Politecnico di Torino

Polithecnic of Turin

Master on ICT for Smart Societies
A.Y. 2024/2025
Graduation Session October 2025

Optimized Cuffless Blood Pressure Measurement Using ECG, PPG and Linear Regression

Supervisors:

Professor Michela Meo
Professor Guido Pagana

Candidate:

Luca Barotto

Abstract

Hypertension is a major risk factor for cardiovascular disease (CVD), which remains the leading cause of mortality worldwide. Continuous and accurate monitoring of blood pressure (BP) plays a fundamental role not only in early diagnosis but also in patient therapeutic management and the prevention of associated complications. However, currently used methods have significant limitations: invasive systems, while highly accurate, can only be used in hospital settings and involve risks and discomfort; noninvasive cuff-based systems, on the other hand, provide only intermittent readings and do not allow dynamic BP analysis during daily activities. To overcome these limitations, in recent years there has been growing interest in cuffless techniques that exploit physiological signals easily acquired through wearable sensors. This thesis explores a methodology for continuous and noninvasive estimation of BP based on combined electrocardiogram (ECG) and photoplethysmogram (PPG) analysis. From these signals, two key parameters are extracted: heart rate (HR) and pulse transit time (PTT). They are known in the literature for their correlation with variations in blood pressure.

The work introduces several significant innovations: the use of wearable Shimmer devices to acquire signals from subjects under controlled conditions, as an alternative to data sets available online (MIMIC and others), and the design of a processing pipeline including signal preprocessing, automatic peak detection, and robust extraction of HR and PTT. Subsequently, through the application of regression techniques, a mathematical relationship was defined between these parameters and the reference BP values obtained using calibrated devices. The expected results include the development of a reliable framework capable of estimating blood pressure in real time with an error within the thresholds established by international guidelines (AAMI/ISO/ESH), the validation of the method against clinical standards and the optimization of algorithms to reduce motion artifacts.

Looking ahead, such an approach could contribute to the evolution of wearable healthcare technologies, allowing continuous, discreet, and personalized BP monitoring even outside the clinical setting. This would represent an important step in the treatment of hypertension and the prevention of CVD, improving the quality of life of patients and reducing the overall social and health impact of these diseases. Therefore, an aspect of the project intended for the integration of this type of algorithm on wearable devices: the European Persimmon project and shimmer3.

Acknowledgements

Before proceeding with the description of my thesis project, I would like to take a moment to express my sincere gratitude to those who have been close to me throughout my university career and who have enabled me to achieve this important milestone.

First of all, I would like to thank Professor Guido Pagana and Dr Debora Beneduce who, with their expert guidance, have supported me in the realisation of this project. I am grateful to them for their patience, availability and rigorous precision throughout the entire thesis writing process. I would like to thank the Links Foundation, which offered me the opportunity to develop this project in a dynamic and stimulating environment, rich in innovation.

Infinite thanks go to my family: my mum, dad, aunt and Alessandro. They have always supported me, both financially and morally, backing every decision I have made, starting with my choice of studies, and offering me valuable advice along the way.

A special thank you and a fond thought go to Alice, my girlfriend, who with her infinite patience has constantly supported and put up with me, sharing both the difficult and the happier moments, for which I can never thank her enough.

I would like to thank my classmates Marco, Simone, Alessandro, Federico, Augusta, Yasmin, Matteo, and Mattia, with whom I shared projects, joys, and difficulties during this intense journey.

Thanks also go to my former classmates from my bachelor's degree, Jessica, Fabrizio, Gabriele, Ottavia (Hovawart's partner) and Valentina, with whom I shared unforgettable moments.

I would like to thank the "Group of Three Marco": "The Lion", "Lord" and "Lugghi", with whom I have maintained a friendship since high school. A special mention goes to Lugghi, affectionately known as "il Bro", for his wise life advice.

I would like to thank my dear friends Daniel, Edoardo, Alessandro, Lorenzo, Marius (known as "The Gentlemen"), Marica, Cristina and Ferri for the happy moments we shared and for always being by my side.

Finally, I would like to thank my football teammates: Was, Stefano, Luca and Saher, and my karate teammates: Ivan, Gabriele, Viviana and my Master Riccardo. I thank them for providing me with a stimulating outlet and for the precious opportunity to share my sporting

passions with them.

To conclude, I dedicate this work to myself: a recognition of the sacrifices and tenacity that, despite the difficulties, have allowed me to achieve this goal with determination, proving to me that perseverance is the key to achieving one's objectives.

Contents

Acronyms table

Acronym	Description
AAMI	Association for the Advancement of Medical Instrumentation
BP	Blood Pressure
CO	Cardiac Output
CVD	Cardiovascular Disease
DBP	Diastolic Blood Pressure
ECG	Electrocardiogram
ESH	European Society of Hypertension
HR	Heart Rate
IoT	Internet of Things
ISO	International Organization for Standardization
LED	Light Emitting Diode
MAP	Mean Arterial Pressure
MAE	Mean Absolute Error
MIMIC	Medical Information Mart for Intensive Care
ML	Machine Learning
PAT	Pulse Arrival Time
PERSIMMON	Personalised Smart Patch for Multimodal Monitoring
PP	Pulse Pressure
PPG	Photoplethysmogram
PTT	Pulse Transit Time
RMSE	Root Mean Square Error
SBP	Systolic Blood Pressure
SHIMMER	Sensing Health with Intelligence, Modularity, Mobility and Experimental Reusability
SVM	Support Vector Machine

Table 1: List of acronyms used in the project, in alphabetical order.

1 Introduction

The Cardiovascular disease (CVD) represents a major public health challenge globally. Among the various risk factors, hypertension stands out as the most modifiable and, at the same time, one of the most common causes of serious complications. For this reason, accurate and continuous blood pressure measurement plays a strategic role, both in prevention and clinical management.

1.1 Clinic context: the problem to be addressed

CVDs are currently the leading cause of mortality and morbidity worldwide [1] [2]. According to the World Health Organization (WHO), over 17 million people die from these diseases every year, and epidemiological projections estimate that this number is expected to increase in the coming decades.

The causes of this growth are multiple: the progressive aging of the population, the increasing prevalence of obesity and diabetes mellitus, and the spread of sedentary lifestyles and unhealthy eating habits. Added to this is the impact of socioeconomic inequalities, which influence access to early diagnosis, adequate treatment, and prevention programs. In this scenario, arterial hypertension emerges as the main modifiable risk factor, with a significant epidemiological impact. It is closely linked to the onset of acute events such as ischemic and hemorrhagic strokes and myocardial infarction, but also to chronic complications such as heart failure and progressive renal failure [3]. Despite being a well-known and theoretically easily measurable risk factor, hypertension continues to pose a global public health challenge: it is estimated that approximately one in three adults worldwide has hypertension, and less than half receive adequate treatment. Accurate and continuous monitoring of blood pressure therefore plays a crucial role, not only to identify abnormalities early, but also to evaluate the effectiveness of pharmacological treatments, adapt therapies based on blood pressure trends and reduce the risk of cardiovascular and renal complications [4] [5]. However, despite the widespread availability of antihypertensive drugs, blood pressure control (BP monitoring) in the population remains inadequate: a significant proportion of individuals are unaware of their condition, while among those who have received a diagnosis, a significant portion do not adhere to treatment or are not regularly monitored [6] [7].

This combination of late diagnosis, poor adherence, and intermittent monitoring contributes to maintaining the high disease burden, underscoring the need for innovative solutions for blood pressure control at the individual and population levels.

1.2 Limitations of Traditional Methods

Currently, BP measurement is performed primarily using two approaches: non-invasive, intermittent methods, and invasive, continuous methods. However, both have significant limitations that affect their applicability and effectiveness.

The cuff sphygmomanometer [8] [9], based on the auscultatory method (with a stethoscope and detection of Korotkoff sounds) or the oscillometric method (use of electronic pressure sensors), represents the gold standard in clinical measurement of ambulatory and home blood pressure measurements. Although generally reliable, this method provides only intermittent measurements, usually at intervals of several minutes or hours [10]. This means that the dynamics of blood pressure throughout the day, influenced by physical activity, stress, sleep, or hormonal changes, is not fully captured. Furthermore, the need to inflate the cuff can cause discomfort, especially in elderly or frail patients, and limits its use in continuous monitoring settings. Figure 1 illustrates its behavior.

Figure 1: Sphygmomanometer

In the other extreme are invasive methods, such as arterial catheterization, which allow for direct and continuous blood pressure measurement using a transducer connected to an arterial access. These systems offer maximum accuracy and are the clinical gold standard, especially in intensive care settings or critically ill patients. However, the invasive nature of the procedure involves significant risks, such as infection, thrombosis, bleeding, and pain, as well as the need for highly specialized personnel and dedicated hospital facilities [11] [12]. For these reasons,

these methods cannot be applied on a large scale or adopted for daily monitoring of the general population.

There is therefore a "technological gap" between these two approaches: on the one hand, a non-invasive but fragmented method; on the other, an accurate but risky and impractical method. Bridging this gap represents one of the most urgent and promising challenges in the field of cardiovascular monitoring like the figure 2.

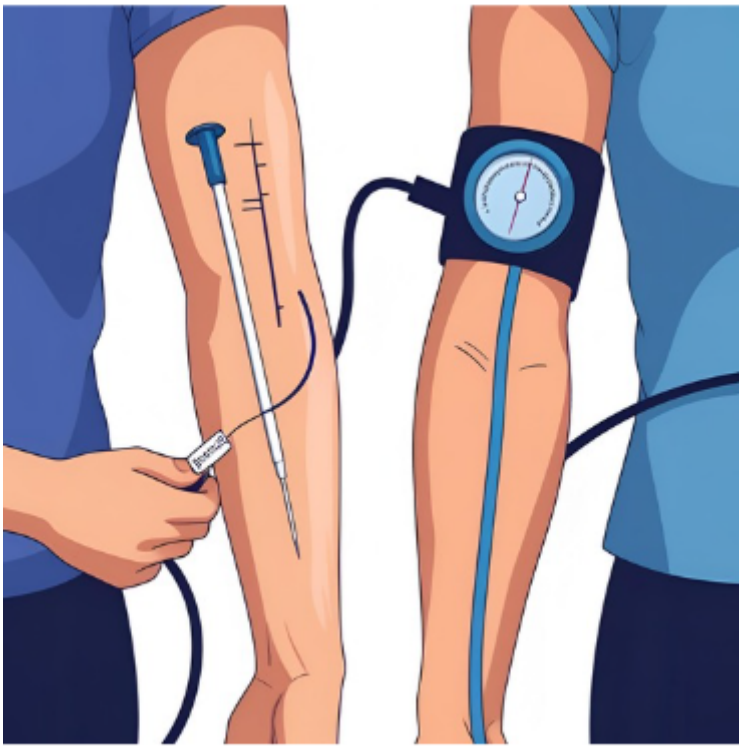


Figure 2: Difference between invasive and noninvasive method for the cardiovascular monitoring.

1.3 The Need for Cuffless Solutions

In recent years, scientific research has increasingly focused on developing alternative methods for monitoring blood pressure, with the aim of developing non-invasive, reliable, wearable solutions capable of providing real-time data [13] [14]. This type of technology could revolutionize the treatment of hypertension, allowing monitoring not only in the clinical setting but also at home and in daily life, improving the quality of patient's life and reducing healthcare costs [15].

The idea behind these approaches is to exploit physiological signals already widely used in the medical field, which are capable of indirectly reflecting on variations in blood pressure. In particular, parameters extracted from signals such as the electrocardiogram (ECG) and photoplethysmogram (PPG) have shown significant potential for cuffless BP estimation.

1.3.1 ECG and PPG for Smarter Blood Pressure Tracking

The ECG is a well-established diagnostic tool has been used for decades to assess the electrical activity of the heart. The ECG waveform provides valuable information on heart rate and rhythm, as well as the propagation of electrical impulses through the different cardiac chambers. Detection of the R peak, for example, provides a stable and easily identifiable time reference, which can be used as a starting point for temporal calculations related to the cardiac cycle [16].

PPG, on the other hand, is a signal obtained through an optical sensor that measures volumetric variations in blood flow at the peripheral level, typically on the finger or wrist. This signal is now widely used thanks to its integration in commonly used wearable devices, such as smartwatches and fitness trackers, which use it to estimate heart rate and oxygen saturation [17] [18]. Its ease of acquisition, non-invasive nature, and increasing miniaturization of sensors make it an ideal candidate for continuous monitoring applications. The combination of ECG and PPG allows the calculation of parameters such as heart rate (HR) and pulse transit time (PTT), both of which correlate with blood pressure dynamics. Recent studies have shown that changes in blood pressure are reflected in measurable changes in PTT, paving the way for predictive models capable of estimating blood pressure values from these signals. Although these approaches are still being validated and present significant challenges such as the need to reduce motion artifacts and ensure accurate calibration. In fact, they represent one of the most promising prospects for the future of cardiovascular monitoring [19] [20] [21].

1.4 Thesis Objectives

In this context, this thesis aims to explore and develop a method for the non-invasive and continuous estimation of blood pressure, based on the combined analysis of ECG and PPG signals recorded with the reference Shimmer devices. Using these systems allows us to work with "real" data, and to evaluate the method's practical transferability to clinical and everyday settings. A detailed description of the Shimmer devices' hardware and software features, as well as their acquisition methods, will be provided in the following chapters, particularly in the section dedicated to methodology (Chapter: 3) The entire work has been carried out at the LINKS Foundation. It aims to be a driver of innovation with high social and economic impact, based on a solid integration of cutting-edge technological knowledge and a multidisciplinary vision. Through a practical approach that extends from design to prototyping, and thanks to a dense network of collaborations with academic institutions and industrial entities both nationally and internationally, LINKS creates a fertile ecosystem where diverse skills meet and interconnect. The Foundation's guiding vision recognizes the intimate connection between technological progress, society, and the environment, aiming to address the structural challenges of our time from a systemic perspective. This ecosystemic perspective combines technological innovation with the principles of sustainability, equity, and territorial development, with the

ultimate goal of generating tangible benefits for the community.

The LINKS Foundation therefore acts as a bridge between research and practical application, transforming advanced ideas into real solutions that meet the needs of contemporary society, always respecting a harmonious balance between humanity, technology, and nature. The Foundation thus represents a model of how technical and scientific excellence can be combined with a responsible vision of progress, where innovation becomes a tool for the shared improvement of living conditions and the protection of the common good [22]. The main idea of the project is the acquisition of signals via wearable sensors, processing to extract HR and PTT parameters, and the application of regression techniques to establish a quantitative relationship with reference blood pressure values, to estimate the trend over time of diastolic and systolic pressure: DBP and SBP values. Then, the future development is the possibility to use the algorithm on portable devices for continuous and real-time monitoring.

The work is therefore organized as follows: Chapter 2 is dedicated to the state of the art, describing the physiological foundations, traditional methods, and emerging techniques for monitoring blood pressure. Chapter 3 illustrates the proposed solution, including the sensors, acquisition, pre-processing, and parameter extraction methods. Chapter 4 presents the experimental results obtained and their evaluation, while Chapter 5 is dedicated to a critical discussion of the results and future perspectives. Lastly, Chapter 6 is dedicated to the conclusions.

2 State Of The Art: Materials and Methods

The study of blood pressure and the methodologies for its non-invasive estimation requires a solid understanding of both basic physiological aspects and biomedical signal acquisition and processing techniques. This chapter provides a critical and structured review of the existing literature, with the aim of outlining the state of the art for existing blood pressure monitoring techniques.

The chapter is organized into five main sections.

- **Section 2.1 – Physiological Background**: The fundamentals of the cardiovascular system will be reviewed, with particular attention to the definition of blood pressure and its main components: systolic blood pressure (SBP), diastolic blood pressure (DBP), and mean arterial pressure (MAP). The relationship between blood pressure, cardiac output, and peripheral resistance will also be discussed, highlighting how these parameters are influenced by physiological and pathological factors.

- **Section 2.2 – Biomedical Signals of Interest**: This section focuses on the two key signals used in cuffless approaches: the electrocardiogram (ECG) and the photoplethysmogram (PPG). The principles of signal generation, their morphology and fiducial points of interest (e.g. the R-peak in the ECG and the foot or peak in the PPG) will be introduced. The main factors that can degrade signal quality, such as physiological noise and motion artifacts, and the strategies commonly adopted to mitigate them will also be analyzed.

- **Section 2.3 – Blood Pressure Measurement Techniques**: This section presents current blood pressure measurement methodologies. We will begin with invasive methods, considered the gold standard for accuracy but limited by their high invasiveness, and then analyze non-invasive techniques based on cuff-based devices (manual sphygmomanometer, oscillometric). Finally, we will introduce the concept of cuffless monitoring, discussing its advantages and challenges compared to established clinical practices.

- **Section 2.4 – Innovative Approaches for Blood Pressure Estimation**: This section focuses on methods that take advantage of parameters derived from ECG and PPG signals, specifically heart rate (HR), pulse transit time (PTT), and pulse arrival time (PAT). We will analyze the main regression models (linear, polynomial, logarithmic) and the most recent techniques based on machine learning and deep learning (SVM, Random Forest, CNN, LSTM). Multimodal approaches that combine different biomedical signals will also be described, as well as the main open issues, including the need for calibration, model generalization, and high interindividual variability.

- **Section 2.5 – Datasets and Validation**: The final section will focus on databases available in the literature and performance evaluation criteria. The most popular public databases, such as MIMIC, and proprietary data collections will be discussed. Commonly used validation metrics (RMSE, MAE, Bland-Altman) and international standards (AAMI, BHS, ISO) will be presented. Finally, a comparison of the main published works will be provided, accompanied by a comparative table summarizing datasets, algorithms, and performance, with a critical discussion of the current state of research and the gaps that remain to be filled.

2.1 Physiologic Background

This section presents the basic physiological principles necessary to understand the functioning of the cardiovascular system and the central role of blood pressure. After a general overview of the organization of the circulatory system and its two main components: systemic and pulmonary circulation, fundamental concepts related to the dynamics of blood flow will be introduced.

Particular attention will be paid to the definition and description of the main components of blood pressure: systolic blood pressure (SBP), diastolic blood pressure (DBP), mean arterial pressure (MAP), and pulse pressure (PP). These parameters form the basis for the correct interpretation of hemodynamic conditions and are key elements for clinical assessment.

2.1.1 Cardiovascular System

The cardiovascular system is the set of organs that allow blood to circulate within our body to provide cells with the necessary nutrients and to eliminate carbon dioxide and other waste products [23]. It is a closed system of blood vessels made up of arteries, capillaries, and veins. In particular, arteries are tubes with a thick layer of elastic tissue and muscle fibers that receive blood from the heart, a muscle that functions as a true blood pump. The arteries branch out into thinner blood vessels until they reach arterioles, through which blood is conveyed into capillaries, small tubes with thin walls permeable to nutrients, gases and waste products. The capillaries carry blood to small blood vessels, the venules, which converge to form veins that return blood to the heart [23]. The graphical representation of the system is shown in figure 3

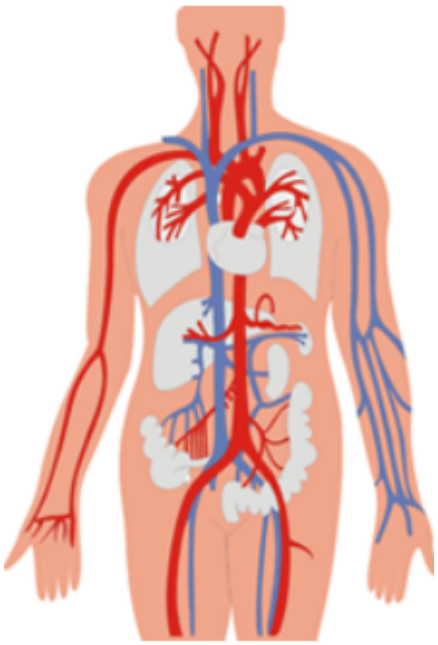


Figure 3: Representation of the Cardiovascular System

The blood circulation goes with the systemic and pulmonary circulation (figure 4). In particular, systemic circulation, or greater circulation, begins in the left ventricle and distributes oxygen-rich blood to all parts of the body via the aorta and its branches. In parallel, the pulmonary circulation transports carbon dioxide-laden blood from the right ventricle to the lungs, where gas exchange occurs [24]. These two, closely integrated, form a closed circuit that maintains the body's homeostasis. The heart acts as a tireless pump, while the blood vessels form an intelligent distribution network, capable of self-regulating according to the needs of the various tissues.

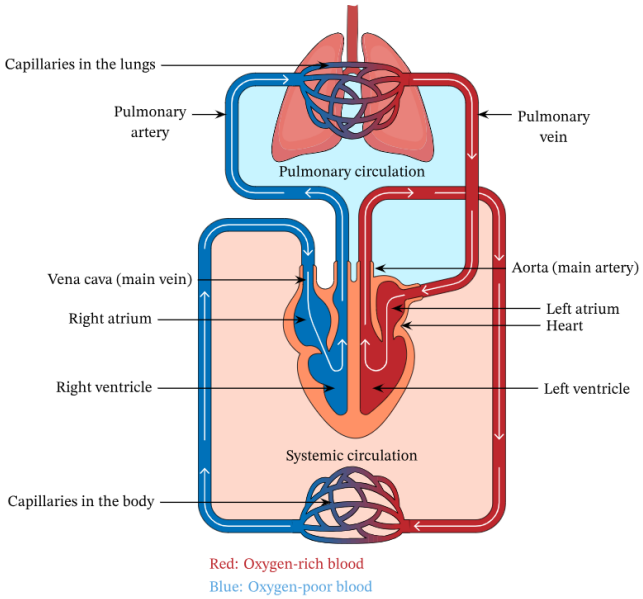


Figure 4: Difference between the Systemic circulation and Pulmonary circulation

Therefore, the cardiovascular system not only allows the distribution of nutrients to organs and the elimination of excess substances, such as carbon dioxide, but also acts as a vehicle for the immune system and other molecules (hormones) that, through blood vessels, are able to reach all parts of our organism [23].

2.1.2 Meaning of MAP, DBP, SBP, PP, CO and TPR in Clinic context

Arterial pressure represents the force exerted by blood against the internal walls of the arteries, determined by the interaction between cardiac output and vascular resistance. This fundamental hemodynamic parameter must remain within adequate values throughout the vascular system, from the arteries to the capillaries, to ensure appropriate blood perfusion to organs and tissues [25] [26].

Blood pressure measurement includes four main components: systolic blood pressure (SBP), which reflects the maximum pressure generated during cardiac contraction; diastolic blood pressure (DBP), which represents the minimum pressure in the arteries during cardiac rest; pulse pressure (PP), which represents the difference between SBP and DBP, which represents the force the heart generates each time it contracts and may also reflect arterial stiffness and mean arterial pressure (MAP), calculated using two possible formulations:

- $$MAP = DBP + \frac{SBP - DBP}{3} \tag{1}$$
- $$MAP = CO * TPR \tag{2}$$

Cardiac output (CO), determined by the product of heart rate and stroke volume, represents the amount of blood pumped by the heart in one minute. Similarly, Total Peripheral Resistance (TPR) is mainly dependent on the tone of the arterioles, which are the main resistance vessels in the circulatory system. The body maintains a delicate balance between these two factors through complex neurohumoral regulatory mechanisms, which act on different time scales to preserve organ perfusion [27] [29].

From a physiological perspective, MAP is the most significant parameter, as it expresses the mean perfusion pressure that ensures blood supply to vital organs. Unlike systolic and diastolic blood pressure values, which reflect the extreme phases of the cardiac cycle, MAP expresses the mean perfusion pressure that ensures constant blood flow through the microcirculation. Maintaining adequate MAP values (generally $\geq$ 65 mmHg) is essential to prevent ischemic damage to organs such as the kidneys, brain, and heart. In intensive care, this parameter guides therapeutic decisions in critically ill patients, with particular attention to shock conditions.

In chronically hypertensive patients, the optimal MAP threshold is higher (75-85 mmHg) due to pre-existing vascular alterations. MAP monitoring is also particularly important in surgical and neurological settings, where inadequate values can lead to serious complications [26] [28]. MAP interpretation must always consider the overall clinical picture, integrating with other hemodynamic and metabolic parameters, with the ultimate goal of ensuring optimal tissue oxygenation rather than simply achieving predetermined numerical values.

2.2 Hypertension, Blood Pressure Monitoring Techniques and ECG, PPG definition

Monitoring of blood pressure (BP) is a fundamental clinical practice for the diagnosis and treatment of numerous cardiovascular diseases, with hypertension representing the most prevalent and impactful condition worldwide. Traditional methodologies fall into two broad categories: invasive systems (IABP), considered the gold standard for continuous measurement in intensive care settings but associated with significant risks, and non-invasive methods, including manual (such as auscultation of Korotkoff sounds) and automated (primarily oscillometric) techniques, although the latter are often inconsistent and prone to error. To overcome the limitations of these techniques, research is increasingly moving toward non-invasive and cuffless technologies. These innovative systems are based on the combined acquisition and analysis of physiological signals. Specifically, the electrocardiogram (ECG) and the photoplethysmogram (PPG) are used to continuously and less intrusively estimate blood pressure. The central physiological parameter in this approach is the Pulse Transit Time (PTT), or the related Pulse Arrival Time (PAT), which is inversely correlated with blood pressure values.

This section illustrates the main BP monitoring techniques, starting with the standard invasive approach and moving on to traditional non-invasive (cuff-based) methodologies, cul-

minating in an analysis of the most promising cuffless methods, which leverage ECG and PPG signal processing, including machine learning models, for accurate, personalized, and real-time estimates of systolic (SBP) and diastolic (DBP) blood pressure. These innovations are particularly relevant in the context of hypertension management, where the need for continuous, reliable, and patient-friendly monitoring solutions remains largely unmet and will be addressed in detail in the following discussion.

2.2.1 Hypertension: definition and why it is important to be monitored

Arterial hypertension is a clinical condition characterized by persistently elevated blood pressure values, generally defined as a systolic blood pressure (SBP) equal to or greater than 140 mmHg and/or a diastolic blood pressure (DBP) equal to or greater than 90 mmHg [3]. This condition represents a major modifiable risk factor for cardiovascular disease, including coronary heart disease, stroke, heart failure, and chronic kidney disease.

According to the most recent epidemiological data, hypertension affects approximately onethird of the adult population globally, and its prevalence is constantly increasing, driven by demographic aging, the growing incidence of obesity and diabetes, and the spread of sedentary lifestyles [7]. A particularly critical aspect is that a significant proportion of hypertensive subjects are unaware of their condition, while among those who have received a diagnosis, many do not regularly follow pharmacological therapy or are not subjected to adequate clinical monitoring [6]. Blood pressure monitoring therefore plays an essential role for several reasons. First, it allows for the early identification of at-risk individuals, reducing the likelihood that undiagnosed hypertension will progress to serious acute events. Second, it allows for the assessment of the response to pharmacological and non-pharmacological treatments, optimizing therapy based on blood pressure trends [4]. Finally, regular monitoring is essential for preventing long-term complications, as effective blood pressure control is associated with a significant reduction in cardiovascular mortality.

In summary, hypertension represents not only an epidemiologically relevant problem but also a clinical challenge requiring accurate, reliable, and, if possible, continuous measurement tools to support patients and the healthcare system in the long-term management of this condition.

2.2.2 Invasive Blood Pressure Monitoring: Indications, Technique, and Complications

Invasive blood pressure monitoring (IABP) represents the standard for continuous blood pressure measurement in critically ill patients. This approach is used primarily in intensive care and during highly complex surgical procedures, providing a beat-to-beat assessment of blood pressure and allowing the analysis of cardiac output through the study of the pulsatile waveform profile [31]. The main indications include:

- Patients with hemodynamic instability or severe hypotension
- Need for frequent blood gas analysis in respiratory failure
- Evaluation of the effects of vasoactive drugs to optimize therapy
- Inability to use noninvasive methods due to skin or joint lesions [30]

The invasive blood pressure monitoring procedure begins with the placement of a catheter in a peripheral artery. The measurement system consists of several essential components that work synergistically: an arterial cannula is connected to special non-compressible tubing, through which a heparinized saline solution flows continuously (figure 5). This hydraulic system is connected to a pressure transducer based on the Wheatstone bridge principle, which converts pressure variations into electrical signals that are subsequently processed and displayed on a dedicated monitor showing both the characteristic pulsatile wave and the numerical values of systolic, diastolic, and mean blood pressure [31].

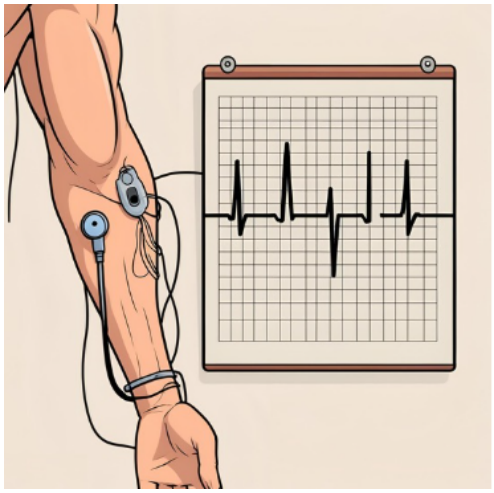


Figure 5: BP monitoring with Invasive method

Compared to non-invasive methods, this approach offers significant clinical advantages. The increased measurement accuracy is particularly valuable in complex situations such as the presence of cardiac arrhythmias or marked hypotensive states. In addition, it represents the ideal solution for patients in whom traditional techniques are inapplicable, such as severe obesity, multiple limb trauma, or extensive burns. A further benefit is the ability to analyze the morphology of the pulsatile waveform in detail, providing valuable information on global hemodynamic status and cardiovascular function [31]. However, like any invasive procedure, monitoring blood pressure is not without risks. Immediate complications include the possibility of hemorrhage, vasospasm, or accidental intra-arterial drug administration. However, in the

long term, thromboembolic events may occur, with an estimated incidence of 3.4 cases per 1000 catheters placed, or device-related infections [32].

To minimize these risks and ensure accurate measurements, rigorous safety procedures are essential. The transducer must be positioned at the correct height, corresponding to the midaxillary line, an anatomical landmark that approximates the level of the right atrium. The system must be kept completely free of air bubbles and periodically calibrated using the zeroing procedure. Careful monitoring of the insertion site, with daily evaluation for local signs of infection or ischemia, completes the necessary precautions [30]. Despite the unquestionable precision of this method, it is important to note that measurement errors can occur. The main sources of inaccuracies include system under- or over-damping, incorrect positioning of the transducer relative to the reference plane, or technical problems in the various components of the measurement circuit. For these reasons, efforts are underway to adopt new methods for blood pressure measurement that surpass or equal the efficacy of invasive techniques while simultaneously ensuring patient safety.

2.2.3 Non-invasive Methods cuff based: Advantages and Disadvantages

Non-invasive blood pressure monitoring (NIBP) uses various methodologies, which can be divided into manual and automated techniques. Among continuous measurement methods, two main approaches are based on the principle of radial artery applanation tonometry and the digital volumetric clamp method, respectively.

Applanation tonometry requires compression of a superficial artery (typically the radial artery) against a contralateral bony structure without completely occluding it. This technique, implemented in devices such as the T-line TL300 blood pressure monitor (figure 6) [32], allows direct measurement of mean arterial pressure (MAP) from the waveform, while systolic and diastolic values are derived by proprietary algorithms. Perioperative studies have shown a good correlation with contralateral invasive measurements, although the presence of motion artifacts represents a significant limitation, particularly relevant in intensive care [32].

The volumetric clamp method, alternatively, initially determines the arterial volume of the finger using infrared transmission plethysmography. By maintaining this volume constant through rapid pressure variations in a digital cuff, the system obtains a continuous blood pressure measurement. This technique also has limitations, particularly its dependence on the quality of the digital signal, which may be suboptimal in critically ill patients [32].

Regarding non-continuous measurements, these can be performed manually or automatically. The manual method, historically based on the use of mercury or aneroid sphygmomanometers, uses a stethoscope placed on the brachial artery to identify Korotkoff sounds during gradual deflation of the cuff [32]. However, the precision of this technique depends on the hearing acuity of the operator, the quality of the stethoscope, and the correct application of the cuff [31]. In contemporary clinical practice, automated oscillometric measurement represents the

Figure 6: T-line TL300 blood pressure monitor for continuous Blood Pressure Monitoring

gold standard for non-invasive blood pressure measurement. This method involves inflating the cuff above systolic blood pressure, followed by gradual deflation, during which the amplitudes of the pressure oscillations are measured. The systolic, diastolic, and mean blood pressure values are then calculated based on the variations in these pulses [32].

However, the oscillometric technique has several practical limitations. Incorrect cuff application, both in terms of size (it should cover two thirds of the upper arm with a width equal to 40% of the brachial circumference) and position (correct alignment with the brachial artery), can result in falsely high (cuff too small) or low (cuff too large) readings. Additional sources of error include repeated and rapid inflation, which causes venous congestion; excessively rapid deflation; leaks or kinks in the tubing; and the tendency to overestimate low blood pressure and underestimate high blood pressures. Cardiac arrhythmias (such as atrial fibrillation) and involuntary movements (tremors, shivers, convulsions) can also compromise the precision of the measurements [32].

Due to the limitations of these techniques, efforts are underway to develop new methods that will make measurements more accurate. An example is methods based on cuffless technologies that use ECG and PPG signals to measure blood pressure.

2.2.4 Cuffless Methods for Blood Pressure Monitoring using ECG, PPG

Because the traditional cuff method for measuring blood pressure (BP) cannot be performed continuously and in real time, a new, non-invasive, accurate and completely cuff-free self-measurement method has emerged [34]. To address this gap, several alternative methods that avoid the use of a cuff have been developed over the past two decades. The most common and widely studied approach is based on Pulse Wave Velocity (PWV), which in turn is derived

from Pulse Transit Time (PTT) using the equation:

$$PWV = \frac{D}{PTT} \tag{3}$$

, where D is the distance between two sensors [34]. PTT, defined as the time taken for a blood pulse to travel between two arterial sites, is negatively correlated with blood pressure. However, much of the research has actually relied on the measurement of Pulse Arrival Time (PAT), which is often used interchangeably with PTT in the literature [33] [34]. PAT is easier to measure, as it refers to the time interval between R_{peak} of the electrocardiogram (ECG) and a specific point (S_{peak}) on the photoplethysmogram (PPG) waveform detected at a peripheral location [34]:

$$PTT = S_{peak} - R_{peak} \tag{4}$$

In addition to PTT/PAT-based models, some researchers have attempted to predict blood pressure based on the analysis of the morphological characteristics of the PPG signal and other physiological variables [33]. For example, a study by Wang et al. [33] used the PAT method by testing different mathematical models (logarithmic, inverse, inverse square) to describe the relationship between PWV and BP. The main disadvantage of approaches using PTT and PAT is the requirement of two perfectly synchronized sensors (e.g., ECG and PPG or two PPGs), which makes the setup more complex and increases the sensitivity to motion artifacts [34]. To overcome these limitations and improve accuracy, recent research is increasingly moving towards the use of machine learning and deep learning models. These methods can both extract complex features from the signals [33] and attempt to estimate directly from the raw ECG and PPG signals, going beyond traditional mathematical equations [34]. It is important to note that, regardless of the method, other physiological factors such as age, body mass index (BMI), sex and posture during measurement can significantly influence the precision of the final result [33].

Therefore, to obtain reliable measurements, it is advisable to follow these guidelines:

- Measure both sensors (ECG, PPG) to ensure the correlation between the signals.
- The patient must be seated and at rest.
- Make sure the electrodes are placed correctly.

Wearable medical sensors that use ECG and PPG signals to measure blood pressure must return the values of diastolic blood pressure (DBP) and systolic blood pressure (SBP). The former, also called minimum pressure, corresponds to the blood pressure when an individual's heart is in the relaxation phase; therefore, it represents the recorded value between one heartbeat and the next, when the cardiac muscle is at rest [35]. In contrast, systolic (or maximum) blood pressure is the value of blood pressure during heart contraction, that is, when it beats and pushes blood

into the arteries [35]. These two pressures are evaluated using the PTT and The HR extraced from the ECG and PPG signals as shown in the equation:

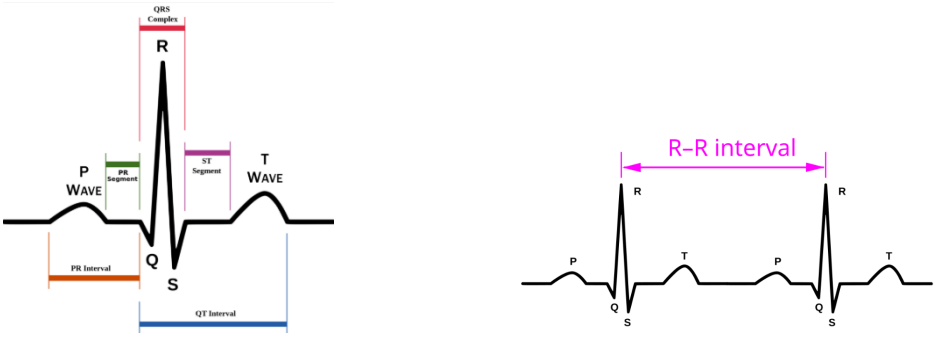
$$\begin{cases} \text{SBP} = \alpha_0 + \alpha \cdot \text{PTT} + \beta \cdot \text{HR} \\ \text{DBP} = \beta_0 + \gamma \cdot \text{PTT} + \delta \cdot \text{HR} \end{cases} \tag{5}$$

In Equation (5), α_0, β_0, α, β, γ and δ represent subject-specific parameters obtained through calibration, providing a comprehensive approach to estimate BP with a more accurate and personalized assessment.

2.2.5 ECG signal

The ECG trace, acquired non-invasively, provides a graphic representation of the electrochemical phenomena that occur in the heart muscle fibers during its cyclic functioning. The most significant element in this context is the QRS complex, a broad oscillation of the signal caused by the ventricular depolarization process. This complex is identified by three distinct components, called Q, R, and S waves (Figure 7a)[36]. By measuring the time interval Δt ($R'_{peak} - R_{peak}$) that separates two successive R peaks (which mark activation of the left ventricle), the Heart Rate (HR) can be precisely determined, as illustrated in Figure 7b and calculated with the formula:

$$HR = \frac{60}{\Delta t} \tag{6}$$



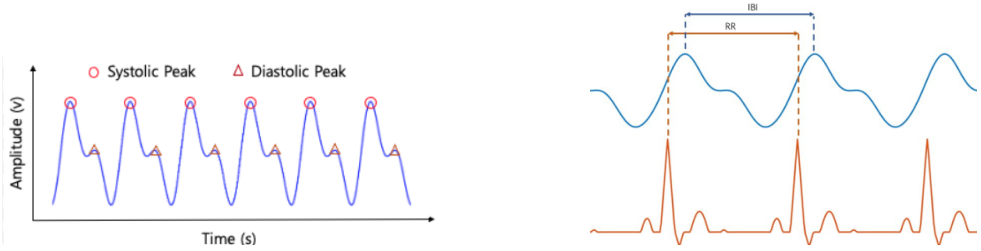
(a) ECG signal (b) HR interval

Figure 7: Illustration of the ECG signal and the corresponded Hr interval

2.2.6 PPG signal

Photoplethysmography (PPG) is a low-cost optical method that detects changes in blood volume within the skin capillaries [37]. Its signal is characterized by a rhythmic pulsatile component ('AC'), generated by blood flow fluctuations related to each single heartbeat (Figure 8) [37]. From the analysis of this waveform, it is possible to derive the Pulse Transit Time (PTT) by

calculating the time difference between R_{peak} of the ECG signal and the point of maximum slope (S_{peak}) on the immediately following PPG wave (Figure 8).



(a) PPG signal (b) PTT interval

Figure 8: Illustration of the PPG signal and the evaluation of PTT

Both ECG and PPG signals are widely used in clinical practice to monitor various physiological parameters (e.g., PPG for oxygen saturation and respiratory rate, ECG for cardiac output). Blood pressure is a vital parameter of extreme importance; this study illustrates the methodology for its estimation through the joint processing of these two physiological signals.

2.3 Innovative approaches for blood pressure estimation

In recent years, the growing availability of biomedical signals acquired through wearable devices and non-invasive techniques has spurred the development of innovative approaches for estimating blood pressure (BP). In addition to traditional direct measurement methods, research has focused on the quantitative analysis of physiological parameters derived from signals such as the electrocardiogram (ECG) and photoplethysmogram (PPG), exploiting their relationship with hemodynamics. The primary goal is to obtain a continuous, accurate, and non-invasive estimate of BP, overcoming the limitations of cuff-based or invasive techniques.

In this context, various modeling and computational approaches have been explored. Initial attempts have relied on classical regression models, such as linear, polynomial, or logarithmic regression, which attempt to establish simple mathematical relationships between surrogate variables (e.g. HR, PTT) and reference blood pressure values. Although these methods have the advantage of being interpretable and easily implementable, they often fail to capture the complexity of physiological interactions. To address this limitation, research has introduced traditional machine learning techniques, such as Support Vector Machines (SVM), Ridge Regression, Random Forest, and Regression Trees. These algorithms are capable of modeling non-linear relationships and managing heterogeneous datasets, offering better performance than traditional models, especially in the presence of noise and interindividual variability.

A further step forward has been made with the application of deep learning approaches, such as convolutional neural networks (CNN), recurrent neural networks (RNN), and their

advanced variants (e.g., LSTM). These models allow automatic learning of complex features from ECG and PPG signals, avoiding the need for manual feature extraction and showing great potential, especially in continuous monitoring scenarios. At the same time, interest is emerging in multimodal applications, which combine multiple physiological signals (ECG, PPG, accelerometers, respiratory signals, etc.) to improve estimation accuracy and reduce sensitivity to artifacts. The integration of different information sources represents a promising perspective for making systems more robust and adaptable to real-world conditions.

Despite progress, some critical issues remain that hinder the widespread clinical adoption of these approaches. These include the need for individual calibration, the poor generalizability of models trained on limited datasets, and the high physiological variability between subjects and conditions. Furthermore, managing motion artifacts and adapting models to uncontrolled contexts remain challenges. This section will analyze in detail the main innovative approaches for cuffless blood pressure estimation, assessing their potential and limitations, with the aim of outlining the state of the art and the main research gaps.

2.3.1 Classical regression Methods: Linear Regression, Logarithmic Regression, Polynomial regression, Gaussian Process regression (GPR)

Cuffless estimation of blood pressure (BP) was initially developed through the use of classical regression methods, chosen for their computational simplicity, easy interpretability, and the ability to provide a first quantitative link between physiological characteristics derived from biomedical signals (e.g. ECG, PPG) and blood pressure values. The main classical methods can be identified as:

1. **Linear Regression**: It is the simplest model and was among the first to be applied in the cuffless study of BP. The model can be written as:

$$BP = \beta_0 + \beta_1 \cdot PTT + \beta_2 \cdot HR + \epsilon \tag{7}$$

   where PTT (Pulse Transit Time) is the transit time of the pulsatile wave between ECG and PPG, strongly correlated with arterial stiffness and therefore with blood pressure [38]; HR (Heart Rate) represents the heart rate, often added as a corrective variable [39]; and ϵ is the error term. The main advantage of this approach is linearity: the coefficients β have a direct meaning and allow the influence of each physiological parameter on blood pressure to be interpreted. For example, a reduction in PTT of a few milliseconds is associated with a proportional increase in SBP, consistent with physiology [40]. However, linear regression suffers from noise in the signals, which leads to unstable estimates, non-linear relationships between PTT and BP, which are not captured, and a strong dependence on individual calibration [41]. To improve robustness, some work has used robust linear regression, which minimizes the impact of typical outliers of PPG signals acquired under real-world conditions [42].

2. **Logarithmic Regression**: It arises from more realistic physiological models since the relationship between PTT and blood pressure is not linear but follows a logarithmic trend resulting from the elastic properties of blood vessels. The most common form is:

$$BP = a \cdot ln(PTT) + b \tag{8}$$

where the parameters a and b are experimentally estimated [43]. This model is based on the Moens-Korteweg law, which links the velocity of the blood pressure wave to arterial elasticity and the exponential relationship between pressure and vessel volume [44]. In practice, logarithmic regression better describes the behavior at high blood pressure values, where arterial stiffness increases disproportionately. The advantages of this approach include better physiological adherence and the ability to model simple nonlinear relationships. However, it also has limitations such as lower interpretability than linear regression and the need for an accurate initial calibration for each subject [45].

3. **Polynomial Regression**: It extends the linear model by adding higher-order terms of the form:

$$BP = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2 + ... + \beta_n \cdot x^n + \epsilon \tag{9}$$

where x can represent PTT, HR, or other extracted characteristics from biomedical signals. The use of quadratic and cubic terms allows capturing curvatures and non-linearities in the relationship between PTT and BP [46]. For example, a study showed that quadratic models reduced estimation error by 2–3 mmHg compared to linear regression, especially in subjects with high blood pressure variability [46]. However, polynomial models have significant limitations, including the risk of overfitting, especially with small data sets, poor generalization among different populations, and the difficulty of physiological interpretation of high-order terms [47]. For this reason, polynomial regression is often used as a benchmark or as a preliminary step to identify potential nonlinear trends before applying more complex models.

4. **Gaussian Process Regression (GPR)**: It represents an evolution of classical methods towards a non-parametric and probabilistic approach. Instead of assuming a specific form for the function, as happens in linear or logarithmic models, GPR models the distribution of possible functions that can describe the input-output relationship through the form:

$$f(x) \sim GP(m(x), k(x, x')) \tag{10}$$

where m(x) is the mean function and k(x, x') is the covariance function (kernel), which defines the similarity between two points [48]. The major advantage of GPR is that, in addition to providing a precise estimate of blood pressure, it also provides a confidence interval, which is particularly useful in clinical settings to assess the reliability of the

prediction. Applications to BP estimation have shown promising results: with a Matérn 5/2 kernel, RMSEs of approximately 4.3 mmHg for SBP and 2.3 mmHg for DBP are obtained [48]; while with a Rational Quadratic kernel and optimal feature selection, the RMSE is around 10.7 mmHg for SBP and 8.0 mmHg for DBP [49]. However, GPR has some significant limitations, including high computational complexity, especially on large datasets, and the need to carefully choose and calibrate the kernel [50]. Despite these challenges, GPR is now considered a "bridge" between classical models and advanced machine learning techniques, representing an important step forward in blood pressure modeling.

Classical regression methods have played a fundamental role in the development of cuffless BP estimation systems. Linear and logarithmic regression are still used as baselines due to their simplicity and interpretability. Polynomial regression represents an improvement in the modeling of nonlinear relationships, while GPR represents an advanced but still "classical" approach capable of managing uncertainty and variability.

Despite their advances, these methods have limitations in terms of inter-subject generalizability and often require individual calibration.

2.3.2 Machine Learning Algorithms: SVM, Ridge Regression, Random Forest

Following traditional regression methods, the introduction of Machine Learning Algorithms has allowed a more flexible approach to the complex relationships between biomedical signals (ECG, PPG) and blood pressure values. Unlike linear or polynomial models, ML techniques can capture nonlinear relationships, handle larger datasets, and incorporate a greater number of variables. Among the most used algorithms in this field are:

1. **Support Vector Machine (SVM)**: It represents an advanced methodology for estimating blood pressure from physiological signals. This approach is based on a machine learning framework that aims to identify a regression function capable of effectively generalizing to previously unobserved data. The strength of SVM lies in its ability to handle non-linear relationships between input physiological variables, such as pulse transit time (PTT) and heart rate (HR), and output blood pressure values [51]. The core of the SVM method is to map the data into a higher-dimensional space using kernel functions (equation (11)).

$$f(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot K(x_i, x_j) + b \tag{11}$$

where α_i - α_i^* are Lagrangean coefficients determined during training, $K(x_i, x_j)$ is the kernel function (e.g., RBF, polynomial), b is the bias term.

This transformation allows complex non-linear problems to be converted into linearly separable problems. The Radial Basis Function (RBF) Kernel:

$$K(x_i, x_j) = exp(-\gamma \cdot ||x_i - x_j||^2) \tag{12}$$

has proven particularly effective in this context, thanks to its ability to capture complex relationships between physiological parameters and blood pressure. The γ, in the equation, represents the control of the function amplitude.

One of the main advantages of the SVM approach is its robustness to noise present in physiological signals. This is achieved by introducing a tolerance margin ϵ in the loss function, which allows the model to ignore small deviations in the training data. This feature is particularly valuable in the analysis of biomedical signals, which are often affected by motion artifacts and interference [51]. However, the implementation of SVM presents some significant challenges. The computational complexity of the algorithm grows considerably with the increase in the size of the dataset, requiring non-negligible computing resources. Furthermore, the performance of the model critically depends on the correct selection of the regularization parameters C and the RBF kernel parameter γ. Optimization of these parameters typically requires grid search procedures combined with cross-validation techniques [51]. A further limitation concerns the model's interpretation.

Unlike traditional linear regression methods, where coefficients provide a direct measure of the impact of each physiological variable, SVM operates as a "black box," making it difficult to understand the specific contribution of each parameter to the final estimate of blood pressure [51]. Despite these limitations, experimental results demonstrate that SVM achieves competitive performance in blood pressure estimation.

2. **Ridge Regression**: It is a penalized linear regression technique widely used to estimate blood pressure (BP) from physiological parameters such as pulse transit time (PTT), heart rate (HR), and photoplethysmographic signal (PPG). This method addresses the problem of multicollinearity between predictor variables, a common phenomenon in physiological data where parameters such as PTT and HR are often correlated [52]. Ridge regression modifies the cost function of ordinary linear regression by adding an L2 penalty term (L2 norm of the coefficients). The objective function becomes the following:

$$\min_{\beta} \left\{ \sum_{i=1}^{n} (y_i - X_i\beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{13}$$

where $y_i$is the blood pressure value (e.g., SBP or DBP), X_i is the vector of physiological features (PTT, HR, etc.), β are the regression coefficients, λ is the regularization parameter that controls the amount of the penalty [52].

This modification has the effect of contracting the model coefficients towards zero, thus stabilizing the estimates and improving predictive power on new data. The parameter λ, which controls the strength of the regularization, therefore, becomes crucial to balance the trade-off between bias and forecast variance [52]. In the specific context of blood pressure estimation, the ridge regression demonstrates a particular effectiveness in handling individual physiological variability. Photoplethysmographic (PPG) and electrocardiographic (ECG) signals present intrinsically correlated characteristics, which can lead to numerical instabilities in traditional regression models. The penalty introduced by the ridge regression mitigates this problem, allowing for more robust and reliable estimates [53].

However, the application of ridge regression is not without challenges. Optimal selection of the regularization parameter λ typically requires the use of cross-validation techniques, which can increase the computational complexity of the training process. Furthermore, while coefficient contraction improves the stability of predictions, it complicates the physiological interpretation of the specific contribution of each parameter [52]. Despite these limitations, ridge regression remains a valuable tool in the arsenal of researchers developing systems for non-invasive blood pressure estimation. Its ability to combine relative simplicity of implementation with robust predictive performance makes it particularly suitable for applications in wearable devices, where computational constraints and the variable quality of acquired signals require approaches resilient to noise and multicollinearity [53].

3. **Random Forest**: It has emerged as one of the most promising approaches for non-invasive blood pressure estimation, especially when applied to complex physiological signals such as heart sounds and ballistocardiograms. This ensemble method combines multiple decision trees, each trained on a random subset of data and characteristics, to produce an average prediction that is more robust and accurate than individual trees [54].

   The mathematical formulation of the Random Forest prediction for estimation of the blood pressure can be expressed as:

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} T_b(x) \tag{14}$$

   where $\hat{y}$ represents the predicted blood pressure (SBP or DBP), B is the number of trees in the forest, $T_b(x)$ is the prediction of the b-th decision tree for the input vector x and x is the vector of features extracted from the physiological signals [54]. Each decision tree T_b is constructed using a random subset of training data through a recursive feature space partitioning process. The optimal splitting function for each node is determined by

maximizing the information gain:

$$G(D, f) = I(D) - \sum_{j=1}^{m} \frac{|D_j|}{|D|} I(D_j) \tag{15}$$

where D is the data set at the current node, f is the feature considered for the split, I(D) is the impurity of the node (typically measured by the mean squared error for regression), D_j are the data subsets resulting from the split [54]. One of the main advantages of Random Forest in this context is its ability to handle nonlinear relationships and complex interactions between physiological variables. Unlike linear models, Random Forest does not require prior assumptions about the distribution of data or the linearity of relationships, an essential feature when working with intrinsically complex biomedical signals [54]. Random Forest, through its importance sampling approach, is able to automatically identify the most relevant characteristics for blood pressure estimation, reducing dimensionality issues and improving computational efficiency [54].

This algorithm shows remarkable performance, with systolic blood pressure estimation errors that are competitive with traditional methods. In particular, Random Forest's ability to reduce overfitting by bagging and selecting random characteristics makes it particularly suitable for clinical applications where generalization is essential [54]. However, implementing Random Forest presents some challenges. The "black box" nature of the algorithm makes it difficult to directly interpret causal relationships from a physiological perspective, partially limiting the analysis of the mechanisms underlying blood pressure estimation. Furthermore, the need for a large and representative training data set can be a limitation in contexts with limited data availability [54]. Despite these limitations, the results of studies demonstrate that Random Forest, combined with physiological signals such as heart sounds, represents a solid platform for the development of continuous and non-invasive blood pressure monitoring systems. The ability to integrate multiple pieces of information and handle the intrinsic noise of biomedical signals makes it particularly suitable for home and clinical applications [54].

2.3.3 Deep Learning and Neural Network methods: CNN, RNN, LSTM

Advances in noninvasive, cuff-free blood pressure (BP) estimation have recently embraced deep learning techniques, which offer superior predictive capabilities compared to classical methods, especially when dealing with complex and non-stationary physiological signals. These approaches, although more computationally intensive and less interpretable, are capable of capturing nonlinear relationships and hidden patterns in multidimensional data, paving the way for continuous and highly accurate monitoring systems. The main applied deep learning architectures include:

1. **Convolutional Neural Network (CNN)**: They represent a class of deep learning models that have revolutionized the approach to estimating blood pressure from physiological signals. Originally developed for image processing, CNNs have proven exceptionally effective in analyzing temporal biomedical signals such as the electrocardiogram (ECG) and the photoplethysmogram (PPG), due to their ability to automatically extract relevant features through convolution and pooling operations [55]. The fundamental architecture of a CNN for analyzing physiological signals is composed of several feature layers.

   The convolutional layers apply learnable filters to the input signals, progressively extracting features at different levels of abstraction. This mathematical operation can be represented as:

$$y(t) = (x * w)(t) = \int x(\tau)w(t-\tau)d(\tau) \tag{16}$$

   where x represents the input physiological signal, w is the learnable convolution kernel, and y is the resulting feature map [55]. This is typically followed by a pooling layer, termed maximum pooling, which reduces the spatial dimensionality of feature maps while preserving the most salient information. This layer is crucial for controlling overfitting and improving computational efficiency, particularly important in applications that require real-time processing [55]. CNNs offer significant advantages in the estimation of blood pressure. Their ability to learn hierarchical features directly from raw signals eliminates the need for manual feature engineering, which is traditionally complex and error-prone when processing physiological signals. Furthermore, weight sharing in convolution operations gives the model a certain translational invariance, making it robust to temporal variations in physiological signals [55]. A particularly innovative aspect of applying CNNs to the estimation of blood pressure is the ability to process multiple physiological signals simultaneously. Through multi-input architectures, CNNs can process ECG and PPG signals in parallel, capturing complex non-linear relationships between cardiac electrical activity, peripheral perfusion, and blood pressure values [55].

   Training CNNs for this application requires large and carefully annotated datasets. The optimization process aims to minimize a loss function, typically the mean squared error between predicted and actual systolic and diastolic blood pressure values. Regularization techniques such as dropout and weight decay are commonly used to prevent overfitting and improve model generalization [55]. Despite their effectiveness, CNNs have some limitations. The "black-box" nature of the learned features makes a physiological interpretation of the results difficult. Furthermore, the computational requirement can be significant, especially for deep architectures, although specific optimizations can mitigate this problem [55].

   CNNs have been shown to significantly outperform traditional methods based on handmade characteristics in blood pressure estimation. Recent studies report average errors

in the range of 3–5 mmHg for systolic blood pressure and 2–3 mmHg for diastolic blood pressure, values that meet the standards of the Association for the Advancement of Medical Instrumentation (AAMI) [55]. Integrating CNNs with other neural architectures, such as Long Short-Term Memory (LSTM) networks, represents a promising frontier, combining the ability of CNNs to extract spatial features with that of LSTMs to model long-term temporal dependencies [55].

2. **Recurrent Neural Network (RNN)**: The fundamental equations of an RNN mathematically describe its recursive operation in processing temporal sequences. The state equation:

$$h(t) = \sigma(W_{hh} \cdot h(t-1) + W_{xh} \cdot x(t) + b_h \tag{17}$$

represents the heart of the recursive process [56].

The term $W_{hh} \cdot h(t-1)$ constitutes the memory component of the network, where the weight matrix W_{hh} transforms the previous state $h(t-1)$ to preserve the historical information of the sequence [56]. This mechanism allows the network to keep track of past states, which is essential to model temporal dependencies in physiological signals [56]. The term $W_{xh} \cdot x(t)$ processes the current input $x(t)$, which in the context of blood pressure estimation represents the features extracted from the ECG and PPG signals at time t [56]. The weight matrix W_{xh} learns which input features are most relevant to the specific task [56]. The bias b_h is an additive parameter that allows the network to shift the output independently of the inputs, improving the model's fitting ability [56].

The activation function σ (typically tanh or ReLU) introduces non-linearity into the system, allowing the network to learn complex relationships between inputs and hidden states [56]. This nonlinearity is crucial to capture the complex physiological relationships between cardiovascular signals and blood pressure [56]. The output equation:

$$y(t) = W_{hy} \cdot h(t) + b_y \tag{18}$$

transforms the hidden state $h(t)$ into the desired output $y(t)$ [56]. The weight matrix W_{hy} maps the network's internal state to the estimated blood pressure values, while the bias b_y adds an offset to improve the fit to the data [56]. For example, this mechanism can be applied to estimate blood pressure in beats [56]. This system of equations operates recursively, processing each new sample in the sequence while maintaining a contextual memory of previous states [56]. The recursive nature allows the RNN to capture the temporal dependencies between consecutive heartbeats, modeling the dynamic evolution of hemodynamic parameters during the acquisition of physiological signals [56].

3. **Long Short-Term Memory (LSTM)**: Long Short-Term Memory (LSTM) represents an evolution of Recurrent Neural Networks (RNN) specifically designed to solve the vanishing

gradient problem and capture long-term dependencies in temporal sequences. Introduced by Hochreiter and Schmidhuber in 1997, LSTMs have revolutionized the analysis of physiological signals thanks to their ability to maintain both short-term and long-term memory [56].

The LSTM structure is based on a system of gates that regulate the flow of information through the memory cell. The fundamental equations that govern its operation are:

$$\begin{aligned}
\text{Forget Gate:} \quad & f_t = \sigma(W_f \cdot [h(t-1), x(t)] + b_f) \\
\text{Input Gate:} \quad & i_t = \sigma(W_i \cdot [h(t-1), x(t)] + b_i) \\
\text{Candidate Memory:} \quad & \tilde{C_{(t)}} = \tanh(W_C \cdot [h(t-1), x(t)] + b_C) \\
\text{Memory Update:} \quad & C(t) = f_t * C(t-1) + i_t * \tilde{C_{(t)}} \\
\text{Output Gate:} \quad & o_t = \sigma(W_o \cdot [h(t-1), x(t)] + b_o) \\
\text{Hidden State:} \quad & h_t = o_t * \tanh(C(t))
\end{aligned} \tag{19}$$

The forgot gate f_t determines which information should be forgotten from long-term memory. This gate calculates a value between 0 and 1 for each element in the cell's state, where 0 indicates "completely forget" and 1 indicates "completely retain." Its equation, $f_t = \sigma(W_f \cdot [h(t-1), x(t)] + b_f)$, combines the previous state $h(t-1)$ and the current input $x(t)$ through learnable weights W_f and a bias b_f, transforming the result through a sigmoid function that produces values in the range [0,1] [56]. The input gate (i_t) decides what new information should be stored in the cell. Operating in parallel with the forgot gate, this gate uses the same combination of inputs but with different parameters (W_i, b_i) to produce a vector of values between 0 and 1 that indicate how much each component of the input should be considered for memory updating [56]. The candidate memory ($C(t)$) generates potential values that could be added to the memory. Using a tanh function instead of a sigmoid, it produces values in the range [-1,1], allowing for both increases and decreases in stored values. This component represents new candidate information for the memory update [56]. The memory update operation ($C(t)$) strategically combines information from the forgot gate and the input gate to update the memory state. The multiplication element-by-element of f_t and $C(t-1)$ determines which parts of the previous memory to retain, while the product of i_t and $C(t)$ decides which new values to add. This mechanism allows the LSTM to keep track of relevant information for long periods [56].

The output gate (o_t) regulates which parts of the memory should be read and propagated to the output and to the next state. Calculated as $o_t = \sigma(W_o \cdot [h(t-1), x(t)] + b_o)$, this gate determines which information in the current memory is relevant for immediate

output and for subsequent time steps [56]. Finally, the hidden state ($h(t)$) represents the effective output of the LSTM cell for the current time step. Obtained by multiplying the output gate by the hyperbolic tangent of the memory state, this value captures relevant information from current memory while maintaining a suitable range of values for subsequent processing [56]. LSTMs offer significant advantages for the analysis of physiological signals, particularly in the estimation of blood pressure. Their ability to maintain long-term memory allows the capture of temporal dependencies spanning hundreds of heartbeats, essential for modeling complex hemodynamic phenomena [55] [56]. Resistance to vanishing gradient ensures stable training even for very long sequences, while the temporal selectivity of the gates allows the network to automatically focus on the most clinically relevant events in ECG and PPG signals [56].

Recent studies show that LSTMs achieve average errors of 3.8 to 4.2 mmHg for systolic blood pressure and 2.5 to 3.1 mmHg for diastolic blood pressure, superior to traditional RNNs and hand-made feature-based methods [55] [56]. This precision is particularly valuable in continuous monitoring applications where the ability to capture temporal trends is crucial for the diagnosis and management of hypertensive conditions. Despite their advantages, LSTMs present significant challenges. The computational complexity is significantly higher than traditional RNNs, requiring significant hardware resources for training and real-time inference [56]. The presence of numerous learnable parameters (weights and biases for each gate) increases the risk of overfitting, requiring large annotated datasets for effective generalization [55] [56].

The black-box nature of internal mechanisms represents another additional limitation, making the physiological interpretation of the network decisions difficult and limiting clinical acceptance in contexts where diagnostic transparency is essential [56]. The tuning of hyperparameters is particularly complex due to the non-linear interactions between the various gates and components of the LSTM cell. Hybrid architectures represent the most advanced frontier in LSTM research for biomedical applications. Bidirectional LSTMs processes sequences in both temporal directions, capturing both past and future information for each point in the sequence, significantly improving predictive capacity in blood pressure estimation [55] [56]. Integration with attention mechanisms allows the network to selectively focus on the most relevant events in physiological signals, improving both the accuracy and the interpretation of the results [56]. CNN-LSTM architectures combine the ability of CNNs to extract spatial features from individual heartbeats with the ability of LSTMs to model temporal dependencies between consecutive beats, creating an end-to-end system for the comprehensive analysis of physiological signals [55].

These hybrid architectures have shown superior performance compared to single models, particularly in the process of ECG and PPG signals, where spatial (signal morphology) and temporal (rhythm and variability) information are equally important for accurate blood

pressure estimation [55] [56]. Promising research directions include the development of more computationally efficient architectures, such as Lite-LSTM optimized for embedded and wearable devices [56]. Customization of the model through transfer learning and domain adaptation will allow LSTMs to be adapted to individual physiological characteristics, improving accuracy for specific patient profiles [55] [56]. Interpretability research focuses on the development of explainable AI techniques to make LSTM decisions transparent, which is crucial for clinical acceptance [56]. Integration with domain-specific knowledge through constraint learning mechanisms will allow the incorporation of physiological principles directly into the network architecture [55]. Developments in specialized hardware technologies for LSTM model inference in edge devices will enable efficient deployment on wearable devices for continuous monitoring of blood pressure in real world scenarios [56]. The creation of larger and more diverse datasets will be essential for training models capable of generalizing between different populations and physiological conditions [55] [56]. These converging research directions promise to transform LSTMs from purely academic tools to fundamental components of cardiovascular monitoring of clinical systems, significantly contributing to the prevention and management of cardiovascular diseases through more accurate and continuous blood pressure estimation [55] [56].

2.3.4 Multimodal Techniques

Multimodal techniques represent the cutting edge of non-invasive blood pressure estimation, overcoming the limitations of unimodal methods through the synergistic integration of multiple sources of physiological information. These approaches combine signals such as the photoplethysmogram (PPG), electrocardiogram (ECG), and ballistocardiogram (BCG) with traditional clinical data, creating a holistic analysis system that captures the complexity of the human cardiovascular system. The conceptual foundation of multimodal techniques lies in the ability to compensate for the intrinsic limitations of each individual signal through complementary information from other sources. PPG, for example, provides detailed information on peripheral blood perfusion but is sensitive to motion artifacts, while ECG offers precise cardiac timing but limited hemodynamic information. The integration of these signals allows for a more robust and accurate estimate of blood pressure (Figure 9) [57].

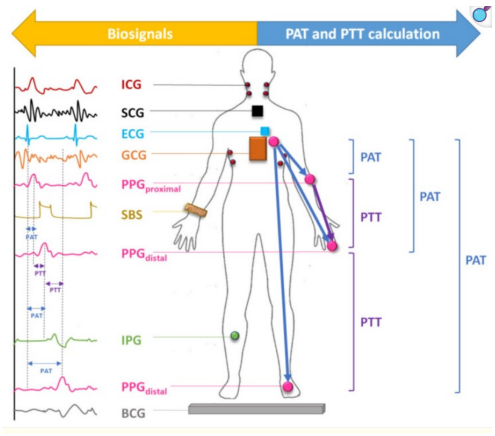


Figure 9: Example of Multimodal Combination of Signals (Ref. [57])

Multimodal integration architectures typically operate at three distinct levels. Raw data fusion directly combines signals before feature extraction, preserving maximum information but requiring precise temporal alignment. Feature fusion extracts the features of each signal separately and then combines them into a unified vector, allowing optimized processing for each modality. Decision-level fusion combines the outputs of specialized models for each signal, maximizing the specific expertise of each module [57]. The advantages of these approaches are multiple and significant. Overall, system robustness improves substantially, as artifacts affecting a single modality can be compensated for by the other modalities. Information completeness increases through the capture of complementary aspects of cardiovascular physiology, from electrical activity to cardiac mechanics to peripheral perfusion. The personalization of the estimates becomes more effective, with the possibility of adapting models to specific patient characteristics through the integration of demographic and clinical data [57].

Practical applications of these techniques show promising results. In the reference paper [57], the integration of PPG with demographic and clinical data improved hypertension detection by 15% compared to unimodal methods, demonstrating the added value of the multimodal approach. The combination of temporal and morphological signals allowed the capture of both beat-to-beat variations and long-term hemodynamic trends. Implementation challenges remain significant, particularly regarding precise temporal synchronization between heterogeneous signals, increased computational complexity, and the need for large and diverse training datasets. However, ongoing advances in signal processing and machine learning effectively address these challenges [57]. Future research directions include integration with emerging signals, such as millimeter wave radar for contactless sensing, development of adaptive fusion architectures that can self-optimize based on the quality of available signals, and implementation on wearable devices for continuous monitoring in home and clinical settings. These developments promise to transform multimodal techniques into routine clinical tools for the prevention and management of cardiovascular diseases [57].

2.4 Dataset and Validation Techniques

2.4.1 Public and Private Dataset

The availability of datasets is a key factor in the development of reliable models for non-invasive blood pressure estimation. Among the most widely used resources is MIMIC (Multiparameter Intelligent Monitoring in Intensive Care), a publicly accessible database that collects clinical signals and parameters from patients admitted to intensive care [58]. This repository has become a reference standard for the scientific community due to its vastness and variety of available data, including electrocardiographic (ECG) signals, photoplethysmographic (PPG), invasive blood pressure, and numerous vital signs. The use of MIMIC allows algorithms to be trained on large and diverse samples, thus improving the robustness and generalizability of the models. However, such data present some critical issues: being collected in complex clinical contexts, the signals are often affected by noise and artifacts, and patients' conditions may not be representative of the general population.

In addition to MIMIC, other public physiology datasets have been made available in recent years, such as those derived from experimental monitoring campaigns in healthy subjects [59]. These databases, while generally smaller in size, offer the advantage of controlled acquisition protocols and higher quality signals, less affected by clinical factors. They are therefore particularly useful for methodological studies and for testing algorithms under laboratory conditions. In addition to public resources, proprietary datasets, collected by research groups through wearable devices, are playing an increasingly important role. A significant example are the Shimmer sensors, used in this thesis to continuously acquire ECG and PPG simultaneously [60].

These systems enable high-quality data collection in non-clinical settings, facilitating the development of models designed for real-world home or outpatient monitoring applications. The main limitation of these datasets is their small sample size, which can limit the generalization of the results, but they offer precise control over acquisition conditions and greater adherence to everyday usage scenarios. Ultimately, integrating public datasets such as MIMIC, experimental archives, and proprietary datasets represents the most robust strategy to develop and validate blood pressure estimation algorithms. Each type of resource offers specific advantages and limitations, but only their combined use can ensure models that are simultaneously accurate, robust, and applicable on a large scale. In the table 2 are shown the advantages and disadvantages of using these two types of datasets.

Database type	Advantages	Disadvantages
Public databases	- Large availability of data (e.g., MIMIC) - Heterogeneous and diverse populations - Common benchmark for the scientific community - Free accessibility	- Variable signal quality (noise, artifacts) - Data often collected in complex clinical settings - May not be fully representative of the general population
Proprietary databases (e.g., Shimmer)	- High-quality signals acquired in controlled conditions - Data collected in realistic scenarios (home-/ambulatory monitoring) - Stronger adherence to practical use of wearable devices	- Smaller sample size - Limited generalization of results - Restricted access, only available to research groups who collect them

Table 2: Comparison between public and proprietary databases for blood pressure estimation

2.4.2 The Role of RMSE, MAE and STD in Evaluating the Accuracy of Blood Pressure Estimation

In the validation of blood pressure estimation techniques, the mean absolute error (MAE), the root mean square error (RMSE) and the standard deviation (STD) are three fundamental statistical metrics for quantifying the precision and reliability of measurements. These indicators are particularly crucial when evaluating medical devices according to international standards such as the ANSI/AAMI/ISO 81060-2:2018 guidelines, as demonstrated in the validation study of Ref. [61]. Mean absolute error (MAE) is defined as the arithmetic mean of the absolute values of the differences between the estimated and reference values. Mathematically, for a set of n measurements, the MAE is calculated as:

$$MAE = (\frac{1}{n}) \cdot \sum |y_i - \hat{y}_i| \tag{20}$$

where y_i represents the reference value (invasive or clinically validated measurement) and $\hat{y}_i$ is the value estimated by the device under test. The MAE provides a direct measure of the average error committed by the device, expressed in the same units of measurement as the target variable (mmHg in the case of blood pressure). A lower MAE value indicates greater

accuracy of the estimation system [61]. Root Mean Square Error (RMSE) is a metric that emphasizes larger errors by squaring the deviations. Its formula is:

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum (y_i - \hat{y}_i)^2} \tag{21}$$

Unlike MAE, RMSE assigns greater weight to larger errors, making it more sensitive to the presence of outliers. This feature makes it particularly useful for identifying anomalous measurements that could compromise the clinical reliability of the device [61]. The Standard Deviation (STD), on the other hand, measures the variability of errors around the mean value. Calculated as:

$$STD = \sqrt{\frac{\sum (e_i - \mu)^2}{(n-1)}} \tag{22}$$

where e_i represents the individual error (y_i - $\hat{y}_i$) and μ is the mean of the errors, STD quantifies the spread of the errors. A low STD value indicates that errors are consistently close to the mean value, suggesting good device stability, while a high value signals high variability and unpredictable performance [61]. The relationship between these metrics can be expressed as: $RMSE^2 = MAE^2 + STD^2$ This equation highlights how the RMSE combines information on both mean error (MAE) and error variability (STD), providing a composite measure of the device's accuracy [61]. In the study Ref. [61], these metrics were used to validate the performance of the HUAWEI smartwatch according to the AAMI/ISO standards, which specifically require MAE be $\leq$ 5 mmHg and STD be $\leq$ 8 mmHg to ensure the clinical accuracy of blood pressure measuring devices. The combined use of MAE and STD allows for a comprehensive evaluation: the MAE captures the overall accuracy of the device, while the STD evaluates its reliability and consistency of measurements [61].

The importance of these metrics lies in their ability to provide an objective and quantitative assessment of device accuracy, essential for the clinical acceptance and commercialization of medical technologies. Specifically, for continuous blood pressure monitoring applications using wearable devices, a low MAE ensures that estimates are clinically meaningful, while a low STD ensures that performance is consistent across different physiological conditions and users [61]. The study demonstrates how the combined analysis of MAE and STD allows us to identify not only the average accuracy of the device, but also its ability to maintain reliable performance in real-world scenarios, where factors such as motion, inter-individual variability, and environmental conditions can influence measurement quality [61].

3 Proposed Solution

This section will illustrate the proposed solution for estimating non-invasive blood pressure from physiological signals acquired by Shimmer devices. The entire process is organized into multiple phases, each of which addresses a specific objective and contributes to the construction of a coherent and reproducible flow from data acquisition to statistical and predictive modeling. We will begin with a description of Shimmer sensors, analyzing their main characteristics and how they are used to synchronously record ECG and PPG signals. Data acquisition methods will be discussed, with particular attention to signal format and the initial pre-processing steps required to ensure proper time management and preliminary noise reduction.

Subsequently, the pre-processing and feature extraction algorithms will be explored in more detail. In this phase, the ECG and PPG signals will be subjected to specific filtering to reduce artifacts and physiological or instrumental interference. It will be shown how R and S peaks are identified from the ECG and PPG signals and how, from these points, it is possible to calculate fundamental quantities such as heart rate (HR) and pulse wave transit time (PTT). Finally, the final section will focus on building the regression model to estimate blood pressure. The various strategies adopted will be illustrated, from linear regression to more complex machine learning and deep learning models, discussing the selection criteria and optimization parameters. The validation phase will be addressed with a methodical analysis of the techniques employed, including cross-validation and train/test splitting, with the aim of assessing the model's robustness and reliability across different available datasets.

3.1 Devices, sensors and software

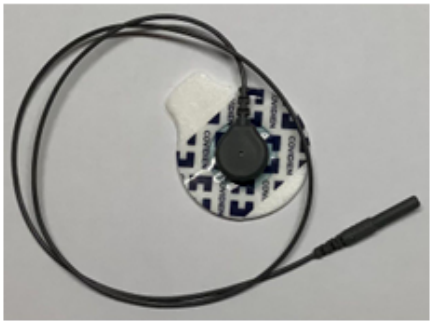
The first step was the collection of physiological signals; we relied on SHIMMER devices (Figure 10). SHIMMER represents an extremely versatile and modular wireless sensor platform, specifically designed for biomedical research applications. Its main appeal lies in its intrinsically wireless nature and its low weight, which make it particularly suitable for monitoring physiological signals in outpatient settings or directly at home [63]. This flexibility allows studying vital signals in conditions that are more natural and less constrained than in a hospital setting.

Figure 10: Shimmer3 Devices

3.1.1 ECG Device

The acquisition of the ECG signal was performed using the Shimmer3 EXG Unit SR47-4-0 module (Figure 11a). To ensure accurate and comfortable measurement, we used Covidien ECG electrodes (Figure 11b). These electrodes are designed for single use and are round in shape. The skin-contact part consists of an Ag/AgCl electrode immersed in a solid hydrogel, secured with a gentle adhesive and connected via a convenient button system.

(a) Shimmer3 EXG Unit SR47-4-0 module



(b) Shimmer3 EXG Unit electrode

Figure 11: Shimmer3 Unit for ECG evaluation

An additional foam backing helps ensure the electrode's stability. These snap-on electrodes use a patented pre-gelled adhesive, enriched with a non-irritating gel, specially formulated to

minimize the risk of allergic reactions. Furthermore, the foam electrode is completely latex-free, making it safe for use on any skin type [64].

To better understand how the Shimmer3 EXG module works, it is helpful to look at its block diagram (Figure 12). This diagram reveals the presence of integrated defibrillation protection, an electromagnetic interference (EMI) filter designed to clean the signal, a sophisticated right-leg drive (RLD) amplifier that effectively counteracts common-mode interference, three programmable gain amplifiers (PGAs) that increase the amplitude of the input signal for better detection, and finally, a high-precision analog-to-digital converter (ADC) transforms the incoming analog signal into a digital representation using a signed 24 bit integer value for each individual sample. This digitization process is crucial for data analysis and processing on a computer.

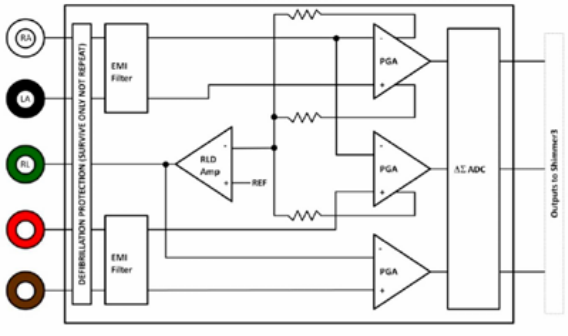


Figure 12: Shimmer3 EXG Unit SR47-4-0 block diagram [65]

3.1.2 PPG Device

For recording PPG signals, we used the Shimmer3 GSR+ Unit SR48-3-0, shown in figure 13. The GSR+ (Galvanic Skin Response) unit is not limited to measuring PPG alone but also provides the connections and preamplification needed to acquire galvanic skin response (GSR) data.

Figure 13: Shimmer3 GSR+ Unit SR48-3-0

This module is particularly suitable for evaluating the electrical characteristics or conductance of the skin and, at the same time, captures the PPG signal, converting it to obtain an accurate estimate of heart rate (HR) using the Shimmer clip (Figure 14) [66]. The integrated optical circuit for pulse measurement includes an integrated amplifier and filter circuit, which take care of initial signal conditioning [67]. The clip itself is equipped with a green LED and a detector positioned next to each other, a configuration that operates in the reflection mode ("adjacent")[62]. In this mode, the light emitted by the LED is reflected by the tissue and detected by the adjacent sensor, allowing changes in blood flow to be measured.

Figure 14: Shimmer3 GSR+ Unit SR48-3-0 Optical pulse clip

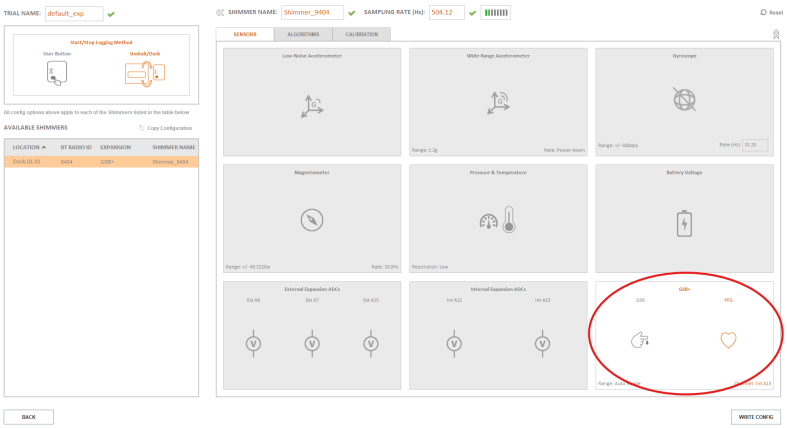
3.1.3 Software

The configuration of the Shimmer devices and the export of acquired data were managed using ConsensysPRO v1.6.0 64bit software. Before proceeding with writing the configuration via the connection card, we carefully set the acquisition parameters. Specifically, in the Shimmer3 GSR+ module, we activated only the PPG sensor (Figure 16a), while in the Shimmer3 EXG module, we selected only the LA-RA lead and enabled the ECG sensor (Figure 16b).

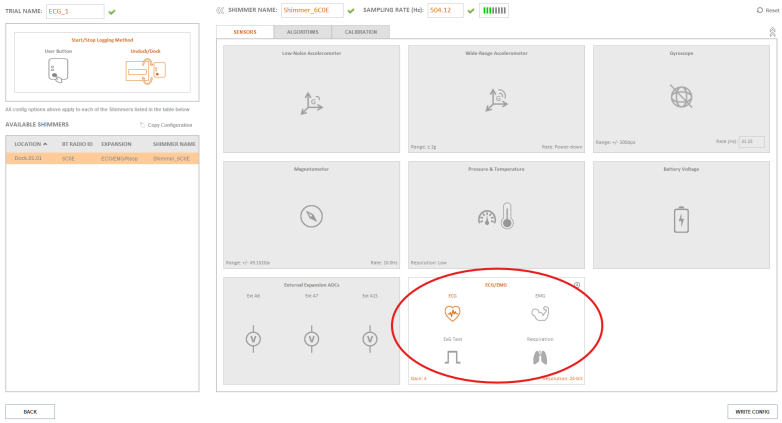
Figure 15: Consensys base unit

In order to ensure precise synchronization between the two modules, we configured the EXG module as master and set the sampling rate to 504.12 Hz (Figure 16c). This choice is motivated by the fact that ECG requires a minimum sampling rate of 500 Hz [68], and the ConsensysPRO software only allows selecting predefined frequency values. On the other hand, PPG requires a minimum sampling rate of 100 Hz [69].

To start recording signals, both Shimmer modules must be undocked from the connection board shown in figure 15. Reversing this procedure, i.e., docking the modules to the board, stops the recording. When the Shimmer modules are undocked, they automatically begin recording signals synchronously, thanks to the Bluetooth connection (Figure 16d).



(a) In Shimmer3 GSR+ screen: only PPG sensor has been turned on for PPG recordings.



(b) Shimmer3 EXG options for ECG recordings

(c) The sampling frequency has been set to 504.12 Hz for both devices.

(d) The Shimmers start to record the physiological signals.

Figure 16: ConsensysPRO v1.6.0 64bit software

This wireless synchronization is essential to analyze the ECG and PPG signals in relation to each other. Once recording is complete and the modules are docked back to the board, all acquired data are transferred to the computer in .mat format, ready for processing and analysis. The procedure followed these steps:

- **Protocols**: To ensure the quality of the acquired data, we followed a strict protocol during

recording. Each participant in the study was asked to sit comfortably and completely relax. As mentioned, the recordings were made at different times of the day, and each recording session lasted approximately 20 minutes.

- **ECG Recording**: The correct placement of the ECG electrodes was essential to obtain a high-quality signal. The electrode configuration is illustrated in detail in figure 17.

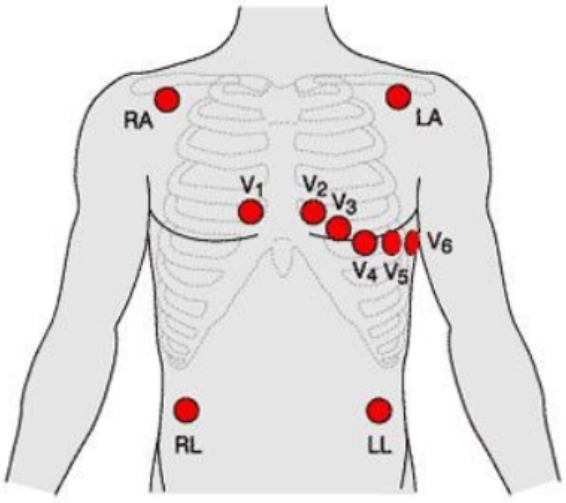


Figure 17: ECG recording phase

The electrodes connected to the white and black pins were placed on the subject's chest, on the right and left sides, respectively. The electrode connected to the green pin was placed on the right side of the pelvis, serving as a reference for the measurement.

- **PPG Recording**: The PPG Shimmer clip was gently attached to the subject's left index finger. To ensure optimal adherence of the clip to the finger and to prevent any interference from external ambient light, we covered the clip with a thick black tie as shown in figure 18.

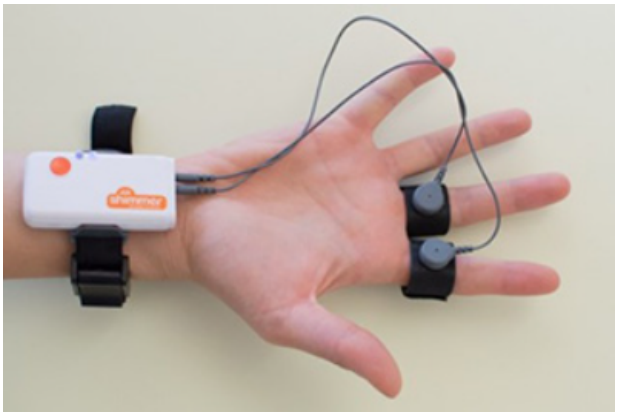


Figure 18: PPG recording phase

This helped improve the quality of the acquired PPG signal.

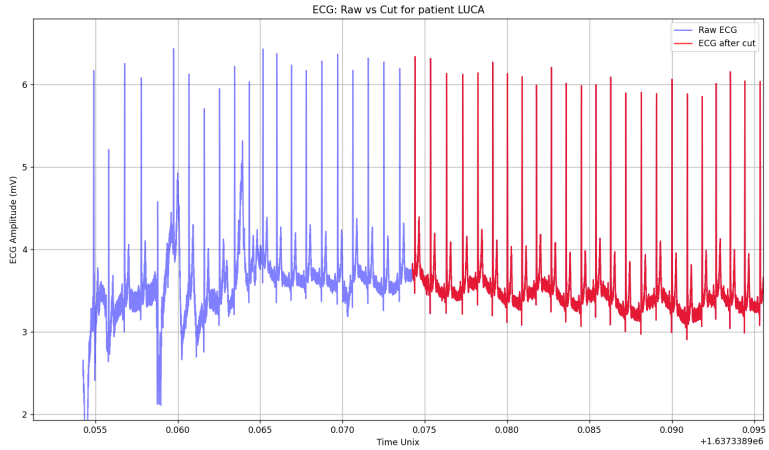
- **Reference Signals**: Blood pressure values were taken every minute for 20 minutes using a blood pressure cuff. This allowed each patient to be assigned reference values for SBP and DBP.

3.2 Algorithm

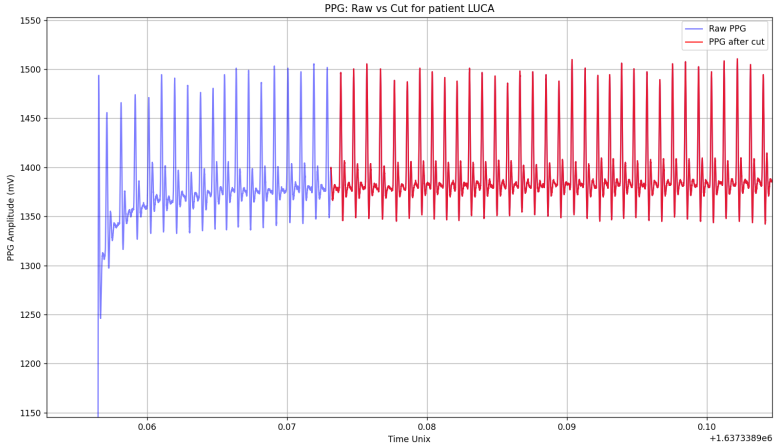
The algorithm used was developed in Python, a versatile high-level programming language. To make it more manageable and clear, we divided it into seven main sections. Each of these sections addresses a specific step in the process of estimating systolic blood pressure (SBP) and diastolic blood pressure (DBP). In practice, the algorithm follows these steps: signal preparation, filtering to remove noise, identifying peaks in the signals, extracting relevant features, preparing comparison data from a reference device, feature reduction to simplify the model, and finally regression analysis to predict blood pressure as is presented in the appendix 6.1.

3.2.1 Signals Preparation

The algorithm begins by reading the ECG and PPG signals from the files, which are loaded into arrays using the "scipy.io.loadmat" function (Ref code 6.1). After opening the files, the data are temporally cleaned: the first 20 seconds of each signal are removed to remove initial noise due to the undocking from the board , and the last 30 seconds to avoid final distortions, as we can observe in figures 19,20.

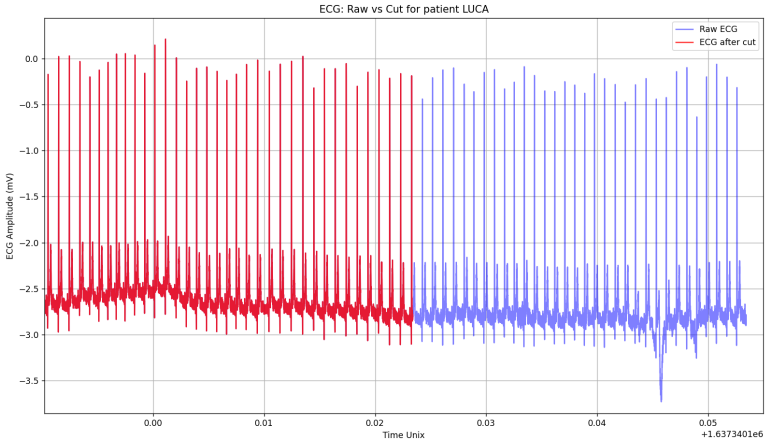


(a) ECG Signal

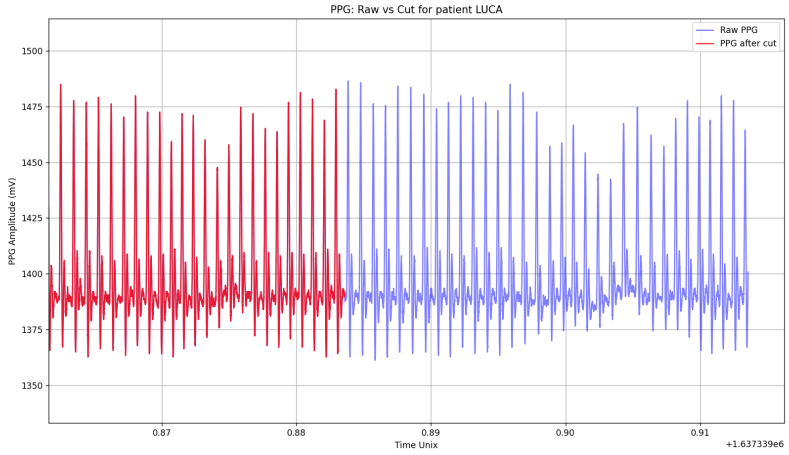


(b) PPG Signal

Figure 19: Signals Before and After the initial 20 sec cleaning



(a) ECG Signal



(b) PPG Signal

Figure 20: Signals Before and After the last 30 sec cleaning

This trimming is performed by calculating the number of samples corresponding to the durations to be removed based on the sampling rate. Then the signals are cut to the same length. The signals are then synchronized: the ECG and PPG data are temporally aligned so that they have the same starting point and length (Figure 21). This process ensures that subsequent analyses are performed on signal segments that perfectly overlap in time, improving the accuracy of feature extraction.

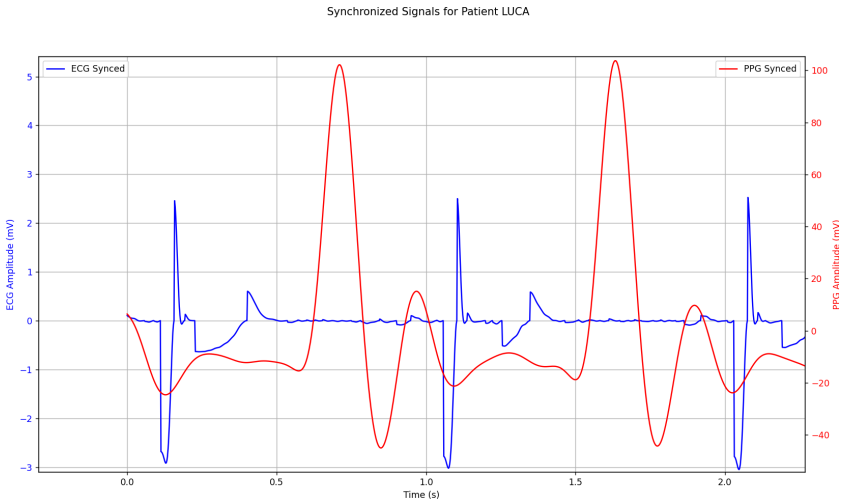


Figure 21: Signals After the Synchronization

3.2.2 Signals Filtering

After loading and initial clipping of the signals, the algorithm proceeds with a fundamental filtering phase to improve data quality and remove unwanted components. In fact, both ECG and PPG signals are affected by 50 Hz noise and by the power variation with respect to the zero (baseline), so some processing is needed.

- **ECG Filtering**: The ECG signal is first normalized by subtracting the mean to eliminate any offsets. Next, a fourth-order Butterworth bandpass filter is applied with a bandwidth of 0.5–40 Hz, which allows only the physiologically relevant frequencies for cardiac analysis to be retained, eliminating both low-frequency noise (baseline wander) and high-frequency noise. If the sampling rate is higher than 100 Hz, a 50 Hz notch filter is also applied to remove interference from the electrical network. Finally, the baseline is removed, bringing the signal back to around zero (Figure 22).

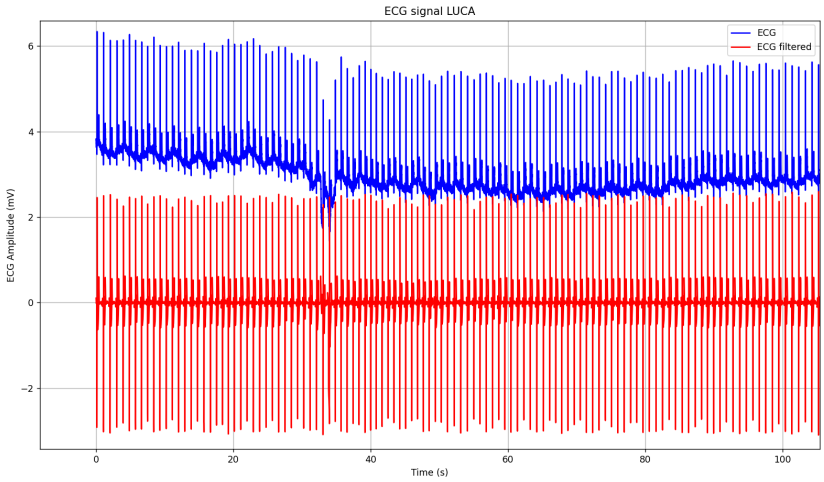


Figure 22: ECG signal pre and post filtration

- **PPG Filtering**: As with the ECG, the PPG signal is first normalized to ensure a comparable scale. It then undergoes a careful filtering sequence to isolate important information and eliminate noise. First, a Butterworth high-pass filter (0.5 Hz) removes background drift and slow variations that could distort the analysis. This cleans the signal, focusing on rapid variations related to cardiac activity. Next, a Butterworth low-pass filter (8 Hz) attenuates high-frequency noise, retaining only the pulsatile component of the signal. This makes the signal cleaner and more reliable, perfect for extracting the features needed to estimate blood pressure (Figure 23).

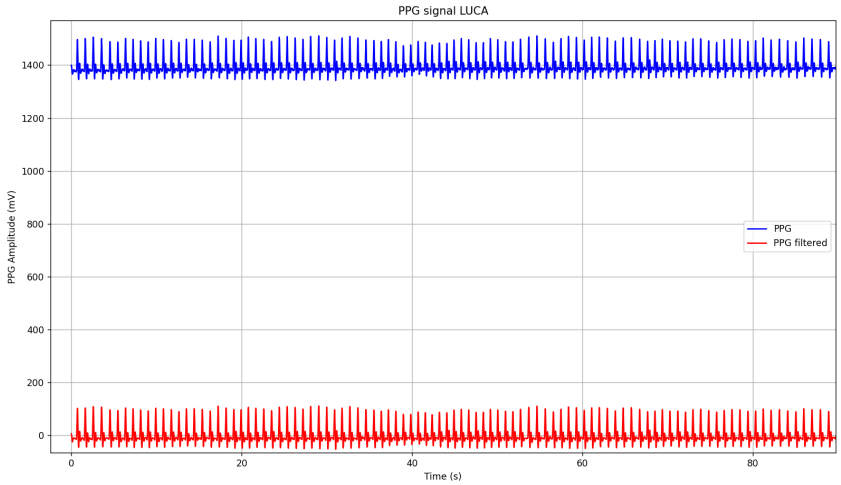


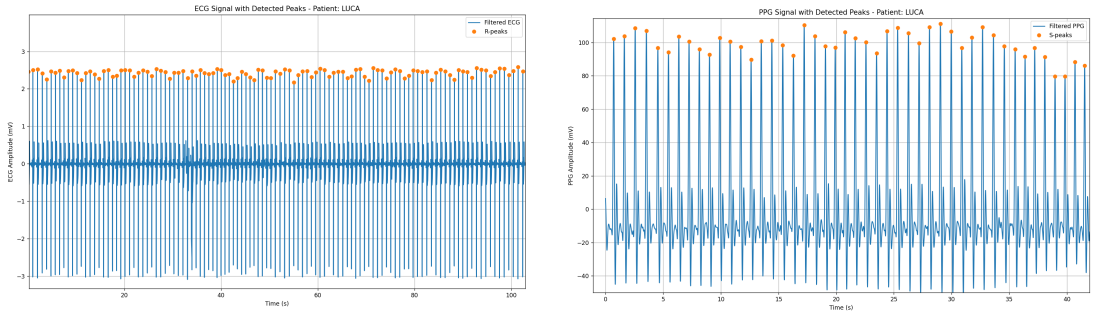
Figure 23: PPG signal pre and post filtration

3.2.3 Peaks detection

After synchronizing and filtering the ECG and PPG signals, the algorithm proceeds to detect the characteristic peaks of each signal, which are essential for extracting physiological features. Regarding ECG peak detection (R-peaks), the filtered and synchronized ECG signal identifies R-peaks, which are the local maxima corresponding to heartbeats. Detection occurs by dividing the signal into time windows (every 0.4 seconds) and searching for the local maximum in each window. This method allows for robust heartbeat identification even in the presence of residual noise.

Similarly, S-peaks, which represent the maximum points of the pulsatile component of blood flow, are identified on the PPG signal. Here too, the signal is analyzed in windows (every 0.5 seconds) to find local maxima. After the initial detection, the peaks are cleaned using two filters: a statistical filter, which only keeps peaks within a certain number of standard deviations from the local mean, eliminating outliers, and a minimum distance filter, which, in the case of peaks that are too close, only keeps the one with the largest value to avoid duplications due to noise.

The detected peaks are displayed overlaid on the filtered signals (Figure 24), allowing verification of the accuracy of the detection and its correspondence with the expected physiological components.



(a) R_{peak} (b) S_{peak}

Figure 24: R_{peak} and S_{peak} superimposed on the signals

This process ensures that the extracted features (such as heart rate and pulse transit time) are calculated on reliable, artifact-free data that represent the patient's actual physiological activity.

3.2.4 Features Extraction

After synchronizing and filtering the ECG and PPG signals, the code proceeds with the extraction of the main physiological features: Heart Rate (HR) and Pulse Transit Time (PTT). The feature extraction process in the code consists of detecting peaks in the signals, calculating HR and PTT, cleaning outliers data, and interpolating the features onto the reference

times, thus obtaining reliable and synchronized time series for subsequent regression analyses and comparison with clinical data.

Regarding HR and PTT calculations, the peak detection is identified by R_{peaks} on the ECG and S_{peaks} on the PPG using peak detection and cleaning functions. For PTT (Pulse Transit Time), for each heartbeat (R_{peaks}), the first subsequent S_{peaks} is searched. The time difference between these two peaks represents the PTT: $PTT = S_{peaks} - R_{peaks}$ or the time taken for the pressure wave to propagate from the heart to the peripheral measurement point. For HR (Heart Rate), the heart rate is calculated as the inverse of the time distance between two consecutive R_{peaks} according to the formula:

$$HR = \frac{60}{t_{R_{i+1}} - t_{R_i}} \tag{23}$$

where the result is expressed in beats per minute (bpm).

After computation, the features are cleaned to eliminate outliers due to detection errors or artifacts: the mean and standard deviation of HR and PTT are calculated, values that deviate by more than one standard deviation from the mean are removed, and the arrays are reduced to the same length to ensure temporal correspondence between HR, PTT, and the timetable. To compare the extracted features with the reference blood pressure values (SBP and DBP), it is necessary to interpolate HR and PTT over the blood pressure measurement times: Interpolation functions (np.interp and CubicSpline) are used to reconcile HR and PTT to the same time points as the reference data (Figure 25).

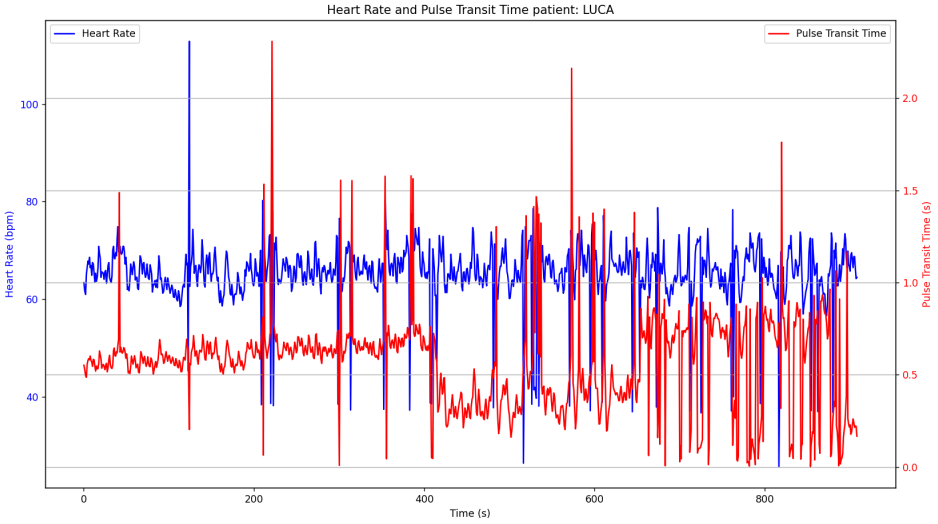


Figure 25: HR and PTT signals after the features extraction

3.2.5 Reference Values Preparation

After extracting features from physiological signals, the code prepares reference systolic (SBP) and diastolic (DBP) blood pressure values for comparison and regression. SBP and DBP values are read from patient-specific CSV files, where each line of the file contains the UNIX timestamp in milliseconds, the SBP values in mmHg, and the DBP values in mmHg. For temporal correction, timestamps are converted from milliseconds to seconds and aligned by subtracting the initial value, resulting in a relative time scale. A 60 ms offset is also subtracted from each timestamp to correct for the instrumental delay between the blood pressure measurement and the physiological signals. For interpolation, SBP and DBP values are interpolated over signal analysis times, such as time points in the timetable of the extracted features, using the np.interp function to obtain blood pressure values corresponding to the same time points as the physiological features. This process ensures that the comparison between signals and references is accurate and synchronized, allowing a reliable evaluation of the regression models.

3.2.6 Features reduction

After extracting and interpolating the physiological features (HR, PTT) and the baseline values (SBP, DBP), the code applies two additional steps to simplify and smooth the data. For feature reduction, features and baseline values are reduced by averaging them over 10-second time windows ($feat.reduction$). This process divides the time series into 10-second intervals and replaces the values within each window with their average. The goal is to reduce noise, local variability, and dimensionality in the data, making the time series more stable and suitable for regression.

Subsequently, after reduction, the resulting (reduced) time series are resampled via interpolation (np.interp) to bring them back to the same instants in the physiological feature timetable. This ensures that all features and reference values are temporally aligned and have the same length, a necessary condition for training and evaluating regression models. These steps are essential to obtain robust and comparable data suitable for statistical analysis and predictive modeling.

3.2.7 Regression Methods

The final step of the algorithm involves defining and using regression techniques to estimate SBP and DBP. The implemented methodology involves a structured process that begins with the preparation of the database and concludes with a robust evaluation of predictive performance.

The process begins with loading data from databases (LUCA and SHIMMER), which can be selected based on specific research needs through a configuration parameter. The analyzes are performed on two different databases:

- **LUCA's database**: This database represents the data taken on myself, using the Shimmer devices.
- **Shimmer's Database**: It represents the data coming from the shimmer patients.

For each patient, essential physiological features are extracted, including Pulse Transit Time (PTT) and Heart Rate (HR), along with reference values for systolic (SBP) and diastolic (DBP) blood pressure using the $extractfeaturesreg$ function (Appendix **??**). All data are then concatenated into unified arrays, thus preparing the data set for subsequent regression analyzes.

Several regression approaches have been implemented to model the relationship between physiological features and blood pressure parameters:

1. **Linear Regression**: It is one of the simplest and most widely used statistical models for estimating the relationship between one or more independent variables (in this case, physiological parameters such as PTT and HR) and a dependent variable, which in our study corresponds to values of systolic blood pressure (SBP) or diastolic blood pressure (DBP). The idea behind this approach is that blood pressure can be approximated as a linear combination of available features, weighted by appropriate coefficients.

   Mathematically, the model can be expressed as:

   $$\hat{y} = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + ... + \beta_n \cdot x_n \tag{24}$$

   where $\hat{y}$ represents the estimated value of the target variable (e.g., SBP), $x_1, x_2, ..., x_n$ are the independent features (such as PTT and HR), while $\beta_1, \beta_2, .., \beta_n$ are the model coefficients. These coefficients are determined to minimize the mean squared error, which is the sum of the squared differences between the predicted and observed values [70].

2. **Least Mean Squares (LMS)**: It represents an algorithmic extension of linear regression. The goal is to minimize the mean squared error between the predicted and actual values using an iterative optimization process. Unlike classical linear regression, which directly calculates the coefficients through an analytical solution, LMS uses a gradient descent approach: the model's weights are initialized arbitrarily and then updated step by step, based on the error in predicting each sample and its derivative with respect to the parameters.

   The update rule is given by:

   $$w_{t+1} = w_t + \eta \cdot e_t \cdot x_t \tag{25}$$

   where w_t represents the weights at step t, η is the learning rate, $e_t = d_t - y_t$ represents the error between the actual value d_t and the predicted value y_t and x_t is the feature vector. This mechanism allows the model to adapt progressively to the data, correcting the parameters as a function of the residual error [71].

3. **Ridge Regression**: It is a variant of linear regression that introduces an L2 regularization term into the cost function, with the aim of controlling model complexity and avoiding overfitting. While in classical linear regression, coefficient estimation is based exclusively on minimizing the sum of the squared errors between actual and predicted values, in this case a penalty proportional to the sum of the squares of the coefficients is added.

   The Ridge objective function can be expressed as:

$$J(w) = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \cdot \sum_{j=1}^{p} w_j^2 \tag{26}$$

   where $\hat{y}_i$ represents the predicted value, y_i the actual value, w_j is the model coefficients, and λ is the regularization parameter that controls the weight of the penalty. If $\lambda = 0$, the Ridge coincides with classical linear regression; higher values of λ reduce the amplitude of the coefficients, leading the model to simpler and more stable solutions [52].

4. **Support Vector Machine (SVM)**: The central idea is to identify a function that approximates the data so that prediction errors remain within a certain tolerance threshold, indicated by the parameter ϵ. Unlike classical linear regression, which directly minimizes the mean squared error, SVM defines an acceptability range: all points falling within this range are not penalized, while those falling outside it contribute to the cost of the model.

   Mathematically, the objective function seeks to minimize the model's complexity, represented by the norm of the coefficients w, while balancing out the largest deviations of ϵ. This leads to a model that is more robust to outliers and capable of generalizing better. Furthermore, SVM relies on the use of support vectors, a subset of the data that actually determines the shape of the regression function: only the points closest to the margins or those that violate the tolerance band affect the final model. A particularly relevant aspect of SVM is the ability to model nonlinear relationships thanks to the introduction of kernel functions. For example, with the Radial Basis Function (RBF) kernel, it is possible to transform the original feature space into a higher-dimensional space, making it easier to construct a regression function that captures the complexity of the data. In this way, SVM combines predictive capability and flexibility, making it suitable for contexts where relationships between variables cannot be described by a simple linear model [51] .

The regression models used within the algorithm are applied after the dataset has been split into a Train and Test set: Classic Train/Test Split and K-Fold Cross-Validation.

- **Classic Train/Test Split**: This approach adopts a 70/30 split of the data, selecting the top 70% of the indices for the training set and the remaining 30% for the test set (Figure 26).

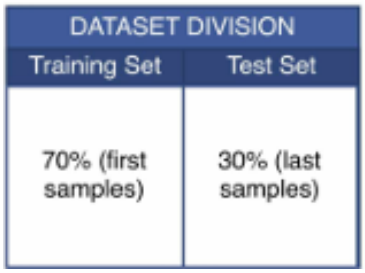


Figure 26: Train and Test Set Split

This approach is simple and computationally efficient, allowing for a quick evaluation of the model's performance on unseen data. However, it has the limitation of being heavily dependent on a single data split, which can lead to unrepresentative results if the data are inadequately shuffled or have non-uniform distributions.

- **K-Fold Cross-validation**: This method divides the dataset into K partitions (folds) of approximately equal size. The iterative process involves using K-1 partitions for each fold to train the model, while the remaining partition is used for testing (Figure 27).

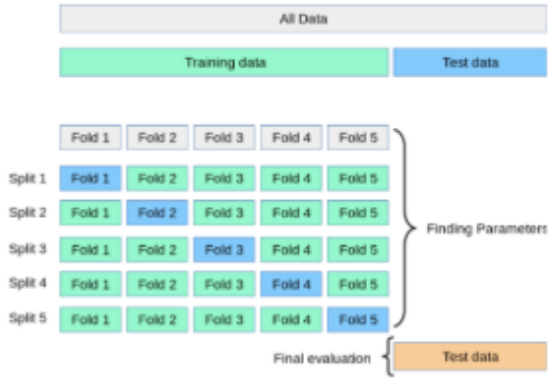


Figure 27: K-Fold Cross Validation

This operation is repeated K times, ensuring that each partition serves as a test set exactly once. The code typically uses scikit-learn's KFold with 3 folds [72].

The main advantages of cross-validation lie in its robustness and statistical reliability, since the model is evaluated on all possible subdivisions of the dataset, reducing the variance of performance estimates. This approach provides a more complete understanding of the generalizability of the model. The main limitation is the increased computational cost due to the need to train and test the model multiple times, with a runtime proportional to the number of folds selected.

The integration of these two methodologies therefore allows for both a rapid initial evaluation through the train/test split and an in-depth and statistically robust analysis through cross-validation, ensuring a complete understanding of the model's predictive performance in different application scenarios. The results of the algorithm are shown in Section 4.

4 Results

This section illustrates the experimental results emerging from the validation of the different blood pressure estimation methods applied to patient **LUCA 1**. The objective of the comparative analysis is to evaluate the performance of the various approaches in terms of accuracy and reliability, measured by the metrics of Mean Absolute Error (MAE) and standard deviation (STD). The overall performance of the noninvasive estimation methods will be examined in depth, with particular attention to the differences between the models before and after cross-coordination. In fact, a sensitivity analysis will be performed on several datasets to evaluate the robustness of the methods in the presence of variable physiological conditions. The results for all patients are shown in the Appendix 6.1.

4.1 Results Validation

To determine how many estimated blood pressure values can be considered valid, it is necessary to refer to the AAMI/ESH/ISO guidelines. These provide an acceptance criterion valid for both 85 and 20 measurements samples: the mean difference between the values measured by the device under test and the reference values must be less than or equal to 5 mmHg, with a standard deviation not exceeding 8 mmHg, for both systolic blood pressure (SBP) and diastolic blood pressure (DBP) [61].

It should also be noted that the reference values were measured using a cuff (sphygmomanometer), whose accuracy is approximately ± 3 mmHg. Therefore, after considering everything, the predicted values can only be considered valid if they satisfy the following conditions:

$$\begin{cases} \text{Mean absolute error (MAE)} <= 5 \text{ mmHg} \\ \text{Standard Deviation (STD)} <= 8 \text{ mmHg} \end{cases} \tag{27}$$

4.2 Algorithm Results

The results are displayed as functions of the regression model and the database used. In fact, the various models exhibit different behaviors based on the subdivision of the dataset.

4.2.1 Standard 70-30 Database division

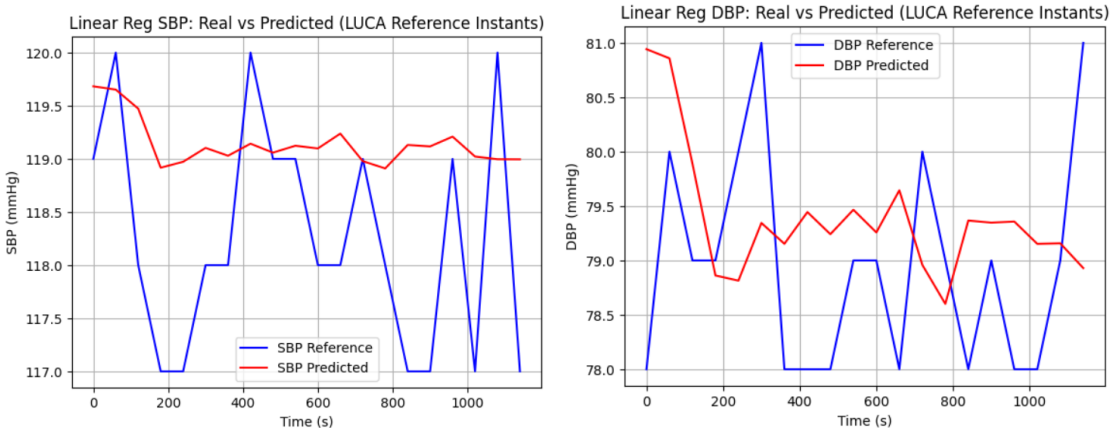
In the initial configuration, the dataset was split according to the classic 70-30 split, maintaining chronological order to ensure temporal consistency and prevent data leakage. The results are shown based on the regression model used:

1. **Linear Regression**: The Linear Regression algorithm was evaluated on four different dataset configurations to analyze its adaptability to contexts with varying levels of complexity and physiological variability. The results show a clear relationship between the size and composition of the training samples and the quality of the estimates produced.

    - Using only the data contained in the patient Luca database, the model is able to capture the relationship between the extracted features (PPG, ECG, etc.) and blood pressure values with good fidelity, thanks to the high coherence and homogeneity of the signals. Under this condition, the estimates are stable and aligned well with the reference values. The quantitative results are reported in Table 3 and in Figures (28).

Average MAE (mmHg) $\pm$ Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	5.32 $\pm$ 5.52	1.94 $\pm$ 2.31

Table 3: Average MAE and STD (mmHg) for Patient Luca 1 using Linear Regression.



(a) SBP (b) DBP

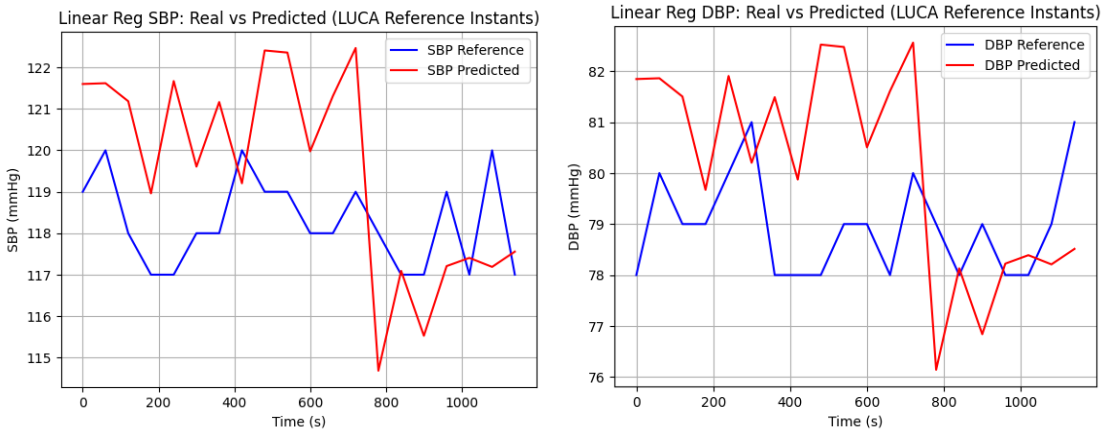
Figure 28: Real Reference Values vs Predicted Vaules of SBP and DBP using the Linear Regression on Luca's Database

- The introduction of 22 Shimmer patients, from patient 20 to patient 42, causes an initial drop in performance. Despite the increased data volume, Linear Regression

struggles to generalize correctly, as the new subjects present physiological dynamics and instrumental noise characteristics different from those of Luca. The model, trained on a heterogeneous mixture, loses the ability to optimally adapt to the specific behavior of the target. Details are available in Table 4 and the figures (29).

Average MAE (mmHg) ± Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	8.91 ± 9.68	3.65 ± 4.10

Table 4: Average MAE and STD (mmHg) for Patient Luca 1 using Linear Regression.



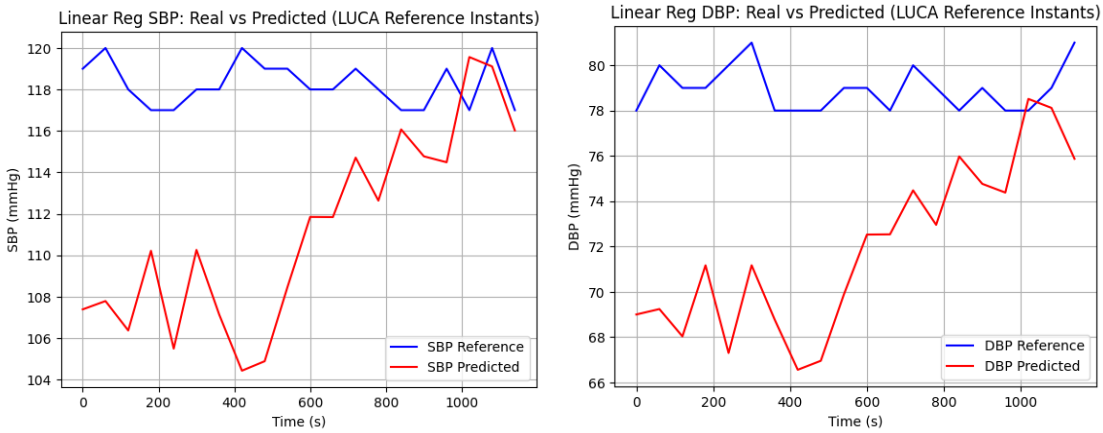
(a) SBP (b) DBP

Figure 29: Real Reference Values vs Predicted Vaules of SBP and DBP using the Linear Regression on Luca's Database + Last 22 Shimmer Patients

- However, if we consider Luca's database combined with the first 20 shimmer patients (excluding the 15th, which is an outlier for the SBP signal), the performance reaches the lowest level. Again, no advantage is observed from the larger sample size; in contrast, the introduction of additional variability that is not consistent with the target (blood pressure levels too low compared to the reference patient) introduces confusion into the model, increasing both the mean error and its variability. The results are reported in Table 5 and in the figures (30).

Average MAE (mmHg) ± Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	15.78 ± 12.15	11.11 ± 6.74

Table 5: Average MAE and STD (mmHg) for Patient Luca 1 using Linear Regression.



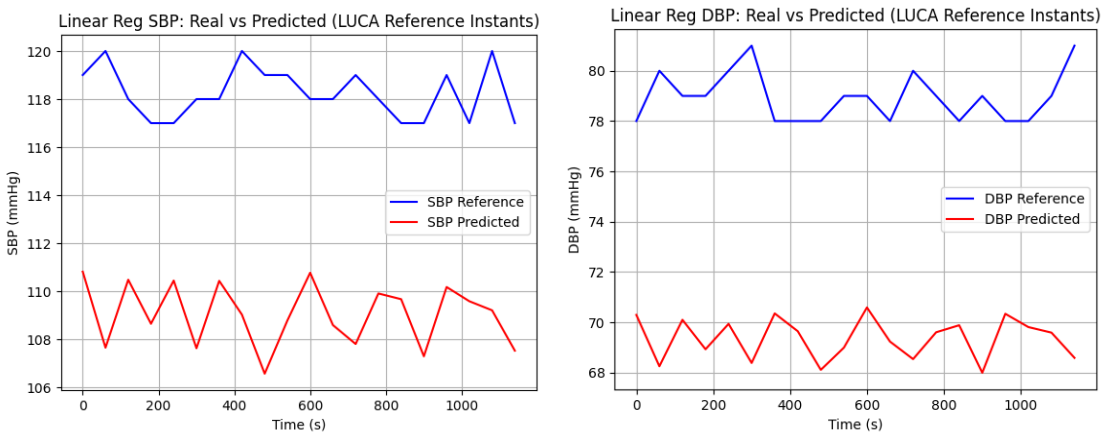
(a) SBP (b) DBP

Figure 30: Real Reference Values vs Predicted Vaules of SBP and DBP using the Linear Regression on Luca's Database + Initial 20 Shimmer Patients

- In the largest configuration, the performance of Linear Regression degradation is further accentuated. The high heterogeneity of the dataset, combined with the lack of physiological similarity between Luca and the Shimmer subjects, leads the model to produce unreliable estimates. The results can be found in the table 6. Furthermore, the strange behavior is also observed in the figures (31).

Average MAE (mmHg) ± Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	14.21 ± 5,28	12.04 ± 4.06

Table 6: Average MAE and STD (mmHg) for Patient Luca 1 using Linear Regression.



(a) SBP (b) DBP

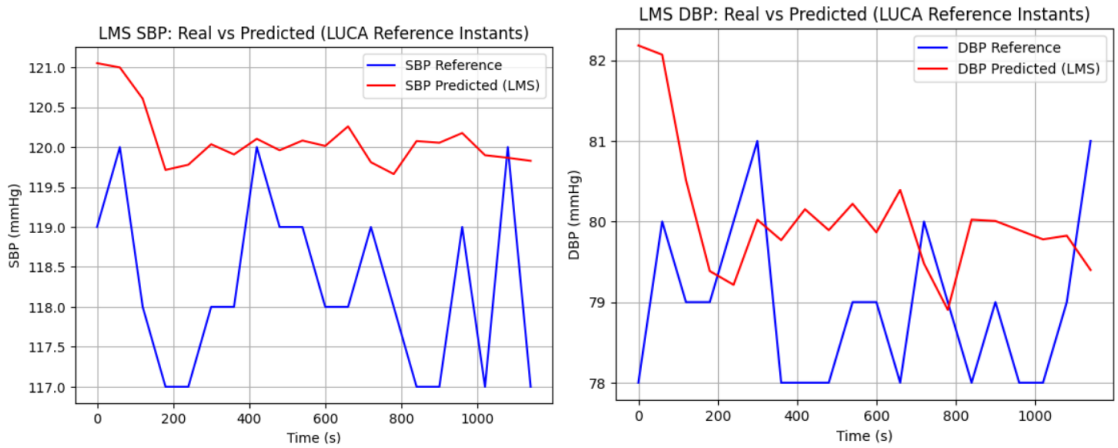
Figure 31: Real Reference Values vs Predicted Vaules of SBP and DBP using the Linear Regression on Luca's Database + All Shimmer Patients

2. **LMS**: Also in this case, the LMS (Least Mean Squares) algorithm, known for its computational simplicity and adaptability to online contexts, was evaluated on four configurations of the dataset, in order to analyze its behavior in the presence of data with different physiological and instrumental characteristics.

   - In the most focused configuration: training and testing exclusively on Luca's data. The LMS algorithm shows the best overall performance. Due to the high temporal and physiological coherence of the signal, the filter is able to adapt its weights stably and precisely, producing estimates well aligned with the reference blood pressure values. Under this condition, the error is low and the variability is minimal. The quantitative results are shown in Table 7 and figures (32).

Average MAE (mmHg) $\pm$ Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	1.92 ± 1.01	1.40 ± 1.17

Table 7: Average MAE and STD (mmHg) for Patient Luca 1 using LMS Regression.



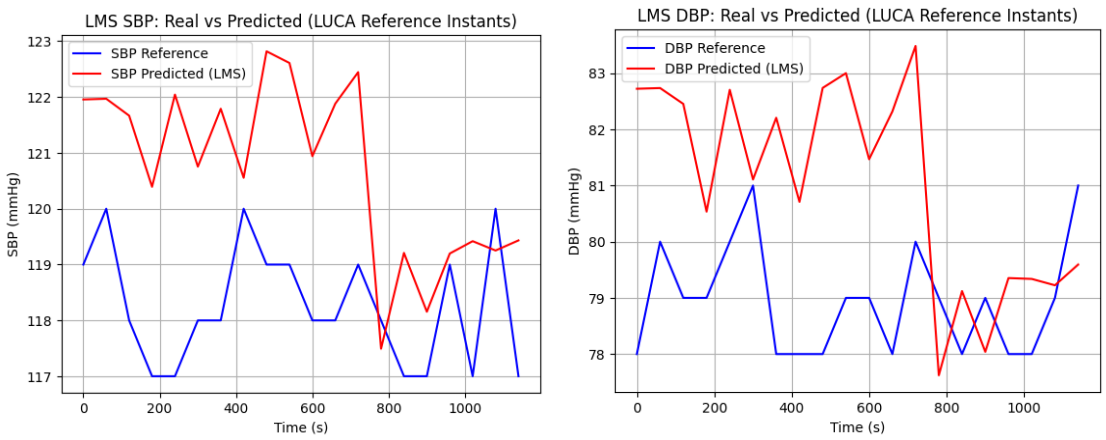
(a) SBP (b) DBP

Figure 32: Real Reference Values vs Predicted Vaules of SBP and DBP using the LMS Regression on Luca's Database

- With the introduction of Shimmer patients 20 to 42, a first significant drop in accuracy was observed. Despite the increased size of training set, the presence of signals with different dynamics (different PPG response, different signal-to-noise ratio, morphological variability of the ECG) prevented the LMS filter from converging on an optimal solution for the target Luca. The model, averaging across heterogeneous patterns, lost the ability to fine-tune to individual behavior. Details are available in Table 8 and figures (33).

Average MAE (mmHg) $\pm$ Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	2.57 $\pm$ 1.54	2.45 $\pm$ 1.94

Table 8: Average MAE and STD (mmHg) for Patient Luca 1 using LMS Regression.



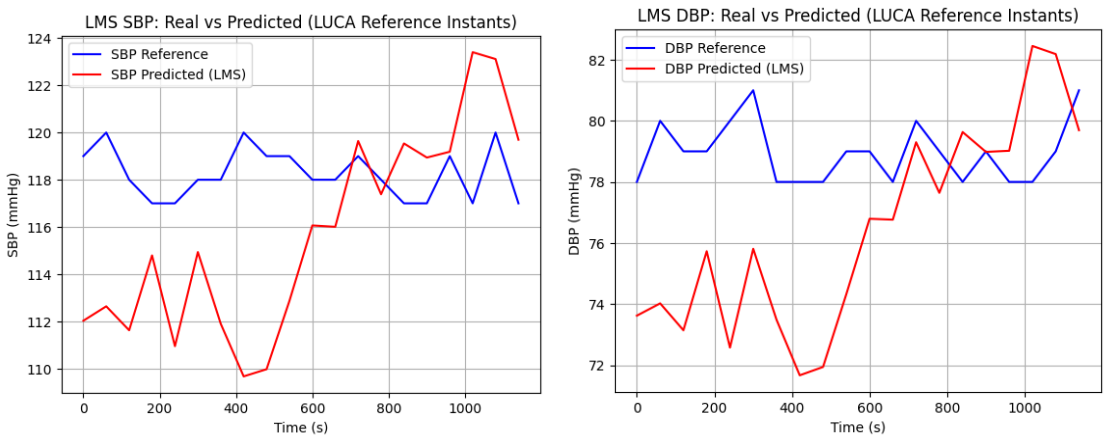
(a) SBP (b) DBP

Figure 33: Real Reference Values vs Predicted Vaules of SBP and DBP using the LMS Regression on Luca's Database + Last 22 Shimmer Patients

- The addition of the first 20 (excluding, also in this case, the patient 15) Shimmer patients further exacerbates the situation that reach lowest level. The algorithm, already destabilized by the first wave of heterogeneous data, now struggles even more to maintain consistency in its estimates. An increase in both the mean error and its fluctuation is observed, indicating poor filter stability in the presence of non-stationary and non-homogeneous inputs. The results are shown in Table 9 and figures (34).

Average MAE (mmHg) $\pm$ Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	4.28 $\pm$ 4.50	3.54 $\pm$ 3.31

Table 9: Average MAE and STD (mmHg) for Patient Luca 1 using LMS Regression.



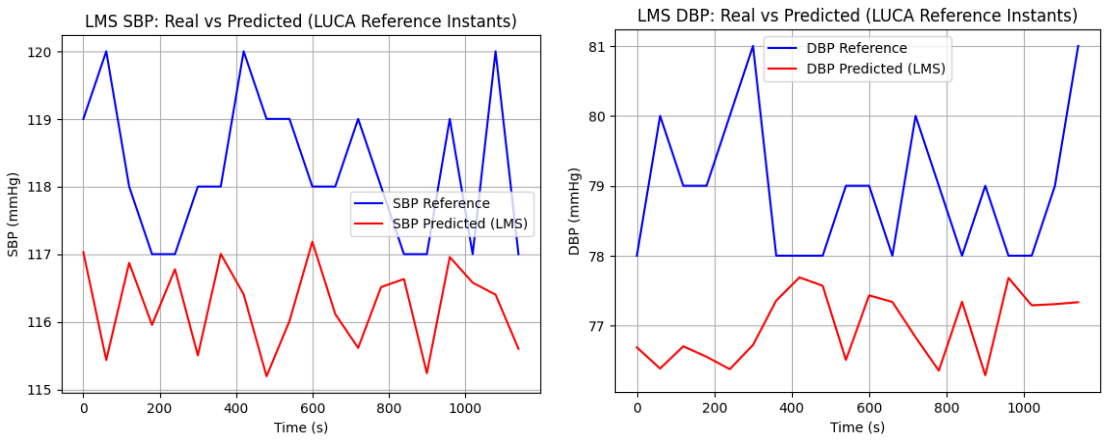
(a) SBP (b) DBP

Figure 34: Real Reference Values vs Predicted Vaules of SBP and DBP using the LMS Regression on Luca's Database + Initial 20 Shimmer Patients

- In the broadest configuration (LUCA + whole Shimmer data), the LMS's performance improve a little. The high inter-subject variability, combined with the lack of physiological similarity with the target Luca, leads the filter to produce unreliable and often systematically biased estimates. The lack of a weighting mechanism or selection of the most relevant data makes the LMS particularly vulnerable in this context. The complete results are available in Table 10 and figures (35).

Average MAE (mmHg) $\pm$ Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	2.00 $\pm$ 1.25	1.96 $\pm$ 1.26

Table 10: Average MAE and STD (mmHg) for Patient Luca 1 using LMS Regression.



(a) SBP (b) DBP

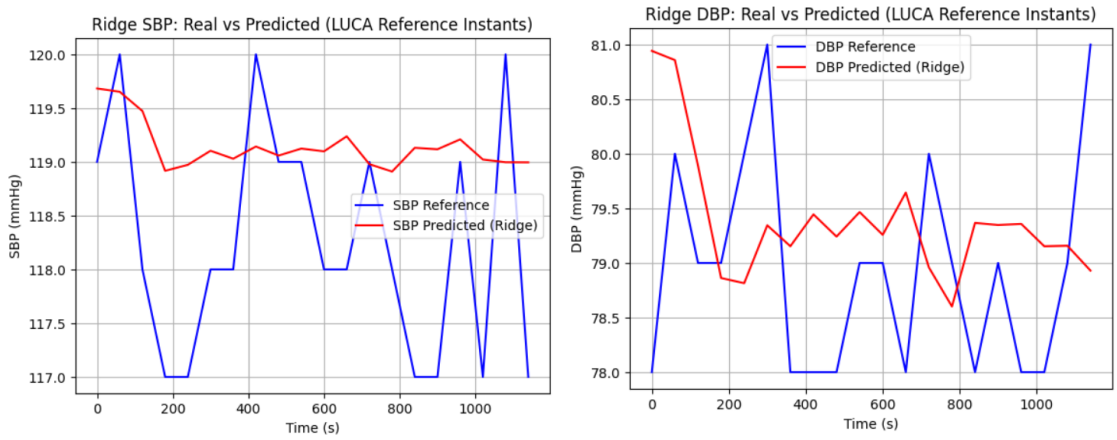
Figure 35: Real Reference Values vs Predicted Vaules of SBP and DBP using the LMS Regression on Luca's Database + All Shimmer Patients

3. **Ridge Regression**:Also in this case, Ridge Regression, known for its ability to regularize coefficients and mitigate the effects of multicollinearity, was evaluated on four different configurations of the dataset, in order to analyze its behavior in the presence of data with different physiological and instrumental characteristics.

   - In the most focused configuration: training and testing done exclusively on Luca's data. Ridge Regression shows the best overall performance. Thanks to the high physiological and temporal coherence of the signals, the model is able to estimate the coefficients in a stable and precise way, producing predictions well aligned with the reference values of the target patient LUCA 1. In this condition, the mean absolute error (MAE) is minimal and the standard deviation of the estimates is particularly low, indicating high predictive reliability. The quantitative results are reported in Table 11 and Figures (36).

Average MAE (mmHg) $\pm$ Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	1.17 $\pm$ 1.01	0.97 $\pm$ 1.09

Table 11: Average MAE and STD (mmHg) for Patient Luca 1 using Ridge Regression.



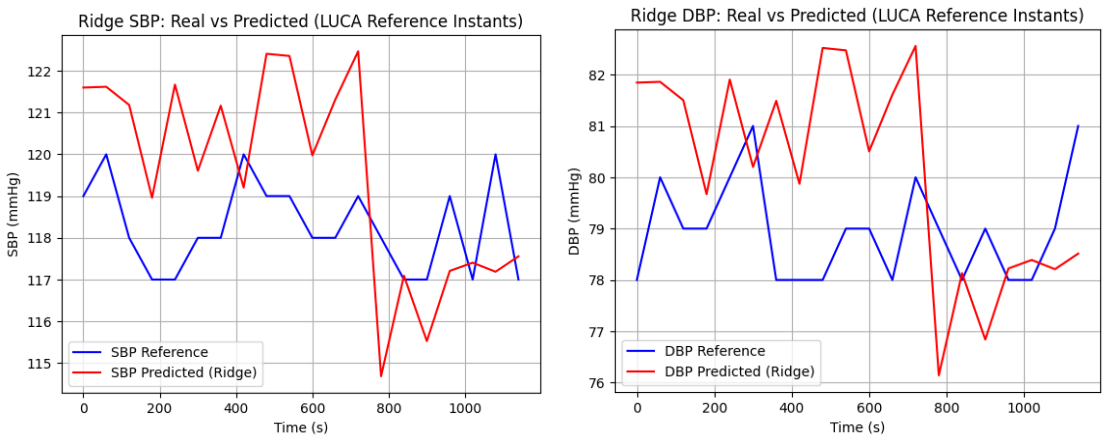
(a) SBP (b) DBP

Figure 36: Real Reference Values vs Predicted Vaules of SBP and DBP using the Ridge Regression on Luca's Database

- With the introduction of the last 22 Shimmer patients, a first significant drop in accuracy is observed. Despite the increase in the size of the training set, the presence of signals with different dynamics such as: different PPG response, variable signal-to-noise ratio, non-homogeneous signal morphology, prevents the model from optimizing the coefficients optimally for the target Luca. The L2 regularization, while stabilizing the system, is unable to compensate for the loss of coherence in the data, and the model, averaging on heterogeneous patterns, loses the ability to fine-tune to individual behavior. Details are available in Table 12 and Figures (37).

Average MAE (mmHg) $\pm$ Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	2.28 $\pm$ 2.25	2.08 $\pm$ 2.14

Table 12: Average MAE and STD (mmHg) for Patient Luca 1 using Ridge Regression.



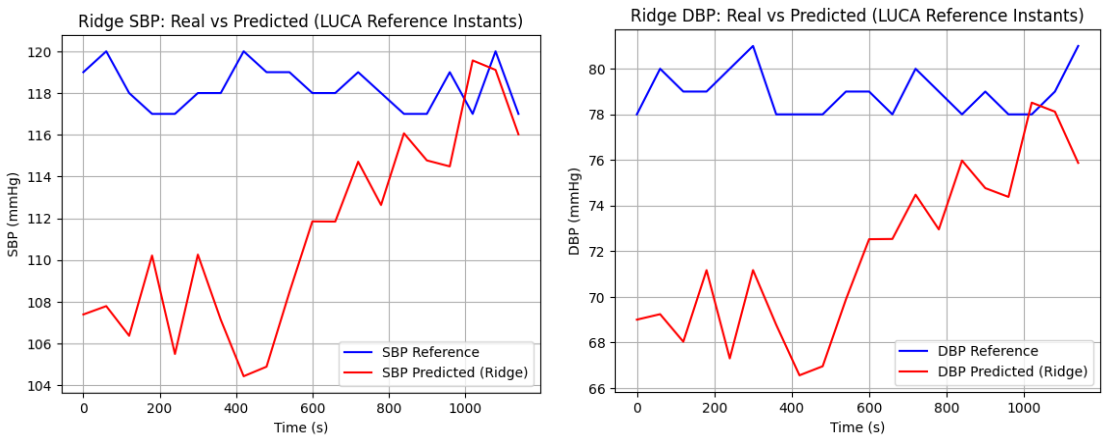
(a) SBP (b) DBP

Figure 37: Real Reference Values vs Predicted Vaules of SBP and DBP using the Ridge Regression on Luca's Database + Last 22 Shimmer Patients

- The addition of the first 20 Shimmer patients (excluding, again, patient 15) further exacerbates the situation. The model, already destabilized by the first wave of heterogeneous data, now struggles even more to maintain consistency in its estimates. An increase in both the mean error and its fluctuation is observed, indicating poor model stability when faced with non-stationary and non-homogeneous inputs. The results are shown in Table13 and Figures (38).

Average MAE (mmHg) $\pm$ Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	7.33 $\pm$ 4.92	7.09 $\pm$ 3.66

Table 13: Average MAE and STD (mmHg) for Patient Luca 1 using Ridge Regression.



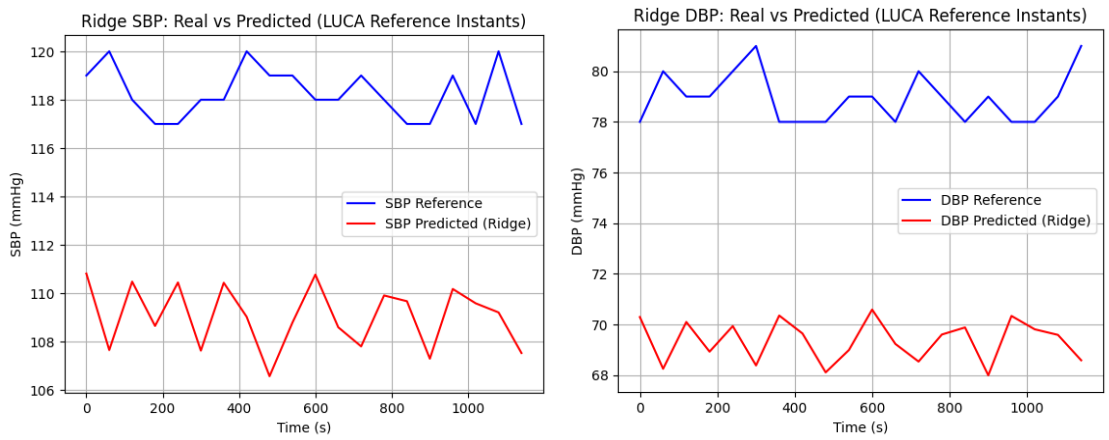
(a) SBP (b) DBP

Figure 38: Real Reference Values vs Predicted Vaules of SBP and DBP using the Ridge Regression on Luca's Database + Initial 20 Shimmer Patients

- In the largest configuration (LUCA + entire Shimmer dataset), contrary to what one might expect, Ridge Regression partially recovers performance, achieving the highest MAE and standard deviation values among all the tested configurations. The complete results are available in Table 14 and Figures (39).

Average MAE (mmHg) ± Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	9.19 ± 1.72	9.60 ± 1.54

Table 14: Average MAE and STD (mmHg) for Patient Luca 1 using Ridge Regression.



(a) SBP (b) DBP

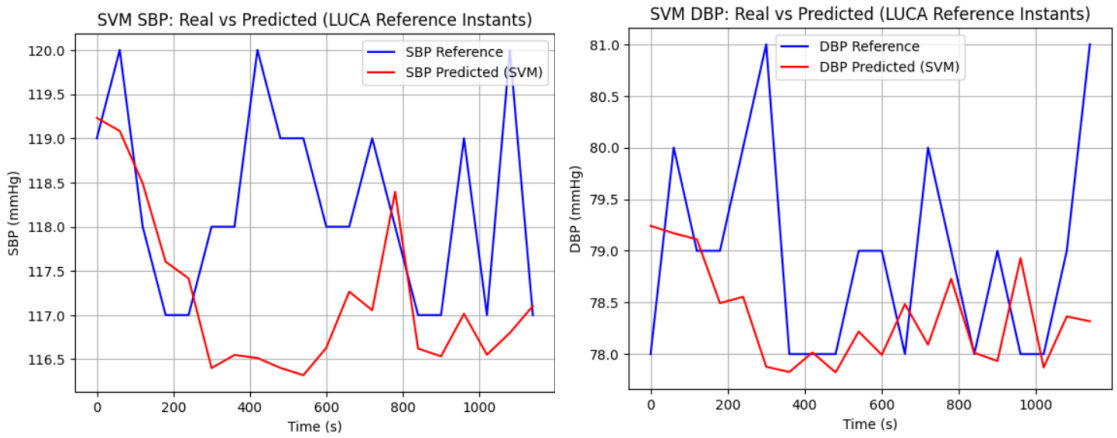
Figure 39: Real Reference Values vs Predicted Vaules of SBP and DBP using the Linear Regression on Luca's Database + All Shimmer Patients

4. **Support Vector Machine (SVM)**: Also, the Support Vector Machine (SVM), known for its robustness in managing high-dimensional spaces and for its ability to maximize the separation margin even in non-perfectly linear contexts (thanks to the use of kernels), was evaluated on four different configurations of the dataset, in order to analyze its behavior in the presence of data with different physiological and instrumental characteristics.

    - In the most focused configuration representing the use of Luca's database, the SVM reaches its peak performance. Thanks to the high temporal and physiological coherence of the signals, the algorithm is able to construct a separation hyperplane (or a decision surface in the case of nonlinear kernels) that is extremely close to the distribution of the target data, producing estimates that are almost perfectly aligned with the reference values of patient LUCA 1. The mean absolute error (MAE) is minimal and the standard deviation of the predictions is negligible, indicating a very high reliability and repeatability of the estimates. The quantitative results are reported in Table 15 and Figures (40).

Average MAE (mmHg) $\pm$ Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	1.19 $\pm$ 1.44	0.88 $\pm$ 1.06

Table 15: Average MAE and STD (mmHg) for Patient Luca 1 using SVM Regression.



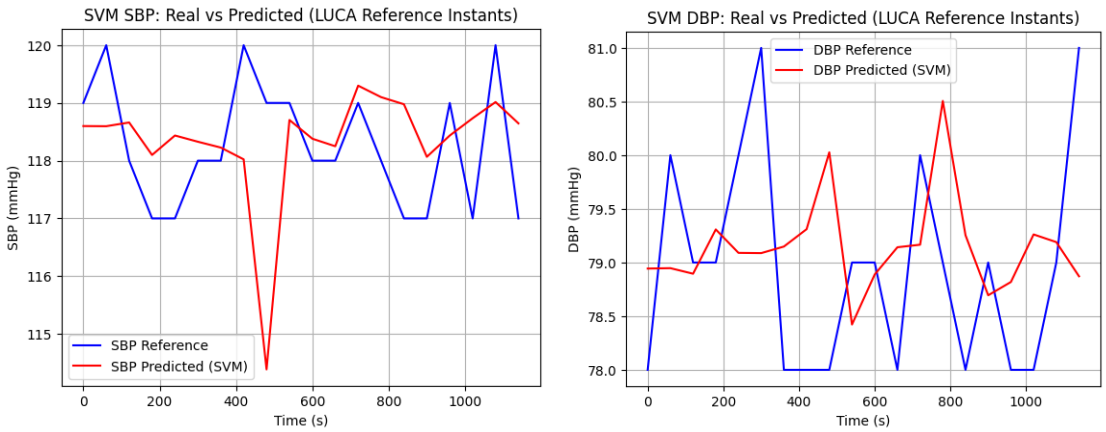
(a) SBP (b) DBP

Figure 40: Real Reference Values vs Predicted Vaules of SBP and DBP using the SVM Regression on Luca's Database

- With the introduction of the last 22 Shimmer patients, although a slight drop in performance was observed (inevitable given the introduced heterogeneity), the SVM still maintained a level of accuracy significantly higher than all the other tested models. Even in the presence of signals with different PPG responses, morphological variability of the ECG, or non-uniform SNR, the SVM managed to maintain good predictive capacity, avoiding the performance collapse observed in other approaches. Details are available in Table 16 and Figures (41).

Average MAE (mmHg) $\pm$ Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	1.12 $\pm$ 1.49	0.99 $\pm$ 1.14

Table 16: Average MAE and STD (mmHg) for Patient Luca 1 using SVM Regression.



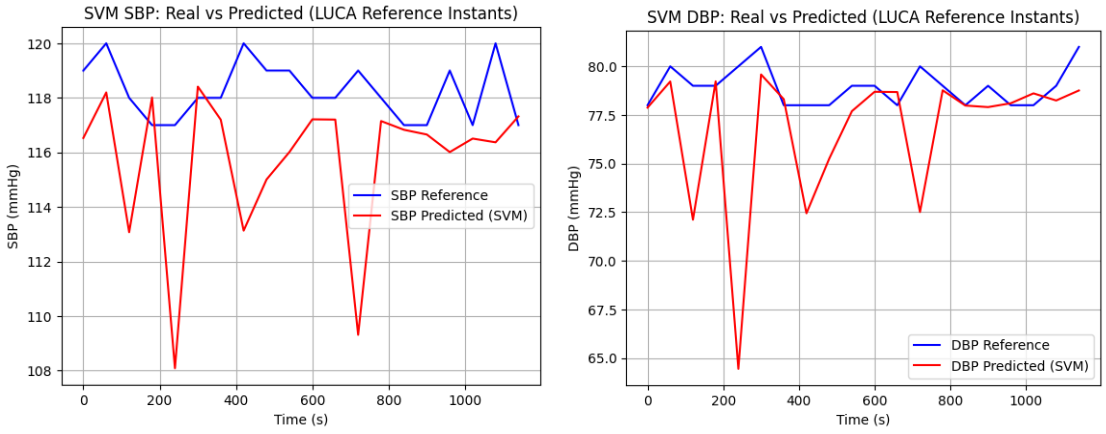
(a) SBP (b) DBP

Figure 41: Real Reference Values vs Predicted Vaules of SBP and DBP using the SVM Regression on Luca's Database + Last 22 Shimmer Patients

- The addition of the first 20 Shimmer patients (again excluding patient 15) fails to significantly destabilize the SVM. In fact, even in this case, the accuracy of the results deteriorates further. The results are shown in Table 17 and Figures (42).

Average MAE (mmHg) $\pm$ Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	2.71 $\pm$ 2.97	2.42 $\pm$ 3.85

Table 17: Average MAE and STD (mmHg) for Patient Luca 1 using SVM Regression.



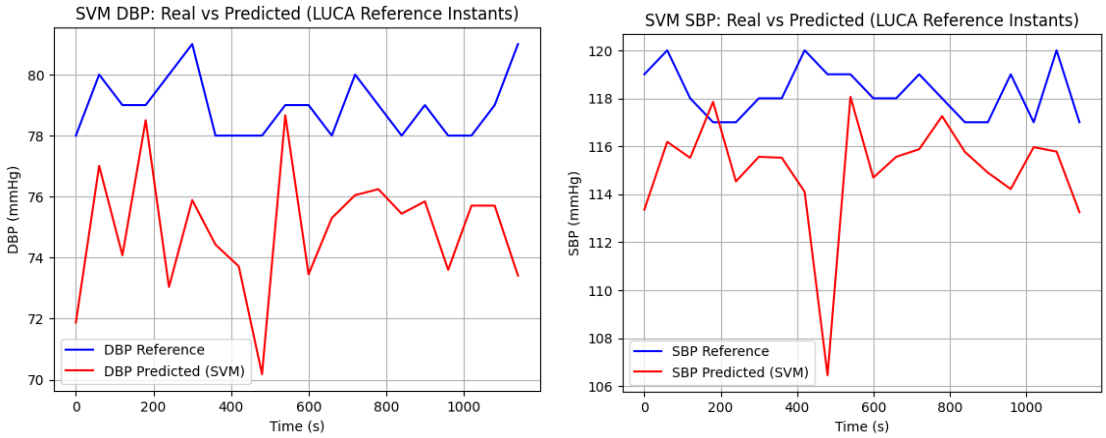
(a) SBP (b) DBP

Figure 42: Real Reference Values vs Predicted Vaules of SBP and DBP using the SVM Regression on Luca's Database + Initial 20 Shimmer Patients

- In the larger configuration (LUCA + entire Shimmer dataset), a deterioration in performance is observed, due to the high inter-subject variability and the presence of physiological patterns not aligned with the LUCA 1 target. However, even in this scenario, the SVM continues to perform best among all the models tested. Although the mean error increases and the standard deviation amplifies, the SVM still manages to produce more stable and less biased estimates than Ridge, LMS, or other linear approaches. The complete results, which unequivocally confirm the relative superiority of the SVM, are available in Table 18 and Figures (43).

Average MAE (mmHg) ± Average STD (mmHg)		
Patient	SBP	DBP
Luca 1	3.33 ± 2.69	4.04 ± 2.02

Table 18: Average MAE and STD (mmHg) for Patient Luca 1 using SVM Regression.



(a) SBP (b) DBP

Figure 43: Real Reference Values vs Predicted Vaules of SBP and DBP using the SVM Regression on Luca's Database + All Shimmer Patients

4.2.2 K-fold Cross-Correlation Dataset Division

As described in section 3, K-Fold Cross-Validation is a more robust and reliable machine learning model evaluation technique than the simple 70-30 train/test split. The method works by dividing the dataset into K parts (called "folds") of similar size. In each iteration, the model is trained in K-1 folds and validated on the remaining fold. This process is repeated K times, so that each fold serves as the test set exactly once. The final performance of the model is given by the average performance achieved in each of the K iterations.

Looking at the results, reported in the tables and figures, the comparison between a metric (e.g., MAE - Mean Absolute Error and STD - Standard Deviation) obtained with a 70-30 split and that obtained with Cross-Validation is not to be understood as an "improvement" or "worsening" of the model itself, but rather as a correction or refinement of the estimate of its actual performance. In fact, we obtained two cases:

1. **Result 70-30 "Bad" (high MAE and STD) that "improves" with CV**: A 70-30 split is strongly influenced by the randomness of the split. It is possible that 30% of the data reserved for testing contains many examples that are particularly difficult to predict (outliers, unusual patterns or, as in this case, data not consistent with the reference patient), while the training set is "easier." This leads to a pessimistic and biased estimate of performance (high MAE and STD). K-Fold CV, by averaging performance across all possible splits, mitigates this risk. If each fold provides a decent MAE, the average will be good, revealing that the model is actually more capable than it appeared from the single unfavorable test set. CV therefore provides a more optimistic and realistic estimate of the model's generalization.

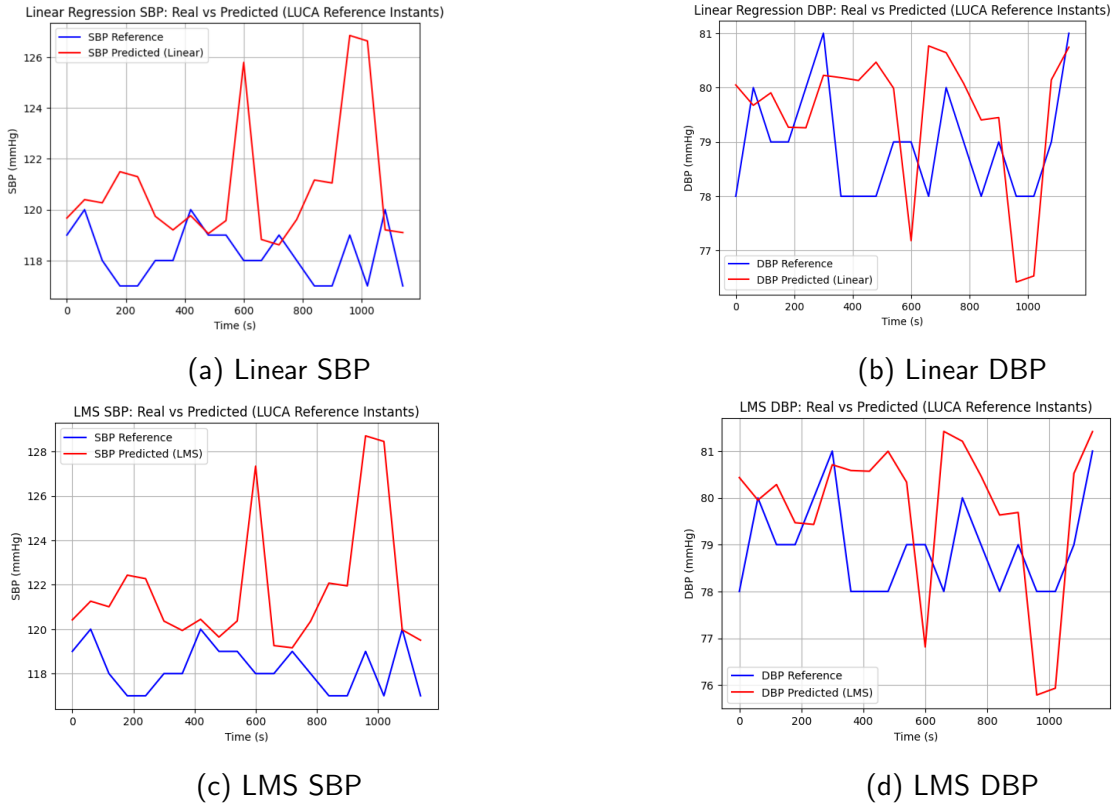
2. **Result 70-30 "Good" (low MAE and STD) which "worsens" with CV**: In contrast, the 70-30 split may be highly favorable. The test set may consist of examples very similar to the training set and particularly easy to predict, the training set may be statistically unrepresentative of the entire dataset, or, as in this case, the data are similar to the reference patient. This leads to an overly optimistic estimate of performance (low MAE). K-Fold CV, by testing the model on all data subsets, unmasks this excessive optimization (overfitting) or fortuitous split. If the model performs poorly in some folds because it was tested on different data, the average MAE will increase. CV therefore provides a more pessimistic, but much more reliable estimate, highlighting the model's potential vulnerability.

The numerical results (MAE) and their variability (std) demonstrating this behavior for the different datasets analyzed are reported in detail in the attached Tables and Figures. The analysis of the standard deviation is particularly important: a high value indicates that the model's performance is very sensitive to the choice of training/test data, confirming the usefulness of CV for obtaining a more stable judgment.

- **LUCA Database**:

	Avg MAE (mmHg) ± Avg STD (mmHg)							
	Linear		LMS		Ridge		SVM	
	SBP	DBP	SBP	DBP	SBP	DBP	SBP	DBP
Luca 1	3.15±0.01	2.45±0.01	3.39±0.02	2.59±0.01	3.15±0.03	2.45±0.01	2.84±0.03	2.22±0.02

Table 19: Average MAE (mmHg) ± Average STD (mmHg) for different models using Luca's database post k-fold cross-validation.



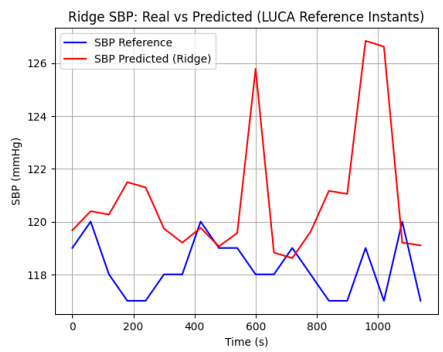
(a) Linear SBP

(b) Linear DBP

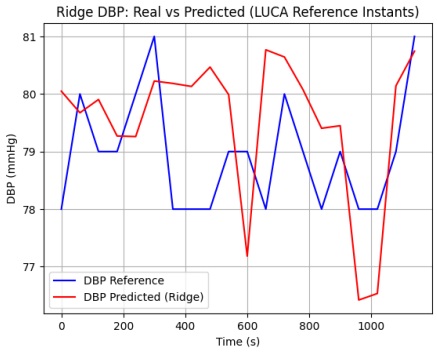
(c) LMS SBP

(d) LMS DBP

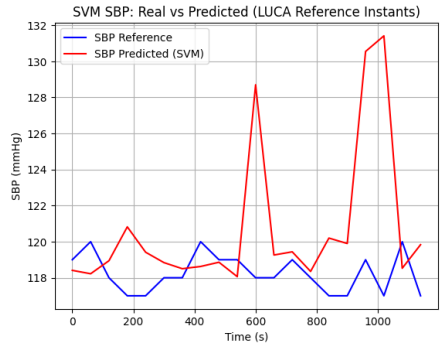
Figure 44: Relation between Reference Values and Predicted Values of SBP, DBP for Linear and LMS Regression Models, post K-fold cross-correlation on Luca's database.



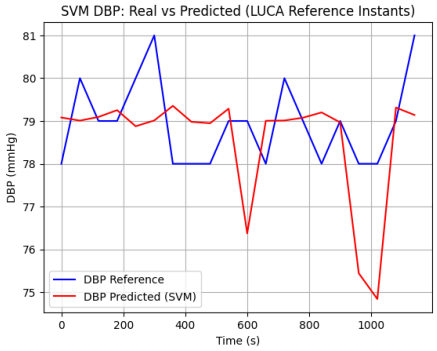
(a) Ridge SBP



(b) Ridge DBP



(c) SVM SBP



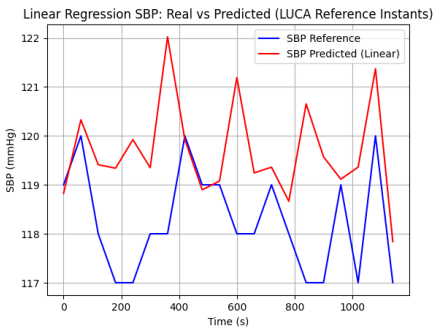
(d) SVM DBP

Figure 45: Relation between Reference Values and Predicted Values of SBP, DBP for Ridge and SVM Regression Models, post K-fold cross-correlation on Luca's database.

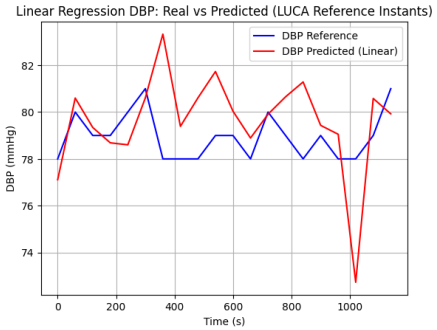
- **LUCA Database + Last 22 Shimmer Patients**

	Avg MAE (mmHg) ± Avg STD (mmHg)							
	Linear		**LMS**		**Ridge**		**SVM**	
	SBP	**DBP**	**SBP**	**DBP**	**SBP**	**DBP**	**SBP**	**DBP**
Luca 1	4.39±0.04	2.25±0.03	4.39±0.08	2.99±0.04	4.39±0.04	2.92±0.03	3.81±0.04	2.61±0.06

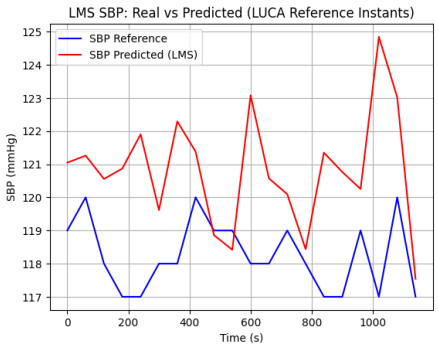
Table 20: Average MAE (mmHg) ± Average STD (mmHg) for different models using Luca's database + Last 22 Shimmer Patients, post k-fold cross-validation.



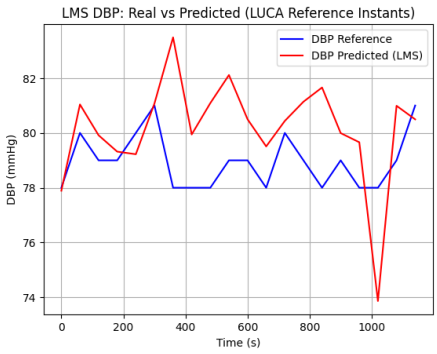
(a) Linear SBP



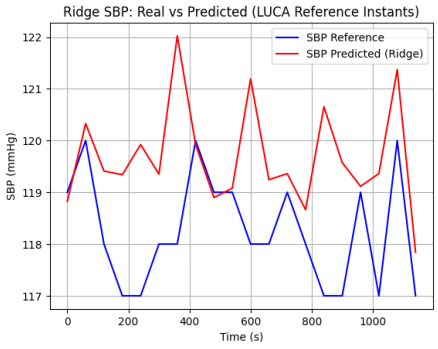
(b) Linear DBP



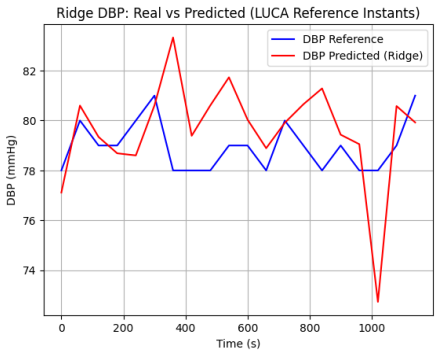
(c) LMS SBP



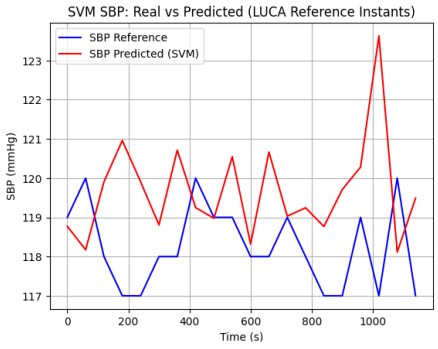
(d) LMS DBP



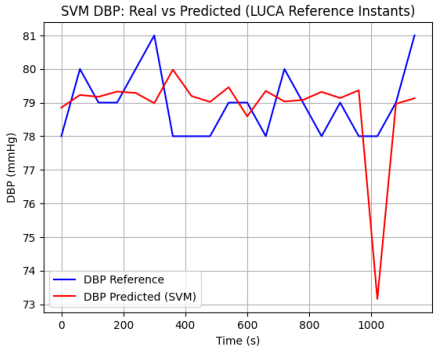
(e) Ridge SBP



(f) Ridge DBP



(g) SVM SBP



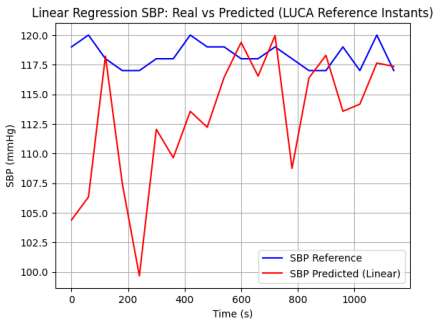
(h) SVM DBP

Figure 46: Relation between Reference Values and Predicted Values of SBP, DBP for the all Regression Models, post K-fold cross-correlation on Luca's database + Last 22 Shimmer Patients.

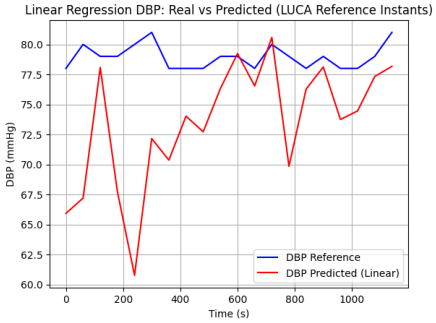
- **LUCA Database + Initial 20 Shimmer Patients**

	Avg MAE (mmHg) ± Avg STD (mmHg)							
	Linear		LMS		Ridge		SVM	
	SBP	DBP	SBP	DBP	SBP	DBP	SBP	DBP
Luca 1	9.80±0.04	8.72±0.02	10.63±0.12	8.73±0.08	9.80±0.04	8.72±0.02	7.83±0.11	6.45±0.08

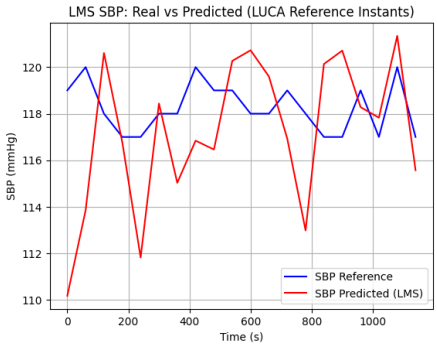
Table 21: Average MAE (mmHg) ± Average STD (mmHg) for different models using Luca's database + Initial 20 Shimmer Patients, post k-fold cross-validation.



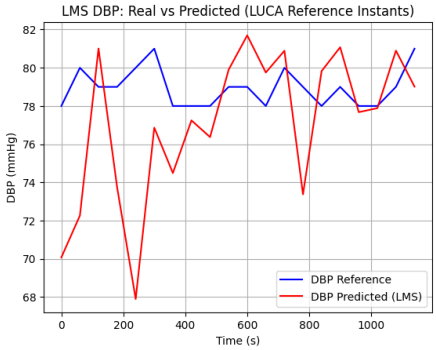
(a) Linear SBP



(b) Linear DBP

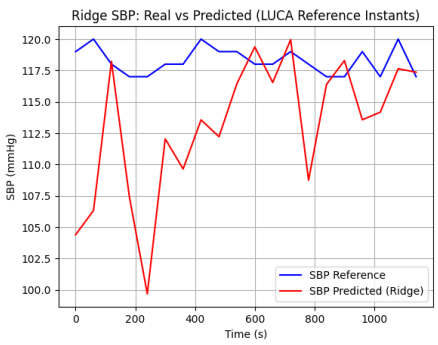


(c) LMS SBP

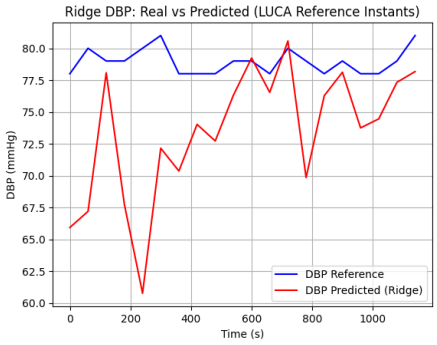


(d) LMS DBP

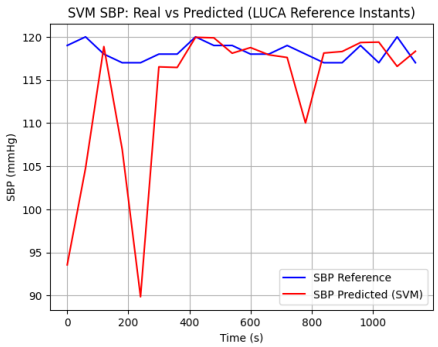
Figure 47: Relation between Reference Values and Predicted Values of SBP, DBP for Linear and LMS Regression Models, post K-fold cross-correlation on Luca's database + Initial 20 Shimmer Patients.



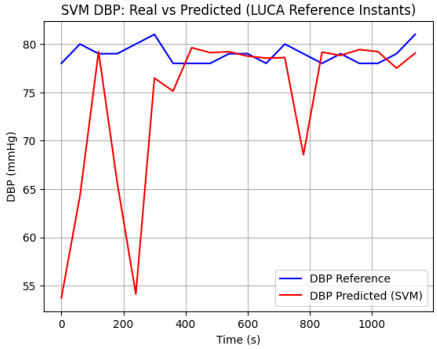
(a) Ridge SBP



(b) Ridge DBP



(c) SVM SBP



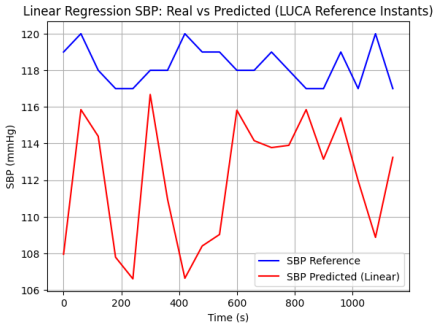
(d) SVM DBP

Figure 48: Relation between Reference Values and Predicted Values of SBP, DBP for Ridge and SVM Regression Models, post K-fold cross-correlation on Luca's database + Initial 20 Shimmer Patients.

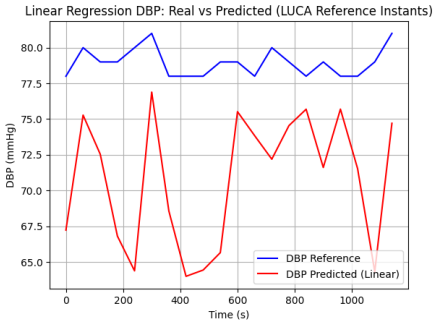
- **LUCA Database + all Shimmer Patients**

	Avg MAE (mmHg) ± Avg STD (mmHg)							
	Linear		**LMS**		**Ridge**		**SVM**	
	SBP	**DBP**	**SBP**	**DBP**	**SBP**	**DBP**	**SBP**	**DBP**
Luca 1	10.88±0.02	8.52±0.02	11.23±0.07	8.19±0.03	10.88±0.02	8.52±0.02	9.17±0.04	6.93±0.03

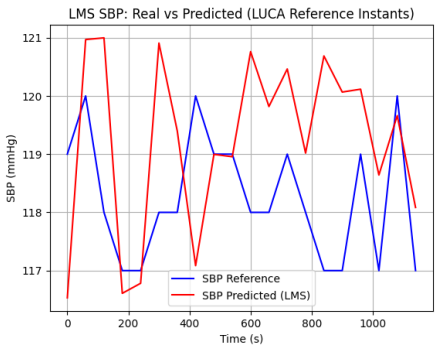
Table 22: Average MAE (mmHg) ± Average STD (mmHg) for different models using Luca's database + All Shimmer Patients, post k-fold cross-validation.



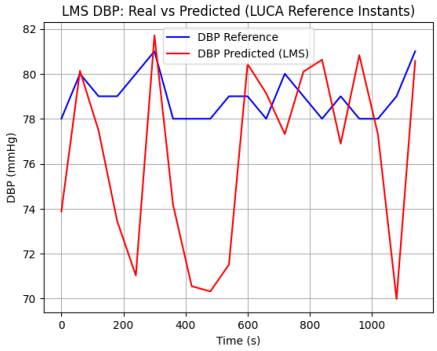
(a) Linear SBP



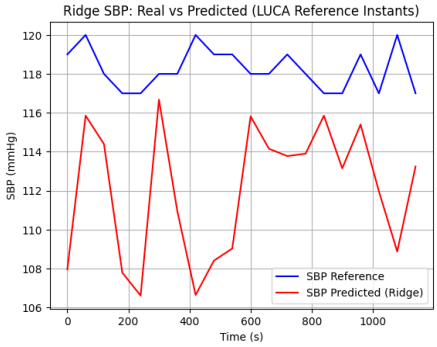
(b) Linear DBP



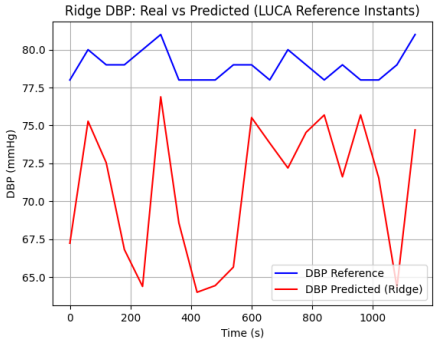
(c) LMS SBP



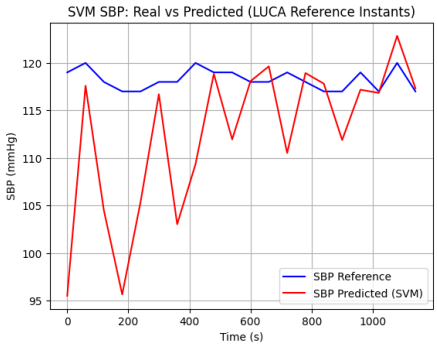
(d) LMS DBP



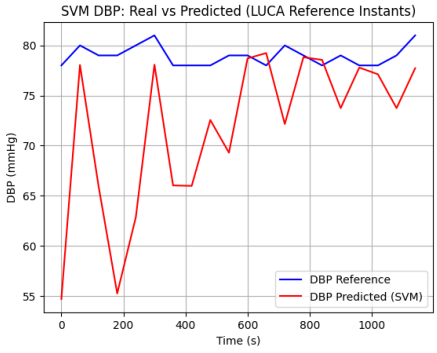
(e) Ridge SBP



(f) Ridge DBP



(g) SVM SBP



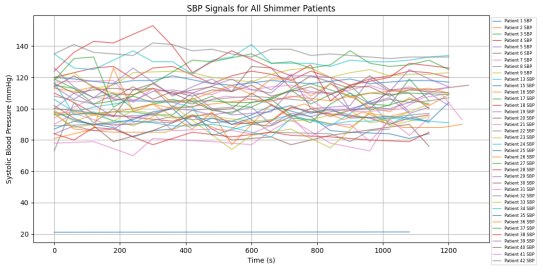
(h) SVM DBP

Figure 49: Relation between Reference Values and Predicted Values of SBP, DBP for the all Regression Models, post K-fold cross-correlation on Luca's database + All Shimmer Patients.

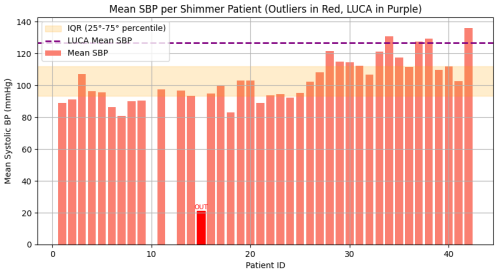
5 Discussion

5.1 Algorithm results Discussion

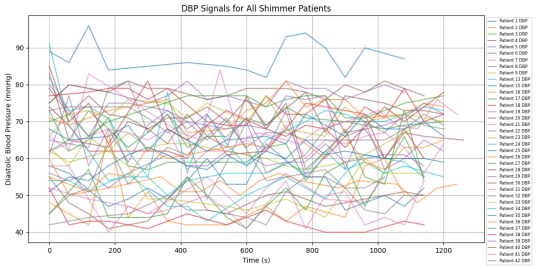
The results obtained in this thesis, analyzed in light of the physiological variability of the patients and the robustness of the algorithms, lead to an unequivocal conclusion: of all the models tested (Linear Regression, LMS, Ridge Regression, and Support Vector Machine), SVM stands out for its absolute superiority, not only in terms of point by point accuracy, but above all in terms of robustness, stability, and generalization ability across real world and heterogeneous contexts. This statement does not arise from a simple comparison of numbers, but from an in depth analysis that links quantitative performance to the intrinsic characteristics of the data and the theoretical principles of the algorithms. This analyzes are made even more evident and convincing by examining the graphs illustrating the distribution of blood pressure values in different patients (Figures: 50a,50c,51a,51c).



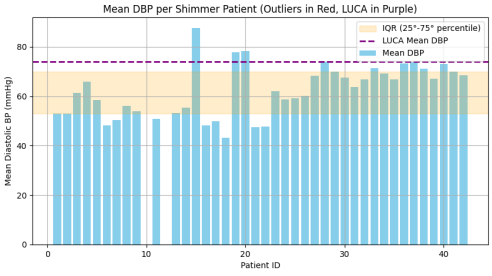
(a) SBP reference values



(b) SBP Bar Graph of refrence values

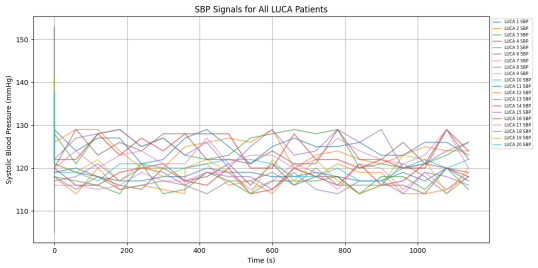


(c) DBP reference values

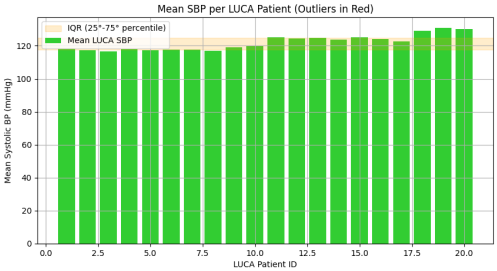


(d) DBP Bar Graph of refrence values

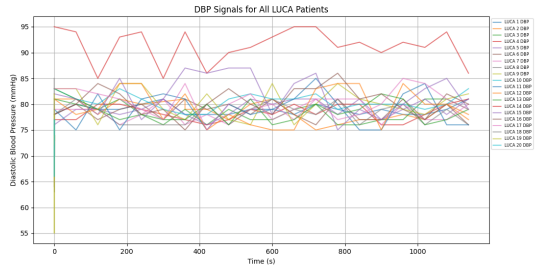
Figure 50: Figures of Shimmer Patients representing the behavior of the SBP, DBP reference values with repect the LUCA 1 mean values.



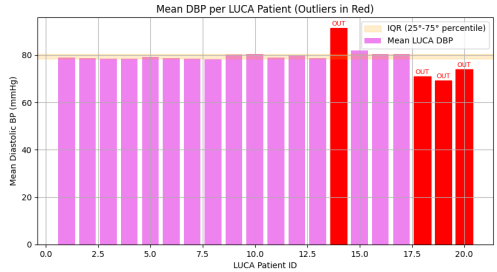
(a) SBP reference values



(b) SBP Bar Graph of refrence values



(c) DBP reference values



(d) DBP Bar Graph of refrence values

Figure 51: Figures of Luca's database Patients representing the behavior of the SBP, DBP reference values with repect the LUCA 1 mean values.

From the graphs we can visualize that, when the models are trained and tested exclusively on homogeneous data from patient Luke 1, they all achieve excellent results, with Ridge and SVM competing for the top spot. However, this scenario is artificial: in clinical practice, a cuffless blood pressure measurement device must work on a heterogeneous population, with patients presenting with profoundly different blood pressure values, signal morphologies, and physiological conditions. And this is where the real competition comes in, and where basic data analysis, made visible by the bar graphs of the average SBP and DBP values, highlighting outliers and the mean value for Luke 1, becomes crucial to interpreting the results.

In fact, the statistical analysis of Shimmer patients' baseline values reveals that: Luca 1 is a specific case with blood pressure values in the mid to high range of the Shimmer distribution. The graphs (50d,50b) clearly show that most Shimmer patients have significantly lower SBP and DBP values . Some can even be classified as statistical outliers, identified by the interquartile range (IQR) method and marked in red in the graphs. These patients have mean diastolic blood pressure values below 60 mmHg, and mean systolic blood pressure values ranging from less than 100 mmHg to over 140 mmHg, representing a high level of variability that is directly reflected in the models' performance. The dashed purple line, representing the mean for Luca 1, appears isolated in this context, almost an exception compared to the norm for the Shimmer sample. This heterogeneity is not noise to be eliminated, but the reality that a clinical model must deal with, and it perfectly explains the behavior observed in the results in Chapter 4. In

contrast, when we used the LUCA patient database, which shows an average trend in line with the values of patient LUCA 1 (Figures 51d,51b), the results are highly accurate.

We can observe, from the results, that when the high variability is introduced into the training set by the Shimmer patients, Linear Regression collapses inexorably. Its errors skyrocket to clinically unacceptable values, well above 10 mmHg, demonstrating extreme fragility when faced with data that do not follow the same distribution as the target. Ridge Regression, despite starting from a solid foundation, also shows a surprising vulnerability: its L2 regularization, so effective on homogeneous data, becomes a limitation when it has to handle irreconcilable physiological patterns, "crushing" the coefficients to the point of rendering the model incapable of adapting (results above the limits set by the international guidelines (AAMI/ISO/ESH)).

LMS, thanks to its iterative adaptive approach, shows remarkable resilience, maintaining relatively low errors even in the presence of heterogeneity. However, its performance is unstable: it deteriorates dramatically with the first 20 Shimmer patients (those with lower blood pressures), then paradoxically "improve" with the entire dataset, suggesting that its behavior is more reactive than proactive, and dependent on the fortuitous composition of the training set. SVM, on the other hand, maintains remarkably consistent and robust behavior. From the first impact with heterogeneous data (the addition of the last 22 Shimmer patients), SVM shows minimal performance degradation, always remaining below 2 mmHg error for both blood pressures. Even in the most extreme configuration, with the entire Shimmer dataset, SVM is the only model that keeps diastolic blood pressure (DBP) within clinical limits (4.04 ± 2.02 mmHg) and systolic blood pressure (SBP) at a reasonable level (3.33 ± 2.69 mmHg), significantly lower than all the others. This superiority is not accidental, but is rooted in the very nature of the algorithm. SVM does not attempt to fit a mean hyperplane to the data, but rather constructs a decision surface that maximizes the separation margin, making it intrinsically robust to outliers and noise. Furthermore, thanks to the use of nonlinear kernels (such as RBF), SVM is able to model complex and nonlinear relationships between features (HR, PTT) and blood pressure values, capturing physiological nuances that linear models ignore.

Validation with K-Fold Cross-Validation confirms and amplifies this superiority. While CV "corrects" the estimates for other models, revealing overfitting or underfitting hidden by the 70-30 split, for SVM the correction is minimal, indicating that its estimates are already intrinsically stable and reliable. Importantly, on the full dataset, SVM is the only model that, with CV, maintains an error for DBP (6.93 ± 0.03 mmHg) close to the clinical threshold, while all the others significantly exceed 8 mmHg.

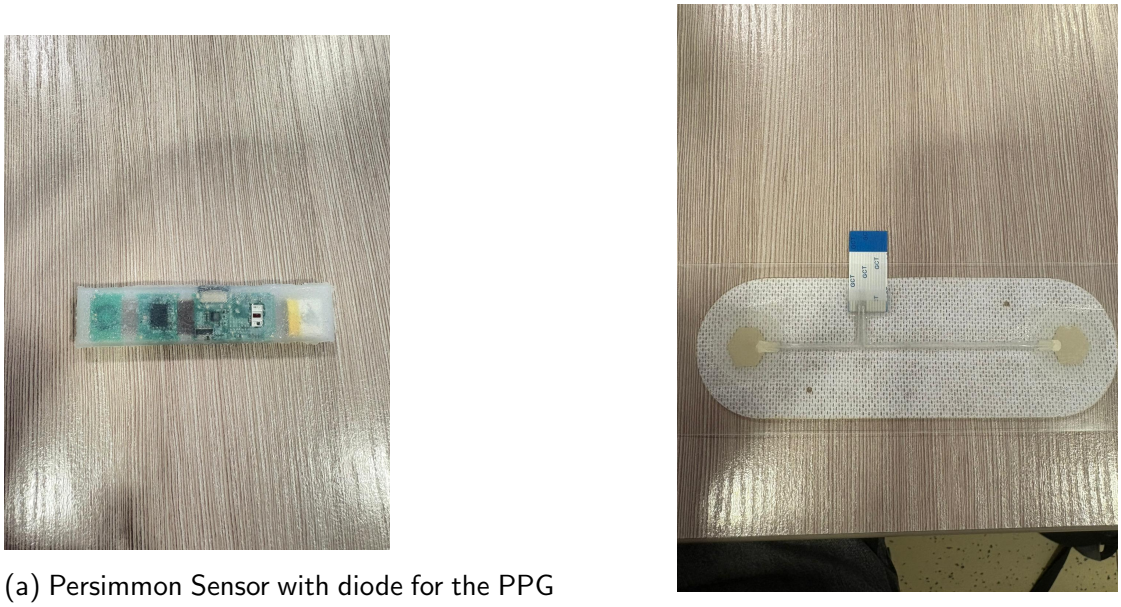
In conclusion, although Ridge and LMS can compete on homogeneous data, SVM is the only algorithm that demonstrates a systematic, robust, and clinically valid ability to generalize to a heterogeneous population. Its architecture, based on margins and kernels, makes it the ideal tool for addressing the complexity and variability of the real world.

5.2 Errors and strategies

However, it is important to note that the relatively high MAE and STD values found in some scenarios, especially in heterogeneous contexts, can be attributed to different sources of error and the strategies adopted to mitigate them:

1. **Intrinsic Physiological Variability**: The high heterogeneity of Shimmer data, with patients presenting blood pressure values across a very wide range, represents the main challenge. Linear models, by their nature, attempt to approximate an average relationship, failing when they must adapt to radically different physiological regimes. Even the SVM, although more robust, must contend with this variability, which physiologically translates into a lower limit on achievable accuracy.

2. **PPG and ECG intersubject variability**: This is due to the subject's physiology [74]. In particular for PPG, whose recording is influenced by the subject's skin conductivity, sometimes the peaks are so low that it is difficult to establish a threshold value to detect them optimally [75]. At the same time, ECG signal variability is also affected by similar problems, particularly when the electrodes are positioned in a non-correct anatomical configuration, leading to potential artifacts and reduced wave amplitude. The sensor, developed by Persimmon, was designed to address these critical issues. It improves detection of PPG S_{peaks} while simultaneously reducing errors resulting from improper Shimmer clip placement and light interference, thanks to the use of a black adhesive patch (Figures: 52). This integrated solution therefore aims to ensure greater reliability of the acquired PPG and ECG signals, under varying physiological and operational conditions.



(a) Persimmon Sensor with diode for the PPG signal

(b) Electrodes

Figure 52: Persimmon Sensor and the Electrodes for The ECG signals .

3. **Instrumental error**: The technical documentation of the EXG module does not specify error margins, since this device allows the recording of the ECG trace without introducing evident alterations due to noise. Consequently, the instrumental uncertainty associated with this module is insignificant. On the contrary, the PPG signal, due to its intrinsic characteristics and acquisition method, is affected by disturbances which also include a systematic measurement error, documented in the literature as an average value calibrated over the entire measurement range [76]. The bias voltage is set to 0.5 V and, considering a typical "low" skin impedance (120 $k\Omega$ corresponding to 8 μS), the bias current is approximately 5 μA; this current value decreases proportionally with the increase in skin conductance [76]. The latest generation Persimmon sensors show the ability to reduce instrumental uncertainty for both signal types, ECG and PPG.

4. **Motion Artifacts**: Although the experimental protocol required subjects to be seated and relaxed, involuntary micro-movements may have introduced artifacts into the signals, especially in the PPG (which is more susceptible) but also in the ECG. Such interference is difficult to completely eliminate through filtering and contributes significantly to increased variability (STD) of the estimates. This issue is particularly relevant with Shimmer devices, where the cables connecting the electrodes to the acquisition module can create additional sources of mechanical disturbance due to accidental displacement of the conductors. In contrast, Persimmon sensors are designed to mitigate these issues through the use of a dedicated adhesive that ensures stable and uniform attachment to the skin,

minimizing both relative sensor movement and cable traction artifacts. This integrated solution improves immunity to mechanical disturbances, preserving signal quality even in the presence of small subject movements.

5. **Synchronization and Time Alignment**: Even a small misalignment between the ECG and PPG signals can cause an incorrect PTT calculation. The 60 ms offset applied to correct for instrument delay is an estimate that may not be perfect for all subjects or hardware configurations, introducing systematic bias.

6. **Cleaning the HR and PTT arrays**: The heart rate (HR) and pulse transit time (PTT) array cleaning phase involves eliminating outliers that exceed the mean $\pm$ standard deviation range. While this process helps improve the accuracy of blood pressure predictions, it inevitably results in the loss of potentially relevant information. Although several strategies have been tested to mitigate this information loss, none have yet been shown to significantly improve the model's predictive performance. To address this issue, Persimmon is developing more advanced signal acquisition technologies that aim to obtain cleaner and more reliable physiological data already during the detection phase.

7. **Calibration and Personalization**: The model was trained on a collective dataset. The lack of specific calibration for the individual patient (LUCA 1) explains part of the residual error. Linear models, lacking intrinsic mechanisms to handle inter-individual differences, suffer dramatically. The SVM, with its RBF kernel, handles these differences better, but the best results would likely be achieved with training or fine-tuning on data specific to the target patient.

5.3 Persimmon Devices

Persimmon devices are conceived as multimodal, personalized, and biodegradable smart patches designed for decentralized personal health monitoring (DPHM). The project's goal is to produce soft, skin-conformable sensory patches capable of detecting relevant physiological parameters (including cardiac parameters) and transferring data to cloud/edge infrastructures for sensor fusion and estimation of parameters such as blood pressure and body temperature. The patches will be produced using additive manufacturing and digital surface mount technology (SMT), using innovative materials such as water-soluble biopolymers and liquid metal interconnects to increase sustainability and conformability [77].

From a mechatronics perspective, a Persimmon device is designed as a multilayer platform composed of: a soft and biodegradable substrate (biopolymer) that adheres to the skin; printed conductive layers (possibly liquid metal) for the electrical traces; digitally positioned SMT components (low-power microcontrollers, ADCs, LEDs, photodiodes for PPGs, front-end amplifiers for ECGs, motion sensors, accelerometers and nano MOS for specific signals); and a wireless

interface to a gateway (with a view to 5G/IoT) or directly to a smartphone. The choice of liquid metals and biodegradable materials is designed to reduce environmental impact and enable multi-use or single-use modules with recycling of active components [77].

5.3.1 ECG and PPG: sensors and detection

For ECG signal detection, the patch integrates skin-contact electrodes, which can be made of printed conductive pads or soft metal films. The analog interface includes a front-end with a low-noise amplifier, a band-pass filter to isolate the useful cardiac frequency range (typically 0.5–40 Hz), and a low-power A/D converter that samples the signal. To maintain a good signal-to-noise ratio on a soft substrate, electrode geometry optimization techniques, surface treatment, and gain adaptation algorithms are employed [77]. While the PPG is acquired via an integrated optical module (LED and photodiode), as shown in figure 53.

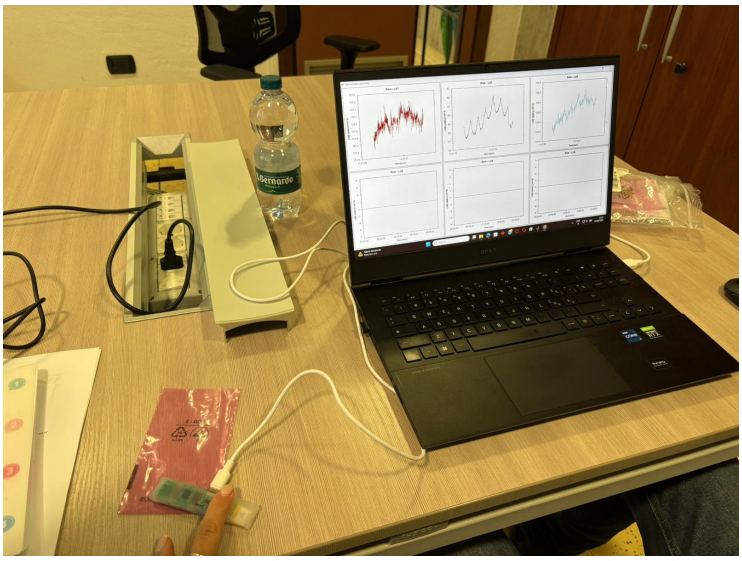


Figure 53: Acquisition of PPG signal through the photodiode

The patch can include LEDs at different wavelengths (e.g., green/red/IR) to improve penetration and sensitivity at different tissue levels; PPG is read via a dedicated analog front-end (photodetector, transimpedance amplifier, low-pass filter) before digitization. Some Persimmon projects also include modules for temperature sensors and gas sensors integrated into the same "multimodal" platform [77].

A key aspect of cuffless pressure estimation is time synchronization between signals. Persimmon patches are designed to operate in multi-nodal networks on the skin: multiple nodes positioned at different points can enable relative measurements (e.g., measuring PTT between two patches) or provide redundancy to reduce artifacts. Synchronization can be managed locally (common clock between modules or gateway synchronization) or at the cloud level after

upload, but the project emphasizes edge AI and the automatic selection of time slots with good signal conditions to reduce the amount of unnecessary data sent over the network [77].

5.3.2 Software and Edge-AI

Persimmon integrates edge-AI solutions directly onto the patch or body gateway for key tasks: selecting "clean" signal windows, reducing motion artifacts (using an accelerometer as a reference and lightweight neural networks to classify signal conditions), intelligent compression and filtering, and local decision-making on when to send data to the cloud. At the cloud level, multimodal sensor fusion (fusion of ECG, PPG, temperature, and motion data) is performed using more computationally intensive algorithms that estimate complex clinical variables, including blood pressure. This hybrid architecture (edge + cloud) is explicitly designed to reduce energy consumption, improve privacy, and reduce operational latency. At the software level, edge-AI intervenes in the quality evaluation and selection phase, while the final pressure estimation can be performed with models calibrated on the cloud or, when required, with lightweight versions directly on the device.

Specifically, SBP and DBP pressures can be precisely calculated by setting the device's characteristics. This allows the specifics to be adjusted based on the patient being examined.

5.3.3 Persimmon Device Manufacturing, Sustainability, Use Cases, and Challenges

Persimmon devices stand out for their focus on sustainability and a circular economy design approach. They are produced using additive manufacturing, which enables flexible electronic circuits to be created on soft substrates, reducing waste. Component assembly is handled by digital surface mount technology (SMT), a cost-effective and precise technique. The use of water-soluble biopolymers makes the patches biodegradable and facilitates the separation of reusable components, thus limiting plastic waste (Figures 54). Liquid metal interconnects, designed to facilitate metal recovery and reuse, further strengthen this approach, enabling devices suitable for both single use and partially reusable use, while keeping costs and environmental impact low [77].

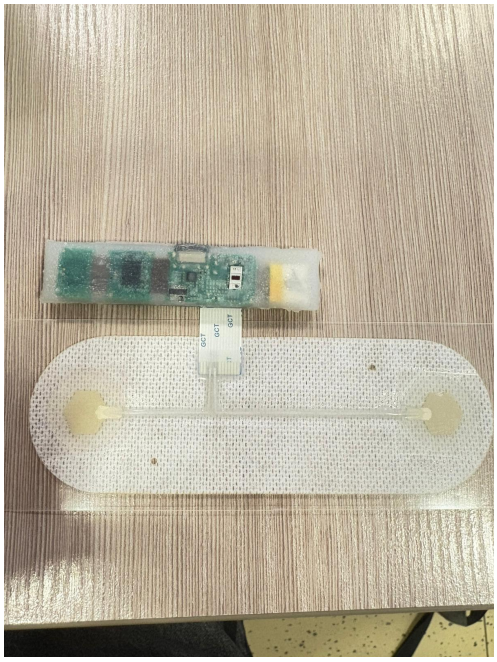


Figure 54: Structure of Persimmon device for the acquisition of the ECG, PPG signals

This technology platform was designed to be tested in particularly demanding real-world scenarios, such as extreme sports like ski mountaineering and swimming (video: [78]), which require high water resistance, tolerance to temperature changes, and signal reliability in critical environmental conditions. At the same time, Persimmon patches are designed for the continuous monitoring of chronic patients on a daily basis, a context that requires stable and secure connectivity. To this end, the system integrates 5G gateways for wearable IoT networks, capable of transmitting physiological data in real time to cloud platforms for signal analysis and fusion. Clinical validation involves campaigns on diverse populations, necessary to calibrate the blood pressure estimation models and ensure compliance with international accuracy and safety standards [77].

Despite promising application prospects, Persimmon devices face several technical challenges to ensure reliable reading of ECG and PPG signals. These include managing motion artifacts, maintaining stable electrode-to-skin contact for ECG, and reducing optical noise for PPG in environments with variable lighting. Added to this are the module's battery life, which must ensure prolonged operation without frequent recharging, and the subject-specific calibration required to accurately estimate blood pressure [77] [78].

So, Persimmon's ECG and PPG reading devices represent an integrated and sustainable platform that combines advanced materials, additive manufacturing, low-power electronics, edge-AI, and sensor fusion cloud to achieve multimodal measurements useful for monitoring blood pressure and other vital signs in the clinical and sports fields.

6 Conclusion

Hypertension is therefore a major risk factor for cardiovascular disease, the leading cause of mortality globally. Continuous and accurate blood pressure (BP) monitoring is therefore essential not only for early diagnosis but also for patient therapeutic management and the prevention of complications.

This thesis addressed the limitations of traditional methods, the invasiveness of clinical systems, and the intermittent nature of noninvasive cuff-based devices, subsequently exploring a "cuffless" methodology for continuous and noninvasive blood pressure estimation. The core of the work was the development of a framework based on the combined analysis of electrocardiogram (ECG) and photoplethysmogram (PPG) signals, acquired through Shimmer3 wearable devices, from which the key physiological parameters of heart rate (HR) and pulse transit time (PTT) were extracted. The approach incorporated several innovations: the acquisition of signals under controlled conditions as an alternative to online datasets, the design of a comprehensive signal processing pipeline, and the application of regression techniques to model the relationship between these parameters and reference blood pressure values. The results showed that the Support Vector Machine (SVM) algorithm stands out for its robustness and generalisation capacity, keeping the error (MAE) within clinically acceptable thresholds for diastolic blood pressure (DBP), in line with international AAMI/ISO/ESH guidelines [61].

This confirms the potential of the proposed approach for the development of a reliable system for real-time estimation.

6.1 Future prospects

Looking ahead, research is evolving along two main lines: hardware integration and software refinement. From a hardware perspective, the goal is to implement the algorithm on nextgeneration wearable platforms that are more compact and comfortable for prolonged use, such as those developed in the European Persimmon project. Although these sensors have different technical specifications from Shimmer, they have demonstrated encouraging performance and greater adaptability to everyday life, paving the way for large-scale, long-term monitoring outside the clinical setting [77].

On the software side, despite the excellent performance of SVM, there is ample room for improvement by exploring more complex algorithmic architectures. The use of recursive algorithms (such as Kalman filters) and neural networks (RNN/LSTM) appears particularly promising for more effectively capturing the dynamic and temporal nature of blood pressure. This would not only improve the accuracy of systolic (SBP) and diastolic (DBP) estimates, but also lay the foundations for increasingly personalised, predictive and integrated monitoring in patients' lives. In conclusion, the experimental results and development prospects paint a solid picture for the clinical and commercial adoption of cuffless blood pressure monitoring systems.

Integration with projects such as Persimmon represents the natural evolution of this research, with the aim of bringing continuous blood pressure measurement beyond the hospital environment and directly into patients' daily lives, contributing significantly to improving their quality of life and reducing the social impact of cardiovascular disease.

Appendix A: Additional Results

Results of All patients pre and post K-Fold cross correlation considering both Shimmer and Luca Dataset

Table 23: Metrics Pre K- Fold Cross-Correlation (MAE and STD)

Model	Linear				LMS				Ridge				SVM			
Signal	DBP		SBP		DBP		SBP		DBP		SBP		DBP		SBP	
Metric	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD
SHIMMER_1	10.70	14.07	17.86	21.66	10.39	13.91	16.09	20.56	10.70	14.07	17.86	21.66	9.69	13.14	18.26	22.55
SHIMMER_2	11.16	14.32	16.88	19.95	10.86	14.14	15.01	18.57	11.16	14.32	16.88	19.95	9.88	12.98	16.97	20.45
SHIMMER_3	10.59	13.55	12.01	13.44	10.30	13.38	10.24	11.94	10.59	13.55	12.01	13.43	9.47	12.42	12.38	14.48
SHIMMER_4	9.31	11.43	15.30	17.71	9.02	11.24	13.46	16.21	9.31	11.43	15.30	17.71	8.04	10.01	15.30	18.15
SHIMMER_5	27.58	248.99	33.50	275.49	26.00	231.69	23.35	163.96	27.58	248.97	33.49	275.45	8.16	10.64	13.44	15.78
SHIMMER_6	11.15	14.87	19.98	24.94	10.82	14.70	18.21	23.95	11.15	14.87	19.98	24.94	10.13	13.94	20.50	25.95
SHIMMER_7	10.08	13.01	15.22	17.77	9.78	12.83	13.42	16.47	10.08	13.01	15.22	17.77	8.97	11.92	15.52	18.58
SHIMMER_8	9.75	12.70	16.89	20.29	9.42	12.53	15.11	19.17	9.75	12.70	16.89	20.29	8.77	11.82	17.39	21.33
SHIMMER_9	9.78	12.57	14.93	17.35	9.47	12.38	13.12	16.05	9.78	12.57	14.93	17.35	8.63	11.40	15.24	18.20
SHIMMER_11	98.92	1551.51	114.05	1715.58	92.44	1444.11	72.03	1015.63	98.91	1551.33	114.04	1715.38	8.86	11.83	15.96	19.26
SHIMMER_13	9.63	12.56	15.60	18.42	9.29	12.39	13.85	17.31	9.63	12.56	15.60	18.42	8.70	11.78	16.08	19.49
SHIMMER_14	46.13	640.70	57.59	710.45	43.19	593.09	39.84	418.94	46.12	640.63	57.58	710.36	10.12	13.95	19.29	24.16
SHIMMER_16	36.11	484.46	41.45	536.76	33.92	448.45	28.19	315.51	36.11	484.40	41.45	536.69	10.17	14.01	14.16	16.79
SHIMMER_17	10.17	13.14	12.64	14.25	9.89	12.99	10.79	12.71	10.17	13.14	12.64	14.25	8.96	11.85	12.87	15.06
SHIMMER_18	20.36	107.30	25.14	118.94	19.51	99.69	20.13	71.98	20.36	107.29	25.14	118.93	12.47	17.56	17.90	22.08
SHIMMER_19	3.58	3.94	13.61	15.61	3.02	3.71	11.80	14.33	3.58	3.94	13.61	15.61	2.33	3.12	14.07	16.69
SHIMMER_20	3.60	4.14	13.31	15.14	3.21	3.93	11.52	13.78	3.60	4.14	13.31	15.14	2.44	3.01	13.71	16.15
SHIMMER_21	10.24	13.31	15.09	17.61	9.95	13.16	13.29	16.34	10.24	13.31	15.09	17.61	9.16	12.26	15.46	18.52
SHIMMER_22	11.86	15.74	14.64	17.07	11.54	15.52	12.74	15.50	11.86	15.74	14.64	17.07	10.46	14.09	14.65	17.39
SHIMMER_23	19.27	216.94	27.68	241.86	18.02	201.14	19.75	144.45	19.27	216.92	27.68	241.83	4.68	5.55	12.99	15.22
SHIMMER_24	32.31	369.85	41.06	417.59	30.26	340.48	29.88	254.67	32.31	369.81	41.06	417.54	9.62	12.14	16.96	20.27
SHIMMER_25	8.72	11.11	12.14	13.59	8.39	10.91	10.36	12.15	8.72	11.11	12.14	13.59	7.58	9.95	12.59	14.70
SHIMMER_26	7.44	8.80	10.70	12.07	7.15	8.56	9.13	10.57	7.44	8.80	10.70	12.07	6.43	7.63	11.17	13.26
SHIMMER_27	5.56	6.55	10.69	11.64	5.19	6.33	8.86	10.06	5.56	6.55	10.69	11.64	4.48	5.50	11.08	12.74
SHIMMER_28	5.00	5.84	8.92	8.49	4.67	5.59	6.05	6.60	5.00	5.84	8.92	8.49	3.91	4.62	8.78	9.69
SHIMMER_29	4.88	5.75	9.14	10.04	4.58	5.52	7.41	8.35	4.88	5.75	9.14	10.04	3.85	4.54	9.63	11.29
SHIMMER_30	7.11	8.77	9.10	9.97	6.79	8.54	7.23	8.15	7.11	8.77	9.10	9.97	5.94	7.41	9.40	10.98
SHIMMER_31	7.36	8.99	9.22	10.07	7.06	8.80	7.46	8.38	7.36	8.99	9.22	10.07	6.31	7.93	9.62	11.28
SHIMMER_32	6.39	7.66	11.49	12.86	6.06	7.38	9.74	11.30	6.39	7.66	11.49	12.86	5.19	6.23	11.74	13.74
SHIMMER_33	5.52	6.48	9.78	7.84	5.19	6.24	6.73	5.92	5.52	6.48	9.78	7.84	4.47	5.35	9.66	9.10
SHIMMER_34	4.70	5.45	13.40	7.67	4.38	5.16	10.22	6.20	4.70	5.45	13.40	7.67	3.70	4.25	13.08	8.60
SHIMMER_35	6.68	7.97	8.44	9.05	6.35	7.73	6.49	7.35	6.68	7.97	8.44	9.05	5.73	7.04	8.76	10.42
SHIMMER_36	4.18	4.81	10.17	11.11	3.87	4.60	8.49	9.63	4.18	4.81	10.17	11.11	3.25	3.84	10.89	12.63
SHIMMER_37	3.55	4.14	10.24	7.58	3.17	3.86	7.13	5.70	3.55	4.14	10.24	7.58	2.48	3.05	10.13	8.78
SHIMMER_38	3.38	3.75	10.81	7.41	2.80	3.51	7.69	5.53	3.38	3.75	10.81	7.41	2.16	2.79	10.53	8.72
SHIMMER_39	8.19	9.97	12.91	14.61	7.88	9.75	11.25	13.26	8.19	9.97	12.91	14.61	7.15	8.88	13.35	15.71
SHIMMER_40	5.42	6.28	10.76	12.05	5.13	6.05	8.92	10.29	5.42	6.28	10.76	12.05	4.15	4.80	10.77	12.71
SHIMMER_41	5.14	6.00	15.66	18.35	4.85	5.79	13.82	17.00	5.14	6.00	15.66	18.35	3.96	4.66	15.89	19.07
SHIMMER_42	6.16	7.37	11.94	7.53	5.73	7.14	9.19	5.92	6.16	7.37	11.94	7.53	4.88	6.14	12.33	8.70
LUCA_1	3.63	4.05	8.71	9.18	3.14	3.80	6.53	7.17	3.63	4.05	8.71	9.18	2.39	3.26	8.73	10.02
LUCA_2	3.46	3.86	8.65	9.39	2.95	3.66	6.59	7.57	3.46	3.86	8.65	9.39	2.29	2.85	8.83	10.49
LUCA_3	3.60	3.98	8.70	9.35	3.07	3.79	6.61	7.50	3.60	3.98	8.70	9.35	2.27	2.85	8.80	10.47
LUCA_4	3.51	3.82	8.87	8.85	2.87	3.62	6.33	7.07	3.51	3.82	8.87	8.85	2.23	2.80	8.81	10.29
LUCA_5	3.55	4.12	8.75	9.86	3.15	3.89	6.71	7.82	3.55	4.12	8.75	9.86	2.17	2.73	8.49	10.45
LUCA_6	3.64	4.23	8.80	9.79	3.20	3.96	6.72	7.79	3.64	4.23	8.80	9.79	2.18	2.80	8.59	10.35
LUCA_7	3.60	4.16	8.69	9.66	3.20	3.93	6.63	7.69	3.60	4.16	8.69	9.66	2.24	2.85	8.57	10.37
LUCA_8	3.78	4.41	8.89	10.04	3.45	4.18	6.88	8.01	3.78	4.41	8.89	10.04	2.30	2.93	8.65	10.61
LUCA_9	3.72	3.97	8.71	9.33	3.10	3.77	6.32	7.28	3.72	3.97	8.71	9.33	2.46	2.84	8.68	10.07
LUCA_10	3.79	3.64	9.00	8.60	2.99	3.44	6.19	6.69	3.79	3.64	9.00	8.60	2.41	2.68	8.77	9.80
LUCA_11	3.80	4.05	10.18	7.71	3.19	3.85	7.24	5.75	3.80	4.05	10.18	7.71	2.54	3.01	10.19	8.88
LUCA_12	3.82	3.94	10.13	7.84	3.17	3.74	7.13	5.91	3.82	3.94	10.13	7.84	2.58	3.01	9.92	9.02
LUCA_13	3.69	4.25	9.76	7.99	3.30	4.04	6.97	5.99	3.69	4.25	9.76	7.99	2.34	2.89	10.05	8.97
LUCA_14	6.51	4.82	9.27	8.21	5.51	4.81	6.57	6.19	6.51	4.82	9.27	8.21	5.75	5.76	9.70	9.09
LUCA_15	4.01	3.90	9.79	7.92	3.27	3.69	6.96	5.92	4.01	3.90	9.79	7.92	2.91	3.18	10.23	8.84
LUCA_16	3.85	3.68	10.14	7.78	3.05	3.51	7.15	5.85	3.85	3.68	10.14	7.78	2.53	2.86	10.02	8.95
LUCA_17	3.88	3.80	9.75	7.93	3.13	3.61	6.76	5.98	3.88	3.80	9.75	7.93	2.45	2.82	9.63	9.12
LUCA_18	3.38	3.75	10.81	7.41	2.80	3.51	7.69	5.53	3.38	3.75	10.81	7.41	2.16	2.79	10.53	8.72
LUCA_19	4.70	5.45	13.40	7.67	4.38	5.16	10.22	6.20	4.70	5.45	13.40	7.67	3.70	4.25	13.08	8.60
LUCA_20	3.55	4.14	10.24	7.58	3.17	3.86	7.13	5.70	3.55	4.14	10.24	7.58	2.48	3.05	10.13	8.78

Table 24: Metrics Post K-Fold Cross-Correlation (MAE and STD)

Model	Linear				LMS				Ridge				SVM			
Signal	DBP		SBP		DBP		SBP		DBP		SBP		DBP		SBP	
Metric	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD
SHIMMER_1	5.54	0.04	7.89	0.06	6.43	0.04	8.69	0.03	5.54	0.04	7.89	0.06	4.65	0.02	6.52	0.06
SHIMMER_2	5.60	0.06	7.30	0.02	6.56	0.06	7.90	0.07	5.60	0.06	7.30	0.02	4.90	0.12	6.41	0.08
SHIMMER_3	5.44	0.03	5.54	0.02	6.27	0.02	5.45	0.03	5.44	0.03	5.54	0.02	4.56	0.05	4.89	0.02
SHIMMER_4	4.74	0.05	6.63	0.05	5.29	0.04	6.94	0.09	4.74	0.05	6.63	0.05	4.24	0.00	5.86	0.07
SHIMMER_5	4.89	0.02	5.93	0.03	5.48	0.06	5.92	0.03	4.89	0.02	5.93	0.03	4.44	0.04	5.70	0.02
SHIMMER_6	5.79	0.08	8.89	0.15	6.72	0.09	10.02	0.13	5.79	0.08	8.89	0.15	4.60	0.06	6.87	0.09
SHIMMER_7	5.23	0.02	6.69	0.03	5.93	0.03	7.00	0.04	5.23	0.02	6.69	0.03	4.27	0.05	5.60	0.02
SHIMMER_8	5.14	0.08	7.46	0.12	5.81	0.07	8.07	0.07	5.14	0.08	7.46	0.12	4.23	0.06	6.08	0.11
SHIMMER_9	5.11	0.01	6.57	0.02	5.74	0.01	6.85	0.05	5.11	0.01	6.57	0.02	4.23	0.05	5.55	0.03
SHIMMER_11	5.25	0.05	6.89	0.06	5.95	0.08	7.21	0.04	5.25	0.05	6.89	0.06	4.67	0.03	6.58	0.06
SHIMMER_13	5.10	0.05	6.95	0.07	5.76	0.04	7.35	0.06	5.10	0.05	6.95	0.07	4.17	0.06	5.68	0.09
SHIMMER_14	5.81	0.03	8.24	0.12	6.72	0.08	9.01	0.17	5.81	0.03	8.24	0.12	5.03	0.10	7.55	0.13
SHIMMER_16	5.78	0.02	8.24	0.12	6.76	0.06	9.01	0.17	5.78	0.02	8.24	0.12	5.05	0.01	6.01	0.02
SHIMMER_17	5.19	0.07	5.71	0.04	5.98	0.10	5.62	0.04	5.19	0.07	5.71	0.04	4.47	0.04	5.07	0.04
SHIMMER_18	6.89	0.08	7.67	0.11	8.20	0.07	8.21	0.13	6.89	0.08	7.67	0.11	5.56	0.05	6.94	0.06
SHIMMER_19	2.94	0.04	6.11	0.07	3.01	0.05	6.26	0.04	2.94	0.04	6.11	0.07	2.61	0.07	5.18	0.04
SHIMMER_20	2.99	0.04	6.00	0.05	3.04	0.05	6.06	0.03	2.99	0.04	6.00	0.05	2.68	0.07	5.27	0.04
SHIMMER_21	5.31	0.04	6.68	0.04	6.11	0.02	6.98	0.04	5.31	0.04	6.68	0.04	4.61	0.02	5.87	0.05
SHIMMER_22	5.95	0.04	6.32	0.04	6.90	0.05	6.46	0.05	5.95	0.04	6.32	0.04	4.76	0.02	5.48	0.02
SHIMMER_23	3.55	0.06	5.78	0.00	3.61	0.07	5.74	0.01	3.55	0.06	5.78	0.00	3.42	0.05	5.53	0.02
SHIMMER_24	5.42	0.03	7.25	0.09	6.00	0.02	7.62	0.06	5.42	0.03	7.25	0.09	4.59	0.06	6.66	0.10
SHIMMER_25	4.67	0.03	5.54	0.03	5.12	0.03	5.52	0.01	4.67	0.03	5.54	0.03	3.78	0.05	4.72	0.03
SHIMMER_26	4.15	0.01	5.19	0.02	4.38	0.02	5.08	0.03	4.15	0.01	5.19	0.02	3.50	0.07	4.43	0.03
SHIMMER_27	3.50	0.03	5.08	0.03	3.55	0.04	4.94	0.01	3.50	0.03	5.08	0.03	3.14	0.04	4.47	0.01
SHIMMER_28	3.31	0.05	4.29	0.03	3.35	0.04	4.25	0.07	3.31	0.05	4.29	0.03	2.88	0.08	3.83	0.03
SHIMMER_29	3.32	0.03	4.69	0.01	3.35	0.04	4.58	0.04	3.32	0.03	4.69	0.01	2.93	0.07	4.07	0.04
SHIMMER_30	3.99	0.06	4.60	0.01	4.19	0.04	4.50	0.04	3.99	0.06	4.60	0.01	3.38	0.06	4.03	0.00
SHIMMER_31	4.15	0.04	4.70	0.03	4.43	0.06	4.55	0.05	4.15	0.04	4.70	0.03	3.64	0.04	4.14	0.05
SHIMMER_32	3.72	0.05	5.28	0.04	3.86	0.03	5.20	0.05	3.72	0.05	5.28	0.04	3.16	0.08	4.47	0.05
SHIMMER_33	3.48	0.02	4.38	0.07	3.55	0.02	4.28	0.10	3.48	0.02	4.38	0.07	3.01	0.05	3.95	0.05
SHIMMER_34	3.25	0.04	5.39	0.02	3.28	0.05	5.63	0.07	3.25	0.04	5.39	0.02	2.87	0.07	4.64	0.02
SHIMMER_35	3.88	0.03	4.46	0.02	4.06	0.02	4.37	0.06	3.88	0.03	4.46	0.02	3.19	0.07	3.88	0.06
SHIMMER_36	3.18	0.04	5.03	0.03	3.22	0.05	4.92	0.03	3.18	0.04	5.03	0.03	2.83	0.05	4.37	0.03
SHIMMER_37	2.95	0.05	4.51	0.03	3.01	0.06	4.43	0.09	2.95	0.05	4.51	0.03	2.62	0.09	4.02	0.04
SHIMMER_38	2.90	0.05	4.55	0.06	2.99	0.07	4.47	0.11	2.90	0.05	4.55	0.06	2.57	0.08	4.00	0.06
SHIMMER_39	4.44	0.05	5.87	0.03	4.79	0.02	5.92	0.01	4.44	0.05	5.87	0.03	3.67	0.05	4.91	0.05
SHIMMER_40	3.41	0.06	5.05	0.00	3.35	0.06	4.86	0.05	3.41	0.06	5.05	0.00	3.13	0.06	4.55	0.01
SHIMMER_41	3.38	0.06	6.84	0.08	3.35	0.06	7.21	0.04	3.38	0.06	6.84	0.08	3.11	0.07	6.00	0.07
SHIMMER_42	3.58	0.02	5.38	0.01	3.52	0.03	5.74	0.08	3.58	0.02	5.38	0.01	3.36	0.02	4.90	0.02
LUCA_1	2.94	0.03	4.39	0.04	3.01	0.05	4.39	0.09	2.94	0.03	4.39	0.04	2.62	0.06	3.90	0.03
LUCA_2	2.95	0.04	4.47	0.04	3.01	0.05	4.39	0.08	2.95	0.04	4.47	0.04	2.66	0.07	4.02	0.04
LUCA_3	2.96	0.05	4.47	0.05	3.02	0.06	4.49	0.08	2.96	0.05	4.47	0.05	2.66	0.07	4.06	0.04
LUCA_4	2.93	0.05	4.41	0.04	3.01	0.07	4.36	0.07	2.93	0.05	4.41	0.04	2.65	0.07	3.96	0.04
LUCA_5	2.92	0.05	4.38	0.03	2.96	0.06	4.37	0.08	2.92	0.05	4.38	0.03	2.62	0.09	3.92	0.03
LUCA_6	2.92	0.03	4.39	0.03	2.95	0.06	4.28	0.08	2.92	0.03	4.39	0.03	2.62	0.07	3.93	0.02
LUCA_7	2.95	0.05	4.40	0.03	2.98	0.06	4.32	0.08	2.95	0.05	4.40	0.03	2.64	0.07	3.91	0.04
LUCA_8	2.97	0.03	4.42	0.03	2.98	0.05	4.40	0.08	2.97	0.03	4.42	0.03	2.66	0.08	3.97	0.03
LUCA_9	3.00	0.05	4.37	0.05	3.07	0.06	4.38	0.10	3.00	0.05	4.37	0.05	2.71	0.07	3.93	0.04
LUCA_10	2.98	0.06	4.33	0.03	3.06	0.07	4.29	0.09	2.98	0.06	4.33	0.03	2.66	0.07	3.91	0.04
LUCA_11	3.02	0.05	4.63	0.07	3.09	0.05	4.62	0.10	3.02	0.05	4.63	0.07	2.74	0.07	4.20	0.03
LUCA_12	3.04	0.05	4.58	0.03	3.13	0.07	4.57	0.10	3.04	0.05	4.58	0.03	2.74	0.08	4.13	0.04
LUCA_13	2.96	0.05	4.69	0.03	3.00	0.06	4.79	0.09	2.96	0.05	4.69	0.03	2.67	0.07	4.22	0.03
LUCA_14	4.11	0.02	4.60	0.02	4.67	0.06	4.68	0.07	4.11	0.02	4.60	0.02	3.65	0.07	4.18	0.02
LUCA_15	3.12	0.03	4.66	0.04	3.24	0.06	4.69	0.10	3.12	0.03	4.66	0.04	2.83	0.07	4.23	0.04
LUCA_16	3.03	0.05	4.59	0.03	3.13	0.07	4.62	0.09	3.03	0.05	4.59	0.03	2.72	0.07	4.12	0.03
LUCA_17	3.02	0.04	4.49	0.04	3.12	0.06	4.50	0.10	3.02	0.04	4.49	0.04	2.71	0.06	4.02	0.05
LUCA_18	2.90	0.05	4.55	0.06	2.99	0.07	4.47	0.11	2.90	0.05	4.55	0.06	2.57	0.08	4.00	0.06
LUCA_19	3.25	0.04	5.39	0.02	3.28	0.05	5.63	0.07	3.25	0.04	5.39	0.02	2.87	0.07	4.64	0.02
LUCA_20	2.95	0.05	4.51	0.03	3.01	0.06	4.43	0.09	2.95	0.05	4.51	0.03	2.62	0.09	4.02	0.04

Patients with their corresponding Database Pre and Post K-Fold Cross-Correlation

Table 25: Metrics Pre Cross-Correlation (MAE and STD) - Luca

Model	Linear				LMS				Ridge				SVM			
Signal	DBP		SBP		DBP		SBP		DBP		SBP		DBP		SBP	
Metric	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD
LUCA_1	1.94	2.30	5.12	5.31	2.16	2.47	5.08	5.74	1.94	2.30	5.12	5.31	2.05	2.85	4.61	3.93
LUCA_2	2.04	2.40	5.34	5.91	2.15	2.53	5.24	6.23	2.04	2.40	5.34	5.91	2.19	2.87	4.70	4.33
LUCA_3	2.10	2.46	5.46	5.97	2.25	2.63	5.45	6.32	2.10	2.46	5.46	5.97	2.20	2.88	4.70	4.52
LUCA_4	1.96	2.35	5.22	5.57	2.05	2.45	5.06	5.90	1.96	2.35	5.22	5.57	2.17	2.88	4.86	4.32
LUCA_5	1.98	2.34	5.06	5.74	2.24	2.55	5.10	6.12	1.98	2.34	5.06	5.74	2.02	2.71	4.50	4.52
LUCA_6	2.01	2.38	5.09	5.72	2.24	2.57	5.06	6.07	2.01	2.38	5.09	5.72	2.03	2.76	4.54	4.44
LUCA_7	2.12	2.45	5.10	5.71	2.33	2.65	5.07	6.07	2.12	2.45	5.10	5.71	2.11	2.80	4.52	4.36
LUCA_8	2.26	2.57	5.27	5.94	2.60	2.82	5.32	6.32	2.26	2.57	5.27	5.94	2.16	2.79	4.68	4.74
LUCA_9	2.19	2.51	4.95	5.29	2.25	2.64	4.69	5.66	2.19	2.51	4.95	5.29	2.40	3.03	4.64	4.12
LUCA_10	2.01	2.25	4.99	4.93	1.98	2.31	4.60	5.29	2.01	2.25	4.99	4.93	2.41	2.97	4.92	3.69
LUCA_11	2.27	2.64	6.51	3.95	2.32	2.74	5.80	4.29	2.27	2.64	6.51	3.95	2.51	3.16	6.38	2.74
LUCA_12	2.31	2.66	6.23	4.13	2.32	2.72	5.53	4.45	2.31	2.66	6.23	4.13	2.58	3.24	6.16	3.01
LUCA_13	2.23	2.54	6.25	4.11	2.50	2.79	5.41	4.40	2.23	2.54	6.25	4.11	2.23	2.87	6.06	2.99
LUCA_14	5.58	5.70	5.85	4.17	5.22	5.35	5.05	4.48	5.58	5.70	5.85	4.17	6.14	6.62	5.77	3.01
LUCA_15	2.56	2.90	6.37	3.95	2.49	2.87	5.54	4.24	2.56	2.90	6.37	3.95	2.88	3.54	6.30	2.81
LUCA_16	2.19	2.46	6.21	4.08	2.14	2.50	5.48	4.41	2.19	2.46	6.21	4.08	2.53	3.14	6.29	2.96
LUCA_17	2.19	2.50	5.76	4.20	2.19	2.56	5.03	4.55	2.19	2.50	5.76	4.20	2.47	3.10	5.83	2.86
LUCA_18	1.84	2.22	6.86	3.77	1.90	2.27	6.22	4.08	1.84	2.22	6.86	3.77	2.15	2.83	6.39	2.56
LUCA_19	3.81	3.83	9.73	5.22	3.96	3.97	8.93	5.30	3.81	3.83	9.73	5.22	3.79	3.77	9.51	4.99
LUCA_20	2.30	2.46	6.30	3.68	2.45	2.58	5.52	3.97	2.30	2.46	6.30	3.68	2.37	2.82	6.19	2.52

Table 26: Metrics Pre Cross-Correlation (MAE and STD) - Shimmer

Model	Linear				LMS				Ridge				SVM			
Signal	DBP		SBP		DBP		SBP		DBP		SBP		DBP		SBP	
Metric	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD
SHIMMER_1	9.38	7.90	12.76	10.59	9.65	7.78	13.31	10.11	9.38	7.90	12.76	10.59	6.76	6.38	11.71	8.25
SHIMMER_2	9.09	7.28	8.73	7.03	9.40	7.17	9.32	6.52	9.09	7.28	8.73	7.03	7.30	6.58	8.33	5.32
SHIMMER_3	8.45	6.87	2.80	2.20	8.77	6.77	2.30	2.50	8.45	6.87	2.80	2.20	6.25	5.91	4.34	4.13
SHIMMER_4	4.43	4.02	4.56	3.94	4.79	3.93	5.22	3.53	4.43	4.02	4.56	3.94	5.35	5.58	4.94	3.27
SHIMMER_5	11.48	54.98	19.19	161.24	10.77	46.01	24.28	204.44	11.48	54.98	19.19	161.24	5.44	5.47	2.56	3.18
SHIMMER_6	10.67	9.03	17.74	14.68	10.92	8.89	18.31	14.22	10.67	9.03	17.74	14.68	7.74	7.05	16.45	12.01
SHIMMER_7	7.73	6.56	6.41	5.78	8.01	6.44	7.01	5.36	7.73	6.56	6.41	5.78	5.79	5.79	5.67	4.23
SHIMMER_8	7.55	6.61	10.91	9.38	7.80	6.48	11.46	8.94	7.55	6.61	10.91	9.38	5.50	5.62	9.91	7.15
SHIMMER_9	7.15	6.13	5.82	5.39	7.42	6.01	6.42	5.00	7.15	6.13	5.82	5.39	5.23	5.36	5.03	3.91
SHIMMER_11	36.60	343.66	92.43	1001.34	31.68	282.52	116.23	1275.70	36.60	343.66	92.43	1001.34	5.67	5.59	6.99	5.03
SHIMMER_13	7.49	6.63	8.52	7.68	7.73	6.50	9.07	7.27	7.49	6.63	8.52	7.68	5.27	5.54	7.29	5.55
SHIMMER_14	22.29	145.15	47.80	407.06	20.43	119.19	57.51	520.31	22.29	145.15	47.80	407.06	7.78	7.04	14.12	10.33
SHIMMER_16	18.60	109.15	27.43	310.56	17.33	89.58	34.42	396.51	18.60	109.15	27.43	310.56	8.93	7.75	3.58	3.48
SHIMMER_17	7.44	6.33	1.98	2.53	7.73	6.22	1.99	2.68	7.44	6.33	1.98	2.53	6.08	5.96	2.90	3.52
SHIMMER_18	17.71	27.20	18.81	68.18	17.53	23.49	21.24	86.59	17.71	27.20	18.81	68.18	12.99	10.64	11.23	8.14
SHIMMER_19	8.51	4.93	4.29	4.73	7.67	5.07	4.76	4.43	8.51	4.93	4.29	4.73	10.59	9.06	3.29	3.50
SHIMMER_20	8.52	4.50	2.63	3.39	7.66	4.62	3.19	3.14	8.52	4.50	2.63	3.39	10.76	8.77	2.65	3.22
SHIMMER_21	8.11	6.87	6.36	5.81	8.39	6.75	6.91	5.37	8.11	6.87	6.36	5.81	5.50	5.83	5.24	4.16
SHIMMER_22	11.09	8.99	4.38	4.48	11.37	8.86	4.99	4.24	11.09	8.99	4.38	4.48	8.99	7.75	4.03	3.62
SHIMMER_23	8.80	47.16	15.06	143.97	7.29	40.15	18.32	181.09	8.80	47.16	15.06	143.97	4.55	5.10	2.55	3.33
SHIMMER_24	15.40	79.10	30.33	251.77	14.90	68.82	36.68	314.29	15.40	79.10	30.33	251.77	6.17	5.61	8.19	4.72
SHIMMER_25	5.30	4.85	2.15	2.54	5.55	4.72	1.97	2.64	5.30	4.85	2.15	2.54	3.82	4.26	3.47	3.70
SHIMMER_26	1.74	2.19	5.16	2.94	1.84	2.14	4.57	3.48	1.74	2.19	5.16	2.94	3.76	4.46	7.01	5.56
SHIMMER_27	3.80	1.69	5.33	2.90	2.96	1.75	4.62	3.35	3.80	1.69	5.33	2.90	5.84	5.88	6.67	5.44
SHIMMER_28	4.85	2.00	12.12	7.27	3.99	2.08	11.38	7.76	4.85	2.00	12.12	7.27	6.72	5.91	13.54	10.03
SHIMMER_29	5.06	2.20	8.27	4.67	4.19	2.27	7.54	5.19	5.06	2.20	8.27	4.67	7.07	6.24	9.68	7.40
SHIMMER_30	1.78	2.32	9.05	5.22	1.91	2.21	8.36	5.78	1.78	2.32	9.05	5.22	3.52	4.42	10.43	7.94
SHIMMER_31	2.01	2.49	8.52	4.75	2.27	2.39	7.87	5.27	2.01	2.49	8.52	4.75	3.90	4.67	10.45	7.85
SHIMMER_32	2.46	1.46	3.86	2.39	1.65	1.41	3.20	2.79	2.46	1.46	3.86	2.39	4.47	4.65	5.28	4.64
SHIMMER_33	3.58	1.55	13.92	8.59	2.73	1.57	13.16	9.06	3.58	1.55	13.92	8.59	5.87	5.45	15.47	11.48
SHIMMER_34	6.15	2.19	22.87	13.58	5.23	2.25	22.21	14.02	6.15	2.19	22.87	13.58	8.87	6.40	24.93	16.36
SHIMMER_35	1.71	2.06	10.09	5.80	1.58	1.98	9.31	6.23	1.71	2.06	10.09	5.80	3.90	4.42	11.99	8.88
SHIMMER_36	6.43	3.08	5.68	3.05	5.57	3.18	4.94	3.49	6.43	3.08	5.68	3.05	9.20	7.62	7.65	6.08
SHIMMER_37	8.72	3.84	16.24	9.62	7.81	3.92	15.57	10.08	8.72	3.84	16.24	9.62	11.06	7.74	18.04	12.37
SHIMMER_38	8.49	4.46	15.53	9.79	7.63	4.57	14.69	10.19	8.49	4.46	15.53	9.79	10.54	8.02	17.02	12.61
SHIMMER_39	3.47	3.44	1.52	2.42	3.81	3.35	2.17	2.42	3.47	3.44	1.52	2.42	2.99	3.89	2.43	3.19
SHIMMER_40	5.88	2.26	6.53	3.59	4.99	2.33	6.13	4.22	5.88	2.26	6.53	3.59	5.89	5.98	7.17	5.65
SHIMMER_41	5.47	2.32	6.86	6.04	4.61	2.41	7.42	5.59	5.47	2.32	6.86	6.04	6.63	6.23	6.45	4.59
SHIMMER_42	3.55	1.61	19.80	14.05	2.73	1.67	19.28	14.81	3.55	1.61	19.80	14.05	3.47	4.30	19.72	15.93

Table 27: Metrics Post K-fold Cross-Correlation (MAE and STD) - Luca

Model	Linear				LMS				Ridge				SVM			
Signal	DBP		SBP		DBP		SBP		DBP		SBP		DBP		SBP	
Metric	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD
LUCA_1	2.45	0.00	3.15	0.01	2.59	0.00	3.39	0.01	2.45	0.00	3.15	0.01	2.22	0.02	2.84	0.03
LUCA_2	2.47	0.01	3.28	0.02	2.61	0.00	3.43	0.01	2.47	0.01	3.28	0.02	2.27	0.01	2.97	0.03
LUCA_3	2.47	0.00	3.33	0.02	2.61	0.01	3.50	0.02	2.47	0.00	3.33	0.02	2.27	0.01	3.00	0.03
LUCA_4	2.45	0.01	3.24	0.02	2.63	0.01	3.40	0.02	2.45	0.01	3.24	0.02	2.25	0.02	2.92	0.03
LUCA_5	2.43	0.02	3.17	0.01	2.54	0.03	3.37	0.00	2.43	0.02	3.17	0.01	2.23	0.02	2.89	0.02
LUCA_6	2.43	0.01	3.17	0.01	2.57	0.01	3.35	0.01	2.43	0.01	3.17	0.01	2.23	0.01	2.89	0.00
LUCA_7	2.46	0.00	3.18	0.01	2.58	0.01	3.37	0.02	2.46	0.00	3.18	0.01	2.25	0.01	2.88	0.01
LUCA_8	2.48	0.03	3.22	0.00	2.56	0.03	3.41	0.01	2.48	0.03	3.22	0.00	2.27	0.02	2.94	0.03
LUCA_9	2.53	0.01	3.16	0.01	2.67	0.01	3.38	0.02	2.53	0.01	3.16	0.01	2.33	0.00	2.92	0.00
LUCA_10	2.49	0.01	3.12	0.01	2.68	0.01	3.34	0.03	2.49	0.01	3.12	0.01	2.27	0.01	2.90	0.00
LUCA_11	2.55	0.02	3.47	0.00	2.71	0.02	3.73	0.04	2.55	0.02	3.47	0.00	2.36	0.03	3.22	0.03
LUCA_12	2.57	0.00	3.42	0.01	2.76	0.02	3.67	0.04	2.57	0.00	3.42	0.01	2.37	0.01	3.16	0.02
LUCA_13	2.49	0.01	3.56	0.01	2.58	0.01	3.87	0.04	2.49	0.01	3.56	0.01	2.28	0.01	3.24	0.01
LUCA_14	3.91	0.02	3.47	0.01	4.49	0.02	3.77	0.03	3.91	0.02	3.47	0.01	3.42	0.04	3.20	0.01
LUCA_15	2.69	0.02	3.57	0.01	2.91	0.01	3.85	0.04	2.69	0.02	3.57	0.01	2.47	0.02	3.25	0.01
LUCA_16	2.55	0.00	3.38	0.01	2.74	0.00	3.66	0.04	2.55	0.00	3.38	0.01	2.34	0.01	3.14	0.01
LUCA_17	2.54	0.00	3.25	0.01	2.73	0.00	3.52	0.04	2.54	0.00	3.25	0.01	2.33	0.01	3.01	0.01
LUCA_18	2.40	0.02	3.38	0.00	2.60	0.02	3.59	0.04	2.40	0.02	3.38	0.00	2.16	0.02	2.97	0.02
LUCA_19	2.80	0.02	4.45	0.01	2.89	0.03	4.89	0.04	2.80	0.02	4.45	0.01	2.49	0.01	3.69	0.05
LUCA_20	2.46	0.01	3.36	0.02	2.61	0.02	3.56	0.03	2.46	0.01	3.36	0.02	2.20	0.01	3.03	0.01

Table 28: Metrics Post Cross-Correlation (MAE and STD) - Shimmer

Model	Linear				LMS				Ridge				SVM			
Signal	DBP		SBP		DBP		SBP		DBP		SBP		DBP		SBP	
Metric	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD	MAE	STD
SHIMMER_1	7.70	0.20	8.66	0.16	7.50	0.25	8.44	0.20	7.70	0.20	8.66	0.16	5.86	0.17	6.93	0.19
SHIMMER_2	7.35	0.21	7.46	0.14	7.12	0.24	7.21	0.17	7.35	0.21	7.46	0.14	5.85	0.22	6.30	0.21
SHIMMER_3	7.33	0.15	6.13	0.21	7.13	0.20	5.92	0.22	7.33	0.15	6.13	0.21	5.76	0.22	5.71	0.24
SHIMMER_4	6.35	0.14	6.60	0.20	6.19	0.17	6.41	0.20	6.35	0.14	6.60	0.20	5.63	0.16	5.89	0.18
SHIMMER_5	7.22	0.17	7.17	0.23	7.20	0.18	7.21	0.23	7.22	0.17	7.17	0.23	6.82	0.20	6.75	0.25
SHIMMER_6	8.00	0.22	9.88	0.19	7.80	0.27	9.71	0.24	8.00	0.22	9.88	0.19	5.89	0.23	7.46	0.18
SHIMMER_7	7.20	0.12	7.05	0.22	7.01	0.18	6.84	0.24	7.20	0.12	7.05	0.22	5.70	0.18	5.98	0.24
SHIMMER_8	7.28	0.23	8.15	0.14	7.09	0.29	7.93	0.17	7.28	0.23	8.15	0.14	5.65	0.26	6.55	0.21
SHIMMER_9	7.06	0.19	6.91	0.24	6.88	0.24	6.70	0.25	7.06	0.19	6.91	0.24	5.64	0.22	5.90	0.26
SHIMMER_11	7.96	0.13	8.96	0.18	7.97	0.14	8.98	0.19	7.96	0.13	8.96	0.18	7.57	0.19	8.88	0.18
SHIMMER_13	7.32	0.24	7.64	0.12	7.14	0.27	7.42	0.14	7.32	0.24	7.64	0.12	5.60	0.22	6.19	0.16
SHIMMER_14	9.03	0.13	11.43	0.13	9.08	0.14	11.53	0.11	9.03	0.13	11.43	0.13	8.39	0.19	10.95	0.15
SHIMMER_16	9.03	0.15	7.79	0.13	9.11	0.14	7.82	0.14	9.03	0.15	7.79	0.13	8.42	0.27	7.60	0.12
SHIMMER_17	7.17	0.17	6.12	0.20	6.98	0.20	5.97	0.23	7.17	0.17	6.12	0.20	5.82	0.19	5.63	0.24
SHIMMER_18	10.80	0.09	10.40	0.09	10.97	0.10	10.46	0.09	10.80	0.09	10.40	0.09	9.53	0.19	9.18	0.18
SHIMMER_19	7.09	0.15	6.67	0.18	6.84	0.20	6.50	0.22	7.09	0.15	6.67	0.18	6.08	0.22	5.80	0.20
SHIMMER_20	7.00	0.12	6.34	0.25	6.76	0.18	6.20	0.27	7.00	0.12	6.34	0.25	6.02	0.19	5.69	0.24
SHIMMER_21	7.37	0.22	7.11	0.24	7.17	0.27	6.90	0.26	7.37	0.22	7.11	0.24	5.74	0.27	5.96	0.28
SHIMMER_22	7.92	0.15	6.79	0.22	7.76	0.18	6.64	0.24	7.92	0.15	6.79	0.22	6.10	0.14	5.83	0.23
SHIMMER_23	6.56	0.22	6.70	0.24	6.38	0.22	6.75	0.24	6.56	0.22	6.70	0.24	5.66	0.23	6.22	0.27
SHIMMER_24	7.88	0.16	9.18	0.10	7.82	0.17	9.17	0.09	7.88	0.16	9.18	0.10	7.37	0.22	8.70	0.15
SHIMMER_25	6.70	0.16	6.16	0.21	6.54	0.22	5.97	0.24	6.70	0.16	6.16	0.21	5.51	0.20	5.71	0.23
SHIMMER_26	5.85	0.10	6.52	0.14	5.67	0.15	6.29	0.17	5.85	0.10	6.52	0.14	5.29	0.13	5.96	0.16
SHIMMER_27	6.26	0.19	6.70	0.13	5.97	0.22	6.49	0.18	6.26	0.19	6.70	0.13	5.47	0.23	6.17	0.19
SHIMMER_28	6.27	0.16	8.16	0.20	5.97	0.19	7.96	0.23	6.27	0.16	8.16	0.20	5.48	0.22	6.80	0.23
SHIMMER_29	6.33	0.16	7.36	0.18	6.04	0.19	7.16	0.21	6.33	0.16	7.36	0.18	5.57	0.22	6.54	0.19
SHIMMER_30	6.01	0.17	7.49	0.22	5.82	0.22	7.30	0.25	6.01	0.17	7.49	0.22	5.38	0.21	6.57	0.25
SHIMMER_31	6.05	0.15	7.44	0.18	5.89	0.21	7.23	0.21	6.05	0.15	7.44	0.18	5.36	0.18	6.42	0.23
SHIMMER_32	6.01	0.16	6.28	0.20	5.77	0.19	6.05	0.22	6.01	0.16	6.28	0.20	5.32	0.18	5.82	0.23
SHIMMER_33	6.17	0.20	8.67	0.25	5.89	0.25	8.49	0.30	6.17	0.20	8.67	0.25	5.39	0.27	7.03	0.22
SHIMMER_34	6.30	0.16	10.80	0.17	6.00	0.19	10.62	0.23	6.30	0.16	10.80	0.17	5.47	0.21	8.23	0.24
SHIMMER_35	5.97	0.16	7.68	0.17	5.76	0.21	7.50	0.21	5.97	0.16	7.68	0.17	5.30	0.19	6.46	0.21
SHIMMER_36	6.57	0.20	6.77	0.19	6.30	0.23	6.56	0.22	6.57	0.20	6.77	0.19	5.72	0.25	6.13	0.22
SHIMMER_37	6.74	0.09	9.04	0.15	6.46	0.14	8.81	0.19	6.74	0.09	9.04	0.15	5.74	0.16	7.14	0.17
SHIMMER_38	6.87	0.14	8.96	0.20	6.60	0.18	8.78	0.25	6.87	0.14	8.96	0.20	5.81	0.22	6.90	0.19
SHIMMER_39	6.21	0.21	6.10	0.21	6.06	0.24	5.96	0.22	6.21	0.21	6.10	0.21	5.34	0.18	5.54	0.22
SHIMMER_40	6.22	0.12	6.83	0.20	5.93	0.15	6.59	0.22	6.22	0.12	6.83	0.20	5.48	0.18	6.23	0.22
SHIMMER_41	6.32	0.17	7.15	0.26	6.02	0.21	6.93	0.29	6.32	0.17	7.15	0.26	5.55	0.24	6.19	0.26
SHIMMER_42	6.24	0.16	10.62	0.12	5.96	0.21	10.46	0.22	6.24	0.16	10.62	0.12	5.48	0.20	8.85	0.39

Appendix B: Python Code

```python
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig
import scipy.io
from scipy.signal import savgol_filter
import os
import pandas as pd
import csv
import math
from scipy.interpolate import interp1d
from scipy.interpolate import CubicSpline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.svm import SVR
from sklearn.linear_model import Ridge
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import KFold



def hl_envelopes_idx(s, dmin=1, dmax=1, split=False):
    """
    Calculates the upper and lower envelope of the signal based on local maxima and minima.
    Returns:
        np.array: Signal filtered with the envelope removed.
    """
    # Find local maxima
    lmax = (np.diff(np.sign(np.diff(s))) < 0).nonzero()[0] + 1

    if split:
        # Calculate the mean of the signal
        s_mid = np.mean(s)
        # Consider only maxima above the mean
        lmax = lmax[s[lmax] > s_mid]

    # Filter local maxima based on dmax
    lmax = lmax[[i + np.argmax(s[lmax[i:i + dmax]]) for i in range(0, len(lmax), dmax)]]

    # Remove the envelope based on local maxima
    s_filt = np.zeros(len(s))
    n = 0
```

```
    for i in range(len(s)):
        if i == lmax[n]:
            s_filt[i] = s[i] - s[lmax[n]]
            if n < len(lmax) - 1:
                n += 1
        else:
            s_filt[i] = s[i] - s[lmax[n]]

    return s_filt

def filter_ecg_signal(ecg_head, fs_ecg, window_size):
    # mean-remove
    ecg_head = ecg_head - np.mean(ecg_head)
    fNy = fs_ecg / 2

    # pass-band 0.5  40   Hz
    b_band, a_band = sig.butter(4, [0.5 / fNy, 40 / fNy], btype='band')
    ecg_filtered = sig.filtfilt(b_band, a_band, ecg_head)

    # Notch 50 Hz (if fs_ecg > 100 Hz)
    if fs_ecg > 100:
        b_notch, a_notch = sig.iirnotch(50 / fNy, Q=30)
        ecg_filtered = sig.filtfilt(b_notch, a_notch, ecg_filtered)

    ecg_filtered = hl_envelopes_idx(ecg_filtered, dmin=1, dmax=1, split=
    False)
    return ecg_filtered

def filter_ppg_signal(ppg_head, fs, window_size):
    window_size_samples = int(window_size * fs)
    ppg_head = ppg_head - np.mean(ppg_head)
    fNy = fs / 2

    # --- Step 1: High-pass filter to remove baseline wander ---
    cutoff_high = 0.5  # Cutoff frequency for high-pass filter (Hz)
    b_high, a_high = sig.butter(4, cutoff_high / fNy, btype='high')
    ppg_high_passed = sig.filtfilt(b_high, a_high, ppg_head)

    # --- Step 2: Low-pass filter to remove high-frequency noise ---
    cutoff_low = 8  # Cutoff frequency for low-pass filter (Hz)
    b_low, a_low = sig.butter(4, cutoff_low / fNy, btype='low')
    ppg_low_passed = sig.filtfilt(b_low, a_low, ppg_high_passed)


    return ppg_low_passed

```

```python
def synchronize_signals(ecg_filtered, ppg_filtered, t_ecg, t_ppg):
    """
    Synchronizes ECG and PPG signals based on their time arrays.
    """
    # Align the start of the signals
    if t_ecg[0] < t_ppg[0]:
        for i in range(len(t_ecg)):
            if t_ecg[i] > t_ppg[0]:
                ind = np.arange(0, i - 1)
                ecg_filtered = np.delete(ecg_filtered, ind)
                t_ecg = np.delete(t_ecg, ind)
                break
    else:
        for i in range(len(t_ppg)):
            if t_ppg[i] > t_ecg[0]:
                ind = np.arange(0, i - 1)
                ppg_filtered = np.delete(ppg_filtered, ind)
                t_ppg = np.delete(t_ppg, ind)
                break


    # Bring signals to the same length
    if len(t_ecg) > len(t_ppg):
        t = t_ppg
        ind = np.arange(0, len(t_ppg))
        ecg_filtered = ecg_filtered[ind]
        t_ecg = t_ecg[ind]
    else:
        t = t_ecg
        ind = np.arange(0, len(t_ecg))
        ppg_filtered = ppg_filtered[ind]
        t_ppg = t_ppg[ind]

    return ecg_filtered, ppg_filtered, t_ecg, t_ppg, t

def peaks_detection(s_filt, ts, time, window_sec=0.2):
    """
    Detects local maxima every window_sec seconds.
    Args:
        s_filt (np.array): Filtered signal.
        ts (np.array): Time array corresponding to the signal.
        time (np.array): Full time array for the signal.
        window_sec (float): Window size in seconds for local maxima search.
    Returns:
        np.array: Detected peak values.
        np.array: Corresponding timestamps of the peaks.
```

```
    """
    peaks = []
    peak_times = []
    start_idx = 0
    n = len(s_filt)
    while start_idx < n:
        # find the end of the current window
        end_time = ts[start_idx] + window_sec
        end_idx = start_idx
        while end_idx < n and ts[end_idx] < end_time:
            end_idx += 1
        # find the local maximum in the current window
        if end_idx > start_idx:
            local_max_idx = np.argmax(s_filt[start_idx:end_idx]) + start_idx
            peaks.append(s_filt[local_max_idx])
            peak_times.append(ts[local_max_idx])
        start_idx = end_idx
    return np.array(peaks), np.array(peak_times)

def clean_peaks(peaks, times, n_std=2, min_distance=0.8):

    stats_window = 1  # seconds
    peaks = np.array(peaks)
    times = np.array(times)
    mask = np.ones(len(peaks), dtype=bool)

    # Static filter: the peak must be within n_std of the local mean
    for i in range(len(peaks)):
        t0 = times[i] - stats_window/2
        t1 = times[i] + stats_window/2
        idx = np.where((times >= t0) & (times <= t1))[0]
        if len(idx) < 2:
            continue
        local_mean = np.mean(peaks[idx])
        local_std = np.std(peaks[idx])
        if not (local_mean - n_std*local_std < peaks[i] < local_mean + n_std
    *local_std):
            mask[i] = False


    peaks = peaks[mask]
    times = times[mask]

    # second filter: minimum distance between peaks
    if len(peaks) > 1: # Check if there are enough peaks to filter
        keep = []
```

```
        i = 0
        while i < len(peaks):
            #if the peaks are too close, keep the one with the highest value and remove the other
            close_idx = np.where((times > times[i]) & (times - times[i] < min_distance))[0]
            if close_idx.size > 0:
                group_idx = np.concatenate(([i], close_idx))
                max_idx = group_idx[np.argmax(peaks[group_idx])]
                keep.append(max_idx)
                i = close_idx[-1] + 1
            else:
                keep.append(i)
                i += 1
        peaks = peaks[keep]
        times = times[keep]

    return peaks, times

def feat_reduction(feat, t_fitted):

    row = np.zeros(len(t_fitted))  # Initialize the reduced feature array
    T = 0  # Initialize the time window tracker

    # Loop through the time array
    for i in range(len(t_fitted) - 1):
        if i >= T:
            for j in range(i + 1, len(t_fitted)):
                # Check if the time difference is at least 10 seconds
                if t_fitted[j] - t_fitted[i] >= 10:
                    ind1 = np.arange(i, j)  # Indices within the 10-second window
                    vect_feat = feat[ind1]  # Extract feature values within the window
                    val_feat = np.mean(vect_feat)  # Calculate the mean of the feature values
                    row[ind1] = val_feat  # Assign the mean value to the corresponding indices
                    T = j  # Update the time window tracker
                    break

    # Handle the last window if it's smaller than 10 seconds
    for i in range(len(row)):
        if row[i] == 0:
            ind1 = np.arange(i, len(row))  # Indices for the remaining values
```

```
            val_feat = np.mean(feat[ind1])  # Calculate the mean of the remaining values
            row[ind1] = val_feat  # Assign the mean value to the remaining indices
            break

    return row

def get_at_ref_times(signal, t_uniform, ref_times):
    vals = []
    for t in ref_times:
        idx = np.argmin(np.abs(t_uniform - t))
        vals.append(signal[idx])
    return np.array(vals)

def extract_features_reg(ecg, ppg, t_ecg, t_ppg, ref_file,fs_ecg, fs_ppg):
    # Preprocess
    ecg = ecg[round(20 * fs_ecg):-round(30 * fs_ecg)]
    t_ecg = t_ecg[round(20 * fs_ecg):-round(30 * fs_ecg)]
    ppg = ppg[round(20 * fs_ppg):-round(30 * fs_ppg)]
    t_ppg = t_ppg[round(20 * fs_ppg):-round(30 * fs_ppg)]

    ecg_filtered = filter_ecg_signal(ecg, fs_ecg,window_size=0.5)
    ppg_filtered = filter_ppg_signal(ppg, fs_ppg, window_size=0.5)

    t_ecg = np.linspace(0, len(ecg_filtered)/fs_ecg, len(ecg_filtered))
    t_ppg = np.linspace(0, len(ppg_filtered)/fs_ppg, len(ppg_filtered))
    ecg_synced, ppg_synced, _, _, t_synced = synchronize_signals(
            ecg_filtered, ppg_filtered, t_ecg, t_ppg
        )

    r_peaks, r_times = peaks_detection(ecg_synced, t_synced, t_synced, window_sec=1)
    s_peaks, s_times = peaks_detection(ppg_synced, t_synced, t_synced, window_sec=0.5)
    r_peaks, r_times = clean_peaks(r_peaks, r_times, n_std=2)
    s_peaks, s_times = clean_peaks(s_peaks, s_times, n_std=2)


    ptt, hr, timetable = [], [], []
    T = 0
    for i in range(len(r_peaks)):
        if i >= T:
            for j in range(len(s_peaks)):
                if s_times[j] > r_times[i]:
                    ptt.append(s_times[j] - r_times[i])
```

```
                    for k in range(i + 1, len(r_peaks)):
                        if r_peaks[k] > 0:
                            hr.append(60 / (r_times[k] - r_times[i]))
                            timetable.append(r_times[i])
                            T = k
                            break
                    break

    ptt, hr, timetable = np.array(ptt), np.array(hr), np.array(timetable)
    # Zero elements deletion and arrays cut at the same length
    ind = np.array(np.where(ptt == 0))
    ptt = np.delete(ptt, ind)
    ind = np.array(np.where(hr == 0))
    hr = np.delete(hr, ind)
    ind = np.array(np.where(timetable == 0))
    timetable = np.delete(timetable, ind)

    if len(ptt) > len(hr):
        ind = np.arange(0, len(hr))
        ptt = ptt[ind]
        timetable = timetable[ind]
    else:
        ind = np.arange(0, len(ptt))
        hr = hr[ind]
        timetable = timetable[ind]

    #clean the data
    mean_PTT = np.mean(ptt) #cleaning the data
    dev_PTT = np.std(ptt)
    mean_HR = np.mean(hr)
    dev_HR = np.std(hr)

    for i in range(len(timetable)):
            if (
                ptt[i] > mean_PTT + dev_PTT
                or ptt[i] < mean_PTT - dev_PTT
                or hr[i] > mean_HR + dev_HR
                or hr[i] < mean_HR - dev_HR
            ):
                ptt[i] = 0
                hr[i] = 0
                timetable[i] = 0

    ind = np.array(np.where(ptt == 0))
    ptt = np.delete(ptt, ind)
    ind = np.array(np.where(hr == 0))
```

```
    hr = np.delete(hr, ind)
    ind = np.array(np.where(timetable == 0))
    timetable = np.delete(timetable, ind)

    ref_df = pd.read_csv(ref_file, delimiter=';', header=None)
    ref_time = (ref_df.iloc[:, 0].values - ref_df.iloc[0, 0]) / 1000
    sbp, dbp = ref_df.iloc[:, 1].values, ref_df.iloc[:, 2].values
    timetable_min = timetable / 60

    SBP_fit = np.interp(timetable_min, ref_time/60, sbp)
    DBP_fit = np.interp(timetable_min, ref_time/60, dbp)

    interp_time = timetable

    SBP_fit = np.interp(interp_time, timetable, SBP_fit)
    DBP_fit = np.interp(interp_time, timetable, DBP_fit)
    HR_fit = np.interp(interp_time, timetable, hr)
    PTT_fit = np.interp(interp_time, timetable, ptt)

    sbp_reduced = feat_reduction(SBP_fit, interp_time)
    dbp_reduced = feat_reduction(DBP_fit, interp_time)
    hr_reduced = feat_reduction(HR_fit, interp_time)
    ptt_reduced = feat_reduction(PTT_fit, interp_time)

    sbp_resampled = np.interp(timetable, interp_time, sbp_reduced)
    dbp_resampled = np.interp(timetable, interp_time, dbp_reduced)
    hr_resampled = np.interp(timetable, interp_time, hr_reduced)
    ptt_resampled = np.interp(timetable, interp_time, ptt_reduced)

    min_len = min(len(ptt_resampled), len(hr_resampled), len(sbp_resampled),
     len(dbp_resampled))
    return ptt_resampled[:min_len], hr_resampled[:min_len], sbp_resampled[:
    min_len], dbp_resampled[:min_len]

def lms(X, y, mu=0.001, epochs=1):
    """
    LMS algorithm for linear regression.
    X: input features (n_samples, n_features)
    y: target (n_samples,)
    mu: learning rate
    epochs: number of passes over the data
    Returns: weights, bias, prediction history
    """
    n_samples, n_features = X.shape
    w = np.zeros(n_features)
    b = 0
```

```
    y_pred_hist = []

    for epoch in range(epochs):
        for i in range(n_samples):
            y_pred = np.dot(w, X[i]) + b
            error = y[i] - y_pred
            w += 2 * mu * error * X[i]
            b += 2 * mu * error
            y_pred_hist.append(y_pred)
    return w, b, np.array(y_pred_hist)

class Patient:
    def __init__(self, pid,dataset, fs_ecg=504.12, fs_ppg=504.12):
        self.dataset = dataset
        self.pid = pid
        self.fs_ecg = fs_ecg
        self.fs_ppg = fs_ppg
        self.ecg = None
        self.ppg = None
        self.t_ecg = None
        self.t_ppg = None
        self.ecg_filtered = None
        self.ppg_filtered = None
        self.ecg_synced = None
        self.ppg_synced = None
        self.t_synced = None
        self.r_peaks = None
        self.r_times = None
        self.s_peaks = None
        self.s_times = None
        self.ptt = None
        self.hr = None
        self.timetable = None
        self.sbp = None
        self.dbp = None


    def load_signals(self):

        if self.pid == "LUCA":
            ecg_mat = scipy.io.loadmat(f"{self.pid}_1_ECG.mat")
            self.ecg = ecg_mat['ECG_signal'].flatten()
            self.t_ecg = ecg_mat['ECG_ts'].flatten() / 1000

            ppg_mat = scipy.io.loadmat(f"{self.pid}_1_PPG.mat")
            self.ppg = ppg_mat['PPG_signal'].flatten()
```

```
            self.t_ppg = ppg_mat['PPG_ts'].flatten() / 1000
        else:
            ecg_mat = scipy.io.loadmat(f"SHIMMER_{self.pid}_ECG.mat")
            self.ecg = ecg_mat['signal'].flatten()
            self.t_ecg = ecg_mat['ts'].flatten() / 1000

            ppg_mat = scipy.io.loadmat(f"SHIMMER_{self.pid}_PPG.mat")
            self.ppg = ppg_mat['signal'].flatten()
            self.t_ppg = ppg_mat['ts'].flatten() / 1000

        self.ecg_head = self.ecg.copy()
        self.ts_ecg = self.t_ecg.copy()
        self.ppg_head = self.ppg.copy()
        self.ts_ppg = self.t_ppg.copy()
        # samples frequencies
        fs_ecg = self.fs_ecg
        fs_ppg = self.fs_ppg


        # Remove the first 20 seconds (noise)
        cut_int_ecg = round(20 * fs_ecg)
        cut_int_ppg = round(20 * fs_ppg)
        self.ecg_head = self.ecg_head[cut_int_ecg:]
        self.ts_ecg = self.ts_ecg[cut_int_ecg:]
        self.ppg_head = self.ppg_head[cut_int_ppg:]
        self.ts_ppg = self.ts_ppg[cut_int_ppg:]

        # Remove last 30 seconds (noise)
        cut_end_ecg = round(30 * fs_ecg)
        cut_end_ppg = round(30 * fs_ppg)
        if cut_end_ecg < len(self.ecg_head):
            self.ecg_head = self.ecg_head[:-cut_end_ecg]
            self.ts_ecg = self.ts_ecg[:-cut_end_ecg]
        if cut_end_ppg < len(self.ppg_head):
            self.ppg_head = self.ppg_head[:-cut_end_ppg]
            self.ts_ppg = self.ts_ppg[:-cut_end_ppg]

        self.t_ecg = self.ts_ecg
        self.t_ppg = self.ts_ppg

    def preprocess(self):

        self.ecg_filtered = filter_ecg_signal(self.ecg_head, self.fs_ecg,
    window_size=0.2)
        self.ppg_filtered = filter_ppg_signal(self.ppg_head, self.fs_ppg,
    window_size=0.5)
```

```
440
441         ts_ecg = np.arange(len(self.ecg_head)) / self.fs_ecg
442         ts_ppg = np.arange(len(self.ppg_head)) / self.fs_ppg
443         self.t_ecg = np.linspace(0, len(self.ecg_filtered) / self.fs_ecg,
    len(self.ecg_filtered))
444         self.t_ppg = np.linspace(0, len(self.ppg_filtered) / self.fs_ppg,
    len(self.ppg_filtered))
445
446         plt.figure()
447         plt.plot(ts_ecg, self.ecg_head, 'b-', label='ECG')
448         plt.plot(self.t_ecg, self.ecg_filtered, 'r-', label='ECG filtered')
449         plt.xlabel('Time (s)')
450         plt.ylabel('ECG Amplitude (mV)')
451         plt.legend()
452         plt.grid(True)
453         plt.title(f'ECG signal {self.pid}')
454
455         plt.figure()
456         plt.plot(ts_ppg, self.ppg_head, 'b-', label='PPG')
457         plt.plot(self.t_ppg, self.ppg_filtered, 'r-', label='PPG filtered')
458         plt.xlabel('Time (s)')
459         plt.ylabel('PPG Amplitude (mV)')
460         plt.legend()
461         plt.grid(True)
462         plt.title(f'PPG signal {self.pid}')
463         plt.show()
464
465     def synchronize(self):
466         t_ecg = np.linspace(0, len(self.ecg_filtered) / self.fs_ecg, len(
    self.ecg_filtered))
467         t_ppg = np.linspace(0, len(self.ppg_filtered) / self.fs_ppg, len(
    self.ppg_filtered))
468         self.ecg_synced, self.ppg_synced, _, _, self.t_synced =
    synchronize_signals(
469             self.ecg_filtered, self.ppg_filtered, t_ecg, t_ppg
470         )
471
472         plt.figure()
473         fig, ax1 = plt.subplots()
474         ax1.plot(self.t_synced, self.ecg_synced, 'b-', label='ECG Synced')
475         ax1.set_xlabel('Time (s)')
476         ax1.set_ylabel('ECG Amplitude (mV)', color='b')
477         ax1.tick_params(axis='y', labelcolor='b')
478         ax1.grid(True)
479
480         ax2 = ax1.twinx()
```

```
        ax2.plot(self.t_synced, self.ppg_synced, 'r-', label='PPG Synced')
        ax2.set_ylabel('PPG Amplitude (mV)', color='r')
        ax2.tick_params(axis='y', labelcolor='r')

        fig.suptitle(f'Synchronized Signals for Patient {self.pid}')
        fig.tight_layout()
        plt.legend(loc='upper right')
        ax1.legend(loc='upper left')
        plt.show()

    def detect_peaks(self):
        self.r_peaks, self.r_times = peaks_detection(self.ecg_synced, self.t_synced, self.t_synced, window_sec=0.4)
        self.s_peaks, self.s_times = peaks_detection(self.ppg_synced, self.t_synced, self.t_synced, window_sec=0.5)
        self.r_peaks, self.r_times = clean_peaks(self.r_peaks, self.r_times, n_std=2)
        self.s_peaks, self.s_times = clean_peaks(self.s_peaks, self.s_times, n_std=2)

        plt.figure()
        plt.plot(self.t_synced, self.ecg_synced, label="Filtered ECG (synced)")
        plt.plot(self.r_times, self.r_peaks, 'o', label="R-peaks")
        plt.xlabel("Time (s)")
        plt.ylabel("ECG Amplitude (mV)")
        plt.title(f"ECG Signal with Detected Peaks - Patient: {self.pid}")
        plt.legend()
        plt.grid()
        plt.show()

        plt.figure()
        plt.plot(self.t_synced, self.ppg_synced, label="Filtered PPG")
        plt.plot(self.s_times, self.s_peaks, 'o', label="S-peaks")
        plt.xlabel("Time (s)")
        plt.ylabel("PPG Amplitude (mV)")
        plt.title(f"PPG Signal with Detected Peaks - Patient: {self.pid}")
        plt.legend()
        plt.grid()
        plt.show()

    def extract_features(self):
        n = 0
        T = 0
        found = 0
        ptt = np.zeros(len(self.ecg_synced))
```

```
        hr = np.zeros(len(self.ecg_synced))
        timetable = np.zeros(len(self.ecg_synced))
        for i in range(len(self.r_peaks)):
            if i >= T:
                if self.r_peaks[i] > 0:
                    found = 0
                    for j in range(len(self.s_peaks)):
                        if self.s_peaks[j] > 0 and self.s_times[j] > self.
    r_times[i]:
                            ptt[n] = self.s_times[j] - self.r_times[i]
                            for k in range(i + 1, len(self.r_peaks)):
                                if self.r_peaks[k] > 0:
                                    hr[n] = 60 / (self.r_times[k] - self.
    r_times[i])
                                    timetable[n] = self.r_times[i]
                                    n += 1
                                    T = k
                                    found = 1
                                    break
                            break
        # Clean arrays
        ind = np.where(ptt == 0)
        ptt = np.delete(ptt, ind)
        ind = np.where(hr == 0)
        hr = np.delete(hr, ind)
        ind = np.where(timetable == 0)
        timetable = np.delete(timetable, ind)
        if len(ptt) > len(hr):
            ind = np.arange(0, len(hr))
            ptt = ptt[ind]
            timetable = timetable[ind]
        else:
            ind = np.arange(0, len(ptt))
            hr = hr[ind]
            timetable = timetable[ind]
        self.ptt = ptt
        self.hr = hr
        self.timetable = timetable

        plt.figure()
        ax1 = plt.gca()  # Get the current axis
        ax1.plot(self.timetable, self.hr, 'b-', label='Heart Rate')
        ax1.set_xlabel('Time (s)')
        ax1.set_ylabel('Heart Rate (bpm)', color='b')
        ax1.tick_params(axis='y', labelcolor='b')

```

```
        ax2 = ax1.twinx()  # Create a twin y-axis sharing the same x-axis
        ax2.plot(self.timetable, self.ptt, 'r-', label='Pulse Transit Time')
        ax2.set_ylabel('Pulse Transit Time (s)', color='r')
        ax2.tick_params(axis='y', labelcolor='r')

        plt.title(f'Heart Rate and Pulse Transit Time patient: {self.pid}')
        plt.legend(loc='upper right')
        ax1.legend(loc='upper left')
        plt.grid()
        plt.show()

    def load_reference_bp(self):
        #interpolation of the signals to the same sampling frequency
        self.time_ref_device = []
        self.sbp = []  # Systolic Blood Pressure
        self.dbp = []  # Diastolic Blood Pressure
        if self.pid == "LUCA":
            with open(f"./{self.pid}_ref_1.csv", 'r') as filecsv:
                reader = csv.reader(filecsv, delimiter=";")
                for row in reader:
                    self.time_ref_device.append(int(row[0]))  # First column
    : Time in ms
                    self.sbp.append(int(row[1]))        # Second column:
    Systolic BP
                    self.dbp.append(int(row[2]))  # Third column: Diastolic
    BP
        else:
            with open(f"./SHIMMER_{self.pid}.csv", 'r') as filecsv:
                reader = csv.reader(filecsv, delimiter=";")
                for row in reader:
                    self.time_ref_device.append(int(row[0]))  # First column
    : Time in ms
                    self.sbp.append(int(row[1]))        # Second column:
    Systolic BP
                    self.dbp.append(int(row[2]))  # Third column: Diastolic
    BP


        print("Systolic Blood Pressure:", self.sbp)
        print("Diastolic Blood Pressure:", self.dbp)
        self.time_ref_device = np.array(self.time_ref_device)  # Convert to
    seconds
        self.time_ref_device= self.time_ref_device - 60*np.ones(len(self.
    time_ref_device)) # Remove 60ms delay
        self.time_ref_device = (self.time_ref_device - self.time_ref_device
    [0])/1000
```

```
603         print("Time Reference Device:", self.time_ref_device)
604 
605         # Generate an interpolating time array from ECG and PPG timestamps
606         n = np.where(self.t_ecg >= self.timetable[0])[0][0]  # Find the
    index of the first timestamp in timetable
607         m = np.where(self.t_ecg <= self.timetable[-1])[0][-1]  # Find the
    index of the last timestamp in timetable
608         ind = np.arange(n, m + 1)  # Create an index range from n to m
609         interp_time = self.t_ecg[ind]  # Interpolating time array based on
    ECG timestamps
610 
611         # Interpolate SBP and DBP values
612         SBP_fit = np.interp(interp_time, self.time_ref_device, self.sbp)  #
    Interpolate SBP
613         DBP_fit = np.interp(interp_time, self.time_ref_device, self.dbp)  #
    Interpolate DBP
614 
615         # Interpolate HR and PTT values using cubic splines
616         HR_fit = CubicSpline(self.timetable, self.hr)(interp_time)
617         PTT_fit = CubicSpline(self.timetable, self.ptt)(interp_time)
618 
619         # Reduce dimensionality (10-second intervals)
620         sbp_reduced = feat_reduction(SBP_fit, interp_time)
621         dbp_reduced = feat_reduction(DBP_fit, interp_time)
622         hr_reduced = feat_reduction(HR_fit, interp_time)
623         ptt_reduced = feat_reduction(PTT_fit, interp_time)
624 
625         # Resample reduced arrays back to the timetable
626         self.sbp_resampled_p = np.interp(self.timetable, interp_time,
    sbp_reduced)  # Resample SBP
627         self.dbp_resampled_p = np.interp(self.timetable, interp_time,
    dbp_reduced)  # Resample DBP
628         self.hr_resampled_p = np.interp(self.timetable, interp_time,
    hr_reduced)    # Resample HR
629         self.ptt_resampled_p = np.interp(self.timetable, interp_time,
    ptt_reduced)  # Resample PTT
630 
631 
632 
633     def get_features(self):
634         return self.ptt_resampled_p, self.hr_resampled_p, self.
    sbp_resampled_p, self.dbp_resampled_p, self.timetable
635 
636     def regression_methods(self, fs_ecg=504.12, fs_ppg=504.12):
637         NUM_PATIENTS = 20
638         NUM_SHIMMER = 20
```

```
        all_ptt, all_hr, all_sbp, all_dbp, groups = [], [], [], [], []

        try:
            if self.dataset == 1:
                for pid in range(20, NUM_SHIMMER + 2 ):
                    # SHIMMER
                    ecg_mat = scipy.io.loadmat(f"SHIMMER_{pid}_ECG.mat")
                    ecg = ecg_mat['signal'].flatten()
                    t_ecg = ecg_mat['ts'].flatten() / 1000

                    ppg_mat = scipy.io.loadmat(f"SHIMMER_{pid}_PPG.mat")
                    ppg = ppg_mat['signal'].flatten()
                    t_ppg = ppg_mat['ts'].flatten() / 1000

                    ref_file = f"SHIMMER_{pid}.csv"
                    ptt, hr, sbp, dbp = extract_features_reg(ecg, ppg, t_ecg
    , t_ppg, ref_file, fs_ecg, fs_ppg)
                    all_ptt.append(ptt)
                    all_hr.append(hr)
                    all_sbp.append(sbp)
                    all_dbp.append(dbp)
                    groups.append(np.full(len(ptt), pid))

                for pid in range(1, NUM_PATIENTS ):
                    # LUCA
                    ecg_mat = scipy.io.loadmat(f"LUCA_{pid}_ECG.mat")
                    ecg = ecg_mat['ECG_signal'].flatten()
                    t_ecg = ecg_mat['ECG_ts'].flatten() / 1000

                    ppg_mat = scipy.io.loadmat(f"LUCA_{pid}_PPG.mat")
                    ppg = ppg_mat['PPG_signal'].flatten()
                    t_ppg = ppg_mat['PPG_ts'].flatten() / 1000

                    ref_file = f"LUCA_ref_{pid}.csv"
                    ptt, hr, sbp, dbp = extract_features_reg(ecg, ppg, t_ecg
    , t_ppg, ref_file, fs_ecg, fs_ppg)
                    all_ptt.append(ptt)
                    all_hr.append(hr)
                    all_sbp.append(sbp)
                    all_dbp.append(dbp)
                    groups.append(np.full(len(ptt), 43 + pid))  # 99 for
    LUCA

            elif self.dataset == 2:
                for pid in range(18, NUM_PATIENTS + 13 ):
                    if pid == 10 or pid == 12 or pid == 21 or pid == 20:
```

```
                        continue
                    ecg_mat = scipy.io.loadmat(f"SHIMMER_{pid}_ECG.mat")
                    ecg = ecg_mat['signal'].flatten()
                    t_ecg = ecg_mat['ts'].flatten() / 1000

                    ppg_mat = scipy.io.loadmat(f"SHIMMER_{pid}_PPG.mat")
                    ppg = ppg_mat['signal'].flatten()
                    t_ppg = ppg_mat['ts'].flatten() / 1000

                    ref_file = f"SHIMMER_{pid}.csv"
                    ptt, hr, sbp, dbp = extract_features_reg(ecg, ppg, t_ecg, t_ppg, ref_file, fs_ecg, fs_ppg)
                    all_ptt.append(ptt)
                    all_hr.append(hr)
                    all_sbp.append(sbp)
                    all_dbp.append(dbp)
                    groups.append(np.full(len(ptt), pid))

            elif self.dataset == 3:
                for pid in range(1, NUM_PATIENTS ):
                    if pid == 10 or pid == 12 or pid == 21 or pid == 20:
                        continue
                    ecg_mat = scipy.io.loadmat(f"LUCA_{pid}_ECG.mat")
                    ecg = ecg_mat['ECG_signal'].flatten()
                    t_ecg = ecg_mat['ECG_ts'].flatten() / 1000

                    ppg_mat = scipy.io.loadmat(f"LUCA_{pid}_PPG.mat")
                    ppg = ppg_mat['PPG_signal'].flatten()
                    t_ppg = ppg_mat['PPG_ts'].flatten() / 1000

                    ref_file = f"LUCA_ref_{pid}.csv"
                    ptt, hr, sbp, dbp = extract_features_reg(ecg, ppg, t_ecg, t_ppg, ref_file, fs_ecg, fs_ppg)
                    all_ptt.append(ptt)
                    all_hr.append(hr)
                    all_sbp.append(sbp)
                    all_dbp.append(dbp)
                    groups.append(np.full(len(ptt), 43 + pid))  # 99 for LUCA
            else:
                raise ValueError("Invalid dataset option. Choose 1, 2, or 3.")
        except Exception as e:
            print(f"Error with patient {pid}: {e}")

        # add patient Luca
```

```
        try:
            ptt_luca = self.ptt_resampled_p
            hr_luca = self.hr_resampled_p
            sbp_luca = self.sbp_resampled_p
            dbp_luca = self.dbp_resampled_p
            all_ptt.append(ptt_luca)
            all_hr.append(hr_luca)
            all_sbp.append(sbp_luca)
            all_dbp.append(dbp_luca)
            groups.append(np.full(len(ptt_luca), 99))  # 99 luca id

        except Exception as e:
            print(f"Error with patient LUCA: {e}")

        # Concatenate all data
        features = np.column_stack((np.concatenate(all_ptt), np.concatenate(
    all_hr)))
        targets_sbp = np.concatenate(all_sbp)
        targets_dbp = np.concatenate(all_dbp)
        groups = np.concatenate(groups)

        # division 70%-30% ---
        sz_train = round(0.7 * len(features))
        ind_train = np.arange(0, sz_train)
        ind_test = np.arange(sz_train, len(features))

        X_train = features[ind_train]
        X_test = features[ind_test]
        y_sbp_train = targets_sbp[ind_train]
        y_sbp_test = targets_sbp[ind_test]
        y_dbp_train = targets_dbp[ind_train]
        y_dbp_test = targets_dbp[ind_test]

        # Standardization
        scaler = StandardScaler()
        X_train = scaler.fit_transform(X_train)
        X_test = scaler.transform(X_test)

        # REGRESSION METHODS
        regr = LinearRegression()
        regr.fit(X_train, y_sbp_train)
        sbp_pred = regr.predict(X_test)
        mae_sbp = mean_absolute_error(y_sbp_test, sbp_pred)
        sd_sbp = np.std(sbp_pred - y_sbp_test)

        regr.fit(X_train, y_dbp_train)
```

```
769         dbp_pred = regr.predict(X_test)
770         mae_dbp = mean_absolute_error(y_dbp_test, dbp_pred)
771         sd_dbp = np.std(dbp_pred - y_dbp_test)
772 
773         print(f"Linear Reg SBP MAE: {mae_sbp:.2f} +- {sd_sbp:.2f} mmHg")
774         print(f"Linear Reg DBP MAE: {mae_dbp:.2f} +- {sd_dbp:.2f} mmHg")
775 
776         # take the first patient for plotting (LUCA)
777         sbp = self.sbp_resampled_p
778         dbp = self.dbp_resampled_p
779         t_x = self.t_synced[:len(sbp)]
780 
781         # Reshape predictions to match the length of the reference signals
782         sbp_pred_full = sbp_pred[:len(sbp)]
783         dbp_pred_full = dbp_pred[:len(dbp)]
784 
785         # Resample to a uniform time grid
786         num_points = len(t_x)
787         t_uniform = np.linspace(0, 1200, num_points)
788 
789         sbp_resampled = np.interp(t_uniform, np.linspace(0, 1200, len(sbp)),
     sbp)
790         sbp_pred_resampled = np.interp(t_uniform, np.linspace(0, 1200, len(
    sbp_pred_full)), sbp_pred_full)
791         dbp_resampled = np.interp(t_uniform, np.linspace(0, 1200, len(dbp)),
     dbp)
792         dbp_pred_resampled = np.interp(t_uniform, np.linspace(0, 1200, len(
    dbp_pred_full)), dbp_pred_full)
793 
794         # Plot Linear Regression
795         sbp_pred_at_ref = get_at_ref_times(sbp_pred_resampled, t_uniform,
    self.time_ref_device)
796         dbp_pred_at_ref = get_at_ref_times(dbp_pred_resampled, t_uniform,
    self.time_ref_device)
797 
798         plt.figure()
799         plt.plot(self.time_ref_device, self.sbp,  label='SBP Reference',
    color='b')
800         plt.plot(self.time_ref_device, sbp_pred_at_ref,  label='SBP
    Predicted', color='r')
801         plt.xlabel('Time (s)')
802         plt.ylabel('SBP (mmHg)')
803         plt.title('Linear Reg SBP: Real vs Predicted (Reference Instants)')
804         plt.legend()
805         plt.grid(True)
806         plt.show()
```

```
807
808         plt.figure()
809         plt.plot(self.time_ref_device, self.dbp, label='DBP Reference',
    color='b')
810         plt.plot(self.time_ref_device, dbp_pred_at_ref,  label='DBP
    Predicted', color='r')
811         plt.xlabel('Time (s)')
812         plt.ylabel('DBP (mmHg)')
813         plt.title('Linear Reg DBP: Real vs Predicted (Reference Instants)')
814         plt.legend()
815         plt.grid(True)
816         plt.show()
817
818         # --- LMS ---
819         w, b, _ = lms(X_train, y_sbp_train, mu=0.0001, epochs=10)
820         sbp_pred_lms = np.dot(X_test, w) + b
821         w, b, _ = lms(X_train, y_dbp_train, mu=0.0001, epochs=10)
822         dbp_pred_lms = np.dot(X_test, w) + b
823
824         sbp_pred_lms_full = sbp_pred_lms[:len(sbp)]
825         dbp_pred_lms_full = dbp_pred_lms[:len(dbp)]
826
827         sbp_pred_lms_resampled = np.interp(t_uniform, np.linspace(0, 1200,
    len(sbp_pred_lms_full)), sbp_pred_lms_full)
828         dbp_pred_lms_resampled = np.interp(t_uniform, np.linspace(0, 1200,
    len(dbp_pred_lms_full)), dbp_pred_lms_full)
829
830         sbp_pred_lms_at_ref = get_at_ref_times(sbp_pred_lms_resampled,
    t_uniform, self.time_ref_device)
831         dbp_pred_lms_at_ref = get_at_ref_times(dbp_pred_lms_resampled,
    t_uniform, self.time_ref_device)
832
833         mae_sbp_lms = mean_absolute_error(self.sbp, sbp_pred_lms_at_ref)
834         std_sbp_lms = np.std(self.sbp - sbp_pred_lms_at_ref)
835         mae_dbp_lms = mean_absolute_error(self.dbp, dbp_pred_lms_at_ref)
836         std_dbp_lms = np.std(self.dbp - dbp_pred_lms_at_ref)
837
838         print(f"LMS SBP MAE: {mae_sbp_lms:.2f} +- {std_sbp_lms:.2f} mmHg")
839         print(f"LMS DBP MAE: {mae_dbp_lms:.2f} +- {std_dbp_lms:.2f} mmHg")
840
841         plt.figure()
842         plt.plot(self.time_ref_device, self.sbp, label='SBP Reference',
    color='b')
843         plt.plot(self.time_ref_device, sbp_pred_lms_at_ref, label='SBP
    Predicted (LMS)', color='r')
844         plt.xlabel('Time (s)')
```

```
        plt.ylabel('SBP (mmHg)')
        plt.title('LMS SBP: Real vs Predicted (Reference Instants)')
        plt.legend()
        plt.grid(True)
        plt.show()

        plt.figure()
        plt.plot(self.time_ref_device, self.dbp, label='DBP Reference', color='b')
        plt.plot(self.time_ref_device, dbp_pred_lms_at_ref, label='DBP Predicted (LMS)', color='r')
        plt.xlabel('Time (s)')
        plt.ylabel('DBP (mmHg)')
        plt.title('LMS DBP: Real vs Predicted (Reference Instants)')
        plt.legend()
        plt.grid(True)
        plt.show()

        # --- Ridge ---
        ridge = Ridge(alpha=1.0)
        ridge.fit(X_train, y_sbp_train)
        sbp_pred_ridge = ridge.predict(X_test)
        ridge.fit(X_train, y_dbp_train)
        dbp_pred_ridge = ridge.predict(X_test)

        sbp_pred_ridge_full = sbp_pred_ridge[:len(sbp)]
        dbp_pred_ridge_full = dbp_pred_ridge[:len(dbp)]

        sbp_pred_ridge_resampled = np.interp(t_uniform, np.linspace(0, 1200, len(sbp_pred_ridge_full)), sbp_pred_ridge_full)
        dbp_pred_ridge_resampled = np.interp(t_uniform, np.linspace(0, 1200, len(dbp_pred_ridge_full)), dbp_pred_ridge_full)

        sbp_pred_ridge_at_ref = get_at_ref_times(sbp_pred_ridge_resampled, t_uniform, self.time_ref_device)
        dbp_pred_ridge_at_ref = get_at_ref_times(dbp_pred_ridge_resampled, t_uniform, self.time_ref_device)

        mae_sbp_ridge = mean_absolute_error(self.sbp, sbp_pred_ridge_at_ref)
        std_sbp_ridge = np.std(self.sbp - sbp_pred_ridge_at_ref)
        mae_dbp_ridge = mean_absolute_error(self.dbp, dbp_pred_ridge_at_ref)
        std_dbp_ridge = np.std(self.dbp - dbp_pred_ridge_at_ref)

        print(f"Ridge SBP MAE: {mae_sbp_ridge:.2f} +- {std_sbp_ridge:.2f} mmHg")
```

```
883         print(f"Ridge DBP MAE: {mae_dbp_ridge:.2f} +- {std_dbp_ridge:.2f}
    mmHg")
884 
885         plt.figure()
886         plt.plot(self.time_ref_device, self.sbp, label='SBP Reference',
    color='b')
887         plt.plot(self.time_ref_device, sbp_pred_ridge_at_ref,  label='SBP
    Predicted (Ridge)', color='r')
888         plt.xlabel('Time (s)')
889         plt.ylabel('SBP (mmHg)')
890         plt.title('Ridge SBP: Real vs Predicted (Reference Instants)')
891         plt.legend()
892         plt.grid(True)
893         plt.show()
894 
895         plt.figure()
896         plt.plot(self.time_ref_device, self.dbp,  label='DBP Reference',
    color='b')
897         plt.plot(self.time_ref_device, dbp_pred_ridge_at_ref,  label='DBP
    Predicted (Ridge)', color='r')
898         plt.xlabel('Time (s)')
899         plt.ylabel('DBP (mmHg)')
900         plt.title('Ridge DBP: Real vs Predicted (Reference Instants)')
901         plt.legend()
902         plt.grid(True)
903         plt.show()
904 
905         # --- SVM Regression ---
906         svm = SVR(kernel='rbf', C=1.0, epsilon=0.2)
907         svm.fit(X_train, y_sbp_train)
908         sbp_pred_svm = svm.predict(X_test)
909         svm.fit(X_train, y_dbp_train)
910         dbp_pred_svm = svm.predict(X_test)
911 
912         sbp_pred_svm_full = sbp_pred_svm[:len(sbp)]
913         dbp_pred_svm_full = dbp_pred_svm[:len(dbp)]
914 
915         sbp_pred_svm_resampled = np.interp(t_uniform, np.linspace(0, 1200,
    len(sbp_pred_svm_full)), sbp_pred_svm_full)
916         dbp_pred_svm_resampled = np.interp(t_uniform, np.linspace(0, 1200,
    len(dbp_pred_svm_full)), dbp_pred_svm_full)
917 
918         sbp_pred_svm_at_ref = get_at_ref_times(sbp_pred_svm_resampled,
    t_uniform, self.time_ref_device)
919         dbp_pred_svm_at_ref = get_at_ref_times(dbp_pred_svm_resampled,
    t_uniform, self.time_ref_device)
```

```
920
921         mae_sbp_svm = mean_absolute_error(self.sbp, sbp_pred_svm_at_ref)
922         std_sbp_svm = np.std(self.sbp - sbp_pred_svm_at_ref)
923         mae_dbp_svm = mean_absolute_error(self.dbp, dbp_pred_svm_at_ref)
924         std_dbp_svm = np.std(self.dbp - dbp_pred_svm_at_ref)
925
926         print(f"SVM SBP MAE: {mae_sbp_svm:.2f} +- {std_sbp_svm:.2f} mmHg")
927         print(f"SVM DBP MAE: {mae_dbp_svm:.2f} +- {std_dbp_svm:.2f} mmHg")
928
929         # Plot for SVM
930         plt.figure()
931         plt.plot(self.time_ref_device, self.sbp, label='SBP Reference',
    color='b')
932         plt.plot(self.time_ref_device, sbp_pred_svm_at_ref, label='SBP
    Predicted (SVM)', color='r')
933         plt.xlabel('Time (s)')
934         plt.ylabel('SBP (mmHg)')
935         plt.title('SVM SBP: Real vs Predicted (Reference Instants)')
936         plt.legend()
937         plt.grid(True)
938         plt.show()
939
940         plt.figure()
941         plt.plot(self.time_ref_device, self.dbp, label='DBP Reference',
    color='b')
942         plt.plot(self.time_ref_device, dbp_pred_svm_at_ref, label='DBP
    Predicted (SVM)', color='r')
943         plt.xlabel('Time (s)')
944         plt.ylabel('DBP (mmHg)')
945         plt.title('SVM DBP: Real vs Predicted (Reference Instants)')
946         plt.legend()
947         plt.grid(True)
948         plt.show()
949
950         #cross-correlation for the train and test sets
951
952         kf = KFold(n_splits=3, shuffle=True, random_state=42)
953         mae_sbp_list, mae_dbp_list = [], []
954         mae_sbp_lms_list, mae_dbp_lms_list = [], []
955         mae_sbp_ridge_list, mae_dbp_ridge_list = [], []
956         mae_sbp_svm_list, mae_dbp_svm_list = [], []
957
958
959         for train_index, test_index in kf.split(features):
960             X_train, X_test = features[train_index], features[test_index]
```

```
            y_sbp_train, y_sbp_test = targets_sbp[train_index], targets_sbp[
    test_index]
            y_dbp_train, y_dbp_test = targets_dbp[train_index], targets_dbp[
    test_index]

            scaler = StandardScaler()
            X_train = scaler.fit_transform(X_train)
            X_test = scaler.transform(X_test)

            # Linear Regression
            regr = LinearRegression()
            regr.fit(X_train, y_sbp_train)
            sbp_pred = regr.predict(X_test)
            mae_sbp = mean_absolute_error(y_sbp_test, sbp_pred)
            mae_sbp_list.append(mae_sbp)

            regr.fit(X_train, y_dbp_train)
            dbp_pred = regr.predict(X_test)
            mae_dbp = mean_absolute_error(y_dbp_test, dbp_pred)
            mae_dbp_list.append(mae_dbp)

            # LMS Regression
            w, b, _ = lms(X_train, y_sbp_train, mu=0.0001, epochs=10)
            sbp_pred_lms = np.dot(X_test, w) + b
            mae_sbp_lms = mean_absolute_error(y_sbp_test, sbp_pred_lms)
            mae_sbp_lms_list.append(mae_sbp_lms)

            w, b, _ = lms(X_train, y_dbp_train, mu=0.0001, epochs=10)
            dbp_pred_lms = np.dot(X_test, w) + b
            mae_dbp_lms = mean_absolute_error(y_dbp_test, dbp_pred_lms)
            mae_dbp_lms_list.append(mae_dbp_lms)

            # Ridge Regression
            ridge = Ridge(alpha=1.0)
            ridge.fit(X_train, y_sbp_train)
            sbp_pred_ridge = ridge.predict(X_test)
            mae_sbp_ridge = mean_absolute_error(y_sbp_test, sbp_pred_ridge)
            mae_sbp_ridge_list.append(mae_sbp_ridge)

            ridge.fit(X_train, y_dbp_train)
            dbp_pred_ridge = ridge.predict(X_test)
            mae_dbp_ridge = mean_absolute_error(y_dbp_test, dbp_pred_ridge)
            mae_dbp_ridge_list.append(mae_dbp_ridge)

            # SVM Regression
            svm = SVR(kernel='rbf', C=1.0, epsilon=0.2)
```

```
            svm.fit(X_train, y_sbp_train)
            sbp_pred_svm = svm.predict(X_test)
            mae_sbp_svm = mean_absolute_error(y_sbp_test, sbp_pred_svm)
            mae_sbp_svm_list.append(mae_sbp_svm)

            svm.fit(X_train, y_dbp_train)
            dbp_pred_svm = svm.predict(X_test)
            mae_dbp_svm = mean_absolute_error(y_dbp_test, dbp_pred_svm)
            mae_dbp_svm_list.append(mae_dbp_svm)

        print(f"Linear Reg SBP MAE (CV): {np.mean(mae_sbp_list):.2f} +- {np.std(mae_sbp_list):.2f} mmHg")
        print(f"Linear Reg DBP MAE (CV): {np.mean(mae_dbp_list):.2f} +- {np.std(mae_dbp_list):.2f} mmHg")
        print(f"LMS SBP MAE (CV): {np.mean(mae_sbp_lms_list):.2f} +- {np.std(mae_sbp_lms_list):.2f} mmHg")
        print(f"LMS DBP MAE (CV): {np.mean(mae_dbp_lms_list):.2f} +- {np.std(mae_dbp_lms_list):.2f} mmHg")
        print(f"Ridge SBP MAE (CV): {np.mean(mae_sbp_ridge_list):.2f} +- {np.std(mae_sbp_ridge_list):.2f} mmHg")
        print(f"Ridge DBP MAE (CV): {np.mean(mae_dbp_ridge_list):.2f} +- {np.std(mae_dbp_ridge_list):.2f} mmHg")
        print(f"SVM SBP MAE (CV): {np.mean(mae_sbp_svm_list):.2f} +- {np.std(mae_sbp_svm_list):.2f} mmHg")
        print(f"SVM DBP MAE (CV): {np.mean(mae_dbp_svm_list):.2f} +- {np.std(mae_dbp_svm_list):.2f} mmHg")

        # Interpolate predictions to the reference time points
        sbp_pred_at_ref = np.interp(self.time_ref_device, np.linspace(self.time_ref_device[0], self.time_ref_device[-1], len(sbp_pred)), sbp_pred)
        dbp_pred_at_ref = np.interp(self.time_ref_device, np.linspace(self.time_ref_device[0], self.time_ref_device[-1], len(dbp_pred)), dbp_pred)

        sbp_pred_lms_at_ref = np.interp(self.time_ref_device, np.linspace(self.time_ref_device[0], self.time_ref_device[-1], len(sbp_pred_lms)), sbp_pred_lms)
        dbp_pred_lms_at_ref = np.interp(self.time_ref_device, np.linspace(self.time_ref_device[0], self.time_ref_device[-1], len(dbp_pred_lms)), dbp_pred_lms)

        sbp_pred_ridge_at_ref = np.interp(self.time_ref_device, np.linspace(self.time_ref_device[0], self.time_ref_device[-1], len(sbp_pred_ridge)), sbp_pred_ridge)
        dbp_pred_ridge_at_ref = np.interp(self.time_ref_device, np.linspace(self.time_ref_device[0], self.time_ref_device[-1], len(dbp_pred_ridge)), dbp_pred_ridge)
```

```
1033
1034        sbp_pred_svm_at_ref = np.interp(self.time_ref_device, np.linspace(
    self.time_ref_device[0], self.time_ref_device[-1], len(sbp_pred_svm)),
    sbp_pred_svm)
1035        dbp_pred_svm_at_ref = np.interp(self.time_ref_device, np.linspace(
    self.time_ref_device[0], self.time_ref_device[-1], len(dbp_pred_svm)),
    dbp_pred_svm)
1036
1037        # Plot for Linear Regression
1038        plt.figure()
1039        plt.plot(self.time_ref_device, self.sbp, label='SBP Reference',
    color='b')
1040        plt.plot(self.time_ref_device, sbp_pred_at_ref, label='SBP Predicted
     (Linear)', color='r')
1041        plt.xlabel('Time (s)')
1042        plt.ylabel('SBP (mmHg)')
1043        plt.title('Linear Regression SBP: Real vs Predicted (Reference
    Instants)')
1044        plt.legend()
1045        plt.grid(True)
1046        plt.show()
1047
1048        plt.figure()
1049        plt.plot(self.time_ref_device, self.dbp, label='DBP Reference',
    color='b')
1050        plt.plot(self.time_ref_device, dbp_pred_at_ref, label='DBP Predicted
     (Linear)', color='r')
1051        plt.xlabel('Time (s)')
1052        plt.ylabel('DBP (mmHg)')
1053        plt.title('Linear Regression DBP: Real vs Predicted (Reference
    Instants)')
1054        plt.legend()
1055        plt.grid(True)
1056        plt.show()
1057
1058        # Plot for LMS
1059        plt.figure()
1060        plt.plot(self.time_ref_device, self.sbp, label='SBP Reference',
    color='b')
1061        plt.plot(self.time_ref_device, sbp_pred_lms_at_ref, label='SBP
    Predicted (LMS)', color='r')
1062        plt.xlabel('Time (s)')
1063        plt.ylabel('SBP (mmHg)')
1064        plt.title('LMS SBP: Real vs Predicted (Reference Instants)')
1065        plt.legend()
1066        plt.grid(True)
```

```
1067        plt.show()
1068
1069        plt.figure()
1070        plt.plot(self.time_ref_device, self.dbp, label='DBP Reference',
    color='b')
1071        plt.plot(self.time_ref_device, dbp_pred_lms_at_ref, label='DBP
    Predicted (LMS)', color='r')
1072        plt.xlabel('Time (s)')
1073        plt.ylabel('DBP (mmHg)')
1074        plt.title('LMS DBP: Real vs Predicted (Reference Instants)')
1075        plt.legend()
1076        plt.grid(True)
1077        plt.show()
1078
1079        # Plot for Ridge
1080        plt.figure()
1081        plt.plot(self.time_ref_device, self.sbp, label='SBP Reference',
    color='b')
1082        plt.plot(self.time_ref_device, sbp_pred_ridge_at_ref, label='SBP
    Predicted (Ridge)', color='r')
1083        plt.xlabel('Time (s)')
1084        plt.ylabel('SBP (mmHg)')
1085        plt.title('Ridge SBP: Real vs Predicted (Reference Instants)')
1086        plt.legend()
1087        plt.grid(True)
1088        plt.show()
1089
1090        plt.figure()
1091        plt.plot(self.time_ref_device, self.dbp, label='DBP Reference',
    color='b')
1092        plt.plot(self.time_ref_device, dbp_pred_ridge_at_ref, label='DBP
    Predicted (Ridge)', color='r')
1093        plt.xlabel('Time (s)')
1094        plt.ylabel('DBP (mmHg)')
1095        plt.title('Ridge DBP: Real vs Predicted (Reference Instants)')
1096        plt.legend()
1097        plt.grid(True)
1098        plt.show()
1099
1100        #plot for svm
1101        plt.figure()
1102        plt.plot(self.time_ref_device, self.sbp, label='SBP Reference',
    color='b')
1103        plt.plot(self.time_ref_device, sbp_pred_svm_at_ref, label='SBP
    Predicted (SVM)', color='r')
1104        plt.xlabel('Time (s)')
```

```
        plt.ylabel('SBP (mmHg)')
        plt.title('SVM SBP: Real vs Predicted (Reference Instants)')
        plt.legend()
        plt.grid(True)
        plt.show()

        plt.figure()
        plt.plot(self.time_ref_device, self.dbp, label='DBP Reference', color='b')
        plt.plot(self.time_ref_device, dbp_pred_svm_at_ref, label='DBP Predicted (SVM)', color='r')
        plt.xlabel('Time (s)')
        plt.ylabel('DBP (mmHg)')
        plt.title('SVM DBP: Real vs Predicted (Reference Instants)')
        plt.legend()
        plt.grid(True)
        plt.show()

if __name__ == '__main__':
    patient_id = "LUCA" #"LUCA" or patient_id = 1.. for "SHIMMER_1" # or "SHIMMER_2", etc.
    dataset = 3  # 1 for both datasets SHIMMER, 2 for SMIMMER, 3 for LUCA
    p = Patient(patient_id,dataset)
    p.load_signals()
    p.preprocess()
    p.synchronize()
    p.detect_peaks()
    p.extract_features()
    p.load_reference_bp()
    p.regression_methods()
```

Listing 1: Main Code

```
    import scipy.io
    import matplotlib.pyplot as plt
    import numpy as np
    from ex import filter_ecg_signal, filter_ppg_signal
    # --- Load signals ---
    pid = "LUCA"
    ecg_mat = scipy.io.loadmat(f"{pid}_1_ECG.mat")
    ecg = ecg_mat['ECG_signal'].flatten()
    t_ecg = ecg_mat['ECG_ts'].flatten() / 1000  #bring to seconds

    ppg_mat = scipy.io.loadmat(f"SHIMMER_42_PPG.mat")
    ppg = ppg_mat['signal'].flatten()
    t_ppg = ppg_mat['ts'].flatten() / 1000  #bring to seconds

    fs_ecg = 504.12
    fs_ppg = 504.12

    # --- CUT signals: remove first 20s and last 30s ---
    cut_int_ecg = round(20 * fs_ecg)
    cut_int_ppg = round(20 * fs_ppg)
    ecg_cut = ecg[cut_int_ecg:]
    t_ecg_cut = t_ecg[cut_int_ecg:]
    ppg_cut = ppg[cut_int_ppg:]
    t_ppg_cut = t_ppg[cut_int_ppg:]

    cut_end_ecg = round(30 * fs_ecg)
    cut_end_ppg = round(30 * fs_ppg)
    if cut_end_ecg < len(ecg_cut):
        ecg_cut = ecg_cut[:-cut_end_ecg]
        t_ecg_cut = t_ecg_cut[:-cut_end_ecg]
    if cut_end_ppg < len(ppg_cut):
        ppg_cut = ppg_cut[:-cut_end_ppg]
        t_ppg_cut = t_ppg_cut[:-cut_end_ppg]

    # --- Plot raw and cut ECG together ---
    plt.figure()
    plt.plot(t_ecg, ecg, color='b', alpha=0.5, label='Raw ECG')
    plt.plot(t_ecg_cut, ecg_cut, color='r', alpha=0.8, label='ECG after cut'
    )
    plt.title('ECG: Raw vs Cut for patient LUCA')
    plt.xlabel('Time Unix')
    plt.ylabel('ECG Amplitude (mV)')
    plt.legend()
    plt.grid(True)
    plt.show()
```

```
45
46      # --- Plot raw and cut PPG together ---
47      plt.figure()
48      plt.plot(t_ppg, ppg, color='b', alpha=0.5, label='Raw PPG')
49      plt.plot(t_ppg_cut, ppg_cut, color='r', alpha=0.8, label='PPG after cut'
        )
50      plt.title('PPG: Raw vs Cut for patient LUCA')
51      plt.xlabel('Time Unix')
52      plt.ylabel('PPG Amplitude (mV)')
53      plt.legend()
54      plt.grid(True)
55      plt.show()
56
57      # --- Apply filters to cut signals ---
58      ecg_filtered = filter_ecg_signal(ecg_cut, fs_ecg, window_size=0.2)
59      ppg_filtered = filter_ppg_signal(ppg_cut, fs_ppg, window_size=0.5)
60
61      # --- Plot filtered signals together (pre-synchronization, after cut)
        ---
62      plt.figure()
63      plt.plot(t_ecg_cut, ecg_cut, 'b-', alpha=0.5, label='ECG after cut')
64      plt.plot(t_ecg_cut, ecg_filtered, 'g-', alpha=0.8, label='ECG filtered')
65      plt.title('ECG: Cut vs Filtered (seconds, pre-sync)')
66      plt.xlabel('Time (s)')
67      plt.ylabel('ECG Amplitude (mV)')
68      plt.legend()
69      plt.grid(True)
70      plt.show()
71
72      plt.figure()
73      plt.plot(t_ppg_cut, ppg_cut, 'b-', alpha=0.5, label='PPG after cut')
74      plt.plot(t_ppg_cut, ppg_filtered, 'g-', alpha=0.8, label='PPG filtered')
75      plt.title('PPG: Cut vs Filtered (seconds, pre-sync)')
76      plt.xlabel('Time (s)')
77      plt.ylabel('PPG Amplitude (mV)')
78      plt.legend()
79      plt.grid(True)
80      plt.show()
```

Listing 2: Code to generate graph of ECG and PPG signals pre vs post reduction

```
1
2     import scipy.io
3     import matplotlib.pyplot as plt
4     import numpy as np
5     import csv
6
7     NUM_PATIENT = 42
8     SBP_signals = []
9     DBP_signals = []
10    time = []
11    patient_ids = []
12
13    for i in range(1, NUM_PATIENT + 1):
14        if i == 10 or i == 12:
15            continue
16        sbp = []
17        dbp = []
18        timestamp = []
19        with open(f"./SHIMMER_{i}.csv", 'r') as filecsv:
20            reader = csv.reader(filecsv, delimiter=";")
21            for row in reader:
22                timestamp.append(int(row[0]))  # First column: Time in ms
23                sbp.append(float(row[1]))      # Second column: Systolic BP
24                dbp.append(float(row[2]))      # Third column: Diastolic BP
25        time_s = [(ts - timestamp[0]) / 1 for ts in timestamp]  # Convert ms
     to seconds
26        SBP_signals.append(sbp)
27        DBP_signals.append(dbp)
28        time.append(time_s)
29        patient_ids.append(i)
30
31
32    # Plot all patients' SBP and DBP signals
33    plt.figure(figsize=(12, 6))
34    for idx, (t, sbp, dbp, patient_id) in enumerate(zip(time, SBP_signals,
    DBP_signals, patient_ids), start=1):
35        if idx == 10 or idx == 12:
36            idx +=1
37            continue
38        plt.plot(t, dbp, label=f'Patient {patient_id} DBP', alpha=0.6)
39    plt.title('DBP Signals for All Shimmer Patients')
40    plt.xlabel('Time (s)')
41    plt.ylabel('Diastolic Blood Pressure (mmHg)')
42    plt.legend(fontsize=6, loc='upper left', bbox_to_anchor=(1, 1))
43    plt.grid(True)
```

```
    plt.show()

    plt.figure(figsize=(12, 6))
    for idx, (t, sbp, dbp, patient_id) in enumerate(zip(time, SBP_signals,
    DBP_signals, patient_ids), start=1):
        if idx == 10 or idx == 12:
            idx +=1
            continue
        plt.plot(t, sbp, label=f'Patient {patient_id} SBP', alpha=0.6)
    plt.title('SBP Signals for All Shimmer Patients')
    plt.xlabel('Time (s)')
    plt.ylabel('Systolic Blood Pressure (mmHg)')
    plt.legend(fontsize=6, loc='upper left', bbox_to_anchor=(1, 1))
    plt.grid(True)
    plt.show()

    # mean values
    mean_sbp = [np.mean(sbp) for sbp in SBP_signals]
    mean_dbp = [np.mean(dbp) for dbp in DBP_signals]
    patient_ids = [i for i in range(1, NUM_PATIENT + 1) if i != 10 and i !=
    12]

    with open(f"./LUCA_ref_1.csv", 'r') as filecsv:
            reader = csv.reader(filecsv, delimiter=";")
            for row in reader:
                timestamp.append(int(row[0]))  # First column: Time in ms
                sbp.append(float(row[1]))      # Second column: Systolic BP
                dbp.append(float(row[2]))      # Third column: Diastolic BP
    time_s = [(ts - timestamp[0]) / 1 for ts in timestamp]
    mean_Luca_dbp = np.mean(dbp)
    mean_Luca_sbp = np.mean(sbp)
    # Histogram with quartiles and outliers
    q25_dbp = np.percentile(mean_dbp, 25)
    q75_dbp = np.percentile(mean_dbp, 75)

    # Limits for outliers
    iqr_dbp = q75_dbp - q25_dbp
    lower_dbp = q25_dbp - 1.5 * iqr_dbp
    upper_dbp = q75_dbp + 1.5 * iqr_dbp

    outlier_dbp_idx = [i for i, val in enumerate(mean_dbp) if val <
    lower_dbp or val > upper_dbp]

    plt.figure(figsize=(10, 5))
    bars = plt.bar(patient_ids, mean_dbp, color='skyblue', label='Mean DBP')
    plt.axhspan(q25_dbp, q75_dbp, color='orange', alpha=0.2, label='IQR (25
```

```
    -75   percentile)')
    for idx in outlier_dbp_idx:
        bars[idx].set_color('red')
        plt.text(patient_ids[idx], mean_dbp[idx]+1, 'OUT', color='red', ha='center', fontsize=8)
    plt.axhline(mean_Luca_dbp, color='purple', linestyle='--', linewidth=2, label='LUCA Mean DBP')
    plt.xlabel('Patient ID')
    plt.ylabel('Mean Diastolic BP (mmHg)')
    plt.title('Mean DBP per Shimmer Patient (Outliers in Red, LUCA in Purple)')
    plt.legend()
    plt.grid(True)
    plt.show()


    q25_sbp = np.percentile(mean_sbp, 25)
    q75_sbp = np.percentile(mean_sbp, 75)

    iqr_sbp = q75_sbp - q25_sbp
    lower_sbp = q25_sbp - 1.5 * iqr_sbp
    upper_sbp = q75_sbp + 1.5 * iqr_sbp

    outlier_sbp_idx = [i for i, val in enumerate(mean_sbp) if val < lower_sbp or val > upper_sbp]

    plt.figure(figsize=(10, 5))
    bars = plt.bar(patient_ids, mean_sbp, color='salmon', label='Mean SBP')
    plt.axhspan(q25_sbp, q75_sbp, color='orange', alpha=0.2, label='IQR (25   -75   percentile)')

    for idx in outlier_sbp_idx:
        bars[idx].set_color('red')
        plt.text(patient_ids[idx], mean_sbp[idx]+1, 'OUT', color='red', ha='center', fontsize=8)
    plt.axhline(mean_Luca_sbp, color='purple', linestyle='--', linewidth=2, label='LUCA Mean SBP')
    plt.xlabel('Patient ID')
    plt.ylabel('Mean Systolic BP (mmHg)')
    plt.title('Mean SBP per Shimmer Patient (Outliers in Red, LUCA in Purple)')
    plt.legend()
    plt.grid(True)
    plt.show()

    # --- LUCA Database  ---
```

```
    NUM_LUCA = 20
    LUCA_SBP_signals = []
    LUCA_DBP_signals = []
    LUCA_time = []
    LUCA_ids = []

    for i in range(1, NUM_LUCA + 1):
        sbp = []
        dbp = []
        timestamp = []
        try:
            with open(f"./LUCA_ref_{i}.csv", 'r') as filecsv:
                reader = csv.reader(filecsv, delimiter=";")
                for row in reader:
                    timestamp.append(int(row[0]))
                    sbp.append(float(row[1]))
                    dbp.append(float(row[2]))
            time_s = [(ts - timestamp[0]) / 1000 for ts in timestamp]  # ms
    -> s
            LUCA_SBP_signals.append(sbp)
            LUCA_DBP_signals.append(dbp)
            LUCA_time.append(time_s)
            LUCA_ids.append(i)
        except FileNotFoundError:
            print(f"LUCA_ref_{i}.csv not found, skipping.")

    # Plot all LUCA DBP signals
    plt.figure(figsize=(12, 6))
    for t, dbp, pid in zip(LUCA_time, LUCA_DBP_signals, LUCA_ids):
        plt.plot(t, dbp, label=f'LUCA {pid} DBP', alpha=0.6)
    plt.title('DBP Signals for All LUCA Patients')
    plt.xlabel('Time (s)')
    plt.ylabel('Diastolic Blood Pressure (mmHg)')
    plt.legend(fontsize=6, loc='upper left', bbox_to_anchor=(1, 1))
    plt.grid(True)
    plt.show()

    # Plot all LUCA SBP signals
    plt.figure(figsize=(12, 6))
    for t, sbp, pid in zip(LUCA_time, LUCA_SBP_signals, LUCA_ids):
        plt.plot(t, sbp, label=f'LUCA {pid} SBP', alpha=0.6)
    plt.title('SBP Signals for All LUCA Patients')
    plt.xlabel('Time (s)')
    plt.ylabel('Systolic Blood Pressure (mmHg)')
    plt.legend(fontsize=6, loc='upper left', bbox_to_anchor=(1, 1))
    plt.grid(True)
```

```
    plt.show()

    # LUCA
    mean_luca_sbp = [np.mean(sbp) for sbp in LUCA_SBP_signals]
    mean_luca_dbp = [np.mean(dbp) for dbp in LUCA_DBP_signals]

    # DBP LUCA
    q25_luca_dbp = np.percentile(mean_luca_dbp, 25)
    q75_luca_dbp = np.percentile(mean_luca_dbp, 75)
    iqr_luca_dbp = q75_luca_dbp - q25_luca_dbp
    lower_luca_dbp = q25_luca_dbp - 1.5 * iqr_luca_dbp
    upper_luca_dbp = q75_luca_dbp + 1.5 * iqr_luca_dbp
    outlier_luca_dbp_idx = [i for i, val in enumerate(mean_luca_dbp) if val
    < lower_luca_dbp or val > upper_luca_dbp]



    # SBP LUCA
    q25_luca_sbp = np.percentile(mean_luca_sbp, 25)
    q75_luca_sbp = np.percentile(mean_luca_sbp, 75)
    iqr_luca_sbp = q75_luca_sbp - q25_luca_sbp
    lower_luca_sbp = q25_luca_sbp - 1.5 * iqr_luca_sbp
    upper_luca_sbp = q75_luca_sbp + 1.5 * iqr_luca_sbp
    outlier_luca_sbp_idx = [i for i, val in enumerate(mean_luca_sbp) if val
    < lower_luca_sbp or val > upper_luca_sbp]


    luca_labels = []
    for pid in LUCA_ids:
        if 1 <= pid <= 15:
            luca_labels.append(f'LUCA {pid}')
        elif 16 <= pid <= 18:
            luca_labels.append(f'Massimo {pid}')
        elif 19 <= pid <= 20:
            luca_labels.append(f'Clelia {pid}')
        else:
            luca_labels.append(f'LUCA {pid}')


    plt.figure(figsize=(12, 5))
    bars = plt.bar(LUCA_ids, mean_luca_dbp, color='violet', label='Mean LUCA
     DBP')
    plt.axhspan(q25_luca_dbp, q75_luca_dbp, color='orange', alpha=0.2, label
    ='IQR (25  -75   percentile)')
    for idx in outlier_luca_dbp_idx:
        bars[idx].set_color('red')
```

```
        plt.text(LUCA_ids[idx], mean_luca_dbp[idx]+1, 'OUT', color='red', ha='center', fontsize=8)
    plt.xlabel('LUCA Patient ID')
    plt.ylabel('Mean Diastolic BP (mmHg)')
    plt.title('Mean DBP per LUCA Patient (Outliers in Red)')
    plt.xticks(LUCA_ids, luca_labels, rotation=45, ha='right')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()


    plt.figure(figsize=(12, 5))
    bars = plt.bar(LUCA_ids, mean_luca_sbp, color='limegreen', label='Mean LUCA SBP')
    plt.axhspan(q25_luca_sbp, q75_luca_sbp, color='orange', alpha=0.2, label='IQR (25  -75   percentile)')
    for idx in outlier_luca_sbp_idx:
        bars[idx].set_color('red')
        plt.text(LUCA_ids[idx], mean_luca_sbp[idx]+1, 'OUT', color='red', ha='center', fontsize=8)
    plt.xlabel('LUCA Patient ID')
    plt.ylabel('Mean Systolic BP (mmHg)')
    plt.title('Mean SBP per LUCA Patient (Outliers in Red)')
    plt.xticks(LUCA_ids, luca_labels, rotation=45, ha='right')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

Listing 3: Code to visualize the different behavior of all patients

References

[1] World Health Organization. *Cardiovascular diseases (CVDs) factsheet*. WHO, 2023.

[2] Roth GA, et al. Global burden of cardiovascular diseases and risk factors, 1990-2021. *J Am Coll Cardiol*. 2023;82(25):2406-52.

[3] World Health Organization. *Hypertension*. WHO, 2023.

[4] Whelton PK, et al. 2017 ACC/AHA/AAPA/ABC/ACPM/AGS/APhA/ASH/ ASPC/NMA/PCNA Guideline for the prevention, detection, evaluation, and management of high blood pressure in adults. *Hypertension*. 2018;71(6):e13-e115.

[5] Williams B, et al. 2023 ESH Guidelines for the management of arterial hypertension. *J Hypertens*. 2023;41(12):1874-2079.

[6] Mills KT, et al. Global disparities of hypertension prevalence and control: a systematic analysis of population-based studies from 90 countries. *Circulation*. 2016;134(6):441-450.

[7] NCD Risk Factor Collaboration. Worldwide trends in hypertension prevalence and progress in treatment and control from 1990 to 2019. *Lancet*. 2021;398(10304):957-980.

[8] O'Brien E, et al. Blood pressure measuring devices: recommendations of the European Society of Hypertension. *BMJ*. 2001;322(7285):531-536.

[9] Pickering TG, et al. Recommendations for blood pressure measurement in humans and experimental animals: part 1: blood pressure measurement in humans. *Hypertension*. 2005;45(1):142-161.

[10] Parati G, et al. Home blood pressure monitoring: methodology, clinical relevance and practical application. *Blood Press Monit*. 2002;7(2):91-107.

[11] Saugel B, et al. Continuous non-invasive arterial pressure monitoring: a review of current methods and future perspectives. *Br J Anaesth*. 2014;113(3):339-349.

[12] Maheshwari K, et al. Invasive arterial blood pressure monitoring. *Anesthesiology*. 2020;132(6):1527-1544.

[13] Mukkamala R, et al. Toward ubiquitous blood pressure monitoring via pulse transit time: theory and practice. *IEEE Trans Biomed Eng*. 2015;62(8):1879-1901.

[14] Slapničar G, et al. Continuous non-invasive blood pressure estimation using photoplethysmogram: a review. *Comput Biol Med*. 2019;111:103386.

[15] Heikenfeld J, et al. Wearable sensors: modalities, challenges, and prospects. *Lab Chip*. 2018;18(2):217-248.

[16] Goldberger AL, et al. *Clinical electrocardiography: a simplified approach*. 9th ed. Elsevier, 2017.

[17] Allen J. Photoplethysmography and its application in clinical physiological measurement. *Physiol Meas*. 2007;28(3):R1-R39.

[18] Tamura T, et al. Wearable photoplethysmographic sensors—past and present. *Electronics*. 2014;3(2):282-302.

[19] Zhang G, et al. Cuffless blood pressure estimation from photoplethysmogram and electrocardiogram signals using machine learning techniques. *Sensors*. 2021;21(1):178.

[20] Chen W, et al. Cuffless continuous blood pressure estimation from pulse transit time and heart rate with adaptive calibration. *Biomed Signal Process Control*. 2020;57:101819.

[21] Xing X, Sun M. Optical blood pressure estimation with photoplethysmography and pulse transit time: a review. *Comput Biol Med*. 2016;65:124-136.

[22] Visit in august 2025, url: https://linksfoundation.com/chi-siamo/mission-vision/.

[23] Visited in August 2025 https://www.humanitas.it/enciclopedia/anatomia/apparato-cardiocircolatorio/

[24] Visited in August 2025 https://www.my-personaltrainer.it/fisiologia/sistema-cardiovascolare-cosa-si-intende-e-come-funziona.html

[25] ScienceDirect , visited in August 2025: https://www.sciencedirect.com/topics/immunology-and-microbiology/arterial-pressure

[26] Practical Issues of Hemodynamic Monitoring at the Bedside Author links open overlay panelPatricio M. Polanco MD a, Michael R. Pinsky MD, CM, Dr hc Division of Trauma, Department of Surgery, University of Pittsburgh School of Medicine, F1275 Scaife Hall, 3550 Terrace Street, Pittsburgh, PA 15261, USA Department of Critical Care Medicine, University of Pittsburgh School of Medicine,606 Scaife Hall, 3550 Terrace Street, Pittsburgh, PA 15261, USA November 2006. https://www.sciencedirect.com/topics/medicine-and-dentistry/mean-arterial-pressure

[27] Hypertensive Emergency Author links open overlay panelManish Suneja MD, FASN, M. Lee Sanders MD, PhD Division of Nephrology and Hypertension, Department of Medicine, University of Iowa Hospitals and Clinics, 200 Hawkins Drive, Iowa City, IA 52242, USA

April 2017. https://www.sciencedirect.com/topics/medicine-and-dentistry/mean-arterial-pressure

[28] Guide lines: Surviving Sepsis Campaign (2021) and studios on Critical Care Medicine

[29] Utility of different blood pressure measurement components in childhood to predict adult carotid intima media thickness. The i3C Consortium Study. Juha Koskinen 1,2, Markus Juonala 3,4,5, Terence Dwyer 6, Alison Venn 7, Janina Petkeviciene 8, Indrė Čeponienė 9, Lydia Bazzano 10, Wei Chen 11, Matthew A Sabin 12, Trudy L Burns 13, Jorma SA Viikari 3,4, Jessica G Woo 14, Elaine M Urbina 15, Ronald Prineas 16, Nina Hutri-Kähönen 17, Alan Sinaiko 18, David R Jacobs Jr 19, Julia Steinberger 20, Stephen Daniels 21, Olli Raitakari 1,22, Costan G Magnussen 1,7. feb 2020

[30] Visited in August 2025 - https://deltamed.pro/en/news/invasive-blood-pressure-monitoring

[31] ScienceDirect: Common errors in clinical measurement Ming Wilson MB ChB FRCA is a Consultant Anaesthetist at Salford Royal Foundation Trust, Manchester, UK. November 2017.

[32] ScienceDirect: Hemodynamic monitoring devices: Putting it all together Author links open overlay panelBhiken I. Naik M.B.B.Ch (Assistant Professor), Marcel E. Durieux M.D., Ph.D. (Professor) Department of Anesthesiology, University of Virginia, Charlottesville, VA 22908, USA. September 2014.

[33] National Library of Medicine: Cuff-Less Blood Pressure Prediction from ECG and PPG Signals Using Fourier Transformation and Amplitude Randomization Preprocessing for Context Aggregation Network Training Treesukon Treebupachatsakul 1, Apivitch Boosamalee 1, Siratchakrit Shinnakerdchoke 1, Suejit Pechprasarn 2, Nuntachai Thongpance 2, March 2022

[34] ScienceDirect : Cuffless blood pressure estimation from PPG signals and its derivatives using deep learning models Author links open overlay panel C El-Hajj, P.A Kyriacou Research Centre for Biomedical Engineering, at City, University of London, Northampton Square, London EC1V 0HB, United Kingdom. August 2021

[35] American Heart Association et al. "What is high blood pressure?" In: South Carolina State Documents Depository (2017).

[36] k. I. R. Rajni. "Electrocardiogram signal analysis". In: International journal of Computer Application (2013).

[37] "Photoplethysmography and its application in clinical physiological measurements". In: (2007).

[38] Mukkamala R. et al., "Toward Ubiquitous Blood Pressure Monitoring via Pulse Transit Time," IEEE Trans. Biomed. Eng., 2015.

[39] Gesche H. et al., "Continuous blood pressure measurement by using the pulse transit time: comparison to a cuff-based method," Eur. J. Appl. Physiol., 2012.

[40] Foo J.Y. et al., "Pulse transit time as an indirect marker for variations in cardiovascular related parameters: a review," J. Med. Eng. Technol., 2006.

[41] Hahn J.O. et al., "Cuff-Less and Continuous Blood Pressure Monitoring: A Methodological Review," IEEE Trans. Biomed. Eng., 2021. [5] Kim S.H. et al., "Robust regression approaches in cuffless BP estimation," Biomed. Signal Process. Control, 2019.

[42] Chen W. et al., "Estimation of blood pressure using photoplethysmography and ECG," IEEE EMBS, 2012.

[43] Bramwell J.C., Hill A.V., "Velocity of transmission of the pulse wave," Lancet, 1922.

[44] Zhang Q. et al., "Cuffless blood pressure estimation using pulse transit time and photoplethysmogram intensity ratio," IEEE EMBS, 2017.

[45] Sun S. et al., "Polynomial regression models for blood pressure estimation," Comput. Biol. Med., 2016.

[46] Li Y. et al., "Quadratic regression in cuffless BP monitoring," Sensors, 2018. [11] Foo J.Y., Lim C.S., "Evaluation of polynomial regression approaches for PTT-BP modeling," Biomed. Eng. Online, 2010.

[47] Rasmussen C., Williams C., Gaussian Processes for Machine Learning, MIT Press, 2006.

[48] Ding X. et al., "Continuous Cuffless Blood Pressure Estimation via Gaussian Process," Sci. Rep., 2017.

[49] Bian D. et al., "GPR with feature optimization for BP estimation," Physiol. Meas., 2019.

[50] Hahn J.O., "Critical review of GPR in biomedical signal processing," IEEE Rev. Biomed. Eng., 2020.

[51] Predicting blood pressure from physiological index data using the SVR algorithm Bing Zhang, Huihui Ren, Guoyan Huang, Yongqiang Cheng , Changzhen Hu. February 2019

[52] Hastie, T., Tibshirani, R., Friedman, J. (2009). The Elements of Statistical Learning. Springer.

[53] Kachuee, M., Kiani, M. M., Mohammadzade, H., Shabany, M. (2017). Cuffless Blood Pressure Estimation Algorithms for Continuous Health-Care Monitoring. IEEE Transactions on Biomedical Engineering.

[54] Gonzalez-Landaeta, R., Ramirez, B., Mejia, J. (2022). Estimation of systolic blood pressure by Random Forest using heart sounds and a ballistocardiogram. Scientific Reports, 12(1), 17196.

[55] A computationally efficient CNN-LSTM neural network for estimation of blood pressure from features of electrocardiogram and photoplethysmogram waveforms Author links open overlay panel Stephanie Baker a, Wei Xiang b, Ian Atkinson c , a College of Science and Engineering, James Cook University, Cairns, Queensland, 4878, Australia b School of Engineering and Mathematical Sciences, La Trobe University, Melbourne, Victoria, 3086, Australia Research Centre, James Cook University, Townsville, Queensland, 4811, Australia June 2022.

[56] Yen, C.-T., Chang, S.-N., Liao, C.-H. (2022). Estimation of Beat-by-Beat Blood Pressure and Heart Rate From ECG and PPG Using a Fine-Tuned Deep CNN Model. IEEE Transactions on Biomedical Engineering.

[57] S. Ma, Y. Wu, C. Peng, Z. Zhao, H. Wang, "Continuous blood pressure prediction based on GAF-driven multimodal PPG and ECG signal fusion," Medical Engineering and Physics, vol. 108, p. 104153, Aug. 2025

[58] Johnson A.E.W. et al., MIMIC-III, a freely accessible critical care database, Scientific Data, 2016.

[59] Zhu T. et al., A public dataset for wearable blood pressure monitoring from healthy subjects, IEEE DataPort, 2019.

[60] Burns A. et al., SHIMMER: A wireless sensor platform for non-invasive biomedical research, IEEE Sensors Journal, 2010.

[61] Wang, L., Xian, H., Guo, J., Li, W., Wang, J., Chen, Q., Fu, X., Li, H., Chen, Q., Zhang, W., Chen, Y. "A Novel Blood Pressure 366 Monitoring Technique by Smart HUAWEI WATCH: A Validation Study According to the ANSI/AAMI/ISO 81060-2:2018 367 Guidelines." Frontiers in Cardiovascular Medicine, vol. 9, 2022, p. 923655. DOI: 10.3389/fcvm.2022.923655.

[62] "Photoplethysmography and its application in clinical physiological measurements". In: (2007).

[63] Adrian Burns et al. "SHIMMER™: an extensible platform for physiological signal capture". In: 2010 annual international conference of the IEEE engineering in medicine and biology. IEEE. 2010, pp. 3759,3762.

[64] Visited in August 2025. url: https://www.medicalexpo.it/ prod/shimmer-sensing/product-107788-753934.html.

[65] Visited in August 2025. url: https://bmslab.utwente.nl/ wp-content/uploads/2019/12/Shimmer-ECG-heart-measurements. pdf.

[66] Visited in August 2025. url: https://www.medicalexpo.it/ prod/shimmer-sensing/product-107788-753612.html.

[67] Visited in August 2025. url: https://www.medicalexpo.it/ prod/shimmer-sensing/product-107788-753927.html.

[68] Robert J Ellis et al. "A careful look at ECG sampling frequency and R-peak interpolation on short-term measures of heart rate variability". In: Physiological measurement 36.9 (2015), p. 1827.

[69] Ahyoung Choi and Hangsik Shin. "Photoplethysmography sampling frequency: pilot assessment of how low can we go to analyze pulse rate variability with reliability?" In: Physiological measurement 38.3 (2017), p. 586

[70] Hsieh, C. H., Lin, S. H., Wu, Z. F., Hsu, Y. L. (2018). A linear regression model with dynamic pulse transit time features for noninvasive blood pressure prediction. Biomedical Engineering Online, 17(1), 1–15.

[71] Devi, S. S., Sairamya, N. J., Jacob, L. (2019). Real-time adaptive filtering of biomedical signals using LMS-based algorithms. Scientific Reports, 9, 16234.

[72] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the 14th International Joint Conference on Artificial Intelligence, 2, 1137–1143. Morgan Kaufmann Publishers Inc.

[73] Optimized Cuffless Blood Pressure Measurement Using ECG, PPG and Linear Regression: Francesca Boschi, Monica Visintin ,Valeria Figini ,Guido Pagana. LINKS foundation, Turin, Italy;

[74] Velislav N Batchvarov et al. "QT-RR relationship in healthy subjects exhibits substantial intersubject variability and high intrasubject stability". In: American Journal of Physiology-Heart and Circulatory Physiology 282.6 (2002), H2356–H2363.

[75] UmangYadav,Sherif N Abbas, and Dimitrios Hatzinakos. "Evaluation of PPG biometrics for authentication in different states". In: 2018 International Conference on Biometrics (ICB). IEEE. 2018, pp. 277-282.

[76] In: Visited in September 2025. url:https://bmslab.utwente.nl/wpontent/uploads/2019/12/Shimmer-GSR-Skin-conductivemeasurements.pdf.

[77] In: visited in september 2025 https://cordis.europa.eu/project/id/101129713/it, PERSONALIZED SUSTAINABLE SMART PATCH OMNIFICENCE, Horizon Europe

[78] In: Visited in september 2025: https://youtu.be/Gycd-8CHvf4?si=oVUJH55kkmFqvOTQ