Classification Algorithm for Early Detection of Atrial Fibrillation

The Development of a Supervised Learning Method Using Photoplethysmography Signals for an ARM Processor

Tim van Es
Florens Helfferich

TU Delft

Classification Algorithm for Early Detection of Atrial Fibrillation

The Development of a Supervised Learning Method Using Photoplethysmography Signals for an ARM Processor

by

Tim van Es
Florens Helfferich

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended on June 29, 2021 at 1:30 PM.

Student number:	4475836	(Tim van Es)
	4607155	(Florens Helfferich)
Project duration:	April 19, 2021 – June 18, 2021	
Thesis committee:	prof.dr. K.A.A. Makinwa,	TU Delft
	MSc B. Abdikivanani,	TU Delft, supervisor
	dr.ir. R.C. Hendriks,	TU Delft, supervisor
	dr.ir. S. Vollebregt,	TU Delft

TU Delft

Abstract

Atrial fibrillation (AF) is the most common type of cardiac arrhythmia occurring in around 0.5% of the world population. AF is characterized by the rapid and irregular beating of the atrial chambers of the heart, which can cause lead to strokes and other heart-failures. To prevent these consequences the early detection of AF is paramount. Using photoplethysmography (PPG) heart activity can be measured from which the inter-beat-interval (IBI), the time between heart beats, can be estimated. Using data collected by a PPG sensor the aim is to classify the heart activity as either AF or Normal Sinus Rhythm in real time using machine learning and collect the outcomes for further analysis by medical professionals. For this a classification method is suggested which is able to be implemented on an ARM based processor. Using a Support Vector Machine and 10 features derived from the IBI's and the PPG signal this algorithm achieves the following accuracy metrics: balanced accuracy = 0.853, sensitivity = 0.850, specificity = 0.856 and Matthews Correlation Coefficient (MCC) = 0.643. Compared to similar studies these results are substandard and should be improved.

Preface and Acknowledgments

The past weeks have been exciting, finally finalizing our Bachelor degrees and applying the knowledge we have collected over the past couple of years. In the course of this project we have experienced how, in this ever changing world full of technology, a few people are able to design a technology that could have a positive impact on the lives of millions of others. A technology that would not have been possible a just few years back.

Similarly, the skills that we utilized the most in this project were only obtained last quarter. The knowledge and tools given to us in the Introduction to Machine Learning course are what made this project possible in its current form. To us this was a hands-on experience of personal progression as a way to help others.

Without the help of Bahareh Abdikivanani and Richard Hendriks, our supervisors, this project would not have been possible. We would like to thank them for the time, effort and knowledge they shared with us during our meetings and in their free time.

Furthermore, we want to thank the organizers of the Bachelor Graduation Project and the staff at the Tellegen Hall. They made it possible to discuss the problem and solutions in the more pleasant and familiar setting that is the Tellegen Hall.

And lastly special thanks to the people that were so kind to take time out of their busy daily lives as experts to answer the many questions we had for them on their fields of expertise. Hadi Jamali-Rad from the Circuits and Systems group, Pourya Omidi from Praxasense, Fons Wesselius and Matthijs van Schie from the Department of Cardiology from the Erasmus Medical Center, Eelko Ronner from the Reinier de Graaf hospital.

Tim van Es
Florens Helfferich
June 18, 2021

Contents

1 Introduction

1.1. The Problem: Atrial Fibrillation

Atrial fibrillation (AF) is the most common type of cardiac arrhythmia among the world population. According to the Global Burden of Disease study of 2017, the worldwide prevalence of AF was reported to be 37.57 million cases [1], which composes approximately 0.51% of the worldwide population. AF is characterized by the rapid and irregular beating of the atrial chambers of the heart, which is caused by disorganized electrical signals that propagate through the heart. The frequency and length of AF episodes generally increase over time, making AF a progressive disease.

Over the past decades, several methods of diagnosing a patient with AF have been used - and are still being used in clinical environments. A prominent example of this is 12-lead electrocardiography (ECG) used in hospitals, based on the findings of its founding father Willem Einthoven, MD, PhD [2]. Although these ECGs yield accurate recordings of the heart activity, it demands the patient to be situated in the hospital, which is inconvenient for both parties. Moreover, early AF diagnosis requires monitoring over a long period. To further investigate AF symptoms the patient usually is monitored in the hospital for some period of time or receives a holter, a large ECG recording device. On the basis of such recordings a medical professional such as a cardiologist is able to officially diagnose the patient. Typically the patient only perceives these symptoms after some preceding AF episodes have occurred unnoticed. In addition, AF episodes do not occur at clearly defined moments, some AF events occur only during physical exercise. These conditions complicate the matter of early detection. Furthermore, the medical professional must perform time consuming manual analysis on the recording to diagnose the patient.

However, wearable consumer electronics show that heart activity monitoring and, in limited cases, classification of arrhythmias is viable from a technological and financial standpoint. Such prevention care can alert potential patients and suggest a visit to the physician, which is more preferable than an actual treatment plan once the disease has already done damage.

Current consumer electronics use an ECG signal or, even more prevalent, photoplethysmography (PPG) for acquisition of the heart activity at body extremities such as the wrist. These technologies allow the device to be non-invasive and wearable which makes long term signal acquisition possible.

Ideally, early detection should happen widespread and often to find as many potential patients as possible. This does mean that signal acquisition is done during day-to-day activities that are far from optimal. The wearable technologies are then prone to motion artifacts for example.

The focus of this thesis is on the detection step. Using machine learning algorithms signals are classified as Normal Sinus Rhythm (NSR) or AF. There are however a lot of different approaches for such classification. In light of the aforementioned problems concerning detection, the main question to answer is: **Which algorithm is best suited to handle classification of atrial fibrillation provided PPG signals in the context of a wearable device?**

1.2. Project Division

The goal of this project is to design a wearable device for early detection of AF. The device should be affordable, convenient to wear and non-invasive. The proposed solution uses a continuous PPG recording to monitor heart activity. Acceleration data is recorded to provide a reference signal to movements made with

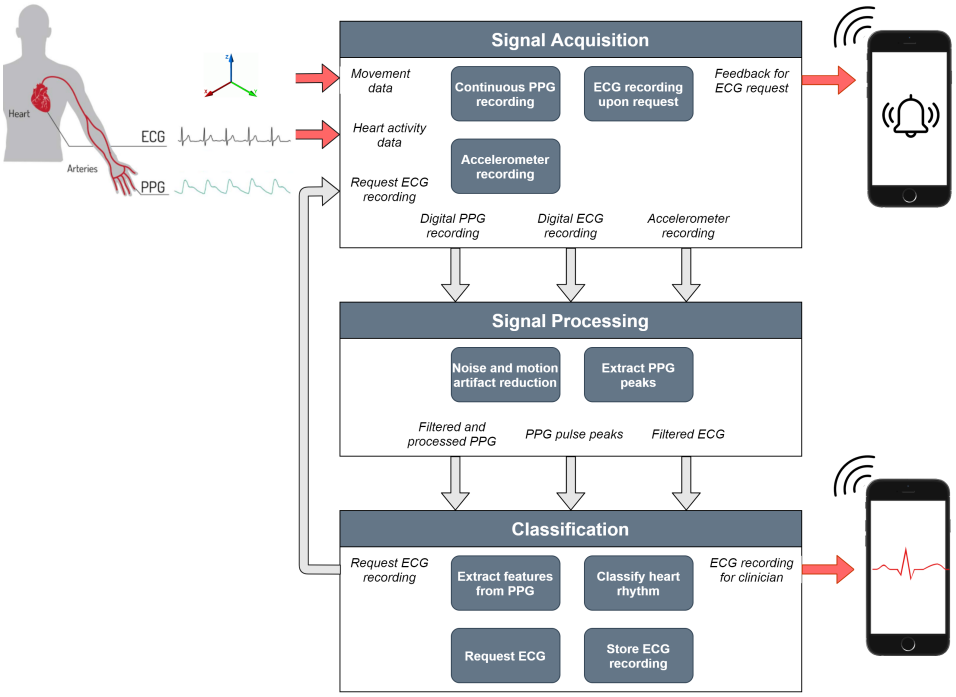

Figure 1.1: Project diagram showing the division of the solution proposal over the three different subgroups.

the device. The PPG signal is processed to remove noise and motion artifacts. Machine learning classification is used to predict the presence of AF in the PPG signal. When AF is detected, the user is warned and has the possibility to record a single lead ECG. This ECG is stored to be reviewed by a medical professional.

The project is divided into the following subgroups:

- Signal acquisition
- Signal processing
- Classification

A block diagram of the interactions between these subgroups is seen in Fig.1.1. The objectives of the signal acquisition subgroup are to acquire continuous PPG and accelerometer recordings. An ECG recording should be made when prompted if AF is detected. All signals should be in a digital format.

The signal processing subgroup is responsible for filtering and processing of these digital signals. Here the PPG signal is enhanced by reducing noise and motion artifacts to output to the classification subgroup. Furthermore, an estimate of the beat-to-beat heart rate is extracted.

The objective of the classification subgroup is to extract features from the filtered PPG signal and the accompanying beat-to-beat heart rate. A machine learning classification algorithm should be designed that uses these features to detect AF episodes in the PPG recordings.

1.3. Thesis Outline

In order to answer the main question, the development progress is separated in parts and described in sequence, each chapter gives background information or elaborates on the design choices that have been made. In Chapter 2 the requirements for the detection system are highlighted to present the boundary conditions for the system. Chapter 3 gives information on AF and the origin of the PPG waveform and the chapter introduces existing classification approaches. Chapter 4 gives an overview of the Classification Pipeline, a framework to develop the necessary algorithms. After all stages of that framework are presented, Chapter 5 discusses the final designs for the classification algorithm and the development process. The results from these algorithms are shown and compared in Chapter 6. Finally, Chapter 7 suggests improvements to the work and Chapter 8 concludes this thesis.

2

Programme of Requirements

The goal of the requested device or system is to find and develop ways of acquiring heart activity signals, improve quality of those signals with preprocessing steps and to feed that to a classification algorithm for automatic detection of AF episodes. The device or system is intended for the consumer market or to be integrated into consumer electronics. This market is vital to the aim of early detection of AF, namely to do early detection as widespread as possible. In this chapter, requirements for the functioning of the system and requirements having to do with the intended market are described and quantified so that they can serve as reference to the design choices made during the development. Due to the data flow inherent to the project division of acquisition, processing and classification functions some assumptions were made based on work of the first two subgroups (signal acquisition and preprocessing for noise and artifact removal) that bound the classification subgroup. Those assumptions are highlighted at the end of this chapter.

2.1. Group Requirements

This is a summary of the requirements for the entire group. The summary serves to highlight the functions that are mandatory for a prototype of the system to work. Furthermore, these requirements bound the system to a context using non-functional requirements targeted at the end users contained in the consumer market.

Functional Requirements

1 The system must measure the user's heart activity.

2 The system must be able to detect AF episodes.

3 The system must generate statements and statistics that are interpretable by a medical professional and are able to aid in the diagnosis of Atrial Fibrillation.

4 The system must give feedback to the user about the classification outcome. With such notification the potential patient is able to take action against the disease or initiate professional help.

5 The device must be able to provide core functionality without interaction with other devices. Core functionality consists of recording and storing measurements, AF classification and user feedback. Such a standalone device greatly enhances wearability by reducing requirements for, e.g., the need for an internet connection for classification.

6 The device should be able to be used during day-to-day activities while maintaining its accuracy. This way continuous detection can give insight in the AF pathology of a potential patient.

Non-Functional Requirements

7 The device must be wearable as an accessory. The eventual system resembling an accessory will help widespread detection.

8 The product can only provide information and advice, no diagnosis is given. In other words the device can not take the place of a medical professional, yet it can aid their work with information.

9 The device should be affordable to be used in a wide audience and therefore components should not cost more than €30,-.

2.2. Requirements for the Classification Subgroup

In the context of the complete requirements mentioned above the classification algorithm design is constrained by the following specifications:

Functional Requirements for Classification

2 The system must be able to detect AF episodes meaning that:

- **2.1** Episodes must be detected using artificial intelligence-based classification also known as machine learning classification.
- **2.2** The classification algorithm must have an accuracy of at least +0.8 Mathews Correlation Coefficient score. This metric is described in Section 4.5.1.
- **2.3** Episodes of at least 30 seconds must be classifiable by the algorithm. Medical professionals usually define AF episodes to be at least 30 seconds long.
- **2.4** Regarding Type I (False Positives) and Type II (False Negatives) errors: the Type II errors are more important to minimize, because that means reducing the times an AF episode is missed.

3 The system must generate statements and statistics that are interpretable by a medical professional and aid the professional in the diagnosis of AF.

- **3.1** The system must provide the times of onset and duration of detected AF episodes.
- **3.2** The system must provide visual representation of ECG recordings made by the user.

5 The device must be able to provide core functionality without interaction with other devices. Core functionality consists of recording and storing measurements, AF classification and user feedback.

Non-Functional Requirements for Classification

8 The product can only provide information and advice, no diagnosis is given.

9 The device should be affordable to be used in a wide audience and therefore components should not cost more than €30,-.

Assumptions based on work and requirements of other subgroups

I The product is able to make a PPG recording of the user. This is the main input signal to the classification process.

II The PPG recording has adequate resolution so that this does not influence classification accuracy.

III The data comes from recordings taken during day-to-day activities. This includes resting, walking and wrist movement.

IV The prototype algorithm for a trained model will run on an ARM Cortex-M7 processor this sets the following constraint: Functions that are required to run the trained model must be available on the hardware or must be constructible with minimal effort.

V Processing power on the device has to be shared between subgroups. However during development the assumption is made that all processing power can be used for the classification process.

3

Background

3.1. Cardiac Activity

At various locations the heart muscles contract as a result of the electrical signals while at other times the muscles relax to let blood flow in. Each heartbeat such a cardiac cycle is initiated by electrical signals that propagate through the heart. To elaborate further the paragraphs below introduce the normal cardiac rhythm and atrial fibrillation.

3.1.1. Normal Sinus Rhythm

The cardiac cycle constitutes of systolic and diastolic phases, the contraction and relaxation phases of the myocardium (heart muscles). The order and periods of these phases is important to the cardiac function of pumping the blood through the body. To start the cardiac cycle the sinoatrial node (SA) generates a dominant electrical impulse and as a reaction the surrounding tissue also releases electrical impulses. Thus the electrical signal propagates through the atria to the atrioventricular node (AV), which in turn propagates the signal to the ventricles that contract strongly to fulfill the pumping function [3], the diastolic phase. The left figure in Fig. 3.1 shows this propagation through the heart going without any deviations.

3.1.2. Atrial Fibrillation

On the right in Fig. 3.1 the propagation of the electrical impulse is irregular. Impulses arriving in the atria cause contractions at higher rates than usual, in other words fibrillation of the atria. This renders the atria inefficient during both the diastolic and systolic phases. Additionally, "the AV conduction system is bombarded with many electrical stimuli, causing inconsistent impulse transmission and an irregularly irregular ventricular rate" [4].

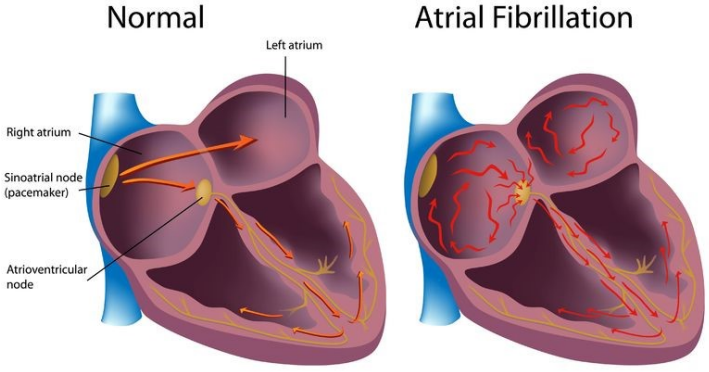

Figure 3.1: The pathway of the electrical signal through the heart is different in case of Normal Sinus Rhythm (left) and than for Atrial Fibrillation (right). Source: [3]

3.2. Photoplethysmography

The Photoplethysmography (PPG) signal is a waveform that is retrieved from light intensity that is reflected of blood in the vascular tree. Their are DC and AC components in the signal. The DC component originates from light absorption in the skin, the venous blood (deoxygenated), and the non-pulsatile arterial blood. The AC component is due to blood pressure in the arterial vessels. Thus the PPG waveform is related to blood pressure therefore related to the heart activity. The following cardiac cycle events are visible:

- The positive slope and systolic peak correspond to the contraction of the left ventricle. The systolic peak is the maximum of the PPG signal. Counter-intuitively this peak corresponds with
- The negative slope starting from the systolic peak and ending in a local minimum (just before the diastolic peak) corresponds with the aortic valves closing.

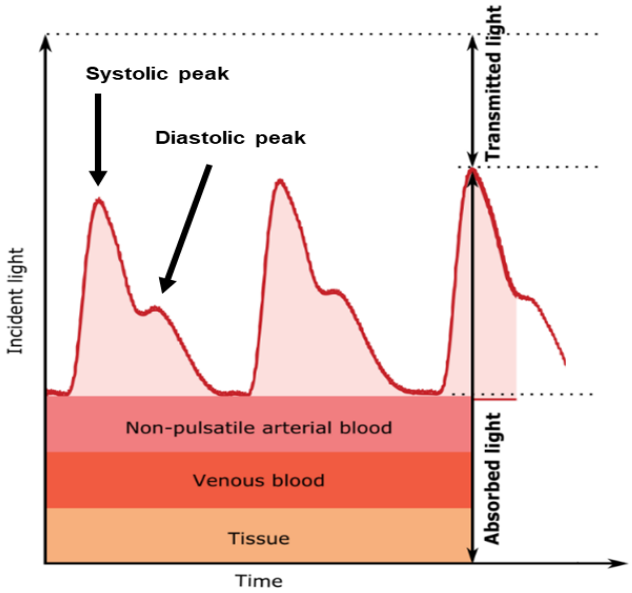

Figure 3.2: Typical PPG waveform and the different locations where the incident light is absorbed by the body. The reflected (transmitted) light is received by the PPG sensor. Source: Adapted from [5]

3.2.1. Noise and Artifacts

One of the downsides of using PPG based sensors to classify heart conditions is the susceptibility to various noise sources including ambient light intrusion on the sensor, limited blood perfusion and motion artifacts [6]. The noise produced by movement has been the subject of many studies which can be categorized in two broad groups: Motion detection followed by data segmentation [6, 7] and noise reduction techniques[8, 9].

Compared to an ECG the PPG signal also lags behind the electrical activity in the heart due to the mechanical movement needed to propagate the pressure wave to the measurement device. This lag however is not detrimental to the ability to estimate the peak to peak intervals from a PPG device while the subject is at rest [10]. Such intervals are sometimes referred to as RR intervals or RRI, which comes from the typical R peaks in an ECG.

3.2.2. Correlation between AF Episode Onset and the Presence of Noise

One of the difficulties of diagnosing patients with AF is the sporadic nature with which its effects can be present. When the arrhythmia occurs recurringly and stops spontaneously it is classified as paroxysmal AF [11]. Therefore, with paroxysmal AF, it is possible that a medical professional misdiagnoses a patient simply because there were no symptoms at the time of inspection. Other classes include persistent and permanent but since these classes don't stop spontaneously they are easier to detect. Since the goal of this study is to detect these sporadic episodes, it is essential to record a signal near-continuously, so as to also record at times when the onset of an episode is likely. According to [12, 13] the most likely events to trigger an episode are

the consumption of alcohol or caffeine, stress, tiredness and physical exercise. Unfortunately, as described above, getting a high quality signal while the subject is in motion is a difficult task due to the presence of noise. This correlation between occurrence of AF episodes and increasing noise means that special filters need to be designed in order to extract the original signal, possibly by integration of various recordings such as acceleration signals or PPG recordings with different wavelength.

3.3. Review of PPG-based Classification Approaches

Three common artificial intelligence approaches to PPG-based classification of AF are: statistical analysis, machine learning and deep learning. Each of these algorithms require a set of rules or boundaries to be fine-tuned using data-points as input like anchoring points and generally involve multiple iterations of learning. Differences between these types originate mostly from the input data processing and decision making on the part of the algorithm that is necessary to come to a well-informed prediction. The next paragraphs function to exemplify model types and their advantages over others.

Statistical analysis (also named threshold-based) refers to identifying and setting thresholds between the features from AF and non-AF input signals. Thereby classification is a matter of observing the input signal with respect to a certain threshold, stationary or dynamic.

Machine Learning (ML) approaches include primarily k-means clustering, support vector machines and decision trees techniques and these "require extensive domain expertise to design features suitable for a comprehensive representation of PPG waveforms and the detection of class-differentiating patterns" as [14] put it.

Deep Learning (DL) models typically take away this knowledge requirement for ML approaches and assign the feature extraction and or selection to the neural network that is the DL model. However, creating a new DL model requires large amounts of training data to be viable. Yet fine-tuning a pre-trained DL model is also considered in DL studies detecting AF highlighted in [14].

4

Classification Pipeline

In this chapter the machine learning approach to a binary classification problem is presented, each part is elaborated in terms of their dependencies with respect to others. This method is introduced to comply with requirement **2.1**, which limits the possible solutions to artificial intelligence based classification methods. Of which machine- and deep learning are subsets.

The classification pipeline refers to the stages that need to be completed before a classification model is able to be put to use. Each stage affects the eventual predictive power or accuracy of the model and thus is given attention in the coming sections. Classification models discussed in this section are selected as a result of the performance of the models presented in Section 3.3. As will be discussed in Section 5.2.1 filtered and processed signals are supplied to the Classification subgroup, therefore no preprocessing on the data will be necessary. Fig. 4.1 shows a block diagram of the proposed pipeline. The last block 'Model comparison' will be executed in the Chapter 6, here the collected metrics will be compared and the best performing classifier will be selected.

For reference a summary of the stages is given:

1. **Dataset**: The collection of entries consisting of data points and labels that represent the input signal(s), required for training the classification model. In classical Machine Learning (ML), these data points are processed to extract features, in Deep Learning (DL) however these data points are used directly by the model.

2. **Feature extraction**: Features are attributes of a single entry from the dataset that could be indicative of the label that was given to that entry. These are extracted from the signal so that they can be used later to predict the label of that specific entry.

3. **Feature selection**: The features that work best in separating the entries can be prioritized or selected above the ones that are less effective. If less features have to be computed for acquiring similar classification results processing power can be saved.

4. **Model training**: By splitting the dataset into training and test data, we can train a model using the features (or raw data in case of DL) and labels that only belong to training dataset (usually this is around 70%) and test the performance on the test dataset. That way we verify that the model works on out-of-training data too.

5. **Prediction and Accuracy**: This is then compared with the actual labels resulting in accuracy metrics showing the effectiveness.

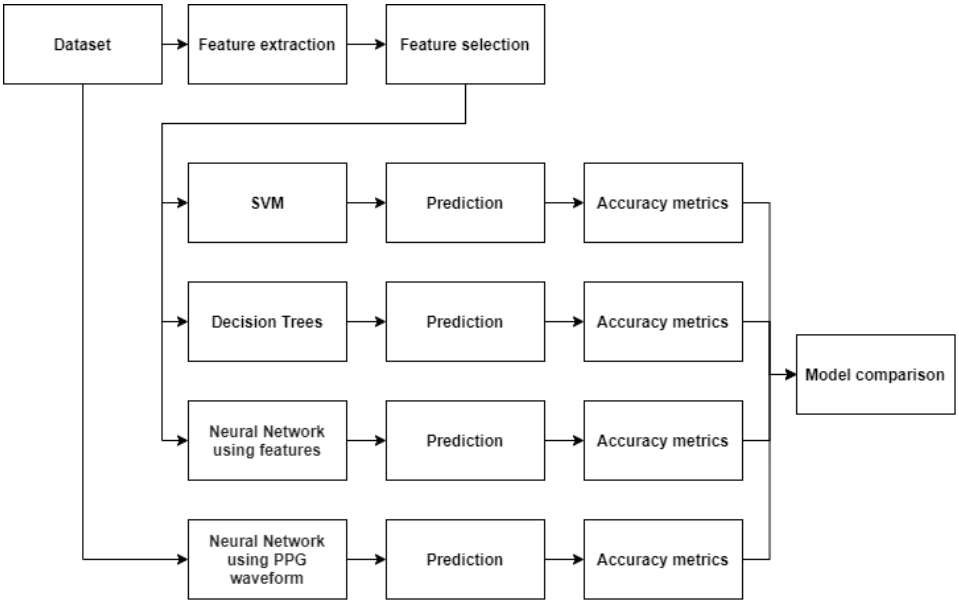

Figure 4.1: Block diagram of the classification pipeline

4.1. Dataset

The data used in a classification algorithm can virtually be of any type; a single number, sequences of numbers, text, images, video or waveforms (e.g. sound, electrical signals), the possibilities are endless. In the case that is relevant for this project a dataset is defined as a set of these data entries of the same type but different values. So a dataset can contain images of the same size but different subjects, or audio recordings together with some metadata like a date and location. This data is represented by matrix $\mathbf{V}$ of size $M \times P$ with M the number of dimensions in the entry (number of features per entry) and P the number of entries.
Since a binary classification problem is considered, all entries from a dataset belong to one of two classes. These classes are given a label represented by 0 or 1. A $1 \times P$ vector $\mathbf{y}$ is constructed from the labels per entry y_p as

$$\mathbf{y} = \{y_p \in \{0,1\}\}_{p=1}^{P} \tag{4.1}$$

The data matrix $\mathbf{V}$ together with the label vector $\mathbf{y}$ form the dataset of P entries.

4.2. Feature Extraction from Waveforms

A classification algorithm needs either a signal or a number of characteristic values derived from an input that allow the signal to be labeled as one of the possible classes. Such characteristic values are called features and in this section the extraction of features from a waveform is discussed.
A large number N of features can be extracted from any waveform, the vector $\mathbf{x}_p$ of size $N \times 1$. The collection of features from all entries is $\mathbf{X}$ with size $N \times P$.

Time Domain Feature Extraction

Within the time domain, features can be extracted that depend on the original time-series of the input signal. Examples of such features could be the maximum value within a timeframe or the 'area under the curve'. These features directly relate to some specifics of the signal that could be useful for classification. However, the true importance of a feature depends on the classes that one is trying to separate. For example, the maximum value in a PPG signal is more likely to be useful to detection of noise than to detection of AF. Statistical features are also easily extracted from a time-series, mean, standard deviation, percentiles are some examples.
One can also calculate features indirectly from the signal by using it as an input for a different function. When zero-crossings of a signal could be interesting, or descriptive for some class, statistics of these crossings and the time in between them are informative as features.

Frequency Domain Feature Extraction

The frequency components of the input signal can also contain features that are of interest to the classification pipeline. The input signal is a digital, thus sampled, representation of the original heart activity and noise sources. Extracting frequency components from a sampled signal $x = [x_0, ..., x_{N-1}]$ can be done using the Discrete Fourier Transformation (DFT):

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \qquad k = 0, .., N-1 \tag{4.2}$$

where X_k are the coefficients belonging to the N different frequency components of the signal. Examples of frequency domain features that are interesting to this project include the amount and amplitude of the high or low frequency contents or a ratio of such. Additionally, features related to periodicity of the signal and the fundamental frequency are retrievable from the auto-correlation of the discrete time signal. "[T]he autocorrelation function $c(\tau)$ describes the correlation that exists between the signal $x(t)$ and shifted versions of it" [15]. For finite discrete time signals $x[n]$ that function is calculated using:

$$c[m] = \sum_{m}^{N-1} x[n]x[n-m] \qquad n = 0, .., N-1 \tag{4.3}$$

where k is the sample delay. The sample delay can be related to time domain using the sample frequency in case the signal originates from a sampled signal. The first peak of the autocorrelation of a periodic signal gives the harmonic frequency of that signal.

4.3. Feature Selection

The large amount of features is likely to contain a few decisive features while some features are assumed to be correlating due to coming from the same source or domain. In order to reduce the dimensionality of a feature matrix **X** feature selection is key and while not diminishing to the model accuracy when applied correctly. This paragraph introduces the general workings for two feature selection algorithms briefly.

The Forward Sequential Feature Selection algorithm (SFS) runs a particular classification algorithm with the full dataset, evaluates which feature contributes the most to the performance of the algorithm and keeps using that feature permanently. In the next iteration the rest of the features are again evaluated and only the best features are kept. The SFS algorithm runs until a set amount of features is kept.

Forests are based on decision trees. Each data entry is eventually put into a so called 'leaf' after passing various thresholds for a number of features associated with that entry. Further explanation of basic trees and forests is provided in the section on tree-based classification (Section 4.4.1). For the forest feature selector it is important to mention that a multitude of decision trees are (randomly) constructed. Thereafter, unlike the SFS algorithm, this selector only does one iteration of fitting parameters per tree to decide which features are the most important. The importance of a feature is related to how homogeneous the entries in each of the two leaves are after thresholding on that specific feature (see Eq. 4.4 and its explanation). For example, a feature that puts all entries from class NSR into a leaf is useful to classification and so is given a high importance value.

4.4. Classification Models, Training and Prediction

This section provides background theory of specific classification models, sometimes referred to as classifiers, that are used in this study.

4.4.1. Tree-based

Tree-based classification algorithms rely on a decision tree to divide the dataset entries using so called stumps, branches and leaves. The stumps sort the received entries using thresholds of a single feature. The weighted gini impurity of a stump indicates the separating ability of the stump. The gini impurity of a leaf with C different classes is defined as:

$$\text{impurity} = 1 - \sum_{i=1}^{C} p_i^2 \tag{4.4}$$

where p_i is the probability of an entry in the leaf coming from the class i. As a little example: in Fig. 4.2 Leaves 1 and 2 have respective impurities of 0.32 and 0.5 indicating to the user that Leaf 1 is more pure (more entries of one class).
As seen in the figure, the leaves are the end stations of the entries that are passed through the branches. This tree goes two decisions deep before the final leaf is reached but in theory there is no limit to how deep a tree can go. Each feature can be used multiple times with different thresholds. However, it is advisable to set a limit on the depth to prevent a tree from overfitting the dataset. Setting a limit on the minimum amount of entries to further the separation with a stump can also work.
Finally, an ensemble of trees can be made by constructing a multitude of trees at random or constructed with different parameters and then use a voting approach per data entry. Such an ensemble is then called a forest.

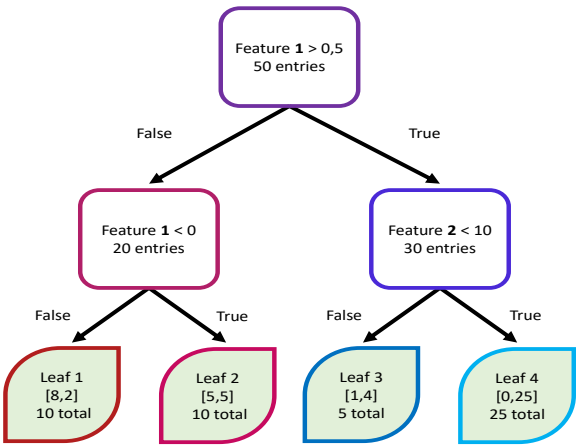

Figure 4.2: Basic representation of a decision tree with depth two. Various data entries end up in leaves at the bottom according to the features associated with each entry. In square brackets is shown the amount of entries belonging to either class.

4.4.2. Support Vector Machines

A Support Vector Machine (SVM) aims to find the hyperplane given by Eq. 4.5 that separates two linearly separable classes with a maximum margin $b + \mathbf{x}^T\boldsymbol{\omega} = \pm 1$ [16]. Where $\mathbf{x}$ are the N input features and w_i the feature touching weights.

$$b + \mathbf{x}^T\boldsymbol{\omega} = 0 \qquad \text{,with} \qquad b = w_0 \quad \text{and} \quad \boldsymbol{\omega} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} \tag{4.5}$$

Finding the values of b and w_i for which the model performs best is done by defining a cost function, describing the error in estimations, which is tried to be minimized. The cost function used for SVM's is usually the 'soft-margin SVM cost',

$$g(b, \boldsymbol{\omega}) = \sum_{p=1}^{P} \max(0, 1 - y_p(b + \mathbf{x}_p^T\boldsymbol{\omega})) + \lambda||\boldsymbol{\omega}||^2 \tag{4.6}$$

which compared to the 'hard-margin' cost function puts less pressure on perfect separation and more on a 'smooth' hyperplane. Here the label for each data point p, y_p is either +1 or -1. The regularization parameter $\lambda \geq 0$ determines the importance of correct classification over maximum margin, the lower λ the 'harder' the separation hyperplane (at $\lambda = 0$ the soft and hard margin costs are equal). Minimizing the cost function $g()$ results in a trained model, which can be used to make predictions.
In case the classes are not linearly separable one can employ the 'kernel trick' [16], where a function called a kernel is used to transform the non-linear problem into a linear one within a higher-dimensional space which can be solved using the method described above. A common kernel is the Radial Basis Function (RBF).

4.4.3. Neural Networks

Analogous the the structure of the human brain a Neural Network (NN) consists out of a dense interconnection of neurons, called nodes. These take the input, which can be features in the case of Machine Learning or

raw data in case of Deep Learning, and extract information used to classify the input. Due to the high level of interconnectivity combinations of inputs that are typical for certain classes are likely to be present and can thus be used to make predictions. The way these models are trained will not be discussed, but a general overview of a trained model is given as a reference.

The basic building block of a NN is the fully connected layer showed in Fig. 4.3. The input $\dot{\mathbf{x}} = [\,1\ x_1\ x_2\ \dots\ x_N\,]^T =$

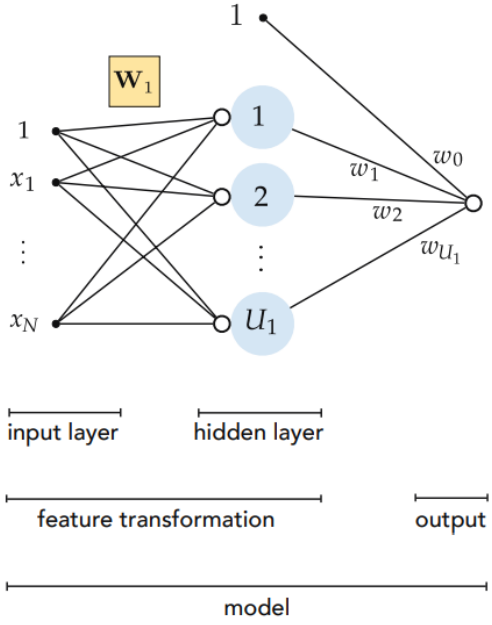

Figure 4.3: Fully connected neural network representation where weight matrix $\mathbf{W}_1$ defines the multiplication factors between the bias and $x_1,...,x_N$ feature space; and the hidden layer nodes U, Source: [16]

$[\,1\ \mathbf{x}^T\,]^T$ is dotted with the weight vector $\mathbf{w} = [\ w_0\ w_1\ w_2\ \dots\ w_N\,]^T$ after which the linear combination of all elements of the result is computed. This is then passed through an activation function $a()$ which is then the output of a single node. So the output of a node will be

$$f(\mathbf{x}) = a\left(w_0 + \sum_{n=1}^{N} w_n x_n\right) \tag{4.7}$$

which can be fed into another node. This way layers can be build with multiple nodes that are fully connected to the next layer with a certain amount of nodes. Fig. 4.3 shows a single hidden layer consisting of U nodes. The matrix W_1 contains all the weights used in the first layer. Since these are randomly initialized every linear combination of inputs with weights will likely yield a different result, some of which are useful for a classification. Combinations that have low values are filtered out by the activation function which is generally a sigmoid, tanh, softmax or ReLu function.

4.5. Model Evaluation and Comparison Methodology

4.5.1. Model Accuracy Metrics for Binary Classification

Accuracy metrics are values that signify the quality of the algorithm. Frequently these are expressed using the final prediction and actual classification labels that were the output and the input, respectively, to the classification algorithm. Notation on such raw outputs are: True Positive (TP) for correct prediction label on the positive class, False Positive (FP) for positive class label wrongly assigned to actual negative class input, False Negative (FN) for negative class label wrongly assigned to actual positive class input and True Negative (TN) for correct prediction label on negative class.

Some common accuracy metrics are described. First, the basic accuracy metric measures the fraction of

correctly classified inputs over the total inputs like this: $accuracy = \frac{TP+TN}{TP+FP+FN+TN}$. Correspondingly, there is the error metric measuring the fraction of incorrect predictions: $error = \frac{FP+FN}{TP+FP+FN+TN}$. These metrics however should not be relied on for a fair assessment of the classification in the common case that there is an imbalanced dataset.

Following are better metrics capable of taking imbalance into account: Sensitivity or recall (Se) describes the portion of positive class inputs that are deemed positive classes by the algorithm and is calculated by Eq. 4.8. Likewise the specificity (Sp) refers to the portion of negative inputs that are predicted negative class labels by the algorithm (equation 4.9). Precision (also known as positive prediction value (PPV) seen in Eq. 4.10) refers to the fraction of actual positive inputs from the total predicted positive labels.

$$Se = \frac{TP}{(TP+FN)} \quad (4.8) \qquad Sp = \frac{TN}{(TN+FP)} \quad (4.9) \qquad precision = \frac{TP}{(TP+FP)} \quad (4.10)$$

Combining two of these metrics gives the balanced accuracy (BA)[16], which basically is the average of sensitivity and specificity like in Eq. 4.11.

$$BA = \frac{1}{2}(Se+Sp) \quad (4.11)$$

The final accuracy metric, named the Mathews Correlation Coefficient (**MCC**), is presented. Chicco [17] mentions the MCC score as the "only binary classification rate that generates a high score only if the binary predictor was able to correctly predict the majority of positive data instances and the majority of negative data instances"[17].

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FN)(TN+FP)}} \quad (4.12)$$

4.5.2. K-fold Cross-Validation

Instead of relying on only one pair of training and testing datasets, the total dataset can be cut into k different folds and different pairings of those folds can be made to make differing training and testing. K-fold cross-validation thereby returns an average evaluation over those k pairs. Fig. 4.4 is a visual representation of this. In that example, the first one-third of the total dataset is used for testing while the last two-thirds are used for training during the first iteration. A variation of this is called stratified where the percentage of fold samples per class is the same as that in the total dataset. That way each fold is representative of the entire dataset.

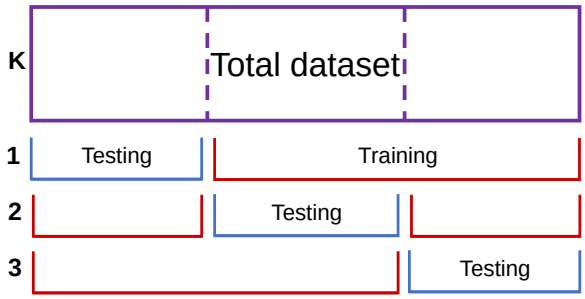

Figure 4.4: Each iteration of the k-fold cross-validation a different set is used to train and test on; here the three iterations of a 3-fold cross-validation is shown.

5

AF Detection Design

This chapter largely follows the same structure as the previous chapter (Chapter 4). The classification pipeline is followed to help the design process. Most of the classical machine learning concepts were realized using the various Python libraries from Scikit Learn[1] and will be referred to as 'sklearn'. For the creation of neural networks the Keras API [2] was used with a TensorFlow [3] backend.

5.1. Dataset

The dataset allows the training process of the classification model. It is desired to input a true representation of real signals in the context that the model is supposed to perform and train accordingly. Therefore, the key aspects are identified for the intended context of PPG-based AF classification on a wrist wearable and are listed below in order of importance to that context:

1. PPG signals are present in the dataset and contains at least AF and NSR labels per signal or subject from which the data is measured.
2. The data contains signals of at least 30 seconds long.
3. The measurements were partially done during (light) movement of the subject.
4. Accelerometer data is present in the dataset.
5. ECG signals are present in the dataset as a reference.

Not all aspects are a must but each addition will contribute to the accuracy of the entire system. Aspects 3 and 4 are important to develop an algorithm that can perform during day-to-day activities as described in requirement 6 in Section 2.1.

5.1.1. The UMMC Dataset

The UMass Medical School Simband Dataset was collected by the UMass Medical Center (UMMC) and was deemed a suitable dataset for model training and validation [18]. For convenience the dataset is referred to as the UMMC dataset. The dataset consists of 8-channel PPG and 3-channel accelerometer recordings using the Samsungs Simband 2 wrist wearable 2 (Samsung Digital Health, San Jose, CA, USA). For reference a 3-channel ECG recording was made simultaneously with a 7-lead holter device. After data collection the PPG segments were visually inspected for 'corruption' and as noisy were applicable. Thereafter, the remaining 30 second segments were labeled based on human adjudication with either 'AF', 'NSR', 'possible NSR', 'NaN' or 'PAV/PCV' label [19]. The latter refers to premature ventricular or atrial contractions, these are different heart arrhythmia that are of no interest to the project therefore those segments were excluded from the final dataset. The data collection protocol included activities like (slow) walking, random wrist movement and resting [19, 20]. The dataset thus includes all criteria for a suitable dataset to develop classification algorithms.

[1] Version 0.24.2; Scikit Learn website: https://scikit-learn.org/stable/
[2] Version 2.4.3; Keras website: https://keras.io/
[3] Version 2.5.0; TensorFlow website: https://www.tensorflow.org/

From the 8-channel PPG signal only the infrared channel using 850 to 970 nm wavelength was unpacked from the UMMC dataset [22]. This channel is chosen because it uses a similar wavelength to that of the sensor on the prototype. Labels are provided in the ground truth tables of the dataset for every 30 seconds. Assuming that an entire 30 second segment of PPG data is indicative for the given label, every continuous PPG recording is cut up in 30 second segments and matched with the corresponding label to comply with requirement **2.3**. One of the subjects was not labeled and thus was left out. The sample rate of the Simband is 128 Hz. That means each signal is cut up in 30 second segments with every segment containing 3840 samples. Each segment was then paired with the appropriate label as provided in the ground truth tables of the dataset. At this point, only the 'AF' and 'NSR' labeled segments were kept and the others were discarded. The table below gives a summary of the segments:

Figure 5.1: The Samsung Simband 2 with multiple channels for ECG and PPG.
Source: [21].

Amount of subjects	39
Total amount of segments	242
total 'AF' labeled	53
total 'NSR' labeled	189
Signal length (samples)	3840
Sample rate (Hz)	128

Table 5.1: Summary of 30 second segments that are used as input to the classification pipeline. As there are significantly more NSR segments than AF the dataset is imbalanced.

The segments however still contain components that do not originate from heart activity and so are filtered out using a band-pass filter. The band-pass filtering is done in the frequency range of 0.2 to 10 Hz with a second order Butterworth filter. In Fig. 5.2 two of the resulting final segments are shown.

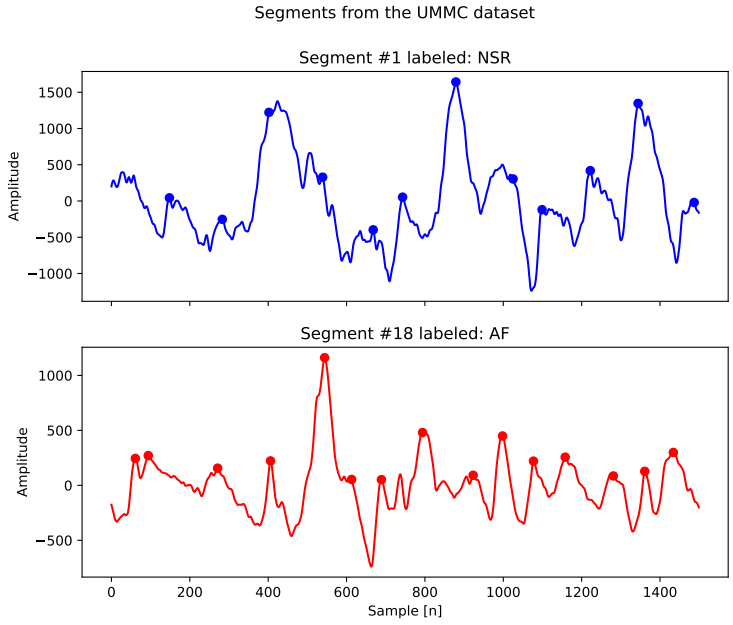

Figure 5.2: The figure shows two segments: the top is labeled Normal Sinus Rhythm and the bottom segment is labeled AF. The segments are truncated to 1500 samples for visualization. Additionally peaks, detected using the peak detection algorithm described in Section 5.2.1, are indicated.

5.2. Feature Extraction and Selection

In this section some of the features that are typically used for AF classification will be described as well as features that were found to be effective but did not directly result from the literature. A subset of these features are Heart Rate Variability (HRV) features, that describe the changes in the time duration in between heart beats [23], which is affected by AF. To extract these features however the time between heartbeats needs to be measured, for this a peak detection algorithm is first developed. Restricting oneself to features derived from a single indirect source, that is the peak detection algorithm, could be a source for errors if the algorithm functions incorrectly or a source for correlated features. Therefore more features are introduced that result from the frequency contents or waveform of the PPG signal.

5.2.1. Peak Detection and Heart Rate Variability Features

Detection of the systolic peaks is the first step in extracting HRV features from a PPG signal. The time between heartbeats, called the inter-beat interval (IBI), is used to derive these HRV features. Because we are often required to use non-invasive methods we are limited to derivations of the IBI from other signals. With an ECG the electrical activity of the heart is measured, from which the IBI can be derived from the time between R-waves, the R-R interval (RRI). Using PPG the IBI's can also be estimated, but from the interval between the systolic peaks, the peak-to-peak interval (PPI). The HRV features derived from RRI's and PPI's are almost equivalent [24, 25] and can therefore be used interchangeably in classification algorithms.

Peak Detection

To estimate the PPI's a simple peak detection algorithm is developed based on [26, 27] that functions according to the following steps: 1) A base line is established by taking a moving average of the signal. Subtracting this baseline from the signal will only leave the variations around the horizontal axis. 2) Regions of interest are identified as areas where the signal is above the horizontal axis. 3) Within each region of interest a local maximum is calculated. 4) Local maxima that are significantly lower than its surroundings are filtered out, these are usually a result of the diastolic peak or movement artifacts. The maxima that are left are then the output of the algorithm. In Fig. 5.2 the peaks detected using this algorithm are indicated on the respective signals. Unless stated otherwise the results produced in the remainder of this report are achieved using this peak detection algorithm.

In a later phase of the project it was decided that the subgroup responsible for the signal processing should take the responsibility for finding the peaks in the PPG signal. With this transition the finding of peaks is taken out of the scope of this project, but a concise explanation is given for the purpose of completeness.

This algorithm uses the same four steps described above but by evaluating short 6 second sliding windows each peaks can be cross-validated, reducing missed and incorrect peaks. Within the short window peaks are detected, the window then shifts one or two seconds and again the peaks are found with some overlap with the first detection. This is done for the entire segment resulting in a collection of possible peak locations, many of which will be duplicates. Peaks with identical locations are more likely to be correctly found and are thus saved. Clusters of peaks that probably result from the same origin are averaged and also saved. Peaks with low support in the total set (singled out) are discarded.

The difference between subsequent indexes of peaks detected by a peak detection algorithm are used to calculate the PPI's which are used in the features from which a classification can be made.

The following HRV features in this section are supported by [23–25, 28, 29] as useful features that can be collected from a PPG signal.

Time domain HRV features

- Average Heart Rate (**HR**) calculated as the average of the instantaneous heart rates, which for each PPI is calculated as one minute over the PPI. Since AF episodes generally are associated with a higher ventricular rate (≈150 BPM [30]), as opposed to 60 BPM at rest.

- Standard deviation of PPI's (**SDNN**) indicates the variability of the time between heart beats.

- Root Mean Square of Successive interval Differences (**RMSSD**) [4]is the root mean square of the difference between adjacent PPI's and reflects the beat-to-beat variance in the heart rate [23]. During AF episodes

a more variable heart rate is experienced for which this is a good feature.

$$RMSSD = \sqrt{\frac{1}{N}\sum(PPI(i) - PPI(i-1))^2} \tag{5.1}$$

- Standard Deviation of Successive Differences (**SDSD**) indicates the variability of the difference between adjacent PPI's.
- Number of successive intervals that differ by more than xx ms (**NNxx**) are features that count how often successive intervals vary by more than a certain duration. The two most frequent durations used in literature are 20 and 50 ms, but NN40 or NN70 [24] are used as well.
- Fraction of NNxx over the total number of samples (**pNNxx**) the previous feature has the limitation that values compare badly if the heart rate of the entries differ a lot. This feature compensates for that by scaling the number over the total length of a segment. However this does not make NNxx features redundant, since recordings are all of the same length there is still useful information in the absolute numbers compared to the relative numbers.

Frequency domain HRV features

Transforming the PPG signal to the frequency domain enables us to examine different frequency bands that are related to fluctuations in the signal resulting from different sources. These bands, as defined by [31], are related to the sympathetic and para-sympathetic nervous system and enables us to discriminate between them.

- Power in low-frequency band (**LF**) is the absolute power confined in the low-frequency band (0.04-0.15 Hz) these frequencies are controlled by the sympathetic nervous and are affected by the rhythm of breathing.
- Peak frequency in low-frequency band (**LF-peak**)
- Power in high-frequency band (**HF**) is the absolute power confined in the high-frequency band (0.15-0.40 Hz) these frequencies result from the para-sympathetic nervous system but are also influenced by breathing.
- Peak frequency in high-frequency band (**HF-peak**)
- Ratio of LF to HF power (**LF/HF**) can estimate the ratio of sympathetic to para-sympathetic activity.
- Power in very-low-frequency band (**VLF**) is the absolute power confined in the band 0.0033-0.04 Hz. This feature is considered but this would require a recording length of at least 5 minutes [23].

Non-linear HRV features

Some of these features result from a Poincaré plot, which is a scatter plot with subsequent PPI plotted against each other. This would thus plot $(PPI(i), PPI(i+1))$ followed by $(PPI(i+1), PPI(i+2))$, $(PPI(i+2), PPI(i+3))$ etc. This could visualize patterns in subsequent PPI's.

- Poincaré (**SD1**) [4] is calculated as the standard deviation of the distance each point of the poincaré plot has to the $y = -x$ axis.
- Poincaré (**SD2**) is calculated equivalent to SD1 but then to the $y = x$ axis.
- Poincaré ratio (**SD1/SD2**) is a measure of the unpredictability of the PPI's [23]
- Sample Entropy (**SampEn**) is, as described in [20] as measure of randomness of the PPI's that is often implemented for detection of arrhythmia from PPG. The python function `sample_entropy` from the `pyentrp` package is used to calculate this feature.
- Coefficient of the Sample Entropy (**CoSEn**) as described by [33] in Eq. 5.2 is a corrected version of the Sample Entropy, the variable r is the tolerance factor set at $0.2 \times \text{std}(PPI)$

$$CoSEn = SampEn - \ln(2r) - (\text{mean}(PPI)) \tag{5.2}$$

[4] It should be noted that the feature SD1 is identical to RMSSD multiplied by $\frac{1}{\sqrt{2}}$ [32], so only one if them must be used to prevent correlation between features.

5.2.2. Autocorrelation Features

Since the underlying mechanisms of the heart are periodic it was thought to use features from the first autocorrelation peak. From the first peak, from now on referred to as 'the peak' in this paragraph, three different features were identified: the peak location, the peak height and the width at half the peak height. The peak location is expressed in the amount of sample delay of the autocorrelation signal. Translating this delay to time delay merely gives another way of extracting a heart rate value and so is expected to have a high correlation with the HRV heart rate feature. However, using the acquired peak location it is trivial to get the autocorrelation peak height. It is thought that the height of the peak may say something about the changes in periodicity of the heart. This is because the height of the peak is dependent on the similarity of the waveform of one period. Two consecutive PPG waveforms that are similar in period will show a higher peak than two from which one waveform is time-dilated. The third autocorrelation feature is the width of the peak at half the peak height. Similar to the height, intuitively the width of the first peak at half the height of that peak may give information to the classification process about the periodicity of the heart activity.

Initially, it was attempted to get the autocorrelation signals and features from the segments as they came from the Dataset stage in the pipeline. However it seems taking the autocorrelation from such long sequences results in useless signals that typically did not exhibit any fundamental frequency and thus had no first peak to detect. To remedy this problem the segments were again segmented into either 4 or 6 subsegments. Thus instead of utilizing 30 second segments 7.5 or 5 second subsegments were used to create the three different features. Shorter subsegments are less prone to changes in periodicity within the targeted 30 second window. Such changes will occur naturally when one stands up to take a walk, making their heart beat faster compared to the 15 seconds ago. The resulting features are taken from 6 subsegments per segment as this is expected to provide more robust and decisive features. So to clarify, after this stage each segment will have 18 features associated with it. Six for each of the three described features.

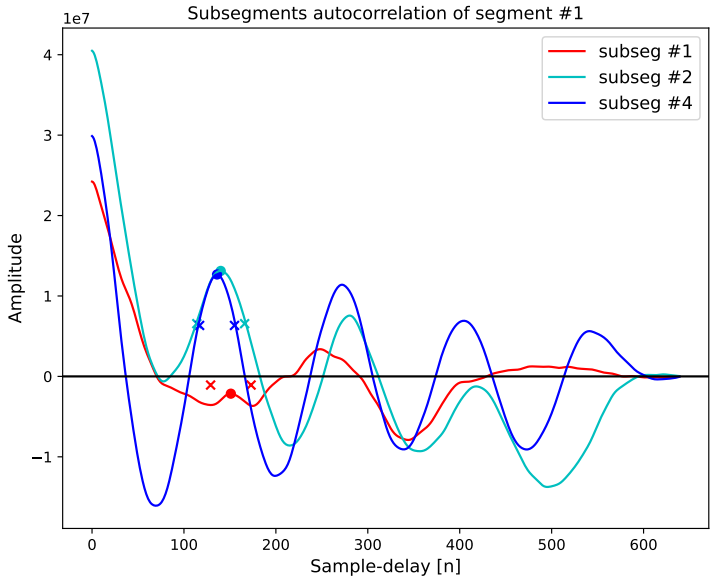

Figure 5.3: Some example autocorrelation signals from segment 1 previously shown in Fig. 5.2. The colored dots represent detected first peaks at the peak height while the crosses represent peak width at half the peak height. While the second and fourth subsegments produce an adequate peak to produce features from, the first subsegment exhibits non-periodicity.

As an example, Fig. 5.3 shows some non-ideal behavior from one of the subsegments (#1 in red). The first subsegment does not have a prominent first peak which points to a lack of periodicity in that subsegment. The incidental advantage of subsegmenting is the redundancy that it brings and thus improvement of these features was deemed unnecessary.

5.2.3. Miscellaneous Features

- Turning Point Ratio (**TPR**) is calculated as the sum of PPI's that are larger than both the previous and subsequent PPI. This is a measure of the 'randomness' of the PPI's.

- Wavelet features, using the wavelet transform a joint time-frequency representation of the signal is composed. The addition of the time aspect of this transform gives this an advantage of the Fourier transform when describing the irregularity of the signal. The coefficients are extracted using the Python `pywt` package, statistics of these coefficients like average, mean, root-mean-square, standard deviation and various percentiles are used as features.

5.2.4. Feature Selection

Two feature selection algorithms have been elaborated, the sequential feature selector and forest feature selector. Yet, only the forest feature selector was deemed useful for eventual implementation for reasons described in this paragraph.

The SFS was designed using a k-nearest neighbours classifier to fit and evaluate the features with. That classifier uses k amount of surrounding data points in the feature space to vote on the eventual label of the data point under inspection. This classifier was chosen because it is different from the classifiers used in the training and prediction stages of the classification pipeline. This is thought to prevent the SFS algorithm from selecting features that are solely useful to one of the classifiers later on used in the actual classification. Therefore none of the classifiers would already start with a bias in the feature space. However, investigation of this thought was eventually postponed under the assumption that feature selection will not make a large difference in the final accuracy of the model. Additionally, since the SFS algorithm on its own requires multiple iterations of the k-nearest neighbors classifier to make a selection, the algorithm is computationally expensive without providing quantified results on which features matter most. On the whole, the judgment is to not use the SFS.

The forest feature selector on the other hand is used. Using sklearns `ExtraTreesClassifier`, a forest of 1000 random trees is made. One thousand trees should be enough to be able to generalize the feature performances. During the development of the feature selector no time was invested in investigating adjustment of parameters. This means any leaf could end with a single entry and that the default depth was selected so that the construction of each tree continues until every leaf is pure.
Applying this selector to the full feature dataset produces percentages relative to the total importances per feature. These are ordered from highest to lowest percentage and plotted together with a cumulative sum in Fig. 5.4.

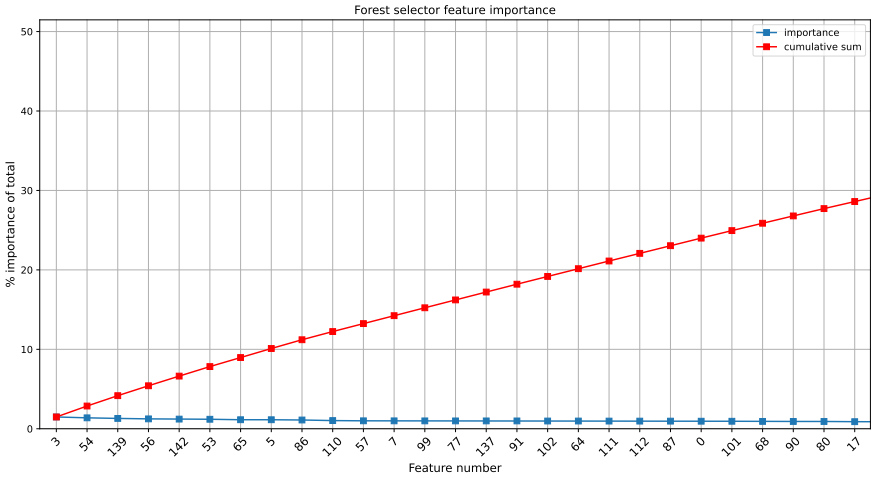

Figure 5.4: Individual feature importances as percentage of the total and the cumulative sum of the features, only the 27 most important features were plotted.

Feature type	Feature number	Description
HRV	3,5,0,7,17	nn50, nn20, sndd, hr, sd2
Wavelets	20-122	wavelet coefficients
Autocorrelation	137-142	half peak widths

Table 5.2: Feature importances as they appear in Fig. 5.4

From the full Fig. A.3, it can be seen that none of the features stand out in importance, creating a cumulative sum function that slowly flattens out. The point of this selection is to find the features which characterize the feature set, yet this selector suggests almost all features are of similar importance. To still reduce the amount of features, moving forward up to the first 25 features from Fig. 5.4 are used (until feature number 90 in the figure) to continue the classification pipeline. Here attention was payed to minimizing the amount of features without reducing the final accuracy.

5.3. Classifier Models and Tuning

In this section the design choices regarding classification models (classifiers) and their tuning is described. The hyperparameter tuning protocol has goals to keep the smallest amount of model complexity possible while prioritizing the best accuracy. This protocol was used for the tree-based and neural network classifiers:

1. The classifier is built into a Python function outputting the MCC score achieved for stratified 5-fold cross-validation.
2. A hyperparameter list is constructed with one changed hyperparameter per run.
3. Iteratively each run outputs the score.
4. Each run is reviewed and empirically the optimum value for that hyperparameter is chosen. If different settings show similar scores the setting that is computationally less complex is chosen.
5. Steps 2 through 4 are repeated until all hyperparameters have been tuned.

A hard constraint in the design of the following algorithms is the possibility the run the trained model on an ARM Cortex-M7 processor which will be used in the prototype. This constraint is a result of requirements **5** and **IV** found in Section 2.2. This prohibits the usage of external devices to make a prediction with the trained classification model and restricts the number of functions available to construct a model with. The following models all comply with these requirements and are implementable on the processor.

5.3.1. Tree-based Classifiers

The theory behind tree-based classifiers is described in Section 4.4.1, in this section this theory is applied to create a classifier that is subject to the constraints of this project. Initially a classifier using one decision tree was constructed but it performed performed poorly due to overfitting the data. In other words, no generalization on the type of features was made. As theory suggested we moved on to a forest ensemble to counteract that overfitting problem.
For this forest the sklearn class named `RandomForestClassifier` was used that constructs the forest in a random way for different subsets of the full dataset [34]. This classifier can only produce trees that are overfitting on at most a fraction of the entire dataset. This together with the voting approach of the ensemble remedies the overfitting on the entire training set. Since the dataset is unbalanced the class weight hyperparameter can prevent the forest from fitting only for the NSR class. In contrast, other forest hyperparameters such as depth, minimum samples per leaf and the fraction of samples to construct individual trees with are not easily set based on intuition. To optimize the forest classifier the hyperparameter tuning protocol was used. This resulted in the best accuracy MCC of 0.326 with the hyperparameter settings shown in Tab. 5.3.

5.3.2. Support Vector Machine

A classification approach using a Support Vector Machine (SVM) is considered since this function is directly supported by the CMSIS-DSP library [5] for ARM processors. The theory behind SVM's is described in Section

[5]Version 1.8.0; CMSIS website: https://www.keil.com/pack/doc/CMSIS/DSP/html/index.html

Balanced accuracy	0.660
MCC	0.326
Fraction subset	0.9
Amount of trees	200
Max depth	3
Minimum samples leaf	3
Class weight	balanced

Table 5.3: The accuracy metrics after hyperparameter tuning ended on the 5 hyperparameters in the lower part of the table.

4.4.2, in this section this theory is applied to create a classifier that is subject to the constraints of this project.

According to the functions available in the CMSIS-DSP library SVM's with Radial Basis Function (RBF) kernel functions are supported. This kernel function is chosen on the basis of its possibility of separating non-linear features and its successful application in other biomedical problems [35]. To initialize this SVM function on the processor the following arguments are needed: number of support vectors, dimension of vector space, intercept, array of dual coefficients, array of support vectors, class ID's and gamma. Which are all easily extracted from a model that is trained in the Python environment. These arguments can be interpreted as a collection of functions specifying the separation hyperplane and gamma, the area of influence a single data point had in the training dataset. The challenge is to train the model and achieve a desirable accuracy, copying the trained model to the processor is trivial using the tutorial provided on the ARM Developer website [36].

The first step in training the SVM model is split into training and test data, for all classifiers a 70/30 ratio of train/test data is chosen. From each feature in the training set the mean μ_n and standard deviation σ_n is determined, these values are then used for standard normalization of the entire set. This is done to prevent a so called 'data leak' from the training to the test set, if the mean and standard deviation of the entire set would be used in normalization the training set alone would likely not be mean centered and of unit variance an thus the test set is known to compensate for that. Something that is impossible when using a trained model in the real world where the training set was not part of the data for which a prediction is to be made. This functionality is implemented using the sklearn StandardScaler within a pipeline that automatically performs standard normalization every time the model is trained.

The sklearn SVM model accepts many arguments that influence the actual values and performance of the trained model. Since the dataset is unbalanced the `class_weight` argument is set to `balanced`. This corrects the importance of correctly classifying each class inversely to the size of that class in the training set. The regularization parameter `C` set the trade-off between maximization of the decision function's margin and correct classification of the training set. Large `C` will try to correctly classify every point at cost of a smaller margin, this will eventually cause the model to 'over fit' on the training set. While smaller `C` favors a larger margin at the cost of more miss-classifications. Finally `gamma` determines the radius of influence of a single data point, for higher `gamma` the radii are smaller and thus cause the decision function to be less sensitive to singled out points, only larger groups cause the function to change. Low values mean a large radius of influence and could mean that outliers have a larger influence on the function.
The optimal values for the arguments `C` and `gamma` are highly dependent on the feature set and should therefore be reconsidered for every addition of a new feature. The default values are `C`= 1.0 and `gamma`= $1/(N \times \text{std}(\mathbf{X}))$, which are a good starting point for further optimization. As a simple estimation of the range of parameters for which the model performs best an algorithm is written that plots the 5-fold cross-validation of the balanced accuracy of the test data within a range of parameters (see an example in Fig. 5.5). To prevent choosing an over-fitted model based on the highest value in the plot alone, the balanced accuracy of the training data is also plotted. At values where the difference is minimal and the test accuracy is acceptable useful parameters are found.

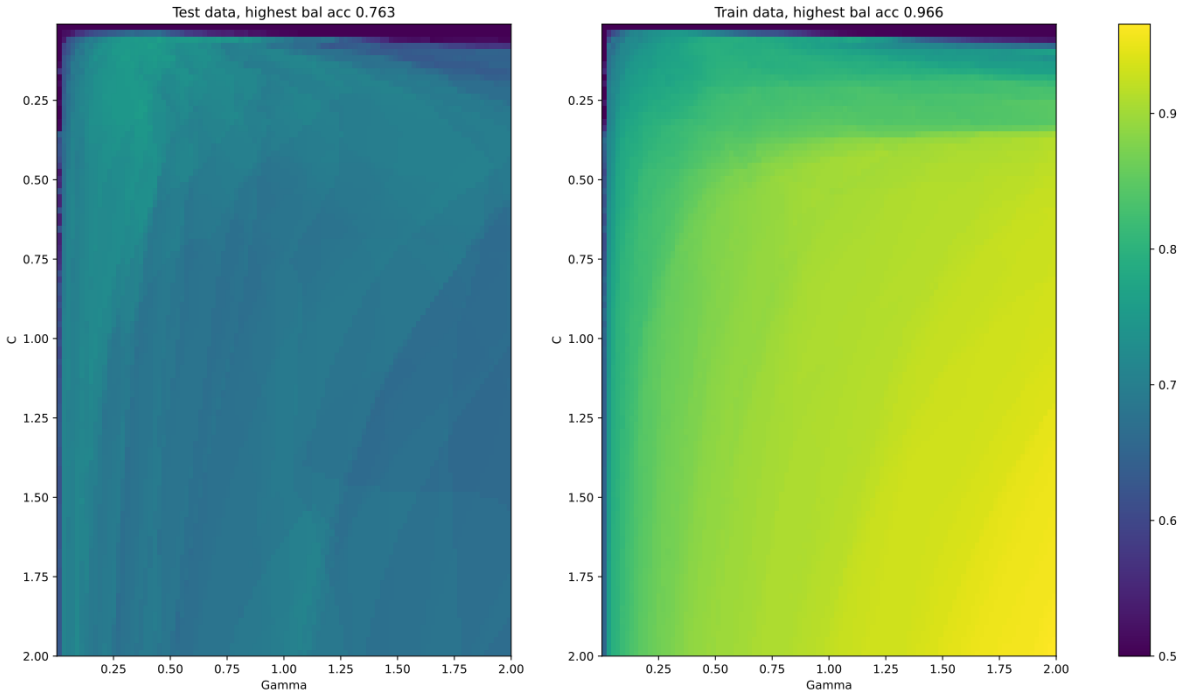

Figure 5.5: Heatmaps of the 5-fold cross-validation of the balanced accuracy for SVM's subject to different hyperparameters for `C` and `gamma`. On the left: the balanced accuracy of the test data. On the right: the balanced accuracy of the training data. Note the small band of hyperparameters for which the model performs best (high accuracy for test data without overfitting).

5.3.3. Neural Networks on Waveform

A DL approach is constructed based on [37], where a convolutional neural network (CNN) with a single long short-term memory (LSTM) layer is used to classify AF en NSR from PPG data with almost perfect precision. While a LSTM layer is not directly supported by the CMSIS-NN library [6] for the ARM processor, it is possible to construct them manually with elements from the NN and DSP CMSIS libraries, as ARM explains [38]. Other required functions are supported by the library, so when a useful model is constructed and trained within the Python environment it can easily be ported to the processor.

The model is composed out of max-pooling layers, a convolutional layer, a LSTM layer and a fully connected layer with one node and a sigmoid activation function to create the output. Max-pooling is used as a down-sampling operation in this context, the input signal is sampled at 128 Hz and after filtering should only contain frequency components up to 10 Hz so a sampling frequency of 20 Hz should suffice. In practice the the signal is down-sampled 6 times to come to a new sampling frequency of 21.3 Hz, this showed the best results after comparison with other cross-validated models at different down-sampling values. After that follows a convolution layer followed by another max-pooling layer that down-samples by a factor two. Which is then fed into the LSTM layer, which, using memory, can learn from temporal changes. Using that output a linear combination is made in the fully connected layer which is the final output after passing through the sigmoid function.

5.3.4. Neural Networks on Features

The final model that was constructed is a simple multilayer perceptron. In this case the neural network consists of an input layer that receives the features, three hidden layers with a dropout in between layers 1 and 2, and a single output node providing the class output label. The model was made using the Keras library for `Dense` and `Dropout` layers. Initially the perceptron was tried with all 142 features as input. This model started of learning extremely slow: the first few epochs seem to classify all segments either AF or NSR thus having a bad accuracy. The progress of such a model per epoch can be plotted using the simple accuracy metric seen in-line in Section 4.5.1. Such a plot is shown in Fig. 5.6 which also shows the initial stationary learning behavior until around epoch 50. The learning rate (of the gradient descent) at the start can actually be accel-

[6] Version 1.3.0; CMSIS website: https://www.keil.com/pack/doc/CMSIS/NN/html/index.html

erated relative to later epochs using Keras' `optimizers` and `LearningRateSchedule`. After tweaking these hyperparameters eventual settings were such that the learning rate starts at a high rate (0.4) and drops by 2% roughly every 6 epochs to prevent the algorithm from overshooting in later stages. Together with training the NN for 500 epochs it was possible to get the network learning. This was not the case for the 25 selected features however and thus the tuning for optimal hyperparameters for the amount of nodes and dropout rates in the layers was done using all 142 features. The final hyperparameters are shown in the Fig. 5.7.

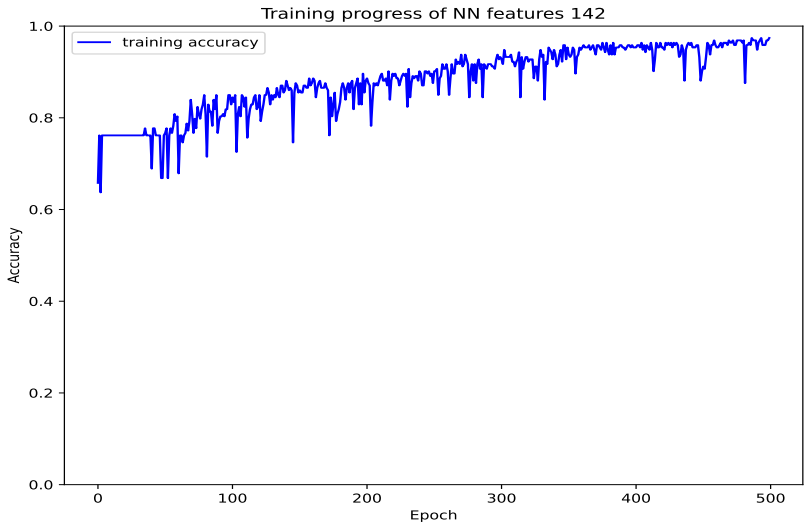

Figure 5.6: A typical training progress of the eventual NN using features from the PPG waveform. At the first few epochs the NN classifies most segments as NSR thus starting of with a value around 0.78, yet the network only starts to learn at around epoch 50.

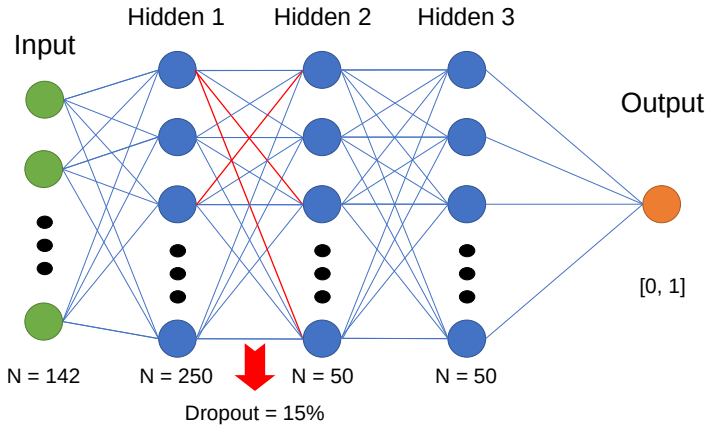

Figure 5.7: This figure shows the general shape of the multilayer perceptron and the amount of nodes for each layer. The total amount of parameters (inter node weights) in the model is 51,151.

6
Results

To investigate the possibilities of low intensity classification based on PPG signals, some common classification approaches and in terms of computational intensity and accuracy. This section describes challenges encountered during training and evaluation the results from the comparison of the models designed in Section 5.3.

6.1. Classification Model Selection

Four different classification models subject to requirements **5** and **IV** are designed in Section 5.3. To come to a conclusion which model is best to be used on the processor these all need to be compared. Whoever, since the models are designed with the implementation in the processor in mind it is assumed that their computational intensiveness will not be a limiting factor. Therefore in this section only the performance of these models are evaluated.

The balanced accuracy is chosen as the main comparison metric, since this is easiest to comprehend for this imbalanced dataset. The size of the feature set and the MCC score will also be part of the consideration since the latter must comply with requirement **2.2**. By repeating a 5-fold cross-validation multiple times with different split locations and taking the average of those model evaluations a fair estimation of the model performance is calculated. This way the 'folds' illustrated in Fig. 4.4 that normally do not overlap are shifted and made to overlap. Removing the chance of 'better' folding locations due to luck in this smaller dataset. The 5-fold cross-validation is evaluated for eight different initializations of the train-test splits using the `seed` argument. The averages for the accuracy metrics for these runs are collected in Tab. 6.1, the sums of all the conditions (TP, FP, FN, TN) are also included. It should be noted that this gives the false impression that the dataset was bigger, while in fact its size one-eighth of the sum of conditions in the table. Additionally the highest and lowest BA and MCC score from the individual cross-validations are given. These indicate the spread that was present in the calculation of the averages due to different initializations.

Model descr.	**TP**	**FP**	**FN**	**TN**	**BA**	**MCC**	**BA_{max}**	**BA_{min}**	**MCC_{max}**	**MCC_{min}**
Random Trees[1]	205	280	219	1232	0.649	0.285	0.649	0.595	0.372	0.210
NN features[1]	139	**169**	285	**1343**	0.608	0.244	0.644	0.553	0.347	0.109
NN waveforms[2]	107	355	317	1157	0.509	0.017	0.536	0.489	0.070	-0.017
SVM all[1]	295	445	129	1067	0.701	0.347	0.743	0.664	0.418	0.283
SVM hrv (first9)	264	606	160	906	0.611	0.184	0.625	0.587	0.208	0.148
SVM25 forest FS	304	367	120	1145	**0.737**	**0.412**	**0.773**	0.722	**0.477**	0.383
SVM10 forest FS	**305**	370	**119**	1142	**0.737**	**0.412**	0.761	**0.726**	0.457	**0.398**

Table 6.1: Model accuracy taken from 40 different runs of the respective model; each model was trained and evaluated on 8 different randomization seeds for 5-fold cross-validation. The metrics are calculated using the definitions given in Section 4.5.1.
[1] a machine learning approach, [2] a deep learning approach

The four top rows in Tab. 6.1 show the accuracies of the models designed in Section 5.3. The lowest performing model, the NN that takes the waveform as input, is the only deep learning approach. With a MCC score of almost zero it performs little better than randomly guessing and is therefore discarded from further

consideration. The machine learning approaches all perform considerably better, but the desired MCC score of 0.8 (requirement **2.2**) isn't achievable yet with the current setup of the project.

On the basis of the BA and MCC score the SVM is the best performing model. Even the lowest BA and MCC scores are higher or comparable to the averages of the other models. Therefore this classification method is chosen to be optimized further.

Using the feature selection methods discussed in Section 5.2.4 a variety of different feature sets applied to the SVM model are assessed. The model's hyperparameters are tuned again with each new feature set before extracting the model accuracies. The most interesting feature sets are the first nine HRV features (see Tab. A.1 for the exact feature names) and the 10 and 25 most significant features using the forest feature selector for which the results are shown on the bottom three rows of Tab. 6.1. With the exception of the model with only HRV features these models perform even better than original model, this observation will be discussed in Section 7.1. These two models, on average, perform almost equally. And are therefore equally good options. When their computational intensity is also considered however a clear optimal model comes to light, the **SVM with the top 10 features** from forest feature selection. Regarding the computational intensity the assumption is made that the intensity scales linearly with the amount of features required. Therefore the lowest number of features is chosen for the models of equal performance.

6.2. Evaluation of the Chosen Model with Improved Data

In the course of this project the responsibility for finding the peaks in the PPG signal, from which all of the HRV features are derived, was handed over to the signal processing subgroup. The improvements they made to the algorithm are discussed in Section 5.2.1, here the influence of the improvements of their work on the classification performance is discussed.

Using the same method as described in the previous section, the performance on the SVM classifier is evaluated again. Only this time using features derived from the processed signal and detected peaks from the other subgroup. The SVM classifier with all features and the chosen model, using the top 10 features, are trained, tuned and evaluated resulting in Table 6.2. Figures A.4 & A.5 are created to aid in the process of finding suitable hyperparameters. The most important results are the BA score of 0.853 and MCC score of 0.643 for the model best suited for implementation on the processor.

SVM with all features			
Metric	value	**Metric**	value
TP	376	**BA**	0.872
FP	194	$\mathbf{BA_{min}}$	0.857
FN	56	$\mathbf{BA_{max}}$	0.891
TN	1342	**MCC**	0.683
ACC	0.873	$\mathbf{MCC_{min}}$	0.653
Se	0.870	$\mathbf{MCC_{max}}$	0.712
Sp	0.874		

SVM with top 10 (forest) features			
Metric	value	**Metric**	value
TP	367	**BA**	0.853
FP	221	$\mathbf{BA_{min}}$	0.840
FN	65	$\mathbf{BA_{max}}$	0.861
TN	1315	**MCC**	0.643
ACC	0.855	$\mathbf{MCC_{min}}$	0.621
Se	0.850	$\mathbf{MCC_{max}}$	0.661
Sp	0.856		

Table 6.2: SVM classifiers using filtered data and peak locations from the signal processing subgroup. Left: model trained and tuned using all available features. Right: model trained and tuned using only the top 10 features from forest features selection method.

7

Discussion of Results

7.1. Tuning of Hyperparameters

Hyperparameters are parameters that describe how a classification model is to be trained. Choosing the wrong parameters can lead to models that are over-fitted (too aggressive) or under-fitted (to passive), but finding the wright parameters is often either a computationally exhaustive task or depends on the experience of the programmer. It is possible to test most of the possible hyperparameters for a classifier, the heatmap (Fig. 5.5) for tuning the SVM required 100×100 5-fold cross-validations (50,000 trained models) to give an area of possible parameters, which still required human interaction to come to a final value. For the evaluation of the final models (Section 6.1) eight 5-fold cross-validations were executed using the optimal hyperparameters predicted by the computationally exhaustive method, but using some manual variations higher accuracies were achieved. This indicates that the plot was too much descriptive of the situation specific to that splitting of the dataset. A solution would be to plot the results for the eight times 5-fold cross-validation, something that was not possible within the scope of the project.

The reason the SVM models with fewer features perform better that the one with all features in Tab. 6.1 is likely because they were tuned separately from each other. The human tuning after plotting the range of possible hyperparameters probably found different local maxima.

7.2. Limitation of the Current Dataset

One of the major issues experienced in extracting features from a PPG signal was the presence of motion artifacts as they were predicted in Section 3.2.1. These artifacts caused the dataset to be only partially useful and even the segments that were not labeled as 'noisy' sometimes contained noise artifacts. Fig. 5.2 shows two segments which should not contain artifacts since they are labeled 'AF' and 'NSR' but their shape is far from the ideal waveform illustrated in Fig. 3.2.

Ideally the model would first have been trained with data from ICU patients since these contain minimal motion artifacts, but since the dataset was recorded for another purpose this was not available. The focus could shift to optimizing for the presence of artifacts. The approach of starting with data with artifacts proved to be difficult, as the results have shown. The act of verification is harder when you cannot be sure if an error occurs due to the unfortunate presence of an artifact or if you made an error. This could have been avoided by selecting a different dataset. But, as to our current knowledge, there was no other dataset readily available that satisfied the requirements of having PPG data and labeled AF segments of relatively short duration (these requirements are discussed in Section 5.1). The other option for a dataset would be to record your own data from patients. But this would have been far out of the scope and (time) budget of the project.

While the NN approach using only the waveform proposed by [37] showed excellent performance (Sp=0.998 and Se=0.999) in their work. In the current study the results proved to be unusable. The main reason for this difference is likely the amount of training data that was available. Gotlibovych et al. collected 180 h of PPG data of which 36h were during AF. A large discrepancy compared to the two hours of data used to train our NN.

7.3. The Collection of PPG Data from a Non-stationary Source

A common approach to dealing with motion artifacts is to discard sections of recordings that contain them [19, 20]. Features derived from PPG with motion artifacts showed to be far less accurate than those derived from ECG data [24] and can therefore better be discarded than used in training and evaluating a classification model. The models used to select a classifier in Section 6.1 do not filter out artifacts and are therefore less likely to perform well in real life situations.
The scarce availability of datasets on PPG signals with AF annotations provided an insight. The most common method found in literature to combat this was to collect data yourself, something that could not be done in this study. The availability or creation of more data could help increase the accuracy of the current model, therefore future work should focus on increasing the dataset.

7.4. Comparison with State of the Art Approach

The results of the current study are compared to the study by Bashar et al. [20] that used the same dataset and classifies NSR and AF. This should be a fair comparison, since all parties had the same amount of data to train their algorithms with. This study uses a very simple decision function composed of an RMSSD and SampEn feature as a classification method. Using their provided performance Table A.2 is constructed. Which indicates that the current algorithm is outperformed by a large margin by a simple decision function.

After reviewing this result, three differences between the Bashar et al. study [20] and the current study are identified:

1. **Preprocessing** Their signals are filtered using a 6th order Butterworth bandpass with cutoff frequencies of 0.5 to 20 Hz versus a 2nd order Butterworth bandpass with cutoff frequencies of 0.2 to 10 Hz in our study.

2. **Peak detection** Bashar et al. uses "local peak and waveform envelope peak detection" whereas this study initially only uses local peak detection before switching to a more robust, yet still dissimilar approach for peak detection of the Signal Processing subgroup (see Section 5.2.1).

3. **Feature extraction** The RMSSD feature is slightly altered by dividing it by the mean of the PPI's of the current segment "because the variability and mean of HR values differ from subject to subject and segment to segment." [20]. Also some specifics for the SampEn feature are not mentioned such as the *embedding dimension* and *tolerance* making it challenging to repeat the results of the study.

In Fig. 7.1 the difference between the extracted features are presented. Notice how in the right panel the features are much less intertwined which indicates how a simple decision function is able to be used as a successful classifier. While in the left panel there exists a large overlap between the features, which means that more features are likely to be required to increase the classification accuracy.

Furthermore notice how the Sample Entropy features of Bashar et al. are centered around 0.5 for the NSR label. This is a sharp contrast to the current study where only few segments scored below 1.0. This means that the PPI's of Bashar et al. are much more similar, likely indicating differences in the results of the peak detection algorithm. Although the altered RMSSD feature (divided by the mean of PPI's) seems interesting it did not alter our results in a significant way once implemented. Thus the difference in the results of the studies likely originates from the peak detection algorithms, which would also be influenced by the filtering step.

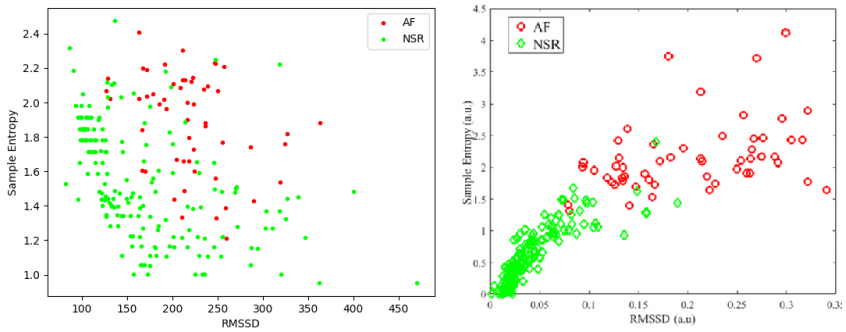

Figure 7.1: Extracted features per entry from current study (left) compared to Bashar et al. [20] (right).

8

Conclusion and Future Work

A definitive answer to the research question cannot be given yet, for this the results have to be validated first by implementing the selected method on the processor first. However, a conclusion is reached on the matter which classification method, subject to constraints of the processor, would be best suited to implement. For this a Support Vector Machine is chosen with ten features selected using a forest feature selection algorithm. Compared to the balanced accuracy and MCC scores of a tree-based classifier, a neural network using features and a neural network using the PPG waveform directly, the SVM performed best on all aspects. The accuracy metrics (balanced accuracy = 0.853, selectivity = 0.850, specificity = 0.856, MCC = 0.643) leave much to be desired though, an MCC score of 0.8 was required as a minimum accuracy for this device that is intended as a medical device. Anything lower that that would likely not suffice for any other applications.

The algorithm produces around four times as few False Negative (FN) as False Positive (FP) classifications. For a medical device this is vital. Letting the user walk freely with missed AF episodes is detrimental to the purpose of early detection. The occasional notification to make an ECG recording in case of a FP is far less harmful.

Future Work

Currently multiple aspects of the project are missing which were required according to our Programme of Requirements. Furthermore in the process of the project new insights have been gained that created additional directions or subjects for further research.

The most notable aspect missing from this report is the implementation of the trained classification model onto the processor. The strict time-constraint of the project unfortunately meant that implementation and validation of the model on the prototype itself had to be skipped. There are no reasons why the model should not work, the implementation has always been in consideration while designing. But within the set time there was no way to verify this.

Also data logging, essential for the application of the product, has not been designed. This is mainly because this should only have to work on the processor itself and since nothing got implemented on the processor, this part wasn't either. Designing this part is not hard and should be finished shortly after classification results are verified on the prototype.

A

Appendix B: Graphs and Statistics

A.1. Extended Graphs

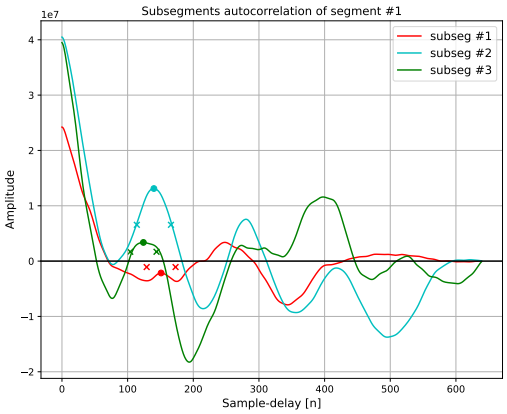

Figure A.1: The first three of six autocorrelation signals from segment 1 (see Fig. 5.2). The colored dots represent detected first peaks at the peak height while the crosses represent peak width at half the peak height.

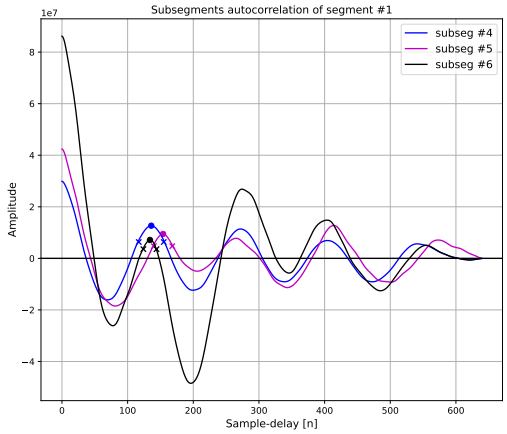

Figure A.2: Continuing with the last three autocorrelation signals from segment 1, see Fig. A.1

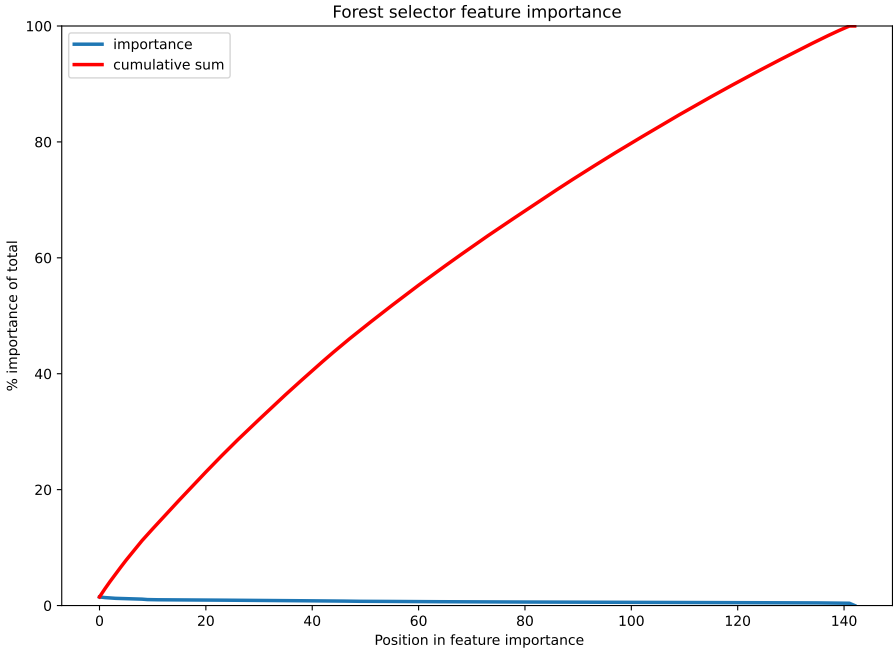

Figure A.3: The full result of the forest feature selector, the features are still ranked from highest to lowest importance yet not a lot of difference can be seen between features.

A.2. Extended Tables

Feature type	**Feature number**	**Description**
Time domain	0-10, 16-18	sndd, rmssd, sdsd, nn50, pnn50, nn20, pnn20, hr, sampen, shannonen, cosen, sd1, sd2, sdr
Miscellaneous	19-21, 123	tpr, auc, absauc, arburg
Wavelets	20-122	wavelet coefficients
Frequency	11-15	peak frequencies (lf, lf_peak, hf, hf_peak, lf_hf_ratio)
	124-142	6 times 3 features from autocorrelation of first peaks

Table A.1: Full layout of the features in the feature data matrix.

	Bashar et al. [20]	Current study
BA	0.951	0.852
ACC	0.955	0.854
MCC	0.857	0.638
Se	0.945	0.849
Sp	0.957	0.856

Table A.2: Comparison of current results with a study using the same dataset

A.3. Hyperparameter Tuning

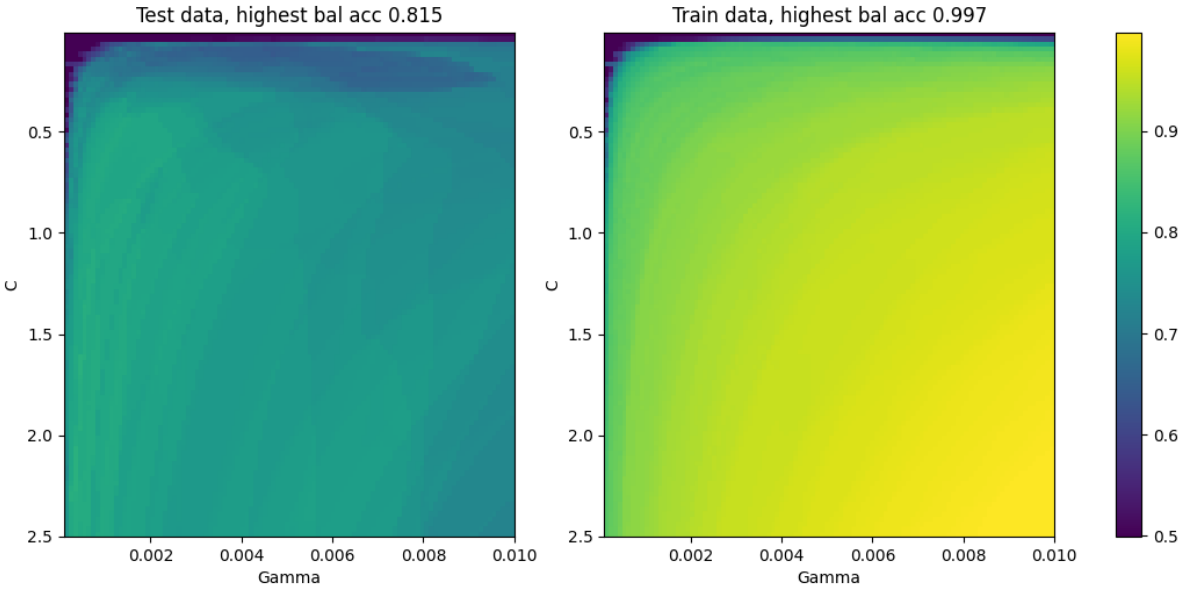

Figure A.4: Heatmaps of the 5-fold cross-validation of the balanced accuracy for SVM's with data supplied by the Signal processing subgroup subject to different hyperparameters for `C` and `gamma`. On the left: the balanced accuracy of the test data. On the right: the balanced accuracy of the training data.

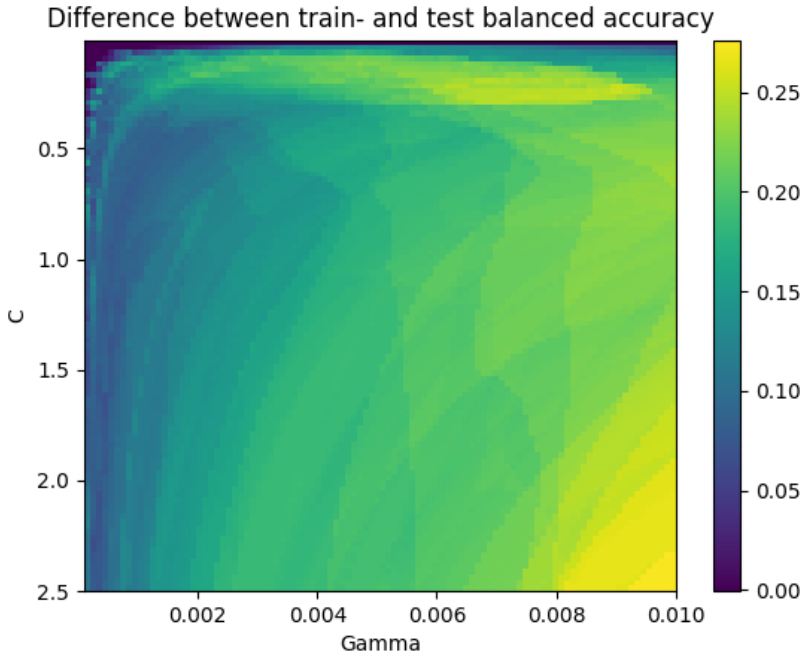

Figure A.5: By plotting the difference between the heatmaps in Fig. A.4 the amount of overfitting is visualized. The larger the difference between the balanced accuracy of model evaluated with the test data and the training data the more the model is only optimized the correctly classify the training data.

Bibliography

[1] GBD 2017 Disease & Injury Incidence and Prevalence Collaborators, "Global, regional, and national incidence, prevalence, and years lived with disability for 354 diseases and injuries for 195 countries and territories, 1990–2017: a systematic analysis for the global burden of disease study 2017," *Lancet*, vol. 392, pp. 1789–1858, 2018.

[2] S. S. Barold, "Willem einthoven and the birth of clinical electrocardiography a hundred years ago," *Cardiac Electrophysiology Review*, vol. 7, pp. 99–104, 2003.

[3] The Society for Cardiovascular Angiography and Interventions. What is atrial fibrillation (afib or af)? Accessed on: 15-06-21. [Online]. Available: http://www.secondscount.org/heart-condition-centers/info-detail-2/what-is-atrial-fibrillation-afib-af#.YMyDk2jzOUk

[4] B. L. Mitchell. (2021) Atrial fibrillation. Accessed on: 16-06-21. [Online]. Available: https://www.merckmanuals.com/professional/cardiovascular-disorders/arrhythmias-and-conduction-disorders/atrial-fibrillation

[5] M. Elgendi, *PPG Signal Analysis: An Introduction Using MATLAB*. CRC Press, Taylor & Francis Group, 2021. [Online]. Available: https://books.google.nl/books?id=EteFtgEACAAJ

[6] R. Couceiro, P. Carvalho, R. P. Paiva, J. Henriques, and J. Muehlsteff, "Detection of motion artifacts in photoplethysmographic signals based on time and period domain analysis," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2012, pp. 2603–2606.

[7] S. K. Bashar, D. Han, S. Hajeb-Mohammadalipour, E. Ding, C. Whitcomb, D. D. McManus, and K. H. Chon, "Atrial fibrillation detection from wrist photoplethysmography signals using smartwatches," *Scientific reports*, vol. 9, no. 1, pp. 1–10, 2019.

[8] D. Pollreisz and N. TaheriNejad, "Detection and removal of motion artifacts in ppg signals," *Mobile Networks and Applications*, pp. 1–11, 2019.

[9] H. Han and J. Kim, "Artifacts in wearable photoplethysmographs during daily life motions and their reduction with least mean square based active noise cancellation method," *Computers in biology and medicine*, vol. 42, no. 4, pp. 387–393, 2012.

[10] N. Selvaraj, A. Jaryal, J. Santhosh, K. K. Deepak, and S. Anand, "Assessment of heart rate variability derived from finger-tip photoplethysmography as compared to electrocardiography," *Journal of medical engineering & technology*, vol. 32, no. 6, pp. 479–484, 2008.

[11] V. Fuster, L. E. Rydén, R. W. Asinger, D. S. Cannom, H. J. Crijns, R. L. Frye, J. L. Halperin, G. N. Kay, W. W. Klein, S. Lévy *et al.*, "Acc/aha/esc guidelines for the management of patients with atrial fibrillation: a report of the american college of cardiology/american heart association task force on practice guidelines and the european society of cardiology committee for practice guidelines and policy conferences (committee to develop guidelines for the management of patients with atrial fibrillation) developed in collaboration with the north american society of pacing and electrophysiology," *Journal of the American College of Cardiology*, vol. 38, no. 4, pp. 1266–1266, 2001.

[12] C. A. Groh, M. Faulkner, S. Getabecha, V. Taffe, G. Nah, K. Sigona, D. McCall, M. T. Hills, K. Sciarappa, M. J. Pletcher *et al.*, "Patient-reported triggers of paroxysmal atrial fibrillation," *Heart rhythm*, vol. 16, no. 7, pp. 996–1002, 2019.

[13] A. Hansson, B. Madsen-Härdig, and S. B. Olsson, "Arrhythmia-provoking factors and symptoms at the onset of paroxysmal atrial fibrillation: a study based on interviews with 100 patients seeking hospital assistance," *BMC cardiovascular disorders*, vol. 4, no. 1, pp. 1–9, 2004.

[14] T. Pereira, N. Tran, K. Gadhoumi, M. M. Pelter, D. H. Do, R. J. Lee, R. Colorado, K. Meisel, and X. Hu, "Photoplethysmography based atrial fibrillation detection: a review," *NPJ digital medicine*, vol. 3, no. 1, pp. 1–12, 2020.

[15] L. F. Chaparro, *Signals and Systems using Matlab*. Oxford: Academic Press, 2015, p. 196.

[16] J. Watt, R. Borhani, and A. Katsaggelos, *Machine Learning Refined: Foundations, Algorithms, Applications*. Cambridge University Press, 2018.

[17] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, pp. 1–13, 2020.

[18] H. Dong, "Cassey2016 UMass Simband Dataset," [Matlab] https://github.com/Cassey2016/UMass_Simband_Dataset, 2021, accessed on 15-6-21.

[19] D. Han, S. K. Bashar, F. Mohagheghian, E. Ding, C. Whitcomb, D. D. McManus, and K. H. Chon, "Premature atrial and ventricular contraction detection using photoplethysmographic data from a smartwatch," *Sensors*, vol. 20, no. 19, p. 5683, 2020.

[20] S. K. Bashar, D. Han, S. Hajeb-Mohammadalipour, E. Ding, C. Whitcomb, D. D. McManus, and K. H. Chon, "Atrial fibrillation detection from wrist photoplethysmography signals using smartwatches," *Scientific reports*, vol. 9, no. 1, pp. 1–10, 2019.

[21] A. Oxford. The next sensor: how your future wearable will know how healthy you are. Accessed on: 18-06-21. [Online]. Available: https://www.stuff.tv/features/next-sensor-how-your-future-wearable-will-know-how-healthy-you-are

[22] S. Simband. Simsense: 2nd generation sensor overview. Accessed on: 15-06-21. [Online]. Available: https://www.simband.io/documentation/sensor-module-documentation/simsense/simsense-2nd-gen.html

[23] F. Shaffer and J. Ginsberg, "An overview of heart rate variability metrics and norms," *Frontiers in public health*, vol. 5, p. 258, 2017.

[24] L. M. Eerikäinen, A. G. Bonomi, F. Schipper, L. R. Dekker, R. Vullings, H. M. de Morree, and R. M. Aarts, "Comparison between electrocardiogram-and photoplethysmogram-derived features for atrial fibrillation detection in free-living conditions," *Physiological measurement*, vol. 39, no. 8, p. 084001, 2018.

[25] N. Selvaraj, A. Jaryal, J. Santhosh, K. K. Deepak, and S. Anand, "Assessment of heart rate variability derived from finger-tip photoplethysmography as compared to electrocardiography," *Journal of medical engineering & technology*, vol. 32, no. 6, pp. 479–484, 2008.

[26] P. van Gent, H. Farah, N. van Nes, and B. van Arem, "Analysing noisy driver physiology real-time using off-the-shelf sensors: Heart rate analysis software from the taking the fast lane project," *Journal of Open Research Software*, vol. 7, no. 1, 2019.

[27] L. Chen, A. T. Reisner, and J. Reifman, "Automated beat onset and peak detection algorithm for field-collected photoplethysmograms," in *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2009, pp. 5689–5692.

[28] S.-M. Shan, S.-C. Tang, P.-W. Huang, Y.-M. Lin, W.-H. Huang, D.-M. Lai, and A.-Y. A. Wu, "Reliable ppg-based algorithm in atrial fibrillation detection," in *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2016, pp. 340–343.

[29] S. Dash, K. Chon, S. Lu, and E. Raeder, "Automatic real time detection of atrial fibrillation," *Annals of biomedical engineering*, vol. 37, no. 9, pp. 1701–1709, 2009.

[30] S. Nattel, "New ideas about atrial fibrillation 50 years on," *Nature*, vol. 415, no. 6868, pp. 219–226, 2002.

[31] Electrophysiology, Task Force of the European Society of Cardiology the North American Society of Pacing, "Heart rate variability: standards of measurement, physiological interpretation, and clinical use," *Circulation*, vol. 93, no. 5, pp. 1043–1065, 1996.

[32] A. B. Ciccone, J. A. Siedlik, J. M. Wecht, J. A. Deckert, N. D. Nguyen, and J. P. Weir, "Reminder: Rmssd and sd1 are identical heart rate variability metrics," *Muscle & nerve*, vol. 56, no. 4, pp. 674–678, 2017.

[33] F. J. Wesselius, M. S. van Schie, N. M. De Groot, and R. C. Hendriks, "Digital biomarkers and algorithms for detection of atrial fibrillation using surface electrocardiograms: A systematic review," *Computers in Biology and Medicine*, p. 104404, 2021.

[34] Scikit Learn, "sklearn.ensemble.RandomForestClassifier," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier, 2021, accessed on 15-6-21.

[35] J. S. Paiva, J. Cardoso, and T. Pereira, "Supervised learning methods for pathological arterial pulse wave differentiation: a svm and neural networks approach," *International journal of medical informatics*, vol. 109, pp. 30–38, 2018.

[36] ARM Ltd., "AI and Machine Learning on Arm: Implement classical ML with Arm CMSIS-DSP libraries," accessed on 16-6-2021. [Online]. Available: https://developer.arm.com/solutions/machine-learning-on-arm/developer-material/how-to-guides/implement-classical-ml-with-arm-cmsis-dsp-libraries/train-an-svm-classifier-with-scikit-learn

[37] I. Gotlibovych, S. Crawford, D. Goyal, J. Liu, Y. Kerem, D. Benaron, D. Yilmaz, G. Marcus, and Y. Li, "End-to-end deep learning from raw sensor data: Atrial fibrillation detection using wearables," *arXiv preprint arXiv:1807.10707*, 2018.

[38] ARM Ltd., "AI and Machine Learning on Arm: Implement classical ML with Arm CMSIS-DSP libraries," accessed on 16-6-2021. [Online]. Available: https://developer.arm.com/solutions/machine-learning-on-arm/developer-material/how-to-guides/converting-a-neural-network-for-arm-cortex-m-with-cmsis-nn/single-page