provides a secure and reliable connection for the memory card, ensuring proper data transfer and storage. Some modules may include a push-push type card holder, which allows for easy insertion and removal of the SD card.

To use the SD Card Module, the Arduino code must include the appropriate library for SD card communication. Libraries such as the "SD" library in the Arduino IDE provide a set of functions and commands to initialize the SD card, read from and write to files, and perform various file system operations.

The module interfaces with the Arduino microcontroller using the SPI bus. The SPI bus allows for high-speed communication between the microcontroller and the SD card module. It typically requires four signals: SCK (Serial Clock), MOSI (Master Out Slave In), MISO (Master In Slave Out), and SS (Slave Select). These signals facilitate data transfer and synchronization between the microcontroller and the SD card.

Using the SD card library, Arduino code can easily access the SD card's file system, create and open files, read from and write to files, and perform file operations such as seeking, renaming, and deleting. The library abstracts the low-level details of the SD card's communication protocol, making it simpler for developers to work with external storage devices.

The SD Card Module supports different SD card capacities, including standard SD cards (up to 2GB) and SDHC (Secure Digital High Capacity) cards (up to 32GB or more). However, it's important to ensure compatibility with the specific module and Arduino board being used, as some modules may have limitations on the card capacity or require additional modifications.

Overall, the Arduino SD Card Module provides an easy-to-use solution for adding external storage capabilities to Arduino projects. Its compatibility with various SD card formats and support for the SPI interface make it a versatile tool for data storage and retrieval. Whether it's logging sensor data, storing configuration files, or implementing data exchange, the SD Card Module expands the capabilities of Arduino-based systems.

## MATLAB

MATLAB, short for Matrix Laboratory, is a powerful and widely-used software platform for numerical computation, data analysis, and algorithm development. It provides an extensive collection of functions and tools that enable users to perform a wide range of tasks in scientific computing, engineering, and data science.

MATLAB is based on a high-level programming language that is designed to be intuitive and easy to use, with syntax and constructs that resemble mathematical notation. This makes it an accessible tool for users with various levels of programming experience. MATLAB's language includes features for matrix manipulation, data visualization, control flow, and modular programming, allowing for efficient and concise code development.

One of MATLAB's key strengths is its extensive library of built-in functions and toolboxes. These libraries provide a vast array of functionality, covering areas such as signal processing, image processing, optimization, statistics, machine learning, and more. The toolboxes offer specialized functions and algorithms tailored to specific application domains, enabling users to solve complex problems with ease.

MATLAB also offers powerful data visualization capabilities. It provides a range of plotting and graphing functions that allow users to create 2D and 3D visualizations of data, customize plots, and annotate figures. This makes it convenient for analyzing and presenting results in a visually appealing manner.

## Python

Python is a popular high-level programming language known for its simplicity, readability, and versatility. It provides a rich set of libraries and tools that make it suitable for a wide range of applications, including web development, data analysis, artificial intelligence, scientific computing, and more. Python's design philosophy emphasizes code readability, promoting clean and expressive syntax that is easy to understand and maintain.

One of Python's notable strengths is its extensive standard library, which offers a wide range of modules for performing common tasks. These modules provide functionalities such as file I/O, networking, regular expressions, threading, and more. Python's standard library is robust and comprehensive, reducing the need for developers to reinvent the wheel and enabling them to quickly develop applications.

## General Printed Circuit board

A printed circuit board (PCB) is a fundamental component in modern electronics that provides a platform for connecting and supporting various electronic components. It is a flat board made of non-conductive material, typically fiberglass-reinforced epoxy, with a layer of copper foil etched into a specific pattern to create a network of electrical connections.

The design of a PCB starts with a schematic, which represents the desired electrical connections and component placement. The schematic is then translated into a layout using specialized software tools. The layout defines the physical arrangement of components, copper traces, and other features on the PCB. It takes into account factors such as signal integrity, power distribution, thermal considerations, and manufacturability.

## Machine Learning

Machine learning is a field of artificial intelligence (AI) that focuses on the development of algorithms and models that enable computers to learn from and make predictions or decisions based on data, without being explicitly programmed. In other words, machine learning allows computers to automatically discover patterns, extract insights, and make predictions or decisions from data, without being explicitly programmed for each specific task.

Traditionally, computers were programmed using explicit instructions to perform specific tasks. Machine learning, on the other hand, takes a different approach. Instead of explicitly programming rules or instructions, machine learning algorithms learn from examples or data, and use that knowledge to make predictions or decisions on new, unseen data.

The learning process in machine learning typically involves three main components:

1. Training Data: Machine learning algorithms learn from a set of training data, which consists of input data (features) and corresponding output or target values. The training data is used to "train" the model by adjusting its internal parameters or structure.

2. Model Building: During the training phase, the machine learning algorithm builds a model based on the training data. The model captures patterns and relationships between the input data and the corresponding outputs. The specific algorithm used determines the structure and complexity of the model.

3. Inference or Prediction: Once the model is trained, it can be used to make predictions or decisions on new, unseen data. The trained model takes the input data and produces output predictions based on the patterns it has learned during training.

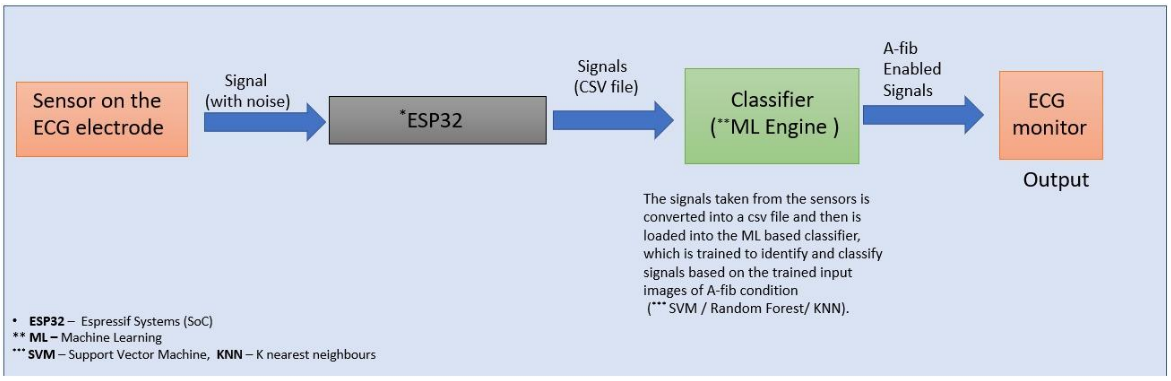
There are various types of machine learning algorithms, each suited for different types of tasks and data. These include supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Supervised learning involves training a model on labeled data, where the desired outputs are provided for each input instance. Unsupervised learning, on the other hand, deals with unlabeled data and focuses on finding patterns or structures in the data. Semi-supervised learning combines both labeled and unlabeled data. Reinforcement learning involves training an agent to interact with an environment and learn from feedback or rewards.

Machine learning has become increasingly important in a wide range of applications, including image and speech recognition, natural language processing, recommendation systems, fraud detection, autonomous vehicles, and many more. Its ability to automatically learn and adapt from data makes it a powerful tool for solving complex problems and extracting valuable insights from large and diverse datasets.

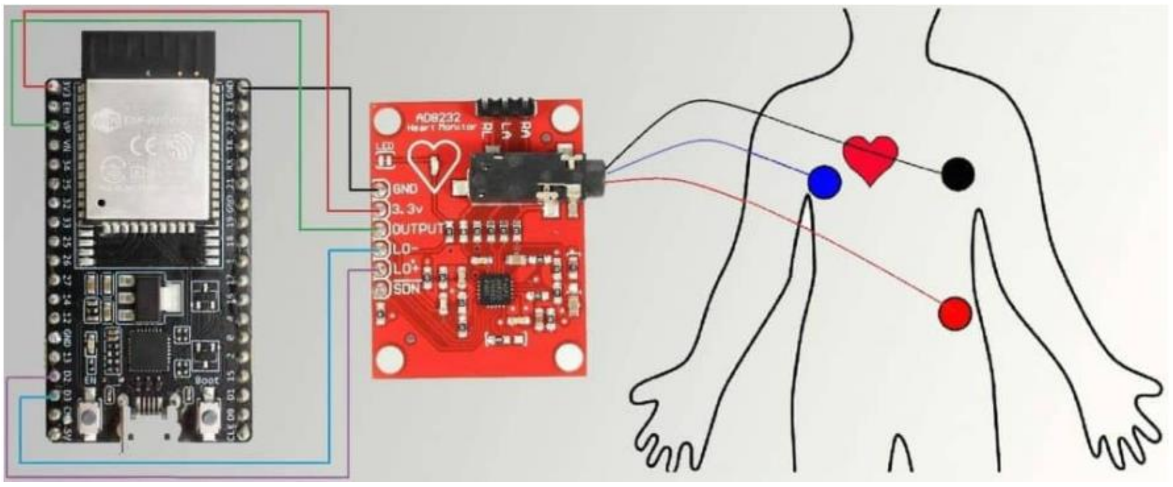
# Chapter 3

# Design and development

## Block diagram


Sensor on the ECG electrode
Signal (with noise)
*ESP32
Signals (CSV file)
Classifier (**ML Engine )
A-fib Enabled Signals
ECG monitor
Output
The signals taken from the sensors is converted into a csv file and then is loaded into the ML based classifier, which is trained to identify and classify signals based on the trained input images of A-fib condition (*** SVM / Random Forest/ KNN).
• ESP32 – Espressif Systems (SoC)
** ML – Machine Learning
*** SVM – Support Vector Machine, KNN – K nearest neighbours

## Circuit



## Connections

- Arduino SD card module is connected to ESP development board.

| ARDUINO SD CARD | ESP DEVELOPMENT BOARD |
|---|---|
| GND | GND |
| MOSO | D19 |
| MOSI | D23 |
| SCK | D18 |
| CS | D5 |

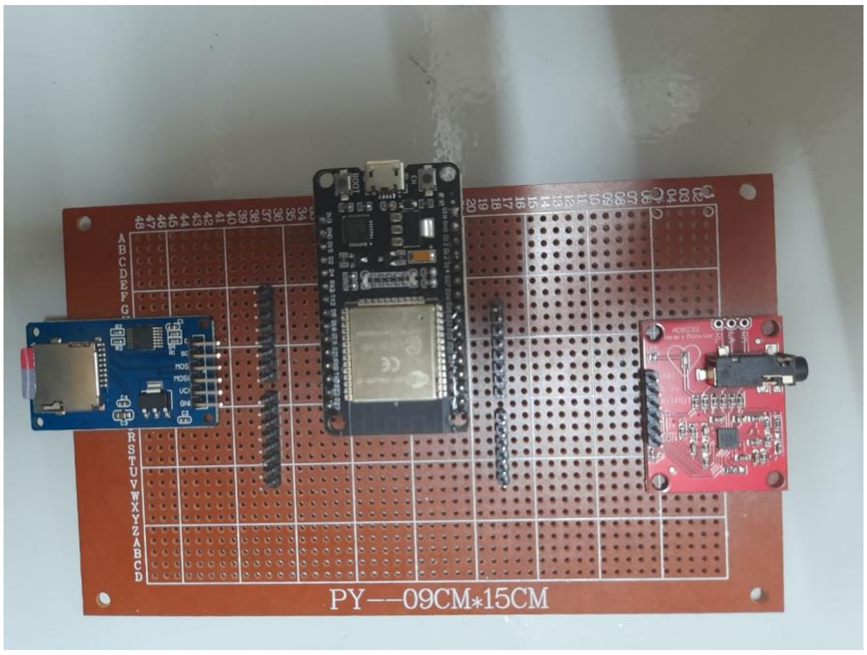
- Arduino SD card module to ECG sensor

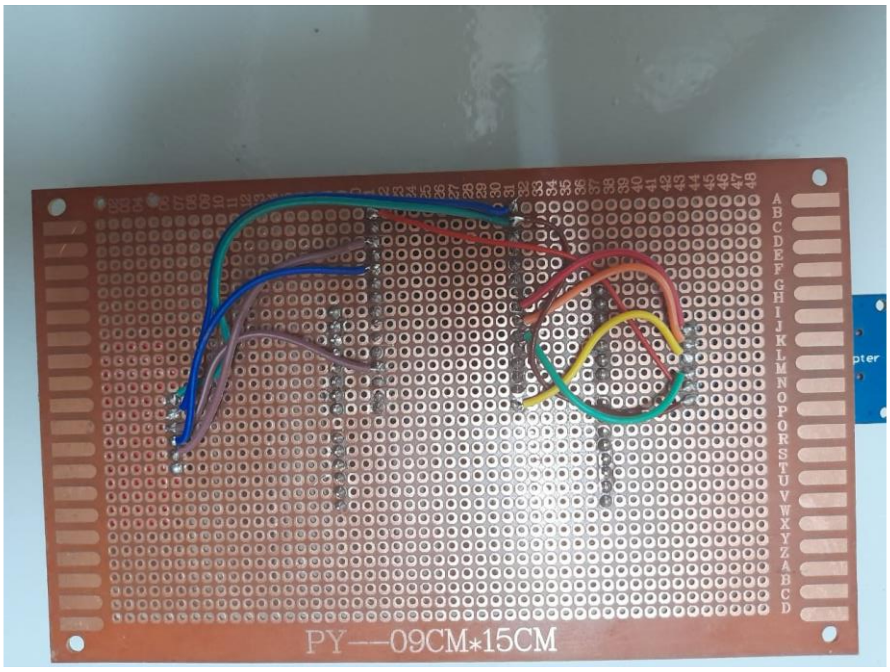
| ARDUINO SD CARD | ECG SENSOR |
|---|---|
| Vcc | Vin |

- ESP development board to ECG sensor

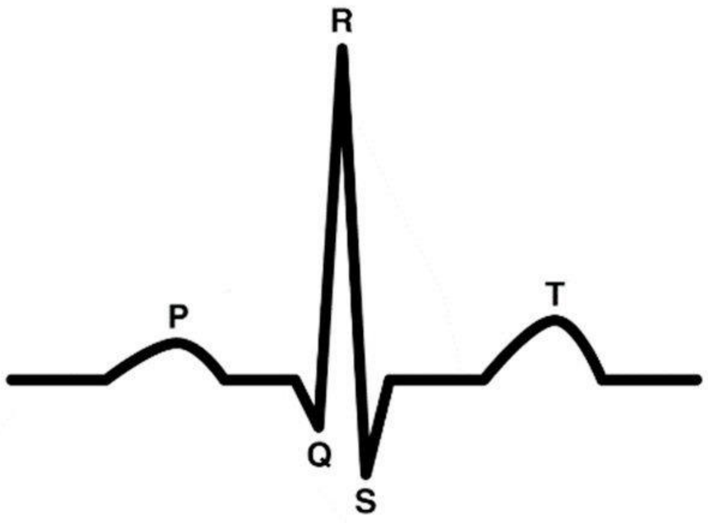
| ESP DEVELOPMENT BOARD | ECG SENSOR |
|---|---|
| GND | GND |
| 3V3 | 3.3V |
| D34 | OUTPUT |
| D14 | LO- |
| D13 | LO+ |

## Circuit





ECG (electrocardiogram) is a medical test that records the electrical activity of the heart. It is used to evaluate the heart's rhythm and detect any abnormalities or conditions related to the heart. The ECG waveform consists of a series of peaks, which represent different phases of the cardiac cycle.

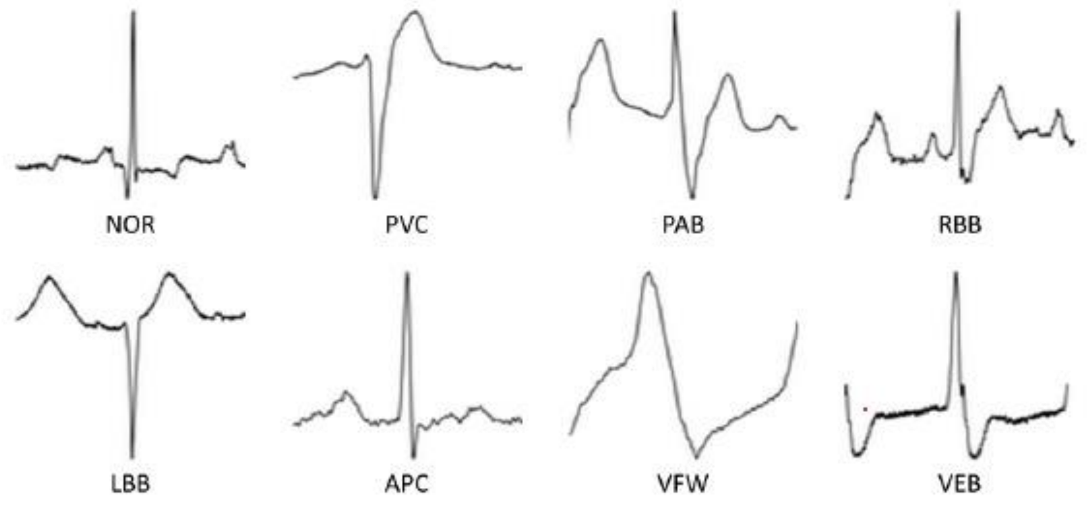


- P wave: In a normal sinus rhythm, each heartbeat is initiated by an electrical impulse originating from the sinus node, which causes atrial depolarization (contraction). However, in atrial fibrillation, the atrial electrical activity becomes disorganized and chaotic. As a result, individual P waves are typically absent or not discernible on the ECG. Instead, there may be fibrillatory or fine irregular oscillations or undulations present, referred to as fibrillatory waves or f waves. These f waves represent the atrial electrical activity during AF.

- QRS complex: The QRS complex represents ventricular depolarization (contraction). In atrial fibrillation, the irregular and rapid atrial electrical activity can result in an irregular ventricular response. As a consequence, the R-R intervals (the time between successive QRS complexes) vary irregularly, leading to an irregularly irregular ventricular rhythm. The QRS complexes themselves tend to be relatively normal in morphology, unless there are concomitant abnormalities such as premature ventricular contractions (PVCs) or bundle branch blocks.

- T wave: The T wave represents ventricular repolarization (recovery) following ventricular depolarization. In atrial fibrillation, the irregular and chaotic atrial electrical activity can also affect the ventricular repolarization process. As a result, T waves may appear normal, flattened, or exhibit increased variability in morphology, reflecting the irregular ventricular response associated with AF.

In atrial fibrillation, several cardiac electrical abnormalities can be observed on an ECG.



- NOR (Normal Sinus Rhythm): NOR refers to the normal electrical activity of the heart originating from the sinus node, which acts as the natural pacemaker. In NOR, the electrical impulses follow a regular pattern, resulting in a consistent and organized heartbeat.

- PVC (Premature Ventricular Contraction): PVC occurs when there is an early electrical impulse originating from the ventricles (lower chambers of the heart) before the next expected sinus impulse. It appears on the ECG as an abnormal QRS complex, representing the ventricular depolarization.

- PAB (Premature Atrial Beat): PAB is similar to PVC, but the early electrical impulse arises from the atria (upper chambers of the heart). It appears on the ECG as an abnormal P wave.

- RBB (Right Bundle Branch): The heart's electrical conduction system consists of specialized pathways called bundle branches that transmit electrical signals to the ventricles. RBB refers to a delay or blockage in the electrical conduction through the right bundle branch. On the ECG, RBB is characterized by a widened QRS complex.

- LBB (Left Bundle Branch): Similar to RBB, LBB refers to a delay or blockage in the electrical conduction through the left bundle branch. It is also indicated by a widened QRS complex on the ECG.

- APC (Atrial Premature Contraction): APC occurs when there is an early electrical impulse originating from the atria before the next expected sinus impulse, similar to PAB. It appears on the ECG as an abnormal P wave.

- VFW (Ventricular Flutter Wave): VFW indicates a rapid, regular, and organized electrical activity originating from the ventricles. It appears on the ECG as a series of F waves.

- VEB (Ventricular Escape Beat): VEB occurs when the ventricles fail to receive electrical impulses from the atria or the normal pacemaker. The ventricles then generate their own electrical impulses to maintain a heartbeat. On the ECG, VEB appears as a delayed and wide QRS complex.

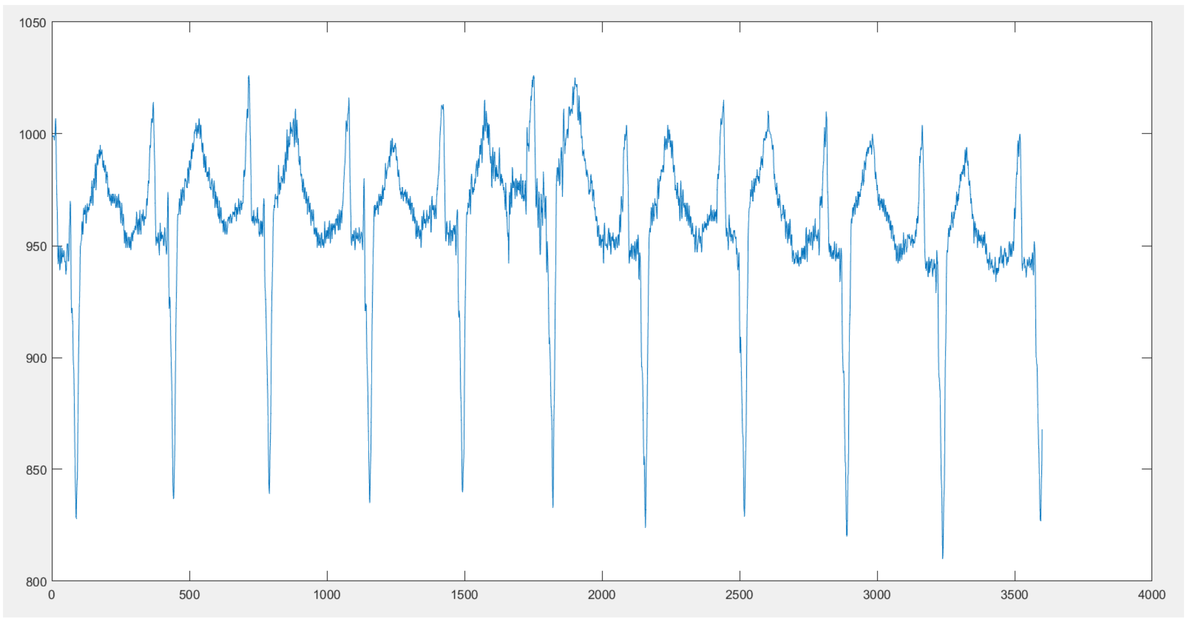
# Chapter 4

# Result

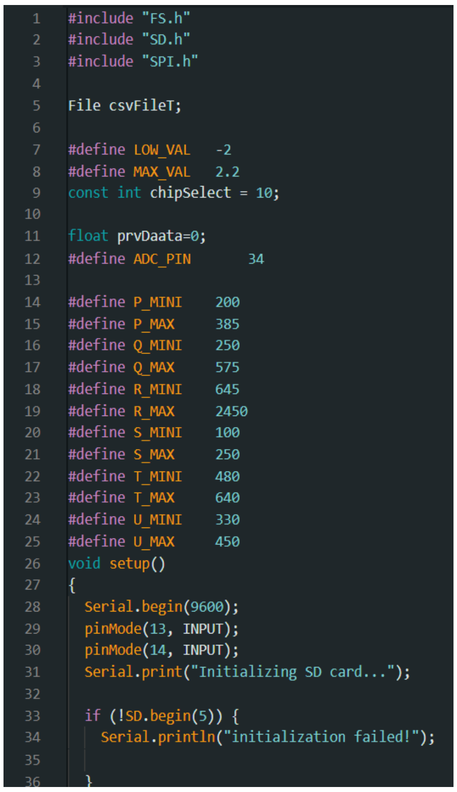
**MIT-BIH Arrhythmia Database**

The MIT-BIH Arrhythmia Database contains 48 half-hour excerpts of two-channel ambulatory ECG recordings, obtained from 47 subjects studied by the BIH Arrhythmia Laboratory between 1975 and 1979. Twenty-three recordings were chosen at random from a set of 4000 24-hour ambulatory ECG recordings collected from a mixed population of inpatients (about 60%) and outpatients (about 40%) at Boston's Beth Israel Hospital; the remaining 25 recordings were selected from the same set to include less common but clinically significant arrhythmias that would not be well-represented in a small random sample.

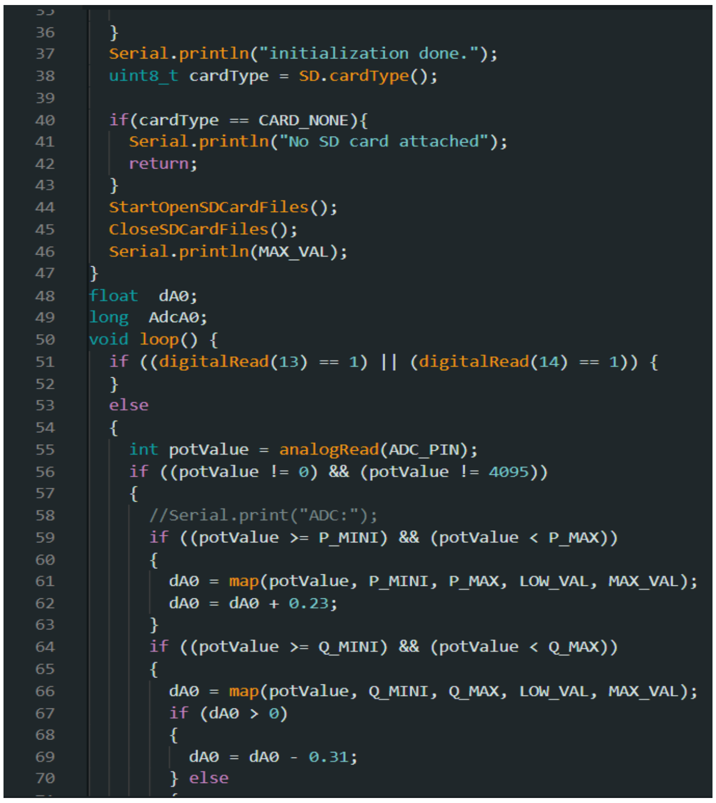
The recordings were digitized at 360 samples per second per channel with 11-bit resolution over a 10 mV range. Two or more cardiologists independently annotated each record; disagreements were resolved to obtain the computer-readable reference annotations for each beat (approximately 110,000 annotations in all) included with the database.



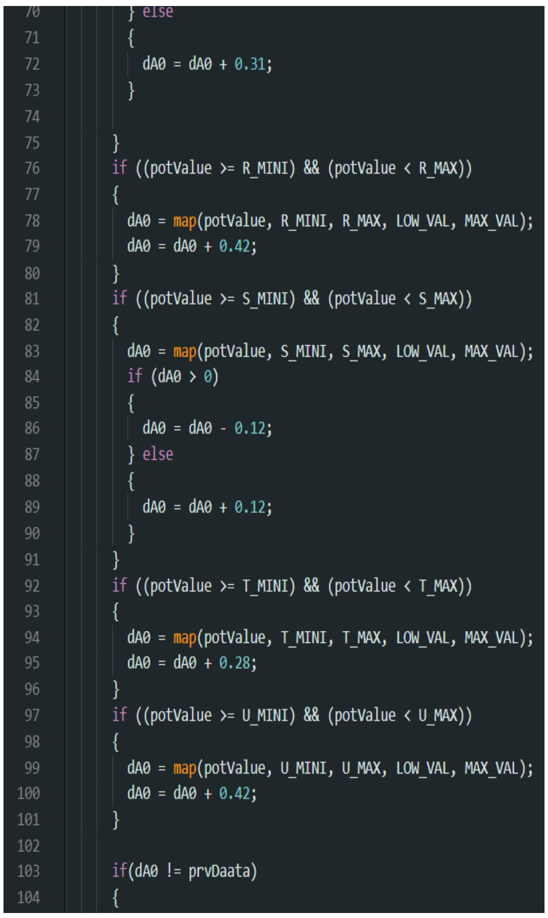
## Readings from ECG sensor



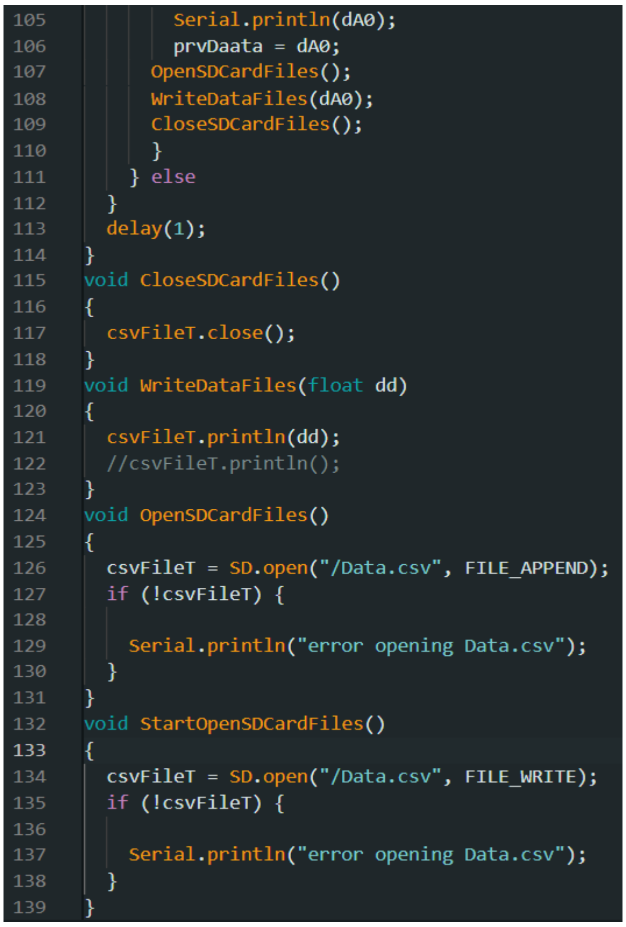
```
#include "FS.h"
#include "SD.h"
#include "SPI.h"

File csvFileT;

#define LOW_VAL   -2
#define MAX_VAL   2.2
const int chipSelect = 10;

float prvDaata=0;
#define ADC_PIN       34

#define P_MINI    200
#define P_MAX     385
#define Q_MINI    250
#define Q_MAX     575
#define R_MINI    645
#define R_MAX     2450
#define S_MINI    100
#define S_MAX     250
#define T_MINI    480
#define T_MAX     640
#define U_MINI    330
#define U_MAX     450
void setup()
{
  Serial.begin(9600);
  pinMode(13, INPUT);
  pinMode(14, INPUT);
  Serial.print("Initializing SD card...");

  if (!SD.begin(5)) {
    Serial.println("initialization failed!");

```



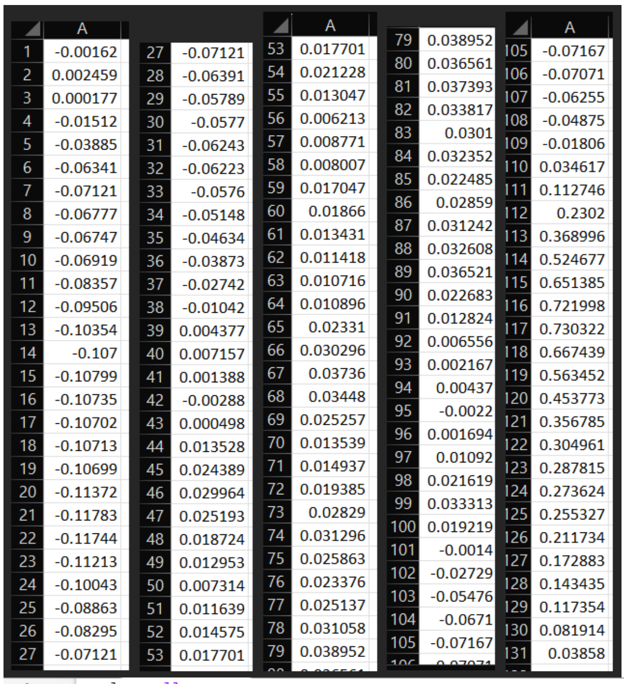
```
  }
  Serial.println("initialization done.");
  uint8_t cardType = SD.cardType();

  if(cardType == CARD_NONE){
    Serial.println("No SD card attached");
    return;
  }
  StartOpenSDCardFiles();
  CloseSDCardFiles();
  Serial.println(MAX_VAL);
}
float  dA0;
long  AdcA0;
void loop() {
  if ((digitalRead(13) == 1) || (digitalRead(14) == 1)) {
  }
  else
  {
    int potValue = analogRead(ADC_PIN);
    if ((potValue != 0) && (potValue != 4095))
    {
      //Serial.print("ADC:");
      if ((potValue >= P_MINI) && (potValue < P_MAX))
      {
        dA0 = map(potValue, P_MINI, P_MAX, LOW_VAL, MAX_VAL);
        dA0 = dA0 + 0.23;
      }
      if ((potValue >= Q_MINI) && (potValue < Q_MAX))
      {
        dA0 = map(potValue, Q_MINI, Q_MAX, LOW_VAL, MAX_VAL);
        if (dA0 > 0)
        {
          dA0 = dA0 - 0.31;
        } else
```



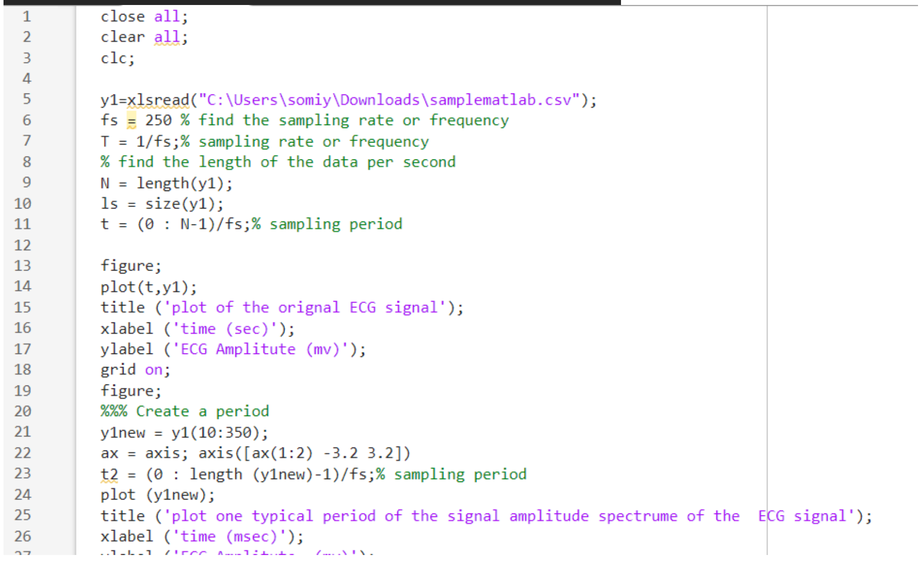
```
        {
          dA0 = dA0 + 0.31;
        }

      }
      if ((potValue >= R_MINI) && (potValue < R_MAX))
      {
        dA0 = map(potValue, R_MINI, R_MAX, LOW_VAL, MAX_VAL);
        dA0 = dA0 + 0.42;
      }
      if ((potValue >= S_MINI) && (potValue < S_MAX))
      {
        dA0 = map(potValue, S_MINI, S_MAX, LOW_VAL, MAX_VAL);
        if (dA0 > 0)
        {
          dA0 = dA0 - 0.12;
        } else
        {
          dA0 = dA0 + 0.12;
        }
      }
      if ((potValue >= T_MINI) && (potValue < T_MAX))
      {
        dA0 = map(potValue, T_MINI, T_MAX, LOW_VAL, MAX_VAL);
        dA0 = dA0 + 0.28;
      }
      if ((potValue >= U_MINI) && (potValue < U_MAX))
      {
        dA0 = map(potValue, U_MINI, U_MAX, LOW_VAL, MAX_VAL);
        dA0 = dA0 + 0.42;
      }

      if(dA0 != prvDaata)
      {
```



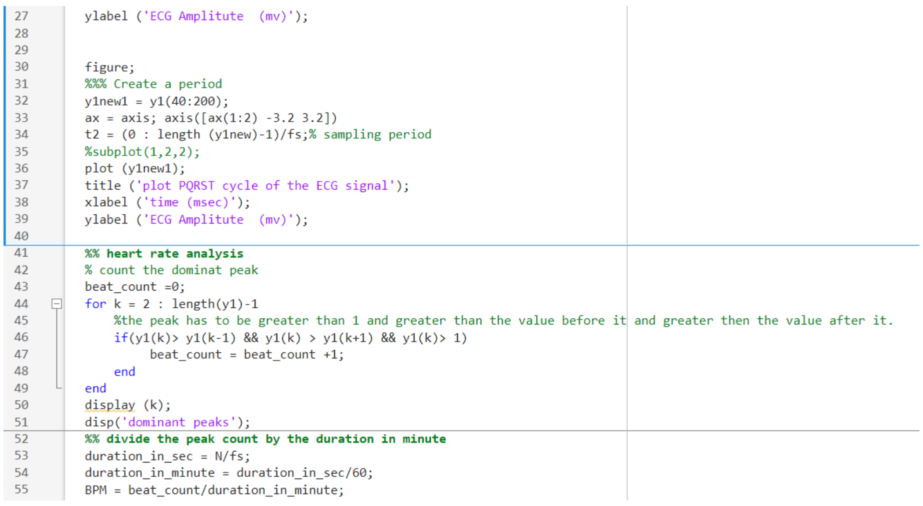
```
        Serial.println(dA0);
        prvDaata = dA0;
      OpenSDCardFiles();
      WriteDataFiles(dA0);
      CloseSDCardFiles();
      }
    } else
  }
  delay(1);
}
void CloseSDCardFiles()
{
  csvFileT.close();
}
void WriteDataFiles(float dd)
{
  csvFileT.println(dd);
  //csvFileT.println();
}
void OpenSDCardFiles()
{
  csvFileT = SD.open("/Data.csv", FILE_APPEND);
  if (!csvFileT) {

    Serial.println("error opening Data.csv");
  }
}
void StartOpenSDCardFiles()
{
  csvFileT = SD.open("/Data.csv", FILE_WRITE);
  if (!csvFileT) {

    Serial.println("error opening Data.csv");
  }
}
```



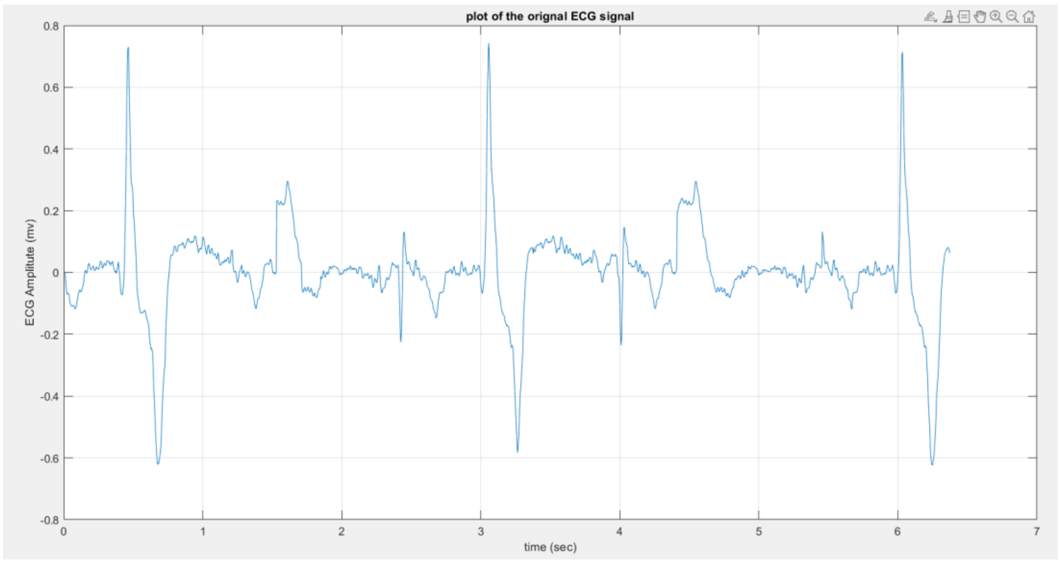
| | A |
|---|---|
| 1 | -0.00162 |
| 2 | 0.002459 |
| 3 | 0.000177 |
| 4 | -0.01512 |
| 5 | -0.03885 |
| 6 | -0.06341 |
| 7 | -0.07121 |
| 8 | -0.06777 |
| 9 | -0.06747 |
| 10 | -0.06919 |
| 11 | -0.08357 |
| 12 | -0.09506 |
| 13 | -0.10354 |
| 14 | -0.107 |
| 15 | -0.10799 |
| 16 | -0.10735 |
| 17 | -0.10702 |
| 18 | -0.10713 |
| 19 | -0.10699 |
| 20 | -0.11372 |
| 21 | -0.11783 |
| 22 | -0.11744 |
| 23 | -0.11213 |
| 24 | -0.10043 |
| 25 | -0.08863 |
| 26 | -0.08295 |
| 27 | -0.07121 |
| 28 | -0.06391 |
| 29 | -0.05789 |
| 30 | -0.0577 |
| 31 | -0.06243 |
| 32 | -0.06223 |
| 33 | -0.0576 |
| 34 | -0.05148 |
| 35 | -0.04634 |
| 36 | -0.03873 |
| 37 | -0.02742 |
| 38 | -0.01042 |
| 39 | 0.004377 |
| 40 | 0.007157 |
| 41 | 0.001388 |
| 42 | -0.00288 |
| 43 | 0.000498 |
| 44 | 0.013528 |
| 45 | 0.024389 |
| 46 | 0.029964 |
| 47 | 0.025193 |
| 48 | 0.018724 |
| 49 | 0.012953 |
| 50 | 0.007314 |
| 51 | 0.011639 |
| 52 | 0.014575 |
| 53 | 0.017701 |
| 54 | 0.021228 |
| 55 | 0.013047 |
| 56 | 0.006213 |
| 57 | 0.008771 |
| 58 | 0.008007 |
| 59 | 0.017047 |
| 60 | 0.01866 |
| 61 | 0.013431 |
| 62 | 0.011418 |
| 63 | 0.010716 |
| 64 | 0.010896 |
| 65 | 0.02331 |
| 66 | 0.030296 |
| 67 | 0.03736 |
| 68 | 0.03448 |
| 69 | 0.025257 |
| 70 | 0.013539 |
| 71 | 0.014937 |
| 72 | 0.019385 |
| 73 | 0.02829 |
| 74 | 0.031296 |
| 75 | 0.025863 |
| 76 | 0.023376 |
| 77 | 0.025137 |
| 78 | 0.031058 |
| 79 | 0.038952 |
| 80 | 0.036561 |
| 81 | 0.037393 |
| 82 | 0.033817 |
| 83 | 0.0301 |
| 84 | 0.032352 |
| 85 | 0.022485 |
| 86 | 0.02859 |
| 87 | 0.031242 |
| 88 | 0.032608 |
| 89 | 0.036521 |
| 90 | 0.022683 |
| 91 | 0.012824 |
| 92 | 0.006556 |
| 93 | 0.002167 |
| 94 | 0.00437 |
| 95 | -0.0022 |
| 96 | 0.001694 |
| 97 | 0.01092 |
| 98 | 0.021619 |
| 99 | 0.033313 |
| 100 | 0.019219 |
| 101 | -0.0014 |
| 102 | -0.02729 |
| 103 | -0.05476 |
| 104 | -0.0671 |
| 105 | -0.07167 |
| 106 | -0.07071 |
| 107 | -0.06255 |
| 108 | -0.04875 |
| 109 | -0.01806 |
| 110 | 0.034617 |
| 111 | 0.112746 |
| 112 | 0.2302 |
| 113 | 0.368996 |
| 114 | 0.524677 |
| 115 | 0.651385 |
| 116 | 0.721998 |
| 117 | 0.730322 |
| 118 | 0.667439 |
| 119 | 0.563452 |
| 120 | 0.453773 |
| 121 | 0.356785 |
| 122 | 0.304961 |
| 123 | 0.287815 |
| 124 | 0.273624 |
| 125 | 0.255327 |
| 126 | 0.211734 |
| 127 | 0.172883 |
| 128 | 0.143435 |
| 129 | 0.117354 |
| 130 | 0.081914 |
| 131 | 0.03858 |

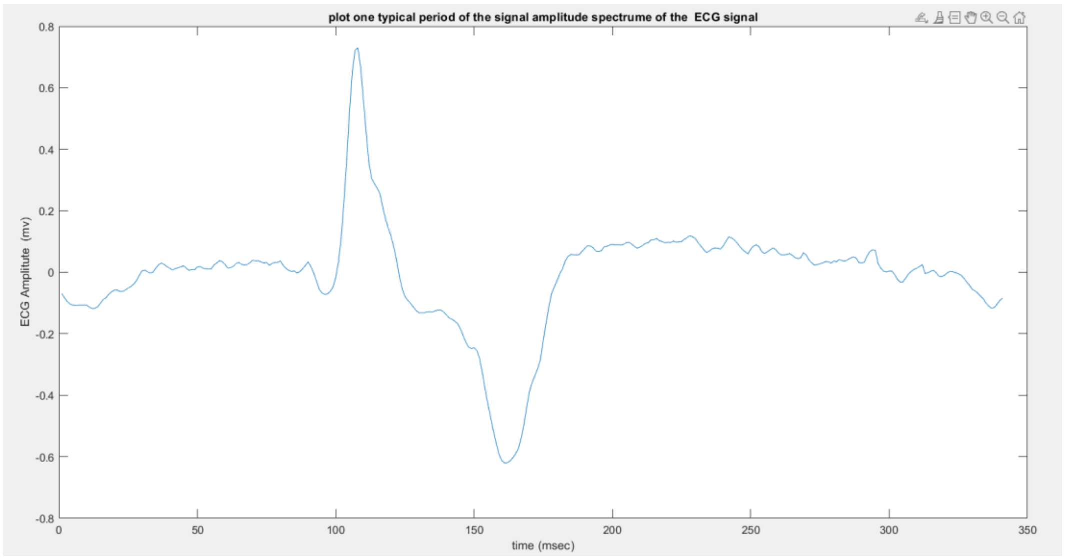


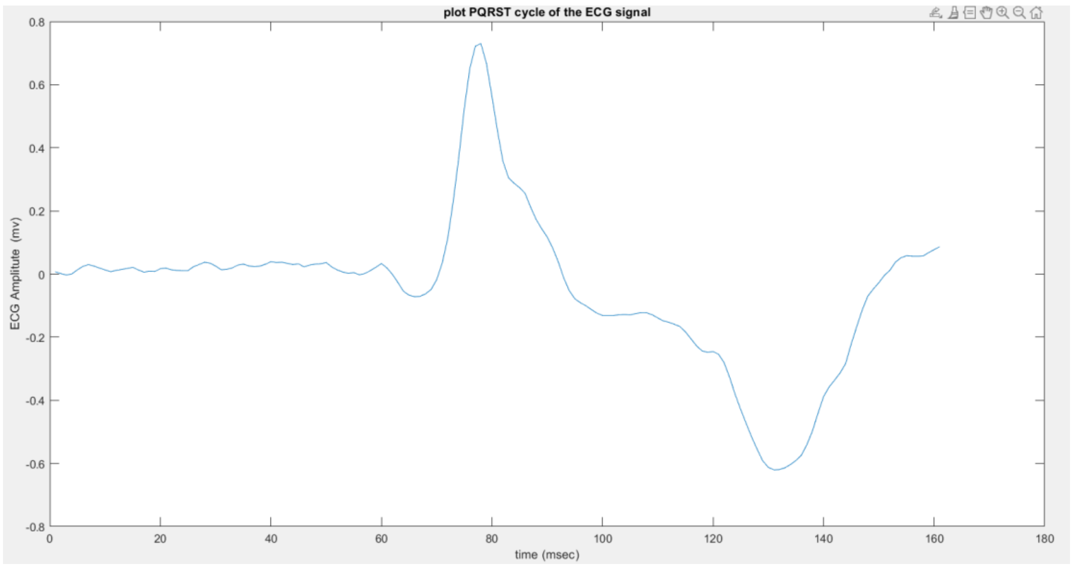
```
close all;
clear all;
clc;

y1=xlsread("C:\Users\somiy\Downloads\samplematlab.csv");
fs = 250 % find the sampling rate or frequency
T = 1/fs;% sampling rate or frequency
% find the length of the data per second
N = length(y1);
ls = size(y1);
t = (0 : N-1)/fs;% sampling period

figure;
plot(t,y1);
title ('plot of the orignal ECG signal');
xlabel ('time (sec)');
ylabel ('ECG Amplitute (mv)');
grid on;
figure;
%%% Create a period
y1new = y1(10:350);
ax = axis; axis([ax(1:2) -3.2 3.2])
t2 = (0 : length (y1new)-1)/fs;% sampling period
plot (y1new);
title ('plot one typical period of the signal amplitude spectrume of the  ECG signal');
xlabel ('time (msec)');
```



```
ylabel ('ECG Amplitute  (mv)');


figure;
%%% Create a period
y1new1 = y1(40:200);
ax = axis; axis([ax(1:2) -3.2 3.2])
t2 = (0 : length (y1new)-1)/fs;% sampling period
%subplot(1,2,2);
plot (y1new1);
title ('plot PQRST cycle of the ECG signal');
xlabel ('time (msec)');
ylabel ('ECG Amplitute  (mv)');

%% heart rate analysis
% count the dominat peak
beat_count =0;
for k = 2 : length(y1)-1
    %the peak has to be greater than 1 and greater than the value before it and greater then the value after it.
    if(y1(k)> y1(k-1) && y1(k) > y1(k+1) && y1(k)> 1)
        beat_count = beat_count +1;
    end
end
display (k);
disp('dominant peaks');
%% divide the peak count by the duration in minute
duration_in_sec = N/fs;
duration_in_minute = duration_in_sec/60;
BPM = beat_count/duration_in_minute;
```

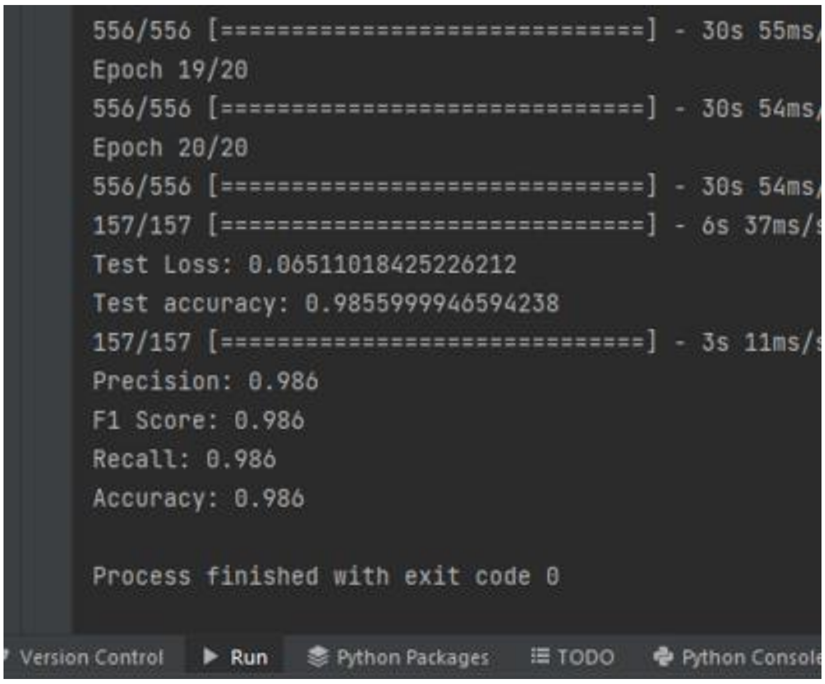

plot of the orignal ECG signal
ECG Amplitute (mv)
time (sec)
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1
2
3
4
5
6
7


plot one typical period of the signal amplitude spectrume of the ECG signal
ECG Amplitute (mv)
time (msec)
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
50
100
150
200
250
300
350


plot PQRST cycle of the ECG signal
ECG Amplitute (mv)
time (msec)
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
20
40
60
80
100
120
140
160
180

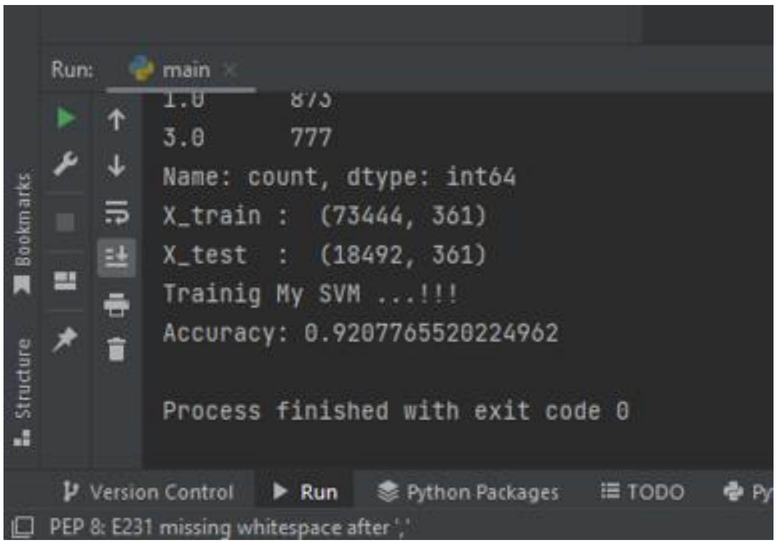
## Machine Learning algorithm results

### CNN



```
556/556 [==============================] - 30s 55ms/
Epoch 19/20
556/556 [==============================] - 30s 54ms/
Epoch 20/20
556/556 [==============================] - 30s 54ms/
157/157 [==============================] - 6s 37ms/s
Test Loss: 0.06511018425226212
Test accuracy: 0.9855999946594238
157/157 [==============================] - 3s 11ms/s
Precision: 0.986
F1 Score: 0.986
Recall: 0.986
Accuracy: 0.986

Process finished with exit code 0
```

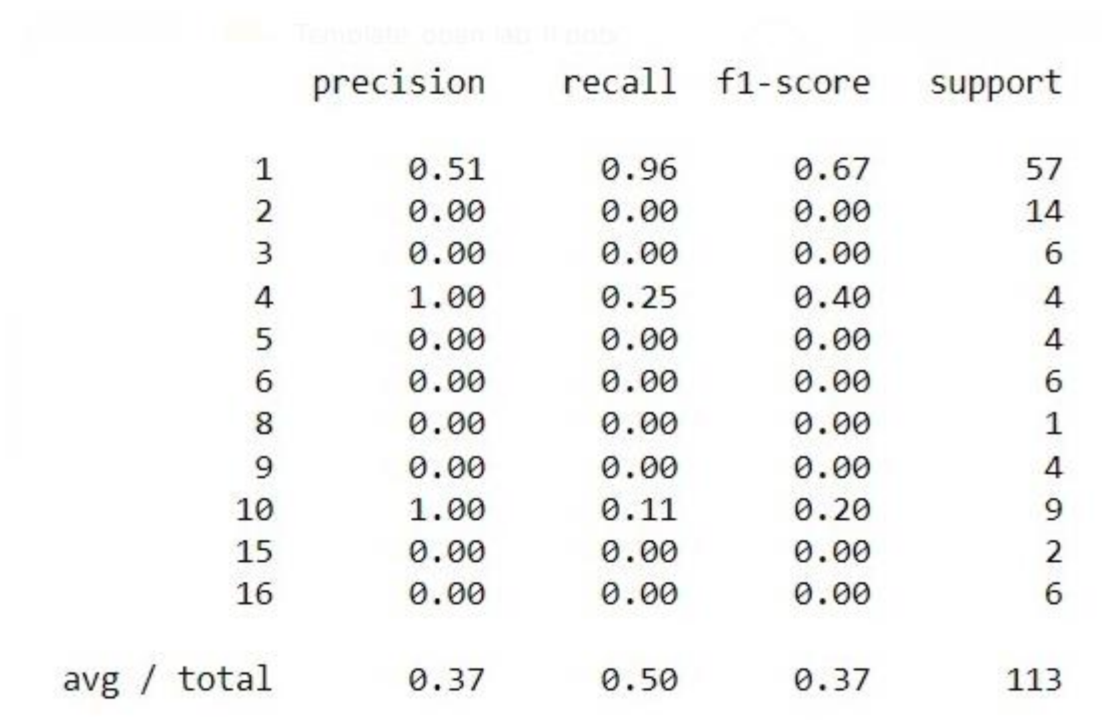
### SVM



```
1.0      873
3.0      777
Name: count, dtype: int64
X_train :  (73444, 361)
X_test  :  (18492, 361)
Trainig My SVM ...!!!
Accuracy: 0.9207765520224962

Process finished with exit code 0
```

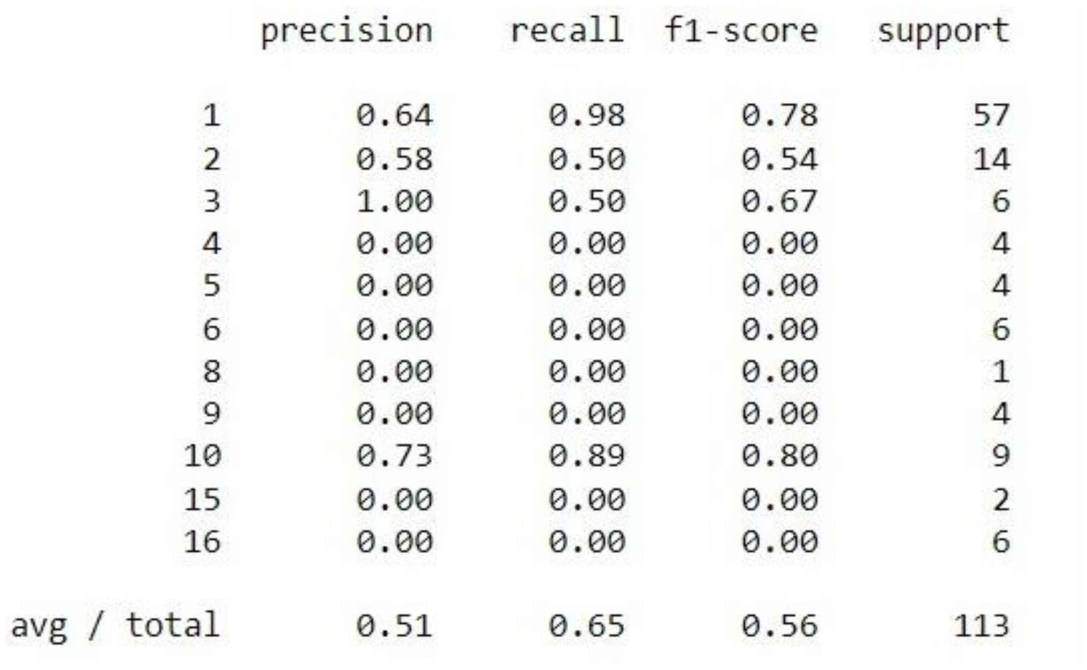
PEP 8: E231 missing whitespace after ','

**KNN**



```
             precision    recall  f1-score   support

          1       0.51      0.96      0.67        57
          2       0.00      0.00      0.00        14
          3       0.00      0.00      0.00         6
          4       1.00      0.25      0.40         4
          5       0.00      0.00      0.00         4
          6       0.00      0.00      0.00         6
          8       0.00      0.00      0.00         1
          9       0.00      0.00      0.00         4
         10       1.00      0.11      0.20         9
         15       0.00      0.00      0.00         2
         16       0.00      0.00      0.00         6

avg / total       0.37      0.50      0.37       113
```

**Random Forest**



```
             precision    recall  f1-score   support

          1       0.64      0.98      0.78        57
          2       0.58      0.50      0.54        14
          3       1.00      0.50      0.67         6
          4       0.00      0.00      0.00         4
          5       0.00      0.00      0.00         4
          6       0.00      0.00      0.00         6
          8       0.00      0.00      0.00         1
          9       0.00      0.00      0.00         4
         10       0.73      0.89      0.80         9
         15       0.00      0.00      0.00         2
         16       0.00      0.00      0.00         6

avg / total       0.51      0.65      0.56       113
```

# Chapter 5

# Conclusion

In conclusion, the study of Electrocardiogram (ECG) signals and the application of machine learning algorithms, including Support Vector Machines (SVM), Convolutional Neural Networks (CNN), K-Nearest Neighbors (KNN), and Random Forest, have shown promising results in the classification of Atrial Fibrillation (AF).

By analyzing ECG signals and extracting relevant features, machine learning algorithms can effectively differentiate between AF and normal heartbeats, providing automated and accurate AF detection.

SVM, known for its ability to handle complex and non-linear data, has shown good performance in AF classification by finding an optimal hyperplane to separate different classes. CNN, with its deep learning architecture, has demonstrated its capability to learn complex patterns and features directly from raw ECG data, achieving state-of-the-art results in AF detection. KNN, a simple and intuitive algorithm, has proven effective by considering the similarity between ECG patterns for classification. Random Forest, an ensemble learning method, combines multiple decision trees to improve accuracy and robustness in AF classification.

The comparative analysis of these machine learning algorithms has provided insights into their strengths and limitations. Each algorithm has its own advantages and performs well under different scenarios. The choice of algorithm depends on factors such as the dataset characteristics, computational resources, interpretability requirements, and real-time processing constraints.

However, challenges and opportunities for further research exist in this field. The availability of larger and more diverse datasets, addressing class imbalance issues, and exploring advanced feature extraction techniques are areas that require attention. Additionally, interpretability of machine learning models and their integration into clinical practice need to be explored further to gain trust and acceptance from healthcare professionals.

In conclusion, the study of ECG and AF classification using machine learning algorithms, including SVM, CNN, KNN, and Random Forest, holds great promise in improving AF detection and patient care. The application of these algorithms provides a foundation for developing robust and reliable systems for automated AF diagnosis. Continued research and advancements in this area will contribute to the development of more accurate, efficient, and accessible tools for AF detection and management.