Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

ANALYSIS OF HEART RATE VARIABILITY INFLUENCE ON HEART RATE TURBULENCE USING BOOSTED REGRESSION TREES IN ACUTE MYOCARDIAL INFARCTION PATIENTS.

Autor: Sandra Cantero Rabanillo

Tutor: Óscar Barquero Pérez

Curso académico 2016/2017

"I am in the right place, at the right time, doing the right thing" (L. Hay)

"When the roots are deep there is no reason to fear the wind" (Proverb)

Acknowledgements

Al primero al que quiero agradecer es a Pedro, porque sin él esto ni siquiera habría sido posible. Por estar ahí, por apoyarme siempre y por hacerme feliz.

A mi tutor, Óscar, por ayudarme incondicionalmente desde el principio, y por toda la paciencia que ha tenido conmigo desde el primer día que me dio clase. Ha sido muy fácil trabajar con alguien como él y me alegro de haberle escogido como tutor.

A mi familia y amigos, por haber formado parte de todo el proceso y por todos sus ánimos cada vez que lo he necesitado.

Y por último, a todos los amigos que he hecho durante mi paso por la universidad. Por todas las horas que hemos pasado juntos, que han sido muchas, y por toda la ayuda que me han brindado desde el principio. No cambiaría a ninguno de ellos.

Mil gracias a todos.

Resumen

La Turbulencia de la Frecuencia Cardiaca (TFC) es el término que describe las perturbaciones en la frecuencia cardíaca después de una Contraccion Ventricular Prematura (CVP) aislada. El patrón fisiológico de la TFC consiste en una breve aceleración de la frecuencia cardíaca (cuantificada por el denominado inicio de la turbulencia, turbulence onset (TO)) seguida de una desaceleración más gradual de la frecuencia cardíaca (cuantificada por la denominada pendiente de turbulencia, turbulence slop (TS)) antes de que el ritmo cardíaco vuelva a un nivel pre-ectópico. La TFC ha demostrado ser importante en la estratificación del riesgo después del Infarto Agudo de Miocardio (IAM).

La TFC suele evaluarse a partir de grabaciones Holter de 24 horas como respuesta media a las CVPs durante períodos más largos, ya que el promedio debe incluir un número suficiente de CVPs para la construcción fiable del tacograma promedio de CVP. A partir de este tacograma, se calculan TO y TS.

La Variabilidad de la Frecuencia Cardíaca (VFC) describe la variación de los intervalos entre latidos, y representa un método no invasivo para evaluar el estado de la actividad del Sistema Nervioso Autónomo (SNA). En la literatura científica se ha reportado una relación significativa entre el SNA y la mortalidad cardiovascular, incluida la muerte súbita cardiaca. La VFC se puede cuantificar utilizando índices temporales, denominados descriptores estadísticos y geométricos.

Algunos estudios han encontrado una relación entre el estado del SNA, a través de la VFC, y el patrón de la TFC, comparando las mediciones de los índices asociados a la VFC y a la TFC en grabaciones de Holter de 24 horas. Sin embargo, este análisis tiene algunos inconvenientes, ya que no tiene en cuenta las condiciones fisiológicas en las que se produce cada tacograma de CVP, debido al promedio calculado.

Proponemos en este TFG, modelar la TFC a través de TS y TO, utilizando Árboles de Regresión, Boosted Regression Trees (BRT), y los índices de la VFC como variables explicativas, para cada tachograma individual. En este caso, enriquecemos el modelo utilizando el sexo y la edad de los pacientes.

Los modelos de BRT son un método de conjunto, basado en la técnica de aprendizaje supervisada que predice valores de respuestas aprendiendo reglas de decisión derivadas de características. Los modelos de BRT combinan de forma progresiva un gran número de modelos de árboles simples para mejorar el rendimiento predictivo, evitando el sobreajustamiento

Los datos utilizados en este TFG provienen de un estudio prospectivo realizado en el Hospital Virgen de la Arrixaca de Murcia (España). Se incluyeron 61 pacientes con IAM. Se realizó una grabación Holter de 24 horas en pacientes con ritmo sinusal estable entre las semanas 2 y 6 después del infarto. El número medio de tacogramas de CVP por paciente fue de 50,7.

Los resultados mostraron que el modelo de BRT puede modelar la influencia de la VFC en la TFC, pudiendo identificar cómo influye el estado del SNA en TO y TS. Se ha encontrado que el control del SNA sobre la TFC disminuye entre pacientes enfermos, y no tiene control en absoluto en los pacientes de IAM de alto riesgo.

Abstract

Heart Rate Turbulence (HRT) is the term that describes the perturbations in heart rate after an isolated Ventricular Premature Complex (VPC). The physiologic pattern of HRT consists of brief heart rate acceleration (quantified by the so-called turbulence onset (TO)) followed by, a more gradual, heart rate deceleration (quantified by the so-called turbulence slope (TS)) before the rate returns to a pre-ectopic level. The HRT has been found valuable in risk stratification after Acute Myocardial Infarction (AMI).

HRT is usually assessed from 24-Hour Holter recordings as an average response to VPCs over longer periods, because the average needs to include a sufficient number of VPCs for reliable construction of the VPC averaged tachogram. From this tachogram, TO and TS are then calculated.

Heart rate variability (HRV) describes the variation in the heart rate and represents a noninvasive method to assess the state of Autonomic Nervous System (ANS) activity. In the scientific literature has been reported a significant relationship between the ANS and cardiovascular mortality, including sudden cardiac death. HRV can be quantified using time-domain indices, called statistical and geometric descriptors.

Some studies have found a relationship between the state of the ANS through HRV and the pattern of the HRT, comparing measures of both HRV and HRT indices over 24-Hour Holter recordings. However, this analysis has some drawbacks, as it does not take into account the physiological conditions in which the VPC tachogram is produced, due to the average procedure.

We propose, in this TFG, to model HRT, through TS and TO, using Boosted Regression Trees (BRT) and HRV time domain indices as explanatory variables, for each individual tachogram. We enrich the model using the sex and age of the patients.

BRT models are an ensemble method, based on supervised learning technique that predict values of responses by learning decision rules derived from features. BRT models adaptively comb large numbers of simple tree models to improve predictive performance, avoiding overfitting.

The data used in this TFG come from a prospective study carried out at the Hospital Virgen de la Arrixaca in Murcia (Spain). The number of patients included was 61 AMI patients. A 24-h Holter recording was performed in patients with stable sinus rhythm between weeks 2 and 6 post-infarction. The average number of VPC-tachograms per patient was 50.7.

Results showed that BRT can model the influence of HRV on HRT, being able to identify how the ANS status influence the TO and TS parameters. We have found that the control of the ANS on HRT decreases between sick patients, and has no control at all in AMI high-risk patients.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of this end-of-degree project is to model the influence of Heart Rate Variability (HRV) on Heart Rate Turbulence (HRT) using Boosted Regression Trees (BRT) in acute myocardial infarction (AMI) patients.

The goal is to analyze the relationship between the different variables that characterize the HRV and the ones that describe the HRT to explain the factors that influence on AMI patients.

Results from this TFG have been accepted to be presented in the International Conference Computing in Cardiology 2017, Rennes, France. See Appendix A

1.1 Motivation

HRT is a short term fluctuation in sinus cycle length that follows a Ventricular Premature Complex or Ventricular Premature Contraction (VPC). The HRT is a measure that quantifies the autonomically-mediated response to ventricular premature beats, an it is important because an abnormal HRT has been associated with an increased risk of cardiac death and it is a predictor of heart failure and mortality after AMI.

A VPC is a relatively common event, where the ventricles contract first and before the atria have optimally filled the ventricles with blood, which means that circulation is inefficient. A VPC interrupts the normal cardiac cycle, so the ventricles of the heart have not had time to fill up to their normal level, before contracting and pumping their content out. This results in a pulse (blood pressure) weaker than expected. HRT is the return to equilibrium of Heart Rate (HR) after a VPC, which have diminished

its normal level. Under normally and healthy conditions, HRT is part of the normal compensation process, and consists of a brief increase in HR after the VPC, that is immediately followed by a slower decrease in HR.

HRT can be estimated by two numerical descriptors: Turbulence Slope (TS) and Turbulence Onset (TO). TO is the amount of sinus acceleration following a VPC (first part of the compensation) and TS is the rate of sinus deceleration that follows the sinus acceleration (second part of the compensation) [32]

On the other part, experimental evidence have recognized a significant relationship between the Autonomic Nervous System (ANS) and cardiovascular mortality, including sudden cardiac death. Therefore, there is a necessity to develop quantitative markers of autonomic activity. In this sense, HRV represents one of the most promising such markers. Contrary to what one might think, the normal resting rhythm of the heart is highly variable rather than being monotonously regular. HRV is the variation in the time interval between heartbeats. It is measured by the variation in the beat-to-beat interval, and it is directly related to the body's interdependent regulatory systems and ultimately, their efficiency and health. It has been shown that low HRV is associated with some cardiac illness: myocardial infarction, atherosclerosis, heart failure and even with aging.

HRV analysis is usually performed on RR-tachograms, also known as interval tachograms, that are time series of the RR-interval durations as a function of the interval number. To measure the HRV there are several methods, but time-domain methods are the simplest ones on computational terms, and they are the ones that we are going to use in this study. They treat the RR-interval sequence as an unordered set of intervals and employ different techniques to express the variance of such data. They can be split into two categories: statistical descriptors, and geometrical descriptors.

The statistical descriptors can be divided in two groups: those derived directly from the RR-intervals, and those derived from the differences between adjacent RR-intervals. The geometrical descriptors on the other part, use the RR-intervals to construct a geometrical pattern and asses the HRV using a parameter of the pattern. They will be explained more in detail in next chapters [3, 20, 31].

Now that we know the individual importance of these two methods associated with heart diseases, we want to model the relationship between them (HRV and HRT) using non-parametric statistical learning model based on ensembled methods, namely, BRT.

In this work, BRT has been chosen as mean algorithm. In previous works, the models used to model HRT as function of the Coupling Interval (CI), the Compensatory Pause (CP) and the Sinus Cycle Length (SCl), were based on Linear Regression methods with non-linear polynomials, and the results have been interesting. In this TFG we are going to include a larger number of variables, since the current study includes the variables already mentioned and also those related to the HRV. For this

reason, we wanted to use a non-linear method that was flexible enough to reflects the complexity of the problem. Furthermore, we also wanted a model that allow us to perform inference analysis. Among these methods, BRT has proven to be very useful, with performance that can match with the more modern models, like Neural Networks (NN) and Support Vector Machines (SVM).

1.2 Objectives

To analyze the statistical relationship between the HRV and the HRT we are going to perform the following steps:

- Develop the tools to perform the HRT analysis on 24-hour Holter recordings.
- Develop the tools to perform the HRV analysis on 24-hour Holter recordings.
- Modeling the relationship between HRT and HRV in AMI patients using 24-hour Holter recordings.

1.3 Methodology and structure of the document

The methodology followed in this project consists of the following stages:

- HRT and HRV bibliographic review , as well as machine learning methods, mainly applications to aid diagnosis and prevention of myocardial infarction, in particular, ensemble methods.
- Compilation and exposure of information related to cardiac turbulence and cardiac variability, together with the description of the systems involved, and associated medical procedures. The techniques used for data collection is detailed, and the statistical technique used to analysis the data of this study, namely BRT, is described.
- Development of the required code in Python programming language, namely modules to:
  - Extract the data from the database.
  - Perform HRT and HRV analysis.
  - Preprocessing data for machine learning analysis.
  - Model data using ensemble methods.

- Presentation of the results obtained through different graphs, that serve as an explanation of the model obtained after the study, along with a descriptive exploratory analysis. Finally, the conclusions and the future lines of investigation are established.

The structure of the document is as follows:

- In Chapter 1 the rationale of this TFG is explained, as well as main objectives, and methodology.
- In Chapter 2 the basic functioning of the heart, specifically the electrical signal. In subsequent sections, the terms HRV and HRT are explained in detail, and the indices used to assess them are described. Finally, the theoretical foundations of ensemble methods, namely BRT, are discussed and developed in detail.
- In Chapter 3 the database analyzed in this TFG is described. In this Chapter, the objective of the study and the methodology followed are also discussed.
- In Chapter 4 the results of the TFG are presented and commented.
- Finally, in Chapter 5 conclusions of and future work are discussed.

Chapter 2

Background

In this chapter, a description of the basic functioning of the heart and the structure of the signal of a heartbeat is given in the two first subsections. After that, two different methods to assess the heart condition, HRV and HRT, are explained. Finally, BRT method are described, a statistical learning method that is the basis for modeling HRV and HRT in this TFG.

2.1 Basic anatomy and function of the heart

The heart is an organ whose main function is to propel the blood into the cardiovascular system. It is located between the lungs in the middle of the chest, behind and slightly to the left of the breastbone (sternum). The heart has four chambers. The upper chambers are called the left and right atria, and the lower chambers are called the left and right ventricles. The blood enters from the veins to the atria, and it is propelled into the arteries from the ventricles [21].

The cardiovascular system delivers the oxygen and nutrients to the cells and removes the carbon dioxide and waste products made by those cells. This system has to assure that an adequate flow of blood reaches the organs, conducting this exchange correctly.

The cardiovascular system consists on two separate circulatory subsystems: the systemic and the pulmonary. The heart connects both subsystems acting as a double pump, one for each circulatory subsystem. Thus, the circulatory and respiratory systems work together to sustain the body with oxygen and to remove carbon dioxide. The right side of the heart receives the blood from the systemic circulatory system and pumps it into the pulmonary circulatory system, whereas the left side of the heart receives the blood from the pulmonary system and pumps it into the systemic sys-

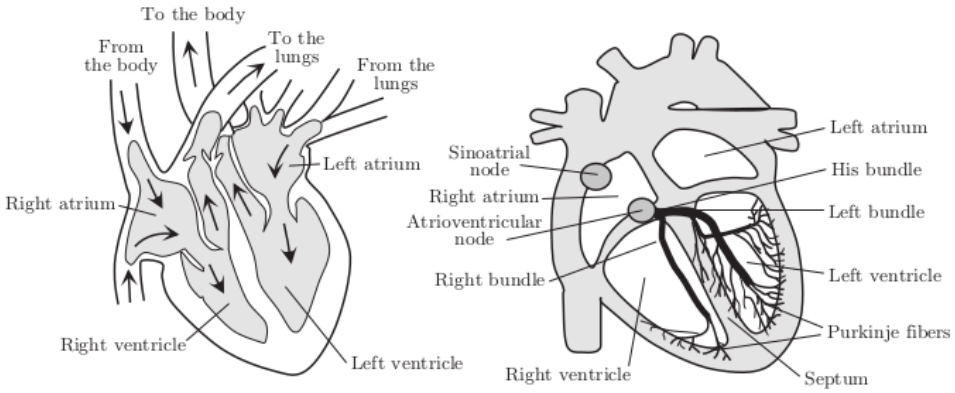


Figure 2.1: Anatomy of Heart, taking from [29].

tem [21].

Pulmonary circulation carries the deoxygenated blood flows into the lungs. In the pulmonary loop, deoxygenated blood exits the right ventricle of the heart and passes through the pulmonary trunk. The pulmonary trunk splits into the right and left pulmonary arteries. These arteries transport the deoxygenated blood to the alveoli in the lungs. There, carbon dioxide is released and oxygen is absorbed. Oxygenated blood then passes from the alveoli into the pulmonary veins. The pulmonary veins transport it to the left atrium of the heart [13].

The systemic circulation is responsible for returning that blood with oxygen and nutrients to the various muscles and organs. In the systemic loop, oxygenated blood is pumped from the left ventricle of the heart through the aorta, the largest artery in the body. The blood moves from the aorta through the systemic arteries to supply body tissues. Here, oxygen and nutrients are released and carbon dioxide and other waste substances are absorbed. Deoxygenated blood then moves into the systemic veins. The systemic veins feed into the inferior and superior vena cava, the largest veins in the body. The vena cava flow deoxygenated blood to the right atrium of the heart [13]. Finally, this deoxygened blood returns back to the heart and into the pulmonary circulation again, performing a complete cycle on the cardiovascular system [21].

2.2 Electrical signal of the heart

The heart pump the blood by cardiac muscle contractions. These contractions are triggered by the propagation of an electrical impulse through the heart muscle (myocardium). To synchronize these cardiac contractions it is necessary that the electrical impulses spread quickly and in an organized way throughout the electrical conduction system of the heart. The graphic representation of the electrical activity of the heart

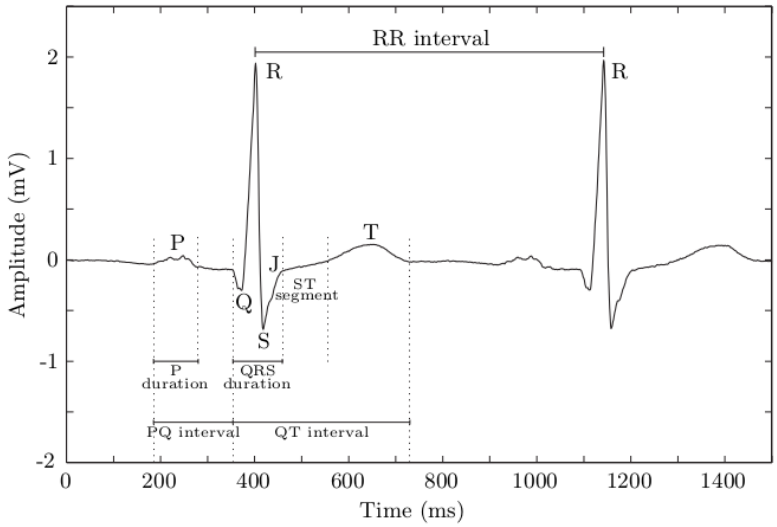


Figure 2.2: Complex QRS, taking from [29].

as a function of time is called Electrocardiogram (ECG) [21].

Each of these electrical impulses begins in a group of cells called the sinus node or sinoatrial (SA) node. The SA node is located in the right atrium, the upper right chamber of the heart. From the SA node, the signal travels through the right and left atria, causing the atria contract and pump the blood into the lower chambers of the heart, the ventricles. The electrical signal moving through the atria is recorded as the P wave on the ECG.

Between the way through the atrial and the ventricles, there is a group of cells called the atrioventricular (AV) node where the electrical signal passes and slows down. This delay is introduced for two reasons. First, it lets a complete atrial depolarization-contraction. Second, it allows the ventricles to finish filling with blood. This part of the process is represented with the line that joins the end of the P wave and the beginning of the Q wave on the ECG.

After the AV node there is a specialized group of heart muscle cells known as the bundle of His. The bundle of His helps to communicate a single rhythm of contraction to all parts of the heart, which provides synchrony. When the electrical signal leaves the AV node, travels along the bundle of His into the right and left bundle branches. Then, the impulse is conducted by the Purkinje fibers at high velocity throughout the ventricles, causing them to contract and move the blood out to the body. This process is recorded as the QRS waves on the ECG. The R wave, the largest one, reflects depolarization of the main mass of the ventricles and also the atrial repolarization.

After the contraction of the ventricles, a period of zero potential between ventricular depolarization and repolarization occurs. It is represented with the ST segment, which is also known as the ST interval, and it is the time between the end of the QRS complex and the start of the T wave. The ventricles then recover their normal electrical state,

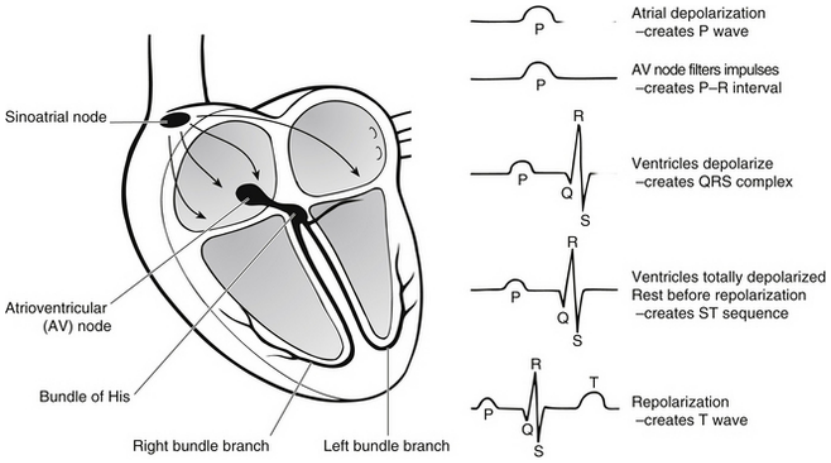


Figure 2.3: Heartbeat process, taking from [7].

which correspond to the T wave on the ECG [8]. The muscle stops contracting to allow the heart to refill with blood. This entire process continues over and over with each new heartbeat [16].

2.3 Heart Rate Variability

In 1965, when Hon and Lee appreciated that in fetal distress cases, before any other appreciable change, occurred alterations in the time between heartbeats intervals previous to the complication, finding a significant relevance of HRV in some heart disorders. Ten years later, Sayers and others looked for the existence of physiological rhythms contained in the beat-to-beat HR signal. The clinical importance of HRV became apparent in the late 1980s when it was confirmed that HRV was a strong and independent predictor of mortality following an AMI [31].

HRV is a term used to describe the phenomenon of the variation in the duration between consecutive heartbeats. The beat-to-beat interval is defined as the time, usually expressed in milliseconds, between normal R to R waves on an ECG, or simply RR interval. HRV is related to the regulation of the ANS on the sinus node, and can evaluate both cardiac health and the state of the ANS responsible for cardiac regulation [23]. A growing number of studies indicate that increased variability in the heart's interbeat interval is physiologically desirable [31].

According to Harald M. Stauss [30], in healthy subjects, the SA node located at the posterior wall of the right atrium initiates each beat of the heart. Due to the unstable membrane potential of the myocytes located in this region, action potentials are generated periodically at a fairly constant frequency. This relatively constant frequency generated by the autorhythmicity of the SA node is modulated by many factors that add variability to the HR signal at different frequencies. When this variability

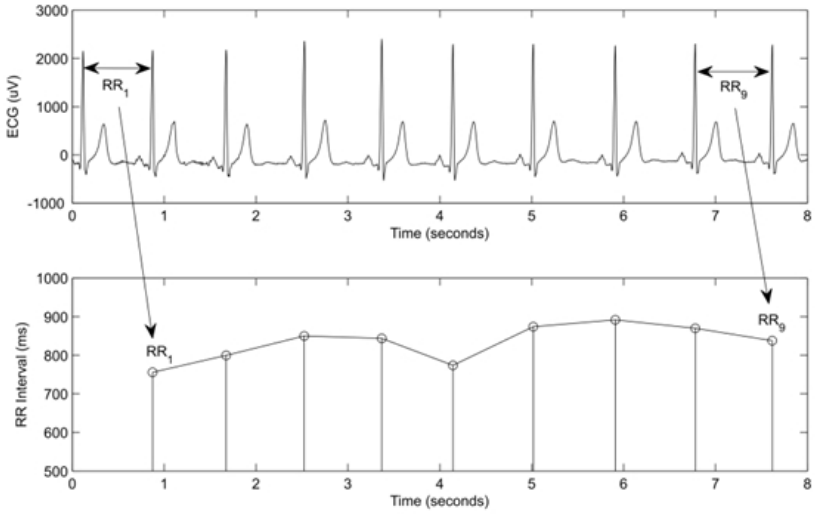


Figure 2.4: Heart Rate Variability, taking from [9].

decreases, we can predict a high probability of heart failure and even mortality in case of myocardial infarction.

Short-term and long-term variations in HR have different physiological origins, and the magnitude of that variation is an indicative of the state of the heart and the autonomic control response. When the autonomic control response is reduced, the HRV reflects this control loss and it makes possible the classification of cardiac sudden death risk groups. Therefore, assessing the HRV would allow to characterize different cardiovascular states, being a noninvasive alternative [20].

2.3.1 HRV measurements

To measure the HRV, we can find several methods. In this study we are going to focus on time domain methods, which are the easiest to perform, and allow us to determine either the HR or the value of the RR-interval at any time. The time domain measures are computed directly on the RR-interval or on the difference time series. To obtain reliable measures of the HRV, long-term recordings, like 24-hour Holter recordings, seem to be appropriate options to calculate the different parameters of HRV. A segment of the HRV signal, commonly a five minutes segment, has also been found to be clinically valid and meaningful. It is important to not compare estimates from segments of different durations, because the results will not be valid as variability is significantly influenced by the length of the signal that is sampled [31]. The time domain methods usually split in two categories, namely, statistical methods and geometrical methods.

Statistical methods

Descriptor	Units	Description
SDNN	ms	Standard deviation of all normal sinus RR intervals
SDANN	ms	Standard deviation of the averages of all normal sinus RR intervals in all 5-minute segments
AVNN	ms	Mean of all normal sinus RR intervals
SDNNindex	ms	Mean of the standard deviations of all normal sinus RR intervals for all 5-minute segments
RMSSD	ms	Root-mean-square of successive normal sinus RR interval difference
NN50		Number of pairs of adjacent NN intervals differing by more than 50 ms
pNN50	%	Percentage of successive normal sinus RR intervals ¿50 ms (percentage of NN50)

Table 2.1: Statistical descriptors of HRV

Table 2.1 shows the time domain measurements to assess the HRV and a brief description.

Time domain measurements are easy to calculate and do not need a high computational load. However, they are very sensitive to noise that affect the signal, specially in long-term recordings, because they are not record in controlled conditions (i.e. 24-hour Holter recordings). The variables derived from the difference between consecutive intervals are associated with short-term variations or high frequencies variations in HR [20]. While the first four variables (SDNN, SDANN, AVNN, SDNNindex) are calculated regardless of the order of the intervals, the last three variables are calculated based on successive intervals (RMSSD, NN50 and PNN50). In this last case, the RMSSD method is preferred to pNN50 and NN50 because it has better statistical properties [31].

Geometrical methods

Geometrical descriptors allow us to work with the RR signal with a robust results in spite of the noise polluting the RR data sequence, unlike statistical descriptors. In geometric methods, the RR-interval is converted into a geometrical pattern whose properties will help us to measure the HRV. There are three basic approaches in geometrical methods [20], namely:

Descriptor	Units	Description
Triangular index	ms	Total number of all NN intervals divided by the maximum of the density function (height of the histogram of all NN intervals)
TINN	ms	Base width of the minimum square difference triangular interpolation of the highest peak of the histogram of all NN intervals
Lorenz plot scattering	ms	Representation of each NN-interval duration versus the duration of the previous interval
Differential index	ms	Difference between the widths of the histogram of differences between adjacent NN intervals measured at selected heights
Logarithmic index		Coefficient ϕ of the negative exponential curve $Kexp^{-\phi t}$ which is the best approximation of the histogram of absolute differences between adjacent intervals

Table 2.2: Geometrical descriptors of HRV

1) The HRV is assessed by some geometrical measurements from the used pattern (e.g. the width of the distribution histogram at the specified level).

2) The geometrical pattern is interpolated with a mathematically defined shape, and the parameters of the shape are used as HRV measures (e.g. the distribution of the histogram is approximated by a triangle, or approximation of the differential histogram by an exponential curve).

3) The general pattern of the geometrical form is classified into one of several predefined categories, which represents different classes of HRV (e.g. linear and triangular shapes of Lorenz plots, elliptic).

Table 2.2 shows the time domain measurements to assess the HRV and a brief description.

The major disadvantage in geometric methods is that they are inappropriate to assess short-time changes in HRV, for what we need a reasonable number of samples to construct the geometric pattern, that is why 24-hour holter recordings are commonly used [31].

The code developed in this TFG to preprocess the RR-intervals signals, as well as to compute HRV indices (statistical and geometric), explained in this section, is presented in Appendix D.

2.4 Heart Rate Turbulence

The term HRT refers a short-term fluctuation in beat-to-beat length after a spontaneous VPC, which can be naturally produced or induced artificially by stimulation. HRT is the process in which the body returns the HR to its normal state after the heart has pumped the blood out too soon without having fully filled the ventricles prior to ejection. This causes that the quantity of blood that reaches the body to be less than expected, and the time between heartbeats after this ectopic one is also lower than expected. In healthy subjects, the brain detects this low release of blood and to compensate it, the SNA speeds speeds up the HR to pump more blood (early acceleration). However, this over-compensation increases blood pressure causing an another reaction, that decreases HR this time (late deceleration) until blood pressure returns to normality [22]. A skipped or abnormal step in the HRT pattern, has been associated with an increased risk of cardiac death and it is a predictor of heart failure and mortality after AMI.

After a VPC, a CP occurs, in which the heart skips one beat (that is to say, the time between the ectopic beat and the next is the time between two normal beats, and this is called a complete CP). After this CP, there is a strong contraction in which the heart pumps more blood (early acceleration) and then a deceleration caused by the CP and the increase of the blood pressure.

2.4.1 HRT measurements

To measure the phenomenon of a HRT, we use VPC-tachograms. A tachogram is a sequence of RR-intervals (intervals between heartbeats) which contains a VPC and picks up the HRT pattern. These sequences include the 5 RR-intervals before VPC, the CI, that is, the VPC itself, the VPC CP, and at least 15 subsequent RR-intervals [22]. It is important to note that to measure correctly the HRT, we need an averaging response of a significant number of VPCs, found in long-term recordings like Holter's recordings. That is because the HRT pattern can be masked by HRV or other factors when it is been measured with isolated VPCs [22]. The usual procedure is to average all VPCs tachograms in order to improve the signal-to-noise ratio and remove background effects.

To make accurate HRT measurements, we need to remove inadequate VPCs from

the recordings before calculating the average of the available VPCs tachograms. To assure a tachogram is valid, it needs to fulfill the following conditions:

- Either the five sinus beats preceding the VPC or the 15 sinus beats following the CP will not include artifacts, some arrhythmia or false classifications.
- 300ms $\leq$ RR-intervals $\leq$ 2000ms.
- The difference between consecutive RR-intervals cannot be higher than 200ms.
- The difference between any RR-interval and the reference interval (mean of the five sinus intervals preceding the VPC) cannot be higher than 20%.
- Prematurity has to be higher than 20% of the reference interval.
- The CP has to be higher than 20% of the reference interval [21].

The two phases of HRT, the early sinus rate acceleration and late deceleration, are quantified by two numerical descriptors, TO and TS respectively. These parameters allow us to quantify the HRT, namely, the initial HR jump (TO) and the rate at which HR returns to normal situations (TS). [22].

To calculate the TO, we use the following expression:

$$TO = \frac{(RR_1 + RR_2) - (RR_{-2} + RR_{-1}) \cdot 100}{RR_{-2} + RR_{-1}}[\%] \tag{2.1}$$

where RR_{-2} and RR_{-1} are the two RR-intervals immediately preceding the VPC coupling interval, and RR_1 and RR_2 are the two RR-intervals immediately following the compensatory pause. The unit of measure of TO is a percentage, that is, %.

TS is defined as the maximum positive regression slope assessed over any 5 consecutive sinus rhythm RR-intervals within the first 15 sinus rhythm RR-intervals after the VPC [22]. TS always needs to be calculated from the averaged tachogram to decrease the noise associated to the Holter recording. Averaging also decreased the TS value with increasing number of VPCs [22]. The unit of measure of TS is millisecond/beat.

The values of TO and TS can be roughly divide into two categories, namely, TO$<$0 and TS$>$2.5 are considered normal values, and TO$\geq$0 and TS$\leq$2.5 are considered abnormal values. In other words, strong sinus acceleration followed by rapid deceleration marks a healthy response [32].

The code developed in this TFG to preprocess the VPC tachograms from the holters recordings, as well as to compute both TO and TS is presented in Appendix C.

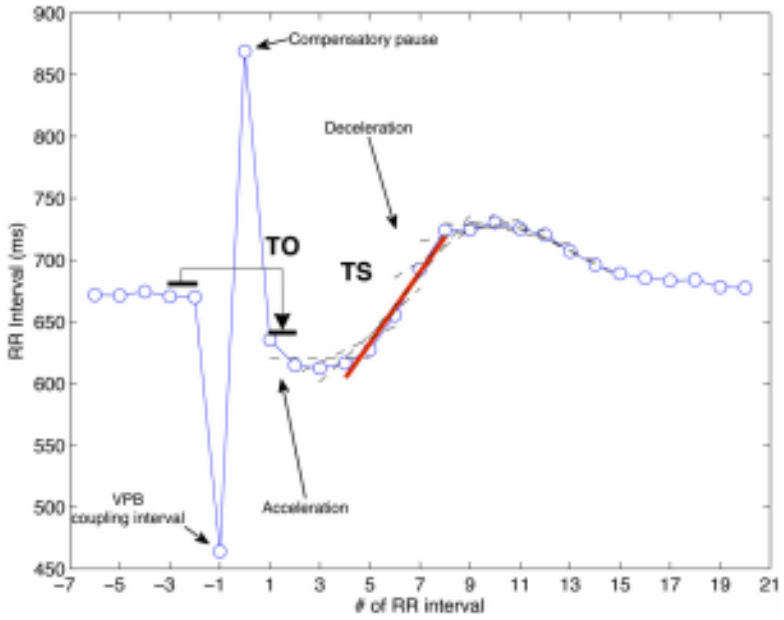


Figure 2.5: Tachogram with TO and TS, taking from [25].

2.4.2 Factors affecting HRT

There are several factors that affect the HRT. For example, aging is associated with a decrease in HRT.

Gender does not influence HRT in healthy control patients or post-infarction patients. But increasing age is associated with a decrease in HRT [22].

HR previous to the VPC has been also found to influence the HRT. At a hight HR, HRT is reduced, but the mechanisms responsible are not completely understood. The observations of previous reports leads to the possibility of correcting HRT indexes for HR, but it is not available data for such a correction [22].

The effects of VPC coupling interval on HRT reflect conflicting results in different studies. On one hand, a relationship between VPC coupling interval and HRT parameters was observed only individually but not in the pooled population. On the other hand, the correlation between TS and TO and normalized Coupling Interval (CI) was negative and positive, respectively, in full agreement with HRT physiologic background [22].

While some studies have found that prematurity of VPC was linearly correlated only with TO, but TS was not affected at all, other studies have reported strong linear correlations of both TO and TS with prematurity of VPC. Contradictorily, different studies have not found correlation between HRT parameters and prematurity of VPC at all [21].

The origin of the VPC has no influence on HRT, but there are other factors (like retrograde atrial depolarization, which may reset the sinus node) that can may change the dynamics of subsequent RR-intervals [22].

2.5 Boosted Regression Trees

2.5.1 Decision Trees

BRTs are based on decision trees, which are a supervised learning technique that predict by learning decision rules derived from features. They can be applied to both regression and classification problems.

2.5.2 Regression Trees

When the data has lots of features which interact in complicated, nonlinear ways, assembling a single global model can be very difficult, and very confusing to interpret. To work with this type of data, we need an alternative approach. Instead of work with all data, we can sub-divide, or partition, the space into smaller regions, and do it again until each region are small enough to be able to fit simple models for them. This is called recursive partitioning. The global model thus has two parts: the recursive partition, and the application of a simple model for each cell of the partition [15].

The recursive partition is represented in the Decision Tree. The interior nodes are labeled with questions, with a 'yes' or 'no' answer (i.e. "AVNN $<$ 50?" or "IsDone == FALSE?") and each question refers to only a single attribute. Which question we ask next depends on the answers to previous questions. These nodes derive in two branches, one corresponding to the 'yes' response, and the other one corresponding to the 'no' response. Each of the terminal nodes or leaves represents the values in data that fall in the corresponding cell of the partition [15]. The process of building a regression tree is divided in two steps:

1) We divide the predictor space, that is, the set of possible values for each observation, into j distinct and non-overlapping regions, R_1,R_2,...,R_j.

2) For each observation x that falls into a particular partition R_j we predict a value that correspond to the estimated response given by the mean of the training observations in the partition. That is, if we have a region R_1, where the mean of the training observations is 77, the predicted value for the observation $x_1 \in R_1$ will be

77 [27].

All decision tree algorithms have the same structure. Basically, it's a greedy divide and conquer algorithm. However, this process does not actually describe how to choose the questions to form the partitions or regions. In theory, the regions or boxes could have any shape. The goal is find the regions R_1,...,R_j that minimize the Residual Sum of Squares (RSS), given by

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{2.2}$$

First we sum across all the partitions of the feature space (first summation) and then we sum across all test observations (indexed by i) in a particular partition (second summation). We then take the squared difference of the response y_i of a particular testing observation with the mean response $\hat{y}_{R_j}$ of the training observations within the j_{th} region.

Unfortunately it is expensive computationally to consider every possible partition of the feature space into J regions. A less demanding method is the Recursive Binary Splitting (RBS), which is a top-down approach because it begins at the top of the tree (at which point all observations belong to a single region) and then, successively splits the predictor space into two new branches further down on the tree. At each step, it chooses the split of the features that minimizes the current RSS, at that particular step. This makes it a greedy algorithm, because it chooses the best split for each iteration of the recursion, rather than looking ahead and continuing to branch before making the evaluations. This is the reason that makes it computationally feasible and practical to use. The process continues splitting into regions until a stopping criterion is reached. A stopping criterion commonly used is the maximum number of observations contained in a region [15, 27].

There are several advantages of using decision trees. The most important are the follow:

- By construction, decision trees produce interpretable if-then-else decision rule-sets, which are easy to understand. Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Tree-based methods are simple and useful for model interpretation.
- Trees can easily handle qualitative predictors without the need to create dummy variables.
- Tree models are able to scale effectively on large datasets.

- There are fast, reliable algorithms to learn these trees.

However, there are also some disadvantages. The most relevant are the follow:

- Trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches.
- Trees can be very non-robust, a small change in the data can cause a large change in the final estimated tree. They are high variance estimators.

The disadvantages of decision trees can be significantly reduced by aggregating many decision trees. They are extremely competitive when utilized in an ensemble setting [15, 27].

2.5.3 Ensemble methods

Ensemble methods refer to a collection of methods by which final predictions are made by aggregating predictions from a number of individual models. The goal of ensemble methods is to combine the predictions of several base estimators in order to improve generalization. Ensemble methods use trees as building blocks to construct more powerful prediction models, via bagging, random forests or boosting. Two families of ensemble methods are usually distinguished:

Averaging methods, like bagging or random forests, where the driving principle is to build several estimators independently and then to average their predictions. The combined estimator is usually better than any of the single base estimator because its variance is reduced.

By contrast, in boosting methods, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble [28].

2.5.4 Boosted Regression Trees

BRT is one of several techniques that aim to improve the performance of single models by fitting and combining many models. BRT creates an ensemble of regression trees using boosting, which improves accuracy.

In boosting, the trees are grown sequentially: each tree is grown using information from previously grown trees. Thus, each tree is dependent on prior trees, and learns by fitting the residual of the trees that preceded it. For BRT, the first regression tree is the one that, for the selected tree size, maximally reduces the loss function. For each following step, the focus is on the part of the response variable that what not explained by the previous tree. Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm. The process is stagewise (not stepwise), meaning that existing trees are left unchanged as the model is enlarged. Only the fitted value for each observation is re-estimated at each step to reflect the contribution of the newly added tree. Boosting does not involve bootstrap sampling; instead each tree is fit on a modified version of the original data set [24, 27].

The boosting approach learns slowly. By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well. The parameter λ, known as learning rate, slows the process down even further, allowing more and different shaped trees to attack the residuals. Note that in boosting, the construction of each tree depends strongly on the trees that have already been grown [27].

Boosting involves combining a large number of decision trees, $\hat{f}_1$,..., $\hat{f}_B$. The algorithm is the follow [27]:

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For b = 1, 2,..., B, repeat:
   a) Fit a tree $\hat{f}^b$ with d splits (d+1 terminal nodes) to the training data (X, r).
   b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$
   c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$
3. Output the boosted model,
   $$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x) \tag{2.3}$$

2.5.5 Configure a Boosted Decision Tree Regression Model

Boosting has several tuning parameters to particularize our model to our preferences. To configure the BRT model we can modified the following parameters:

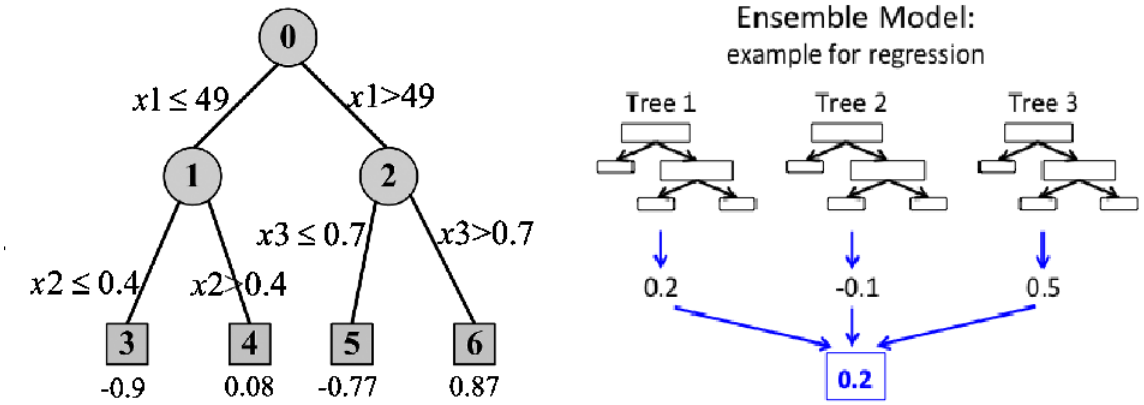


Figure 2.6: BRT example, taking from [5, 6].

1. Loss function, namely, least squares regression, least absolute deviation, quantile regression.

2. Number of trees, B, in the ensemble. By creating more decision trees, we can potentially get better coverage, but training time will increase. Boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all.

3. Depth of the individual regression estimators. The maximum depth, or the number d of splits in each tree, limits the number of nodes in the tree and can improve performance. Often $d = 1$ works well, in which case each tree is a stump, consisting of a single split. In this case, the boosted ensemble is fitting an additive model, since each term involves only a single variable. More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables. The best value depends on the interaction of the input variables.

4. Maximum number of leaves per tree, indicate the maximum number of terminal nodes (leaves) that can be created in any tree. By increasing this value, we potentially increase the size of the tree and get better precision, at the risk of overfitting and longer training time.

5. Minimum number of samples per leaf node, which indicates the minimum number of samples required to create any terminal node (leaf) in a tree. By increasing this value, we increase the threshold for creating new rules.

6. Learning rate, a positive number between 0 and 1 that defines the step size while learning, usually denoted by λ. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. The learning rate determines how fast or slow the learner converges on the optimal solution. If the step size is too big, it might overshoot the optimal solution. If the step size is too small, training takes longer to converge on the best solution. Very small λ can require using a very large value of B in order to achieve good performance.

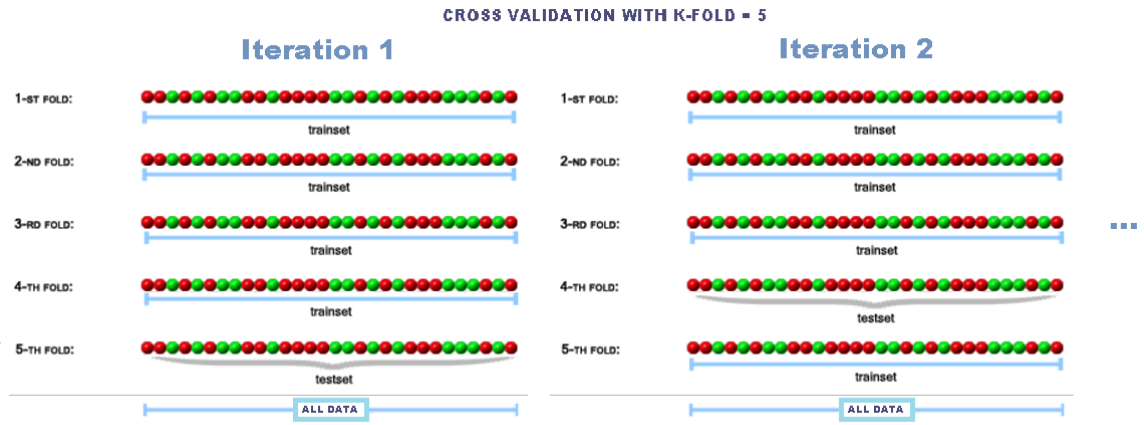


Figure 2.7: K-Fold Cross-Validation procedure.

7. Maximum number of features to consider when looking for the best split. When choosing `maximum_features` < `total_number_of_features` leads to a reduction of variance and an increase in bias. The search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `maximum_features` features [28].

2.5.6 Selection of hyperparameters using Cross Validation

We have already talk about the parameters to configure a BRT model. Now, we are going to see how to choose the optimal values for those parameters in our model. For example, to choose the appropriate number of trees to avoid overfitting, we need to find the optimal value for it. A way of performing hyperparameter optimization is known as *grid search*, which is an exhaustive searching through a specified subset of the hyperparameter space of the learning algorithm. This search is typically done using k-fold cross-validation (K-fold CV) on the training set.

K-fold CV is a model validation technique, used to estimate how accurately a predictive value will perform in practice. K-fold CV is useful when the size of the data is small, and it is not possible to partition the data set into two sets, one set for training and another one for validation. The procedure begin with a randomly partition of the original data into k equal sized subsamples. A common value for k is 5 or 10. Once the dataset is divided into k folds, one of them is chosen as validation set, and the other k-1 folds are used as training set. This procedure is repeated k times until every fold has been chosen once as validation set. Then, the k results obtained can be averaged to produce a single estimation.

In this case, we are going to use *grid search* with a range of possible values for each

parameter. For example, if we want to select the optimal value for the minimum number of samples per leaf node, we can specify a range of values, e.g. $[2, 3, 4]$. The *grid search* technique will use K-fold CV for each of the values in the range, obtaining an error for each, and it will select the set of hyperparameters with lower validation error. There is a trade-off between computantion time and fine grid search of hyperparamenters [18,19].

Chapter 3

Database Description and Experimental Setup

In this section we are going to explain the database used for the current study. Before that, a brief explanation of the medical terms included in the description of the database is given.

3.1 Acute Myocardial Infarction

AMI is a life-threatening condition that occurs when blood flow to the heart is abruptly cut off, causing tissue damage. This is usually the result of a blockage in one or more of the coronary arteries. When these arteries become blocked or narrowed the blood flow to the heart decreases significantly or stop completely. Certain factors may increase the risk of having an AMI: high blood pressure, high cholesterol levels, high triglyceride levels, diabetes and high blood sugar levels, obesity, smoking, age, family history [12].

AMIs require immediate treatment. Angioplasty is a common treatment that restores the normal flow of blood to the heart. An angioplasty is a percutaneous coronary intervention that allows to open partial or totally blocked arteries that supply blood to the heart. All that is needed is to make a small puncture in an arm or a leg and introduce a thin tube called catheter to unblocked the artery by inflating a small balloon placed at the end of the thread. Sometimes, to ensure the artery remains open, the specialist introduce a stent, which is a small, mesh-like device made of metal which acts as a support, or scaffold, and leaves it inside the artery [4, 17].

The chances of recovering from an AMI depend on how much damage there is to the

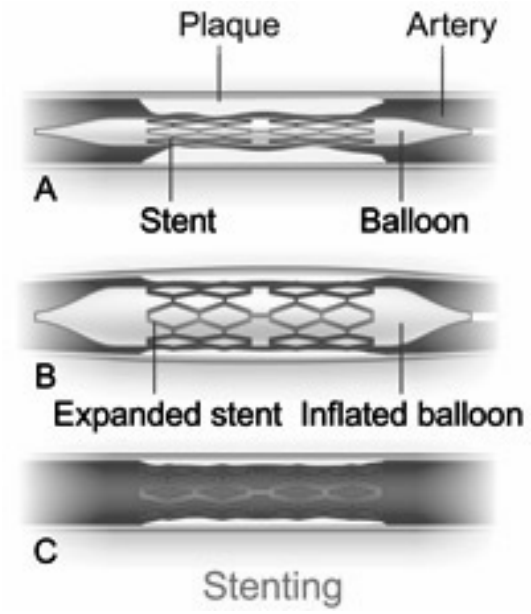


Figure 3.1: Angioplasty procedure, taking from [2]

heart and how quickly the patient receive emergency care. Heart damage also increases the risk of developing arrhythmias, which are abnormal heart rhythms, namely, a change in either the speed or pattern of the heartbeat. To monitor these events the specialist may use a Holter recording, which is a long-term ambulatory ECG. Holter monitoring has been proved to be an effective tool for assessing the frequency and duration of arrhythmias, particularly in patients who are post-AMI [12].

3.2 24-Hour Holter recordings

Sometimes symptoms may not take place during a routine heart examination, or a control is required after a heart problem (like after an AMI), so the specialist needs the patient to be monitored over a 24-hour period.

A Holter monitor is a small machine that continuously records the ECG over 24 hours, or longer periods. Holter monitoring can help with the diagnosis of many conditions affecting the heart and lungs.

To use a Holter the specialist needs to place several sticky electrode patches (small conducting patches) onto the patient's chest. The electrodes are connected with a wire to a portable recording machine (the Holter monitor) which will pick up and record electrical signals produced by every heartbeat [1, 11].

When the recording of ECG signal is finished it is up to the specialist to perform the analysis. The success of the analysis is very closely associated with the record quality.

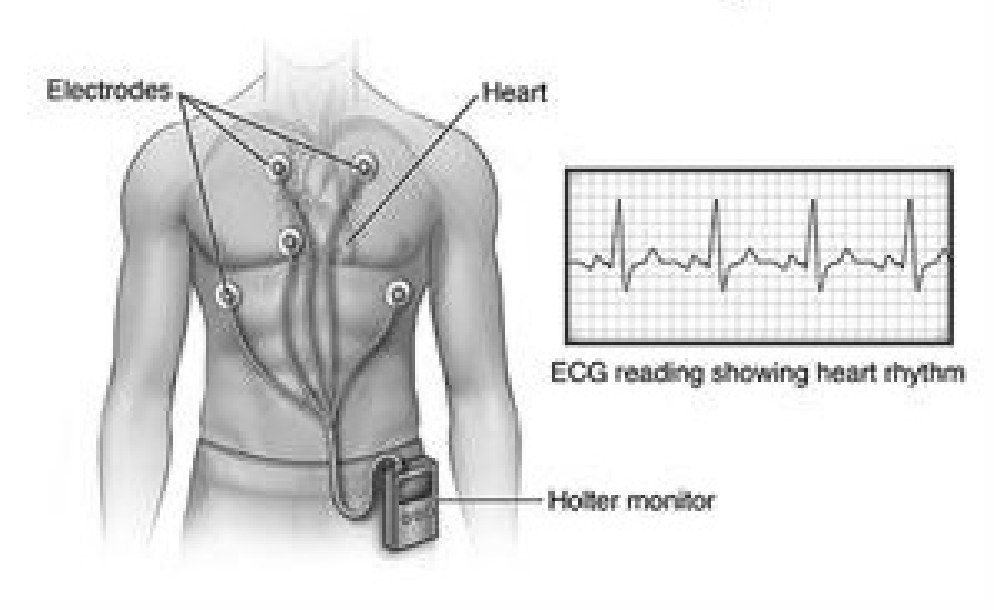


Figure 3.2: Holter monitor with ECG reading, taking from [10].

The quality of the recording depends on different factors, such as muscle tremors, the presence of noise, artifacts or false beat detections, which are a common problem in Holter recordings [21].

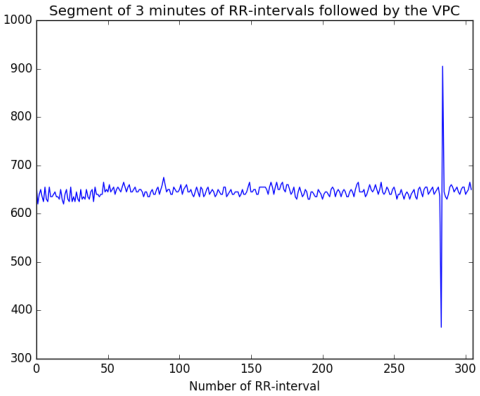
The code developed in this TFG to read Holter recordings and extract RR-interval time signals as well as beat annotations, is presented in Appendix B.

3.3 Database and Experimental Setup

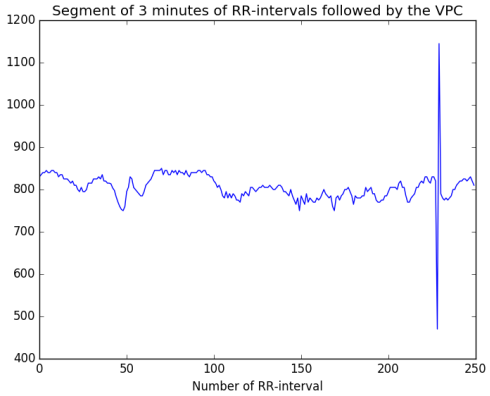
The data used in this TFG come from a prospective study carried out at the Hospital Virgen de la Arrixaca in Murcia (Spain) to evaluate the impact of primary angioplasty on the indication for implantable defibrillator in patients with AMI. A total of 102 post-infarction patients (80 men, age 63.6 $\pm$ 11.5 years) included in a regional primary angioplasty program were studied [26].

The final number of patients included in the study of this TFG was 61 AMI patients, due to several reasons: missing data, low quality of the signal, no presence of the VPC, etc. A 24-h Holter recording was performed in patients with stable SR between weeks 2 and 6 post-infarction. Only patients with at least one VPC during the monitoring period were included in the analysis. The average number of VPC-tachograms per patient was 50.7 (median 12, rank 1-474) [26].

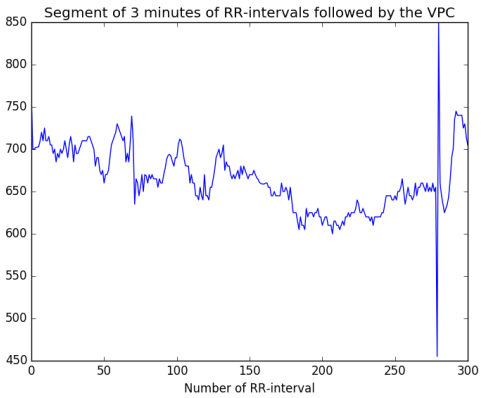
The main objective of this TFG is to model the relationship between HRV and HRT. In order to do that, we analyzed the holter recordings to select individual VPC



(a) Low variability



(b) Medium variability



(c) High variability

Figure 3.3: Segment of RR-intervals of three minutes followed by the VPC tachogram

tachograms. HRV measurements were computed on 3-minutes segments before each valid VPC-tachogram, namely time domain indices. For each VPC the CI, the CP, and the HR prior to the VPC (SCL) are also considering as features of the HRT, which assessed by TS and TO parameters. The aim is to build a model to predict TS/TO using as explicative variables: HRV indices, CI, CP, SCL, age and sex:

$$\hat{TS} = \hat{f}(\boldsymbol{x}) \tag{3.1}$$

$$\hat{TO} = \hat{f}(\boldsymbol{x}) \tag{3.2}$$

where,

$\boldsymbol{x}$ = [SDNN,TriangIndex,SDSD,TINN,AVNN,LogIndex,RMSSD,SCL,CI,CP,Sex,Age]

We propose to model $\hat{f}$ using BRT.

We want to verify the hypothesis that the influence of the ANS, as measured by HRV, on HRT responses is different regarding the condition. So to verify that we split the dataset according to the HRT status: [22].

1. HRT category 1 means TO and TS are normal. TS and TO cutoff values are commonly used in most clinical studies, where TS $>$ 2.5 ms/RR-interval and TO $<$ 0 are considered normal. One subset comprised patients with normal averaged HRT parameters values, hereafter called AMI low-risk:

   TS $\geq$ 2.5 ms/RR-interval and TO $\leq$ 0%

2. HRT category 2 means either TO or TS is abnormal. Another subset comprised patients with normal averaged TS values and abnormal averaged TO values, or normal averaged TO values and abnormal averaged TS values.

   TS $\geq$ 2.5 ms/RR-interval and TO $>$ 0% OR

   TS $<$ 2.5 ms/RR-interval and TO $\leq$ 0%

3. HRT category 3 means both TO and TS are abnormal. The last subset comprised patients with abnormal averaged HRT parameters, both TS and TO values, hereafter called AMI high-risk.

   TS $<$ 2.5 ms/RR-interval and TO $>$ 0%.

Some examples of tachograms studied in this work are shown in Figure 3.3

Chapter 4

Results

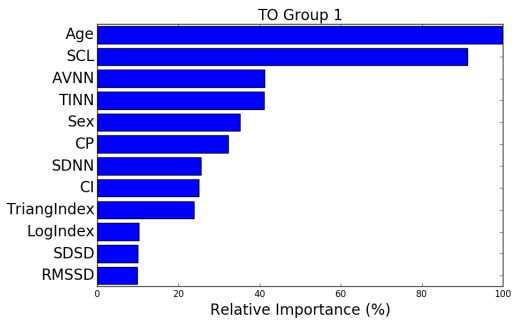
In this chapter, results are presented. First, results modeling TO are detailed, namely, the analysis of the variables using the relative importance, and then the relationship between variables and TO using BRT is depicted by using partial dependence plots. Next, the same analysis using TS is presented.

4.1 Model Inference Analysis

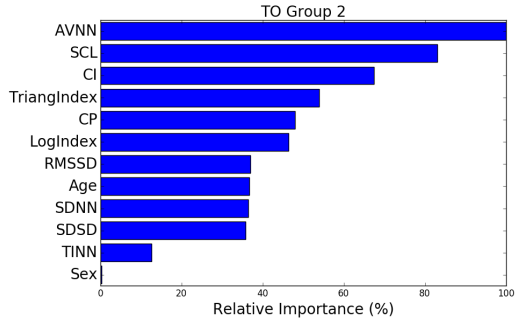
Figure 4.1 shows the relative importance of variables modeling TO using BRT. The relative importance of a feature is associated with the depth at which that feature is used as a decision node in a tree. The more deeper the node, the less relatively important that feature will be. The relative importance is related to the importance of that feature with respect to the predictability of the target variable. The expected fraction of the samples a feature contribute to can be used as an estimate of the relative importance of that feature [14].

Figure 4.1(a) represents the model for group 1, i.e. normal HRT group included in the study in the calculation of TO. For this group the two most important variables are *age* and SCL, which is the inverse HR (in milliseconds), just before the VPC. Including this variable in the model could induce some error, since this variable (almost) is used in the equation to obtain the TO parameters, so is expected to be one of the most relevant variable, distorting the results. Therefore, further work should remove this variable from the model.

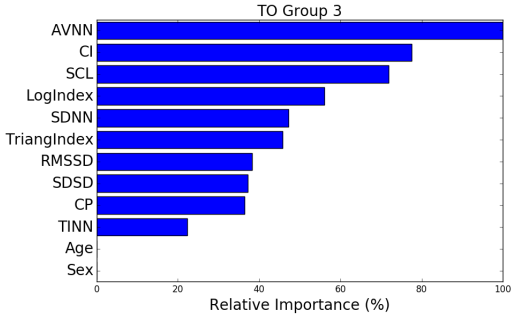
Figure 4.1(b) represents the variable importances for group 2, with subjects at higher risk. Age is not anymore the most important variable, instead, $AVNN$, a measure of the ANS status, is the most important. The *local* physiological variables,



(a) Group 1



(b) Group 2



(c) Group 3

Figure 4.1: TO Relative importance of variables for the different groups

those measured near the VPC, are quite important, since both the CI and the CP are in top positions.

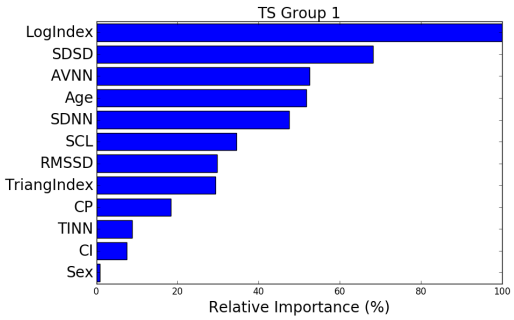
Figure 4.1(c) represents the variables importances for group 3, patients at highest risk. Neither *age* nor *sex* play any role in this model. Again, the variable that most influences the value of TO is $AVNN$, followed by the CI.

In each group the model rank, erroneously, SCL as one of the most important variable.

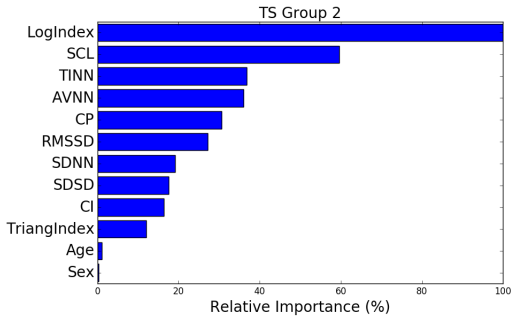
Figure 4.2 shows the relative importance of variables modeling TS using BRT.

Figure 4.2(a) represents the relative importance of variables for group 1. The relative importance is distributed among all variables, as in the case of TO. The most important variables are those related to ANS in the three minutes prior to turbulence, which are those that measure HRV. *Sex* variable in this case is the least relevant variable, unlike the case of TO. Variables related to "local" physiology of the HRT have almost no influence.

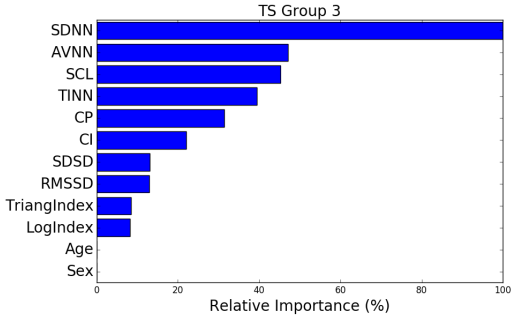
Figure 4.2(b) represents the relative importance of variables for group 2. Variables



(a) Group 1



(b) Group 2



(c) Group 3

Figure 4.2: TS Relative importance of variables for the different groups

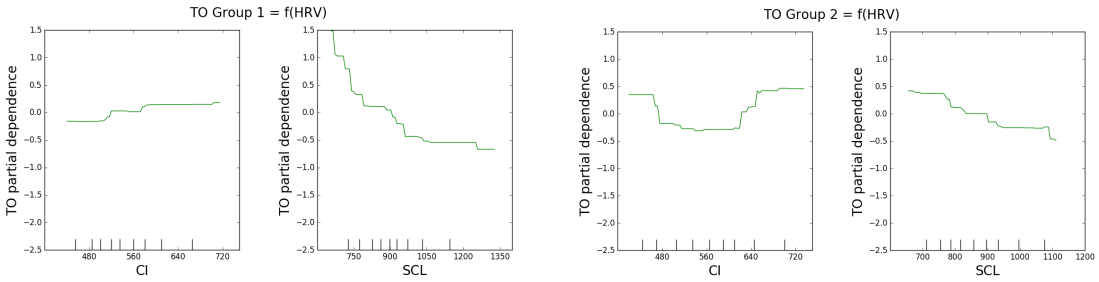
measuring HRV on the three minutes prior to the turbulence are the most influential ones. But unlike the first group, the second most important is SCL, a local measure of VPC, which measures the HR immediately preceding the turbulence. In the case, neither age nor sex have any relevance.

Figure 4.2(c) represents the relative importance of variables for group 3, highest risk patients. This case is similar to the group 2, although SCL loses some relevance. In this case the most significant variable is the standard deviation of the intervals ($SDNN$), which is a measure of the HRV. The local measures of turbulence have more relevance than in the two previous cases, but they are still not the most significant. So in this case, in contrast to healthier subjects, "local" physiology has more impact on the HRT responses.

Figure 4.3 shows the relationship between "local" variables CI and SCL and TO HRT parameters, for the groups 1, 2 and 3 respectively.

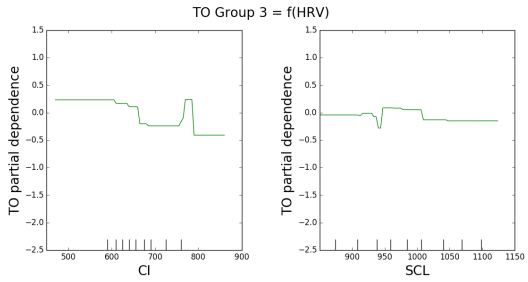
In group 1, the hypothesis of the scientific literature is confirmed, TO grows when CI grows, and TO decreases when SCL grows.

In group 2, although TO decreases when SCL grows as in group 1, CI has no longer strictly increasing behavior in sick patients.



(a) Group 1

(b) Group 2



(c) Group 3

Figure 4.3: Relationship between CI/SCL and TO for the different groups

In group 3, for high-risk patients, the trend of CI is contrary to the case of healthy patients. In high-risk patients TO decreases as CI increases. In the case of TO, it remains approximately constant regardless of the value of SCL.

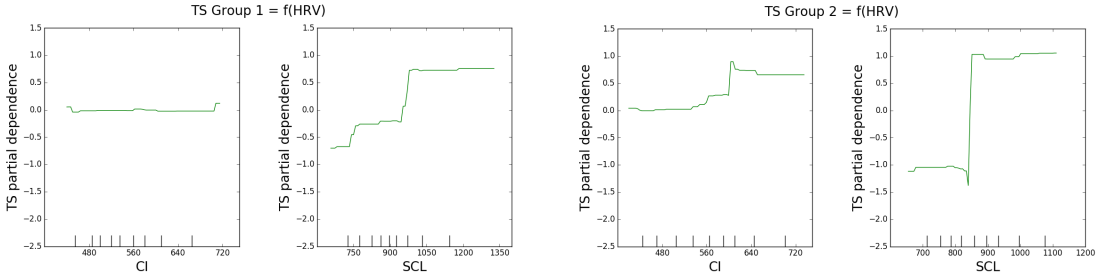
Figure 4.4 shows the relationship between CI/SCL and TS, for the groups 1, 2 and 3 respectively, using partial dependence plots, which are graphical tools to quantify the effect of one variable on the response after accounting for the average effects of the remaining variables in the model. Partial dependence plots enable us to visualize interactions among target features, and they allow an easier insight into the structure and interpretation of the model.

In group 1, the results follow the hypothesis of all the scientific literature that TO decreases when SCL grows. In the case of CI, the behavior should be decreasing, but it remains constant.

In group 2, for sick patients, the SCL behavior remains the same as in healthy patients, but in the case of CI, it is contrary to the hypothesis of the scientific literature.

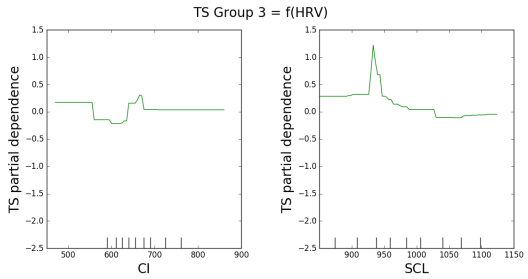
In group 3, for high-risk patients, TS remains constant regardless of the CI value, and in the case of SCL, the behavior is contrary to the scientific literature hypothesis.

Figure 4.5 shows the relationship between the most influential variables and TO, for the different groups.



(a) Group 1

(b) Group 2



(c) Group 3

Figure 4.4: Relationship between CI/SCL and TS for the different groups

In group 1, the most significant variables are age and sex, and also those that are relative to the state of the ANS.

In group 2, for sick patients, in addition to the influence of the variables related to the state of the ANS with similar tendencies to the case of healthy patients, also influence those related to local measures to the turbulence.

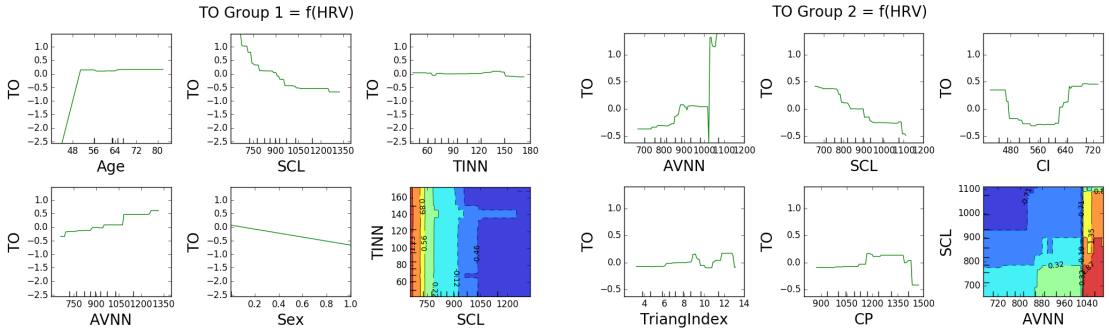
In group 3, for high-risk patients, it occurs similarly to the previous case, but with a slightly different order of variables and a less clear tendency of the signals.

Figure 4.6 shows the relationship between the most influential variables and TS, for the different groups.

In group 1, the most significant variables are those that measure variability and are relative to the state of the ANS, along with age.

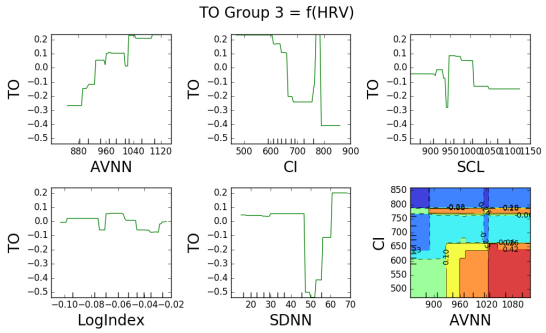
In group 2, for sick patients, in addition to the influence of the variables related to the state of the ANS with similar tendencies to the case of healthy patients, also influence those related to local measures to the turbulence.

In group 3, for high-risk patients, it occurs similarly to the previous case, but with a slightly different order of variables.



(a) Group 1

(b) Group 2



(c) Group 3

Figure 4.5: Partial dependence plots showing the relationship between the most influential variables and TO for the different groups

In order to estimate the accuracy of the BRT model, the MSE and the r^2 statistic were computed, and the results for the HRT descriptors (TO and TS) are shown in Table 4.1, along with the mean value and the standard deviation for each.

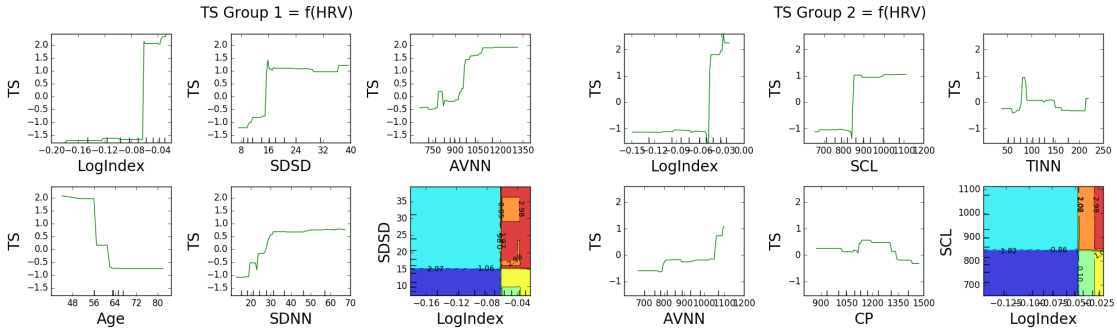
The coefficient of determination r^2 is defined as:

$$r^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{4.1}$$

Where N is the total number of cases in the test set. The numerator corresponds to the RSS given in 2.2, and the denominator corresponds to the Total Sum of Squares (TSS).

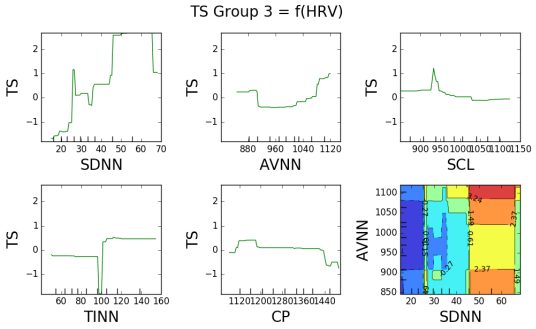
The Mean Square Error MSE is calculated as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.2}$$



(a) Group 1

(b) Group 2



(c) Group 3

Figure 4.6: Partial dependence plots showing the relationship between the most influential variables and TS for the different groups

HRT param-Group	**Mean ± Std**	**MSE**	r^2
TS-1	10.43 ± 9.02	48.18	0.41
TS-2	8.28 ± 6.54	24.92	0.29
TS-3	6.76 ± 5.54	38.43	0.12
TO-1	-1.14 ± 2.82	6.25	0.24
TO-2	-0.41 ± 3.14	9.39	0.19
TO-3	0.02 ± 2.12	5.86	0.00

Table 4.1: Mean values of HRT parameters in each group and performance metrics of the BRT models (mse and r^2).

Chapter 5

Conclusions and Future Work

The objective of the TFG to analyze the influence of HRV on HRT in AMI patients using statistical learning methods.

To achieve this objective the following specific objectives had to be met:

- Implement a module in Python to read data from Holter recordings, and to preprocess the data and to create a structure with RR-interval time series, including the labels for each beat.
- Implement a module in Python to perform HRV analysis on RR-interval time series, namely, for preprocessing stages to identify artifacts and correct them, and for computing HRV metrics (time domain indices).
- Implement a module in Python to perform HRT analysis, namely, for preprocessing stages and for computing HRT parameters.
- Implement a module in Python to integrate clinical data into the analysis.
- Implement a module in Python to perform modeling using BRT.

Results showed that BRT can model the influence of HRV on HRT, being able to identify how the ANS status models the TO and TS parameters.

Differences between the three groups of study have been found. The mean value of TS is bigger in group one, AMI low-risk patients, it decreases for group 2, and even more for group 3, AMI high-risk. The results are logical and conform to the scientific literature hypothesis. The standard deviation is high because we do not show the mean value of TS which is calculated from the mean tachogram for each patient (decreasing

the noise of the signal), but the mean value of all local TS values associated to each tachogram for each patient.

In low risk patients (group 1), HRV indices showed that whenever variability increases the HRT response is stronger, meaning higher values of TS and lower values of TO. Moreover, HRV indices were the most important explaining the value of the HRT parameters, so the status of the ANS in the 3 minutes previous to the VPC is essential to understand the response of the cardiac systems to an VPC. However, this influence is less pronounce on higher risk patients (group 2 and 3), even some of the variables had no importance at all. Even more, the models for those groups yielded a lower r^2, which indicates that the model was not able to explain the variability of TS and TO parameters. In higher risk patients, the physiological status prior to the VPC (assessed by SCL, CP, and CI) played an important role on the HRT response (rank of these variables in the relative importance), however they were less important on group 1, where the five first variables were associated with HRV and age.

The main conclusion is that in lower risk patients (group 1) the HRT response is mainly modeled by the ANS status in the 3 minutes previous to the VPC. However, in higher risk patients (groups 2 and 3), the physiological status prior to VPC plays and important role on the HRT response, noting that neither this physiological status nor ANS status are able to complete explain the HRT response.

We propose next some future work:

- More comprehensive datasets, for both pathological and healthy subjects. With a more complete and larger database, the results will be more accurate and clear. Including several heart conditions can elucidate the role of the HRV on HRT response.
- Includes different HRV metrics, extending the ANS assessment using frequency domain and nonlinear methods. This would help to better account for the ANS status.
- Further analyze the BRT model. Inference on the model can be improve by feature selection (using the BRT model itself) or by previously aggregate different indices into one index, e.g. using principal component analysis. This approach would help the model to deal with the problem of collinearity.
- Include clinical variables in the model, could help to give a complete description of the physiological status of the patient, and allowing to build a model that is able to account correctly for the HRT response
- Use the model to predict cardiovascular outcomes. The main goal is to get a better description of the cardiovascular status to predict the risk of outcomes.

Bibliography

[1] 24 hour holter ecg monitoring. http://www.expressdiagnostics.co.uk/patients/heart-tests/24-hour-holter-ecg/?lang=es. Accessed: 2016-06-30.

[2] Angioplasty. http://www.texasheart.org/HIC/Topics/Proced/angioplasty.cfm. Accessed: 2016-07-03.

[3] Article explains importance of heart rate variability for your health. https://www.heartmath.org/articles-of-the-heart/science-of-the-heart/article-explains-importance-of-heart-rate-variability-for-your-health/. Accessed: 2016-06-30.

[4] Balloong angioplasty and stents. http://www.texasheart.org/HIC/Topics/Proced/angioplasty.cfm. Accessed: 2016-06-30.

[5] Boosting regression trees 1. http://electronicimaging.spiedigitallibrary.org/article.aspx?articleid=1100624. Accessed: 2016-07-03.

[6] Boosting regression trees 2. https://www.origamilogic.com/blog/engineering-insights-machine-learning-methods-behind-marketing-intelligence/. Accessed: 2016-07-03.

[7] Cardiac emergency. https://clinicalgate.com/cardiac-emergencies/). Accessed: 2016-07-03.

[8] Cardiology explained. https://www.ncbi.nlm.nih.gov/books/NBK2204/. Accessed: 2016-06-30.

[9] Heart rate variability. https://clinicalgate.com/cardiac-emergencies/). Accessed: 2016-07-03.

[10] Holter. http://www.hopkinsmedicine.org/healthlibrary/test_procedures/cardiovascular/holter_monitor_92,P07976/. Accessed: 2016-07-03.

[11] Holter monitor. https://medlineplus.gov/ency/article/003877.htm. Accessed: 2016-06-30.

[12] Myocardial infarction healthline. http://www.healthline.com/health/acute-myocardial-infarction#outlook7. Accessed: 2016-06-30.

[13] Pulmonary circulation and systemic circulation: The routes and function of blood flow. https://www.visiblebody.com/learn/circulatory/circulatory-pulmonary-systemic-circulation. Accessed: 2016-06-30.

[14] Relative importance. http://scikit-learn.org/stable/modules/ensemble.html. Accessed: 2016-07-03.

[15] Shaliz's lectures notes. chapter 10: Decision trees. http://www.stat.cmu.edu/~cshalizi/. Accessed: 2016-06-30.

[16] Understanding the heart's electrical system and ekg results. https://www.nhlbi.nih.gov/health/health-topics/topics/hb/understanding. Accessed: 2016-06-30.

[17] What is coronary angioplasty. american heart association. https://www.heart.org/idc/groups/heart-public/@wcm/@hcm/documents/downloadable/ucm_300437.pdf. Accessed: 2016-06-30.

[18] Wiki:cross validation. https://en.wikipedia.org/wiki/Cross-validation_(statistics). Accessed: 2016-07-03.

[19] Wiki:hyperparameter. https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning). Accessed: 2016-07-03.

[20] O. Barquero-Pérez. Heart Rate Variability: A fractal Analysis. Master's thesis, Universidade do Porto - FEUP, Portugal, 2008.

[21] O. Barquero-Pérez. *Robust Signal Processing in Cardiac Signals: Applications in Heart Rate Variability, Heart Rate Turbulence and Fibrillatory Arrhythmias*. PhD thesis, Universidad Rey Juan Carlos - ETSIT, 2014.

[22] A. Bauer, M. Malik, G. Schmidt, P. Barthel, H. Bonnemeier, I. Cygankiewicz, P. Guzik, F. Lombardi, A. Müller, A. Oto, R. Schneider, M. Watanabe, D. Wichterle, and W. Zareba. Heart rate turbulence: standards of measurement, physiological interpretation, and clinical use: International Society for Holter and Noninvasive Electrophysiology Consensus. *Journal of the American College of Cardiology*, 52(17):1353–1365, 2008.

[23] M. P. de Rezende Barbosa, J. N. Júnior, B. M. Cassemiro, A. F. B. Bernardo, A. K. F. da Silva, F. M. Vanderlei, C. M. Pastre, and L. C. M. Vanderlei. Effects of functional training on geometric indices of heart rate variability. *Journal of Sport and Health Science*, 5(2):183 – 189, 2016.

[24] J. Elith, J. R. Leathwick, and T. Hastie. A working guide to boosted regression trees. *Journal of Animal Ecology*, 77(4):802–813, 2008.

[25] F. J. Gimeno-Blanes, M. Blanco-Velasco, O. Barquero-Pérez, A. García-Alberola, and J. L. Rojo-Álvarez. Sudden cardiac risk stratification with electrocardiographic indices - a review on computational processing, technology transfer, and scientific evidence. *Frontiers in Physiology*, 7:82, 2016.

[26] J. González-Carrillo, A. García-Alberola, D. Saura, P. Carrillo, R. López, J. J. Sánchez-Muñoz, J. Martínez, and M. Valdés. Impacto de la angioplastia primaria en la indicación de desfibrilador implantable en pacientes con infarto de miocardio. *Rev Esp Cardiol*, 56(12):52–56, 2003.

[27] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] L. Sornmo and P. Laguna. *Bioelectrical Signal Processing in Cardiac and Neurological Applications*. Elsevier, 2005.

[30] H. M. Stauss. Heart rate variability. *American Journal of Physiology - Regulatory, Integrative and Comparative Physiology*, 285(5):R927–R931, 2003.

[31] Task Force. Heart rate variability. *Circulation*, 93(5):1043–1065, 1996.

[32] M. A. Watanabe. Heart rate turbulence: a review. *Indian pacing and electrophysiology journal*, 3(1):10–22, 2003.

Appendix A

Computing in Cardiology 2017 abstract. Accepted

Analysis of Heart Rate Variability Influence on Heart Rate Turbulence using Boosted Regression Trees in Heart Failure Patients

Óscar Barquero-Pérez, Sandra Cantero, Rebeca Goya-Esteban, Carlos Figuera-Pozuelo, Arcadi García-Alberola, José Luis Rojo-Álvarez

Universidad Rey Juan Carlos, Fuenlabrada, Madrid, Spain

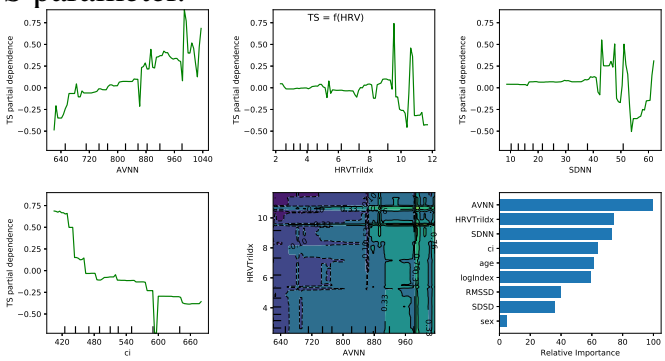
Background. Heart Rate Turbulence (HRT) is a physiological phenomenon used as cardiac risk stratification criterion. The relationship between 24-hour Heart Rate Variability (HRV) and HRT has been documented in the literature. However, the influence of HRV on HRT using individual tachograms has been not addressed.

Objective. Our aim was to propose a nonparametric model based on Boosted Regression Trees (BRT) of turbulence slope (TS) as a function of coupling interval (CI), Age, Sex, and HRV quantified by temporal indices.

Methods. We used a set of 90 holters from decompensated Heart Failure patients (Vpredict study). HRT parameters were estimated on individual ventricular premature complex (VPC) tachograms. HRV was assessed on 3-min segments prior to an individual VPC tachogram. Temporal (statistical and geometrical) indices were used to characterize HRV. We propose to model TS as a function HRV indices using BRT, which is an ensemble approach to build regression models using several small trees, each one learning information not addressed by previous trees.

Results. AVNN, HRV triangular index, SDNN, and CI were the most relevant variables. AVNN and was positively related with TS, while SDNN, HRV triangular index, and CI were negatively related with TS. BRT model accounted also for the interaction of explicative variables.

Conclusions. BRT model allowed to identify the influence of HRV on HRT for Heart Failure patients. It also allowed to quantify the influence of each HRV index on TS parameter.



TS partial dependence and variable importance.

Appendix B

Python Code for Holter

"HolterData.py"

```
# -*- coding: utf-8 -*-
"""
Extracts the data contained in the Holter's file of a patient.
"""
from __future__ import unicode_literals
import codecs
import numpy as np

class HolterData(object):

    def __init__(self):
        self.name = ''           #name of the file
        self.HRegTime = {}       #begin and end time of the recording
        self.HolterInfo = {}     #various information of the recording
        self.RRInts = []         #values of the RR intervals
        self.Labels = []         #types of beat associated to the RR
         ↪ intervals:
                                 #[N:normal, V:ventricular, A:atrial]

    def read_holter_file(self, fileName):
        """
        Extracts the information contained in fileName
        Saves the file's information in a dictionary with the
    ↪ entrances described below:
            - Name: file's name
            - HolterInfo: First line of the file, Holter's various
    ↪ information
            - HRegTime: Begin and end time of the recording
            - RRInt: Values of the RR Intervals
```

```
27              - Labels: Types of beat (N:normal, V:ventricular,
    ↪ A:atrial)
28          """
29          headerHolter = codecs.open(fileName, 'r', "iso-8859-1")
30          line1 = headerHolter.readline()
31          line2 = headerHolter.readline()
32          headerHolter.close()
33
34          #File name (first element of line 1)
35          line1 = line1.split('\t')   #First line of the file
36          self.name = line1[0]        #First element in first line of
            ↪ the file
37          line1 = line1[2:]           #Once saved, remove the name
38
39          #Now we save the rest of the information of line 1 in
            ↪ HolterInfo
40          for elem in line1:
41              name = elem.split(':')[0]
42              value = elem.split(':')[1]
43              self.HolterInfo[name] = value
44
45          #Begin and end time including in line 2
46          line2 = line2.split(' ')
47          self.HRegTime[line2[3]] = line2[4]
48          self.HRegTime[line2[5]] = line2[6]
49
50          #From the third line to the end, we find the information of
            ↪ the RR intervals
51          holter = np.loadtxt(fileName, dtype = str, skiprows = 2,
            ↪ delimiter = '\t')
52          self.RRtimes = np.array(holter[:,0], dtype = str)       #First
            ↪ column
53          self.RRInts = np.array(holter[:,1], dtype = float)      #Second
            ↪ column
54          self.Labels = np.array(holter[:,2], dtype = str)        #Third
            ↪ column
55
56          #Now we add all the values to the dictionary associated to the
            ↪ patient
57          pat = {}
58          pat['Name'] = self.name
59          pat['HolterInfo'] = self.HolterInfo
60          pat['HRegTime'] = self.HRegTime
61          pat['RRInt'] = self.RRInts
62          pat['Labels'] = self.Labels
63          return pat
64
```

Appendix C

Python Code for HRT

"HRT.py"

```
# -*- coding: utf-8 -*-
"""
Extracts from all rr intervals:
    1.  The valid tachograms (associated to the correspondent labels and their ventricular beat position for each)
    2.  The mean tachogram
    3.  The value of the turbulence slop (TS) and the turbulence onset (TO) for each tachogram,
        including the mean tachogram (TS_average, TO_average)
    4.  The RR intervals previous to the tachograms
"""

from __future__ import unicode_literals
import matplotlib.pylab as plt
import scipy as sp
import numpy as np


class HRT(object):

    def __init__(self, RRInts = [], Labels = []):
        self.RRInts = RRInts
        self.Labels = Labels
        self.tachograms = []            #only condition 1 tachograms [tach = 5N + V + 21N],
        self.v_pos_tachs = []           #and their ventricular beat position for each
        self.tachograms_ok = []         #all conditions tachograms,
        self.v_pos_tachs_ok = []        #and their ventricular beat position for each
```

```
26          self.mean_tachogram_ok = []      #mean tachogram of all
            ↪ conditions tachograms
27          self.TO = []                     #turbulence onset for each all
            ↪ conditions tachograms
28          self.TS = []                     #turbulence slope for each all
            ↪ conditions tachograms
29          self.TO_average = None           #turbulence onset for the mean
            ↪ tachogram
30          self.TS_average = None           #turbulence slope for the mean
            ↪ tachogram
31          self.RR_before_V = []            #values of the RR intervals in
            ↪ a period of three minutes (by default) previous to the all
            ↪ conditions tachograms,
32          self.pos_RR_bef_V = []           #and their associated
            ↪ positions
33
34
35      def fill_HRT(self):
36          """
37          Returns a dictionary with all the values of the heart rate
    ↪   turbulence calculated
38          """
39          self.HRT_preprocessing()
40          self.RRBeforeV()
41          self.compute()
42          hrt_pat = {}
43          hrt_pat['RRInts'] = self.RRInts
44          hrt_pat['Labels'] = self.Labels
45          hrt_pat['tachograms_ok'] = self.tachograms_ok
46          hrt_pat['v_pos_tachs_ok'] = self.v_pos_tachs_ok
47          hrt_pat['mean_tachogram_ok'] = self.mean_tachogram_ok
48          hrt_pat['RR_before_V'] = self.RR_before_V
49          hrt_pat['pos_RR_bef_V'] = self.pos_RR_bef_V
50          hrt_pat['TO'] = self.TO
51          hrt_pat['TS'] = self.TS
52          hrt_pat['TS_average'] = self.TS_average
53          hrt_pat['TO_average'] = self.TO_average
54          return hrt_pat
55
56
57      def is_tach_valid(self, tach):
58          """Checks if tach is a valid tachogram following these two
            ↪ conditions:
59              i)  Has a ventricular beat (V) at position 6 ([5])
60              ii) The rest of beats are normal (N)
61          """
62          aux = tach.copy()
63          aux[5] = 'N'
```

```
64          V = np.where(aux == 'V')
65          A = np.where(aux == 'A')
66          AV = np.append(A, V, axis=None)
67          if AV.size > 0:
68              return False
69          else:
70              return True
71
72
73      def paint_tach(self, tachogram):
74          """
75          Paints a tachogram (tach = 5N + V + 21N)
76          """
77          plt.close('all')
78          x = sp.linspace(0, 1, 27)
79          plt.figure()
80          plt.plot(x, tachogram)
81
82
83      def HRT_preprocessing(self, Filter_type = 'Watanabe'):
84          """
85          Function that performs a complete HRT preprocessing steps that
    ↪ includes:
86                  1.  Find all positions of Ventricular beats
87                  2.  Find all VPC-tachograms (5beats pre+VB+PC+20beats
    ↪ post)
88                  3.  Filter all VPC-tachograms; i.e. find all the
    ↪ availables VPC-tachograms
89                      to be used on the HRT analysis
90          This function fills the following variables described at the
    ↪ begining:
91                  i.   tachograms
92                  ii.  v_pos_tachs
93                  iii. tachograms_ok
94                  iv.  v_pos_tachs_ok
95                  v.   mean_tachogram_ok
96          Input arguments:
97              Filter_type = 'Watanabe', it allows to select filter from
    ↪ Bauer papers
98          """
99
100         possible_tachs = []             #all possible tachograms
101         labels_tachs = []               #and their labels
102
103         post_cp_beats = 20              #number of beats post_pc
104         idx_vpc_tach = range(-6,post_cp_beats+1)
105
106         v_pos = self.Labels == 'V'
```

```
107         v_pos = np.argwhere(v_pos == True)
108         # Guarantee last VPC position allows 20 beats after it.
            ↪
109         if v_pos[-1] + post_cp_beats > len(self.RRInts):
110             v_pos = np.delete(v_pos,-1)
111
112
113         for m in range(len(v_pos)):
114             idx_tachs = v_pos[m] + idx_vpc_tach+1
115
116             #absolute idx on the rr-interval time series
117             if idx_tachs[-1] >= len(self.RRInts):
118                 continue
119             possible_tachs.append(self.RRInts[idx_tachs])
120             labels_tachs.append(self.Labels[idx_tachs])
121
122         #Filtering process of tachograms:
123             #Every possible tachogram is going to be "filtered"
124             #to determine if it is a valid tachogram to be used
125             #in HRT analysis
126
127         #Indices relatives for conditions
128         idx_no_vpc = np.asarray(range(-6,-2+1) +
            ↪  range(0,post_cp_beats+1)) + 6
129         idx_no_vpc_no_cp = np.asarray(range(-6,-2+1) +
            ↪  range(1,post_cp_beats+1)) + 6
130
131
132         i = 0
133         for vpc,lab in zip(possible_tachs,labels_tachs):
134             #compute all the conditions that a VPC tachogram must
                ↪  fulfill
135
136             #All beats, except for VPC, must be 'N'
137             cond_1 = np.all(lab[idx_no_vpc] == 'N')
138
139             #Sinus beats of the tachogram of RR intervals >= 300 ms
140             cond_2 = np.all(vpc[idx_no_vpc_no_cp] >= 300)
141
142             #Sinus beats of the tachogram of RR intervals <= 2000 ms
143             cond_3 = np.all(vpc[idx_no_vpc_no_cp] <= 2000)
144
145             #Jumps <= 200 ms from one interval to the next
146             cond_4 = np.all(np.abs(np.diff(vpc[0:5]))<=200) and
                ↪  np.all(np.abs(np.diff(vpc[7:]))<=200)
147
148             #Reference RR interval, mean of the five N RR-intervals
                ↪  precedding the VPC
```

```
149             refRRinterval = np.mean(vpc[0:5])
150 
151             #All the RR-intervals should be lower than
                ↪  1.2*refRRinterval
152             cond_5 = np.all(vpc[idx_no_vpc_no_cp]<= 1.2*refRRinterval)
153 
154             #CVP should be at least 20% shorter than refRRinterval,
                ↪  i.e must be
155             #lower than 80% than refRRinterval
156             cond_6 = vpc[5] <= 0.8*refRRinterval
157 
158             #CP should be greater than 1.2*refRRinterval
159             cond_7 = vpc[6] >= 1.2*refRRinterval
160 
161             #Only condition 1 must be fulfilled
162             if np.all([cond_1]):
163                 self.tachograms.append(vpc)
164                 self.v_pos_tachs.append(v_pos[i])
165 
166             #Every one condition must be fulfilled
167             if
                ↪  np.all([cond_1,cond_2,cond_3,cond_4,cond_5,cond_6,cond_7]):
168                 self.tachograms_ok.append(vpc)
169                 self.v_pos_tachs_ok.append(v_pos[i])
170 
171             i += 1
172 
173         if len(self.tachograms_ok) == 0:
174             #there is no tachograms that fulfill all conditions
175             self.mean_tachogram_ok = np.nan
176         else:
177             self.mean_tachogram_ok = np.mean(self.tachograms_ok,axis=0)
178 
179 
180 
181     def RRBeforeV(self, mins_before_V = 3):
182         """
183         Function that saves the values of the rr intervals during
    ↪  mins_before_V
184         minutes before the ventricular beat.
185         Input arguments:
186             mins_before_V = period of minutes (3 minutes by default)
    ↪  before the tachogram,
187                             in which we want to save the values of the
    ↪  RR intervals
188         """
189         ms_before_V = mins_before_V * 60000 #in miliseconds
190 
```

```
191         rr = self.RRInts
192         v_pos = self.v_pos_tachs_ok
193 
194         sumRR = np.cumsum(rr)
195 
196         for i in v_pos:
197             #for each tachogram, we save all the values of the RR
                ↪  intervals in the given period,
198             #along with their initial position and their final
                ↪  position for localized them
199             limInf = np.where(sumRR > sumRR[i] - ms_before_V)
200             limSup = np.where(sumRR < sumRR[i])
201             pos_aux = np.append(limInf[0][0],limSup[0][-1],axis=None)
202             self.pos_RR_bef_V.append(pos_aux)
203             #NOTE: pos_RR_bef_V = (initial_position, final_position)
                ↪  is a tuple
204             self.RR_before_V.append(rr[pos_aux[0]:pos_aux[1]])
205             pos_aux = np.array([])
206 
207 
208     def TurbulenceSlope(self, VPC_tach):
209         """
210         Computes the Turbulence Slope on the tachogram given by
    ↪  parameter
211         """
212         seg_len = 5
213         posPC = 6
214         posFin = 16
215 
216         slopes = []
217         ordenada = []
218 
219         for m in range(posPC+1, posFin):
220             seg = VPC_tach[m:m+seg_len]
221             p = np.polyfit(range(m, m+seg_len), seg, 1)
222             slopes.append(p[0])
223             ordenada.append(p[1])
224 
225         TS = max(slopes)
226 
227         return TS
228 
229 
230     def TurbulenceOnset(self, VPC_tach, posPC = 6, posFin = 16):
231         """
232         Computes the Turbulence Onset on the tachogram given by
    ↪  parameter
233         """
```

```
234         TO = ((VPC_tach[7] + VPC_tach[8]) - (VPC_tach[3] +
            ↪  VPC_tach[4])) / (VPC_tach[3] + VPC_tach[4]) * 100;
235
236         return TO
237
238
239     def compute(self):
240         """
241         Computes the TS and TO on all conditions tachograms
    ↪  (tachograms_ok) and on the mean tachogram
242         """
243         if np.sum(np.isnan(self.mean_tachogram_ok)) > 0:
244             #if there is no tachograms that fulfill all conditions,
                ↪  the value of the mean tachogram is nan,
245             #so the TS and TO values will be nan too
246             self.TS = np.nan
247             self.TO = np.nan
248             self.TS_average = np.nan
249             self.TO_average = np.nan
250         else:
251             #compute TS and TO for each VPC tachogram
252             for tac in self.tachograms_ok:
253                 self.TS.append(self.TurbulenceSlope(tac))
254                 self.TO.append(self.TurbulenceOnset(tac))
255             #compute the TS and TO from the average VPC tachogram
                ↪
256             self.TS_average =
                ↪  self.TurbulenceSlope(self.mean_tachogram_ok)
257             self.TO_average =
                ↪  self.TurbulenceOnset(self.mean_tachogram_ok)
```

Appendix D

Python Code for HRV

"HRV.py"

```
# -*- coding: utf-8 -*-
"""
Calculates from the RR intervals the statistical time domain variables and the
geometrical variables to characterize the heart rate variability (HRV).
The RR intervals used are all of them previous to valid tachograms according to the
conditions evaluated in the characterization of the heart rate turbulence (HRT).
"""

from __future__ import unicode_literals
import numpy as np
import matplotlib.pylab as plt
from scipy import interpolate

class HRV(object):

    def __init__(self):
        self.HRV_statistical = {}
        self.HRV_geometrical = {}


    def load_HRV_variables(self, rr):
        """
        Returns a dictionary with all the considered values of the heart rate variability
        calculated from the list of the rr intervals passed as a parameter
```

```
        """
        hrv_pat = {}
        avnn = []
        nn50 = []
        pnn50 = []
        rmssd = []
        sdnn = []
        sdsd = []
        hrvTriangIndex = []
        logIndex = []
        tinn = []

        for i,elem in enumerate(rr):
            avnn.append(self.avnn(elem))
            nn50.append(self.nn50(elem))
            pnn50.append(self.pnn50(elem))
            rmssd.append(self.rmssd(elem))
            sdnn.append(self.sdnn(elem))
            sdsd.append(self.sdsd(elem))
            hrvTriangIndex.append(self.hrvTriangIndex(elem))
            logIndex.append(self.logIndex(elem))
            tinn.append(self.tinn(elem))

        hrv_pat['AVNN'] = avnn
        hrv_pat['NN50'] = nn50
        hrv_pat['PNN50'] = pnn50
        hrv_pat['RMSSD'] = rmssd
        hrv_pat['SDNN'] = sdnn
        hrv_pat['SDSD'] = sdsd
        hrv_pat['HRVTriangIndex'] = hrvTriangIndex
        hrv_pat['logIndex'] = logIndex
        hrv_pat['TINN'] = tinn

        return  hrv_pat


############################## HRV Indices
    ##############################

    ## STATISTICAL TIME DOMAIN HRV VARIABLES ##


    def avnn(self, nn):
        """
        Function that computes the AVNN, that is, the average value of
    all NN
        intervals computed over the complete time series that is
    passed as
```

```
70          input parameter.
71          """
72          #Mean of the NN interval series
73          mu = np.mean(nn)
74          return mu
75
76
77
78      def nn50(self, nn):
79          """
80          This function computes the NN50 index, that is, the number of
    ↪ adjacent
81          pairs of NN intervals that are more than 50 msg of the entire
    ↪ time
82          series that is passed as the input parameter.
83          """
84          #Differences between adjacent NN intervals.
85          d = np.diff(nn)
86          #Number of adjacent intervals whose distance is greater than
            ↪ 50ms
87          res= sum(abs(d) > 50)
88          return res
89
90
91
92
93      def pnn50(self, nn):
94          """
95          Function that computes the pNN50 index, that is, the
    ↪ percentage
96          of adjacent pairs of NN intervals that are more than 50 msg of
    ↪
97          the entire time series that is passed as the input parameter.
98          """
99          #Differences between adjacent NN intervals.
100         d = np.diff(nn)
101         #Number of adjacent intervals whose distance is greater than
            ↪ 50ms
102         num = float(sum(abs(d) > 50))
103         #Percentage
104         res = num/len(d)*100
105         return res
106
107
108
109     def pnnX(self, nn, x):
110         """
```

```
111         This function computes the pNNX index, that is, the percentage
    ↪ of
112         adjacent pairs of NN intervals that are more than X msg of the
    ↪ entire
113         time series that is passed as the input parameter.
114         """
115         #Differences between adjacent NN intervals.
116         d = np.diff(nn)
117         #Number of adjacent intervals whose distance is greater than x
        ↪  ms
118         num = float(sum(abs(d) > x))
119         #Percentage
120         res = num/len(d)*100
121         return res
122
123
124
125     def rmssd(self, nn):
126         """
127         Function that computes the RMSSD, that is, the square root of
    ↪ the squared
128         differences between successive NN intervals of the entire time
    ↪ series
129         that is passed as input parameter.
130         """
131         #Differences between adjacent NN intervals.
132         d = np.diff(nn)
133         #Square of the differences between adjacent NN intervals
134         d2 = d**2
135         #Square root mean squared differences between adjacent NN
        ↪  intervals.
136         res = np.sqrt(np.mean(d2))
137         return res
138
139
140     def sdann(self, nn, t = None, window_min = 5):
141         """
142         Function that computes the SDANN, that is, the standard
    ↪ deviation
143         of the means of segments of 5 min of the whole time series
    ↪ that is
144         passed as input parameter
145         """
146         if t == None:
147             t = np.cumsum(nn)/1000.      #in seconds
148
149         tau = window_min * 60;
150         #Rounding down the last element of t divided by tau
```

```
151         numSeg = float(t[-1]/tau)
152         numSeg = np.floor(numSeg);
153         numSeg = int(numSeg)
154 
155         mus = []
156         V_inicio = np.zeros((1, numSeg))
157         V_fin = np.zeros((1, numSeg))
158 
159         #Calculation of the mean of each segment
160         for m in range(numSeg):
161             #Initial and final indices for each segment of 5 min
162             inicio = np.where(t >= (m)*tau)[0]
163             fin = np.where(t <= (m+1)*tau)[0]
164             V_inicio[0][m]= inicio[0]
165             V_fin[0][m] = fin[-1]
166             seg = nn[inicio[0]:(fin[-1])+1]
167             mus.append(np.mean(seg))
168 
169         #Sdann computing
170         stadev = np.std(mus, ddof=1)
171         return stadev
172 
173 
174 
175     def sdnn(self, nn):
176         """
177         Function that computes the SDNN, that is, the standard
    ↪ deviation of
178         all NN intervals computed over the complete time series that
    ↪ is passed
179         as input parameter.
180         """
181         #Standard deviation of the series of NN intervals.
182         stdev = np.std(nn, ddof=1)
183         return stdev
184 
185 
186 
187     def sdnnidx(self, nn, t = None, window_min = 5):
188         """
189         Function that computes the sdnnidx, the mean value of the std
    ↪ of the segment of 5 minutes                    computed over the
    ↪ complete time series that is passed as input parameter
190         """
191         if t == None:
192             t = np.cumsum(nn)/1000.
193 
194         #Obtaining the temporary instants of heartbeats
```

```
195         tau = window_min * 60
196         numSeg = float(t[-1]/tau)
197         numSeg = np.floor(numSeg)
198         numSeg = int(numSeg)
199 
200         stdSeg5min = []
201         V_inicio = np.zeros((1,numSeg))
202         V_fin = np.zeros((1,numSeg))
203 
204         #Calculation of the mean of each segment
205         for m in range(numSeg):
206             #Initial and final indices for each segment of 5 min
207             inicio = np.where(t >= (m)*tau)[0]
208             fin = np.where(t <= (m+1)*tau)[0]
209             V_inicio[0][m]= inicio[0]
210             V_fin[0][m] = fin[-1]
211             seg = nn[inicio[0]:(fin[-1])+1]
212             #Standard deviation of the segment
213             stdSeg5min.append(np.std(seg, ddof=1))
214 
215         #sdnnidx computing
216         res = np.mean(stdSeg5min)
217         return res
218 
219 
220     def sdsd(self, nn):
221         """
222         Function that computes sdsd, that is, standard deviation of
    ↪ the
223         differences between adjacent NN intervals.
224         """
225         #First we obtain the differences of the intervals NN
226         d = np.diff(nn)
227         #The standard deviation is then calculated
228         res = np.std(d, ddof=1)
229         return res
230 
231 
232 
233     ### GEOMETRICAL TIME DOMAIN HRV VARIABLES ##
234 
235     def elipse(self, xc, yc, theta, sd1, sd2, pintar = None):
236         """
237         Function that constructs an ellipse and paints it, with the
    ↪ parameters
238         that are indicated in its entrance
239 
240         Input arguments:
```

```
241             Xc:     coordinate x of the center of the ellipse
242             Yc:     coordinate y from the center of the ellipse
243             Theta:  angle of the coordinate axes of the ellipse, with
    ↪ center c (xc, yc),
244                     with respect to the horizontal
245             Sd1:    length of the x axis of the ellipse
246             Sd2:    length of the y axis of the ellipse
247         Output arguments:
248             X:      points of the ellipse along the x-axis
249             Y:      points of the ellipse along the y-axis
250         """
251        #By default the ellipse is not painted
252         if pintar == None:
253             pintar = 0
254 
255         #Number of points
256         n = 100
257 
258         #Angle variation
259         l = np.arange(0, n+1, dtype=float)
260         ang = l*2*np.pi/n
261 
262         #Construction of the ellipse
263         paso1 = np.matrix([[xc],[yc]]) * np.ones((1,ang.size))
264         paso2 = np.matrix([[np.cos(theta), -np.sin(theta)],
            ↪ [np.sin(theta), np.cos(theta)]])
265         paso3 = np.matrix([np.cos(ang)*sd2,np.sin(ang)*sd1])
266         xy = paso1 + paso2 * paso3
267 
268         xyList = xy.tolist()
269 
270         if pintar:
271             plt.plot(xyList[0], xyList[1], color='r', linewidth=2.0)
272 
273         return xy
274 
275 
276     def hrvTriangIndex(self, rr, flag=None):
277         """
278         Function that computes the triangular index, that is, the
    ↪ total number
279         of intervals rr between the height of the histogram.
280         """
281         if flag == None:
282            flag = 0
283 
284         #Number of bins with fs = 128, recommendation of the ref.
285         fs = 128.
```

```
286         ts = 1/fs*1000. #ms
287 
288         #Bins computing
289         x = np.arange(min(rr), max(rr), ts)
290 
291         #Number of bins for the histogram
292         nhist = x.size
293 
294         #Histogram
295         [N, X] = plt.histogram(rr,nhist)
296 
297         #Only the non-empty bins are taken into account
298         ind = np.where(N != 0)
299         N = N[ind]
300         X = X[ind]
301 
302         #Histogram maximum
303         yo = max(N)
304 
305         res = sum(N)*1./yo
306 
307         if flag:
308         #Graphic representation
309             plt.hist(rr,nhist)
310             plt.title('HRVTriangIndex')
311             plt.xlabel('Duracion intervalos RR [ms]')
312             plt.ylabel('Numero de intervalos RR')
313 
314         return res
315 
316 
317 
318     def logIndex(self, rr, pintar=None):
319         """
320         Function that computes the triangular interpolation of the
    ↪ intervals rr
321         Output arguments:
322             res: exponent of the exponential that best adjusts ->
    ↪ k*exp(-res*t)
323         """
324 
325         if pintar == None:
326             pintar = 0
327 
328          #We create the difference series
329         diffSer = np.diff(rr)
330 
331         #We create the histogram:
```

```
332
333         #number of bins with fs = 128, recommendation of ref.
334         fs = 128.
335         ts = 1/fs*1000. #ms
336
337         #bins computing
338         x = np.arange(rr.min(0), rr.max(0), ts)
339
340         #number of bins for the histogram.
341         nhist = x.size
342
343         [Nabs, X] = np.histogram(abs(diffSer), nhist)
344
345         #non-empty bins
346         ind = np.where(Nabs != 0)[0]
347         Nabs_full = Nabs[ind]
348         X_full = X[ind]
349
350         #Adjusting the exponential k*exp(-phi*t):
351
352         #Constants
353         k = max(Nabs_full)
354
355         #Number of iterations n=4000
356         Niter = 10000
357         phi = np.linspace(-1, 1, Niter)
358
359         #Error
360         error = np.zeros((Niter,1))
361
362         for m in range(Niter):
363             error[m] = sum((Nabs_full - k*np.exp(phi[m]*X_full))**2)
364
365         #Minimum error
366         indmin = np.argmin(error)
367
368         #Phi for best setting of the exponential
369         res = phi[indmin]
370
371         #Graphic representation
372         if pintar:
373             plt.close('all')
374             plt.bar(X_full,Nabs_full)
375             plt.plot(X_full, k*np.exp(res*X_full), 'r')
376
377         return res
378
379
```

```
380
381     def mediasPoincare(self, rr, flag = None):
382         """
383         Function that computes the geometric HRV indices based on the
    ↪ Poincare Plot.
384
385         Output Parameters:
386             sd1:    dispersion of map points perpendicular to the axis
    ↪ of the identity line
387
388             sd2:    dispersion of map points along the axis of the
    ↪ identity line
389
390             cup:    contributions for the decelerations of the heart
    ↪ rhythm by the Poincare points,
391                     based on the asymmetries of the map
392
393             cdown:  contributions for the accelerations of the cardiac
    ↪ rhythm by the points of the Poincare,
394                     based on the asymmetries of the map
395         """
396
397         if flag == None:
398            flag = 0
399
400         #In the input parameter rr are the rr intervals without
            ↪ ectopic,
401         #the vectors x and y (Vid Ref) are defined as:
402         x = rr[:]
403         x = x[:-1] #we removed the last element
404         y = rr
405         y = y[1:] #we removed the first element
406         L = x.size
407
408         #The standard indices sd1 and sd2:
409         sd1 = np.sqrt((1./L) * sum(((x - y) - np.mean(x - y))**2)/2.)
410         sd2 = np.sqrt((1./L) * sum(((x + y) - np.mean(x + y))**2)/2.)
411
412         #Index sd1I (moment of second order around the identity
            ↪ line).
413         sd1I = np.sqrt((1./L) * (sum((x - y)**2)/2.))
414
415         #Quantification of the contributions of the points above and
            ↪ below
416         #the identity line.
417         xy = (x - y)/np.sqrt(2)
418         indices_up = np.where(xy > 0)[0]
419         indices_down = np.where(xy < 0)[0]
```

```
420         sd1up = np.sqrt(sum(xy[indices_up]**2)/L)
421         sd1down = np.sqrt(sum(xy[indices_down]**2)/L)
422 
423         #Finally, the relative contributions
424         cup = sd1up**2/sd1I**2
425         cdown = sd1down**2/sd1I**2
426 
427         #Graphic representations
428         if flag:
429         #poincarePlot
430             plt.plot(x,y,'.')
431 
432         #identity line and the perpendicular
433             xc = np.mean(x)
434             yc = np.mean(y)
435 
436             l1 = (np.tan(np.pi/4)*(x-xc)) + yc
437             l2 = (np.tan(3*np.pi/4)*(x-xc)) + yc
438 
439             xl = np.sort(x)
440             l1 = np.sort(l1)
441 
442             xData = [xl[0],xl[-1]]
443             yData1 = [l1[0],l1[-1]]
444             yData2 = [max(l2),min(l2)]
445 
446             plt.hold(True)
447             plt.plot(xData, yData1, color='r', linestyle=':',
                ↪  linewidth=2.0)
448             plt.hold(True)
449             plt.plot(xData, yData2, color='r', linestyle=':',
                ↪  linewidth=2.0)
450             #Paint more thick the area of the sd
451 
452             #We paint the ellipse
453             plt.hold(True)
454             self.elipse(xc,yc,np.pi/4,sd1,sd2,1)
455 
456         return sd1,sd2,cup,cdown
457 
458 
459 
460     def tinn(self, rr, flag = None):
461         """
462         Function that computes the triangular interpolation of NN
463 
464         Output arguments:
465             res:    width of the triangular interpolation
```

```
466         """
467         #Number of bins with fs = 128, recommendation of ref.
468         if flag == None:
469             flag = 0
470 
471         fs = 128
472         ts = 1./fs*1000 #ms
473 
474         #Bins computing
475         x = np.arange(min(rr), max(rr), ts)
476 
477         #Number of bins for the histogram
478         nhist = x.size
479         #Histogram
480         [N,X] = np.histogram(rr, bins = nhist)
481         #Only the non-empty bins are taken into account
482         ind = np.where(N != 0)
483         N = N[ind]
484         X = X[ind]
485 
486         #Center position of the histogram
487         yo = max(N)
488         k = np.argmax(N)
489         xo = X[k]
490 
491         #Approximation of each half of the histogram
492 
493         #Number of maximum iterations for interpolation
494         Nstep = 4000
495 
496         #First half
497         N1 = N[0:k]
498         X1 = X[0:k]
499 
500         #Second half
501         N2 = N[k+1:]
502         X2 = X[k+1:]
503 
504         #Compute of errors
505         errorsm = np.zeros((Nstep,1))
506         errorsn = np.zeros((Nstep,1))
507 
508         if k == 0:
509             res = np.nan
510         else:
511             mrange = np.linspace(min(X1)/2, max(X1), Nstep)
512             nrange = np.linspace(min(X2), 2*max(X2), Nstep)
513 
```

```
514             for h in range(Nstep):
515
516                 #First half
517                 aux1 = np.where(X1 < mrange[h])
518                 aux2 = np.where(X1 >= mrange[h])
519                 errorsm[h]= sum(N1[aux1]**2) + sum((N1[aux2] -
                    ↪  (yo*X1[aux2]-yo*xo)/(xo-mrange[h]) - yo)**2)
520
521                 #Second half
522                 aux1 = np.where(X2 <= nrange[h])
523                 aux2 = np.where(X2 > nrange[h])
524                 errorsn[h] = sum(N2[aux2]**2) + sum((N2[aux1] -
                    ↪  (yo*X2[aux1]-yo*xo)/(xo-nrange[h]) - yo)**2)
525
526             errorsm = errorsm/N1.size
527             errorsn = errorsn/N2.size
528
529             mm = min(errorsm)
530             km = np.argmin(errorsm)
531             m = mrange[km]
532             nn = min(errorsn)
533             kn = np.argmin(errorsn)
534             n = nrange[kn]
535
536             #Area percentage explained by TINN
537             k = min(abs(X1-m))
538             km = np.argmin(abs(X1-m))
539             k = min(abs(X2-n))
540             kn = np.argmin(abs(X2-n))
541             total = sum(N)
542             explained = sum(N1[km:]) + yo + sum(N2[0:kn+1])
543             tinnpercent = (total-explained)*1./total*100.
544
545
546             res=(n-m)
547
548
549         if flag:
550         #Graphic representation
551             #close all
552             Y1 = (yo*X1 -yo*xo)/(xo-m) + yo
553             aux1 = np.where(X1 < m)[0]
554             Y1[aux1] = np.zeros((aux1.size))
555
556             Y2 = (yo*X2-yo*xo)/(xo-n) + yo
557             aux1 = np.where(X2 > n)[0]
558             Y2[aux1] = np.zeros((aux1.size))
559
```

```
            XX = np.hstack((X1, xo, X2))
            YY = np.hstack((Y1, yo, Y2))

            plt.figure(1)
            plt.hist(rr, nhist)
            plt.plot(XX, YY, color='r', linewidth=2.5)
            plt.xlabel('NN (ms)')
            plt.title('tinn')

        return res


############################## HRV_Preprocessing ##############################



    def threshold_filter(self, rr):
        """
        Function that identifies a rr-interval as non-sinusal following the rule:
            if RR(n) > thr_up or RR(n) < thr_low
                where:
                thr_up = 2000
                thr_low = 300
        Verify this thresholds and find a reference
        Output arguments:
            ind_not_N:  has 1 in the position where there is a non-sinusal beat as
                        classified by the threshold criterion
        """
        ind_not_N = [False]*len(rr)
        ind_not_N = np.array(ind_not_N) #convert to a numpy array

        pos_ind_not_N = (np.where(rr > 2000) and np.where(rr < 300))[0]

        if len(pos_ind_not_N) > 0:
            ind_not_N[pos_ind_not_N] = True

        return ind_not_N



    def beat_label_filter(self, beat_labels, numBeatsAfterV = 4):
        """
        Function that identify non-normal beats, and filter the rr signal to
```

```
604         produces a vector identifying the positions where are
    ↪ non-normal beats.
605 
606         Input arguments:
607             numBeatsAfterV <= 4
608         Output arguments:
609             ind_not_N:  has 1 in the position where there is a
    ↪ non-sinusal beat as
610                         classified by the label information.
611         """
612         ind_not_N =[False] * len(beat_labels) #vector with False in
            ↪ every position
613         ind_not_N = np.array(ind_not_N) #convert to a numpy array
614 
615         pos_ind_not_N = np.where(beat_labels != 'N')[0]
616 
617         if len(pos_ind_not_N) > 0:
618 
619             ind_not_N[pos_ind_not_N] = True
620 
621         #Identify as non normal 3 beats after a ventricular one
622         ind_V = np.where(beat_labels == 'V')[0]
623 
624         if len(ind_V) > 0 :
625 
626             for g in range(1, numBeatsAfterV+1):
627                 #For each group of posterior beats to one
                    ↪ ventricular,
628                 #we eliminate the beat that is 'g' positions behind a
                    ↪ ventricular one
629                 if ind_V[-1] + g < len(ind_not_N):
630 
631                     ind_not_N[ind_V+g] = True
632 
633         return ind_not_N
634 
635 
636 
637 
638     def perct_filter(self, rr, prct):
639         """
640         Function that identifies a rr-interval as non-sinusal
    ↪ following the rule
641 
642         if RR(n) > prct * RR(n-1) then RR(n) is non-sinusal
643 
644         Output arguments:
```

```
645             ind_non_N:  has 1 in the position where there is a
    ↪ non-sinusal beat as
646                         classified by the percentage criterion.
647         """
648
649         ind_not_N = [False]*len(rr)
650         ind_not_N = np.array(ind_not_N) #convert to a numpy array
651         #Construct a matrix with the percentage * RR(n-1)
652
653         percMatrix = np.abs(np.diff(rr) / rr[:-1])
654
655         pos_ind_not_N = np.where(percMatrix > prct)[0]
656
657         if len(pos_ind_not_N) > 0:
658             ind_not_N[pos_ind_not_N+1] = True  #consider the first
                ↪ rr-interval as normal.
659
660         return ind_not_N
661
662
663
664
665     def artifact_ectopic_detection(self, rr, labels, prct,
        ↪ numBeatsAfterV = 4):
666         """
667         Function that calls detection methods to evaluate rr.
668
669         For a rr interval to be valid, it must pass all three
    ↪ detection methods.
670
671         NOTE: RECOMMENDATION use this function with a detrended
    ↪ signal, get
672         better results
673
674         Output arguments:
675             ind_not_N_beats:  has 1 in the position where there is a
    ↪ non-sinusal beat as
676                               classified by the detection methods
677         """
678
679         ind_not_N_1 = np.array(self.beat_label_filter(labels,
            ↪ numBeatsAfterV))
680
681         ind_not_N_2 = np.array(self.perct_filter(rr, prct))
682
683         ind_not_N_3 = np.array(self.threshold_filter(rr))
684
```

```
685         ind_not_N_beats = np.logical_or(ind_not_N_1, ind_not_N_2,
            ↪ ind_not_N_3)
686 
687         return ind_not_N_beats
688 
689 
690 
691 
692     def is_valid (self, ind_not_N,perct_valid = 0.15):
693         """
694         Function that checks if there are more than 15\% of invalid
    ↪ values in the vector,
695         where True is an invalid value
696 
697         Returns True if it contains less than 15\% of invalid values
698         """
699 
700         num_not_valid = sum(ind_not_N == True)
701 
702         # if  percentage of num_not_valid is lower than perc_valid
            ↪ then the RR interval segment
703         #is valid (return True)
704         if num_not_valid*100/len(ind_not_N) <= perct_valid*100:
705             return True
706         else:
707             return False
708 
709 
710 
711 
712     def artifact_ectopic_correction(self, rr, ind_not_N,
        ↪ method='cubic'):
713 
714         """
715         Function that corrects ectopic beat by interpolation.
716 
717         The interpolator method is given by the string method
    ↪ (cubic','linear','nearest').
718         """
719 
720         t_rr = np.cumsum(rr)
721 
722         rr_aux = rr[np.logical_not(ind_not_N)]
723 
724         t_rr_aux = t_rr[np.logical_not(ind_not_N)]
725 
726         #TO_DO verify extrapolation with splines in scipy
727         #Meanwhile: extrapolate using the mean value of the 5 first,
```

```
728         f = interpolate.interp1d(t_rr_aux,rr_aux,method,fill_value =
            ↪  (np.mean(rr_aux[:5]),np.mean(rr_aux[-5:])),bounds_error =
            ↪  False)
729
730         rr_corrected = f(t_rr)
731
732         return rr_corrected
733
```