Research Progress Report

Francisco Bischoff

Contents

```
Knitting as latex? TRUE
```

Last Updated: 2023-06-12 12:51:54 UTC

1 Research title

"Detecting life-threatening patterns in Point-of-care ECG using efficient memory and processor power."

1.1 Author

Francisco Bischoff

1.2 Key-words

anomaly detection, ECG, fading factors, matrix profile, time series, point-of-care

2 Abstract

Currently, Point-of-Care (POC) ECG monitoring works either as plot devices or alarms for abnormal cardiac rhythms using predefined normal trigger ranges and some rhythm analysis, which raises the problem of false alarms. In comparison, complex 12-derivation ECG machines are not suitable to use as simple monitors and are used with strict techniques for formal diagnostics. Thinking

outside the ICU setting, where high-end devices are available for patient monitoring, we aim to identify, on streaming data, life-threatening hearth electric patterns using low CPU and memory, enabling ward monitors, home devices and even wearable devices to be able to identify such events. The study design is comparable to a diagnostic study, where high accuracy is essential. Physionet's 2015 challenge yielded very good algorithms for reducing false alarms. However, none of the authors reported benchmarks, memory usage, robustness test, or context invariance that could assure its implementation on small devices. We expect to identify the obstacles of detecting life-threatening ECG changes within memory, space, and CPU constraints and using the proposed methods, assess the feasibility of implementing the algorithm in the real world and other settings than ICU monitors.

3 Introduction

Currently, Point-of-Care (POC) ECG monitoring works either as plot devices or alarms for abnormal cardiac rhythms using predefined normal trigger ranges. Modern devices also incorporate algorithms to analyze arrhythmias improving their specificity. On the other hand, full 12-derivation ECG machines are complex, are not suited to use as simple monitors, and are used with strict techniques for formal diagnostics of hearth electric conduction pathologies. The automatic diagnostics are derived from a complete analysis of the 12-dimension data after it is fully and well collected. Both systems do not handle disconnected leads and patient's motions, being strictly necessary to have a good and stable signal to allow proper diagnosis. These interferences with the data collection frequently originate false alarms increasing both patient and staff's stress; depending on how it is measured, the rate of false alarms (overall) in ICU is estimated at 65 to 95%[1].

Alarm fatigue is a well-known problem that consists of a sensory overload of nurses and clinicians, resulting in desensitization to alarms and missed alarms (the "crying wolf" situation). Patient deaths have been attributed to alarm fatigue[2]. In 1982, the increase in alarms with "no end in sight"; studies have demonstrated that most alarm signals have no clinical relevance and lead to clinical personnel's delayed response. Ultimately patient deaths were reported related to inappropriate responses to alarms[2].

In April of 2013, The Joint Commission[3] issued the Sentinel Event Alert[4], establishing alarm system safety as a top hospital priority in the National Patient Safety Goal. Nowadays (2021), the subject is still on their list, in fourth place of importance[5].

In February of 2015, the CinC/Physionet Challenge 2015 was about "Reducing False Arrhythmia Alarms in the ICU[6]. The introduction article stated that it had been reported that up to 86% resulting of the alarms are false, and this can lead to decreased staff attention and an increase in patients' delirium[7–9].

This subject draws attention to the importance of correctly identify abnormal hearth electric patterns in order to avoid the overload of clinical staff. Meanwhile, this opens the opportunity of thinking outside the ICU setting, where we still monitoring patients (and ourselves) using devices with low processing power, as for example ward monitors, home devices and wearable devices.

4 Objectives and the research question

While this research was inspired on the CinC/Physionet Challenge 2015, its purpose is not to beat the state of the art on that challenge, but to identify, on streaming data, abnormal hearth electric patterns, specifically those which are life-threatening, using low CPU and low memory requirements in order to be able to generalize the use of such information on lower-end devices, outside the ICU, as ward devices, home devices, and wearable devices.

The main questions is: can we accomplish this objective using a minimalist approach (low CPU, low memory) while maintaining robustness?

5 Principles

This research is being conducted using the Research Compendium principles[10]:

1. Stick with the convention of your peers;
2. Keep data, methods, and output separated;
3. Specify your computational environment as clearly as you can.

Data management follows the FAIR principle (findable, accessible, interoperable, reusable)[11]. Concerning these principles, the dataset was converted from Matlab's format to CSV format, allowing more interoperability. Additionally, all the project, including the dataset, is in conformity with the Codemeta Project[12].

6 Materials and methods

6.1 Softwares

6.1.1 Pipeline management

All steps of the process are being managed using the R package `targets`[13] from data extraction to the final report. An example of a pipeline visualization created with `targets` is shown in Fig. 1. This package helps to keep record of the random seeds (allowing reproducibility), changes in some part of the code (or dependencies) and then running only the branches that need to be updated, and several other features to keep a reproducible workflow avoiding unnecessary repetitions.

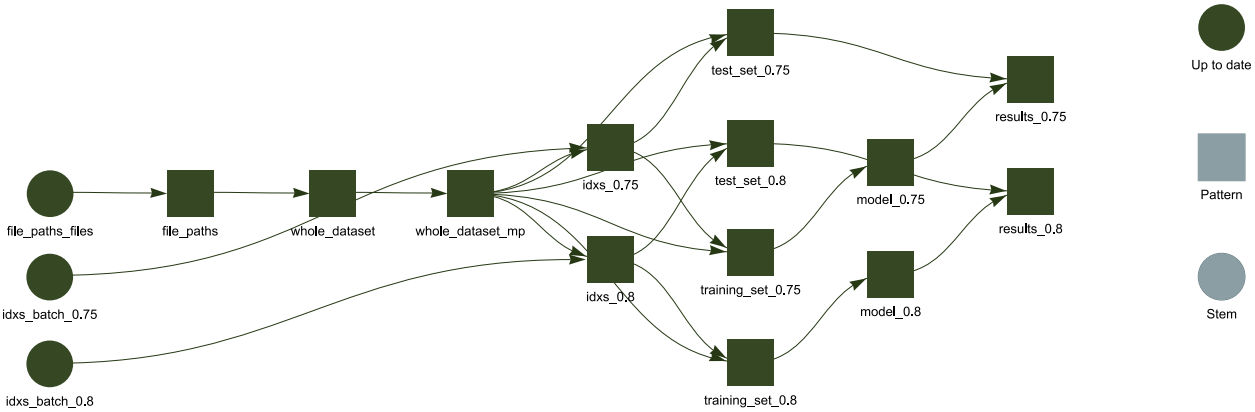
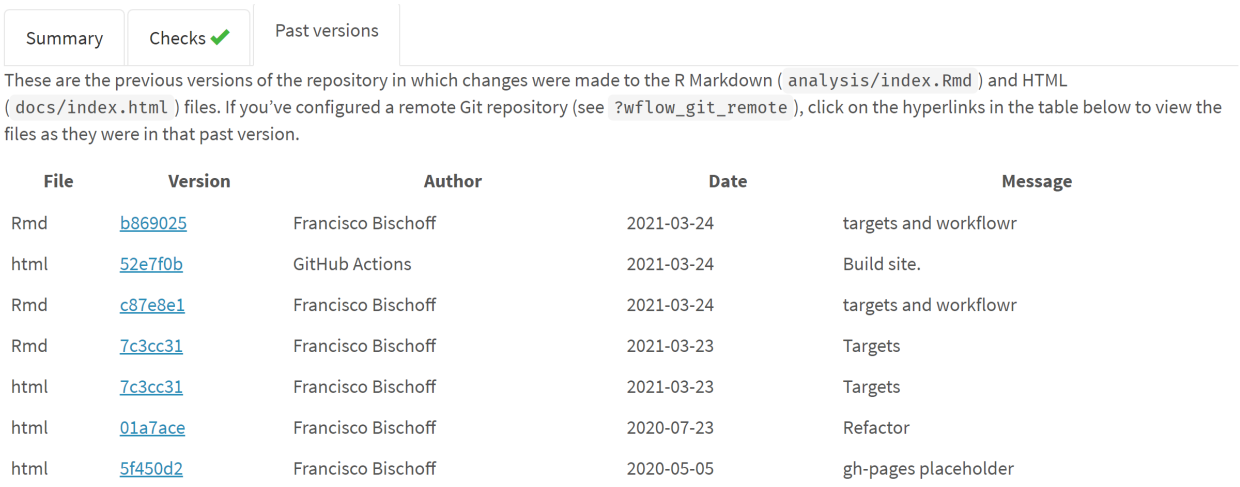


Figure 1: Example of pipeline visualization using `targets`. From left to right we see 'Stems' (steps that do not create branches) and 'Patterns' (that contains two or more branches) and the flow of the information. The green color means that the step is up to date to the current code and dependencies.

6.1.2 Reports management

The report is available on the main webpage[14], allowing inspection of previous versions managed by the R package `workflowr`[15]. This package complements the `targets` package by taking care of the versioning of every report. It is like a Log Book that keeps track of every important milestone of the project, while summarize the computational environment where it was run. Fig. 2 shows only a fraction of the generated website, where we can see that this version passed the required checks (system is up-to-date, no caches, session information was recorded, and others) and we see a table of previous versions.



Summary | Checks ✔ | Past versions

These are the previous versions of the repository in which changes were made to the R Markdown (`analysis/index.Rmd`) and HTML (`docs/index.html`) files. If you've configured a remote Git repository (see `?wflow_git_remote`), click on the hyperlinks in the table below to view the files as they were in that past version.

File	Version	Author	Date	Message
Rmd	b869025	Francisco Bischoff	2021-03-24	targets and workflowr
html	52e7f0b	GitHub Actions	2021-03-24	Build site.
Rmd	c87e8e1	Francisco Bischoff	2021-03-24	targets and workflowr
Rmd	7c3cc31	Francisco Bischoff	2021-03-23	Targets
html	7c3cc31	Francisco Bischoff	2021-03-23	Targets
html	01a7ace	Francisco Bischoff	2020-07-23	Refactor
html	5f450d2	Francisco Bischoff	2020-05-05	gh-pages placeholder

Figure 2: Fraction of the website generated by `workflowr`. On top we see that this version passed all checks, and in the middle we see a table referring to the previous versions of the report..

6.1.3 Modeling and parameter tuning

The well known package used for data science in R is the `caret` (short for **C**lassification **A**nd **RE**gression **T**raining)[16]. Nevertheless, the author of `caret` recognizes several limitations of his (great) package, and is now in charge of the development of the `tidymodels`[17] collection. For sure, there are other available frameworks and opinions[18]. Notwithstanding, this project will follow the `tidymodels` road. Three significant arguments 1) constantly improving and constantly being re-checked for bugs; large community contribution; 2) allows to plug in a custom modeling algorithm that, in this case, will be the one needed for developing this work; 3) `caret` is not in active development.

6.1.4 Continuous integration

Meanwhile, the project pipeline has been set up on GitHub, Inc.[19] leveraging on Github Actions[20] for the Continuous Integration lifecycle. The repository is available at[19], and the resulting report is available at[14]. It is also public available the roadmap and tasks status of this thesis on Zenhub[21].

6.2 Developed software

6.2.1 Matrix Profile

Matrix Profile (MP)[22], is a state-of-the-art[23,24] time series analysis technique that once computed, allows us to derive frameworks to all sorts of tasks, as motif discovery, anomaly detection, regime

change detection and others[22].

Before MP, time series analysis relied on what is called *distance matrix* (DM), a matrix that stores all the distances between two time series (or itself, in case of a Self-Join). This was very power consuming, and several methods of pruning and dimensionality reduction were researched[25].

For brevity, let's just understand that the MP and the companion Profile Index (PI) are two vectors that hold one floating point value and one integer value, respectively, regarding the original time series: (1) the similarity distance between that point on time (let's call these points "indexes") and its first nearest-neighbor (1-NN), (2) The index where this this 1-NN is located. The original paper has more detailed information[22]. It is computed using a rolling window but instead of creating a whole DM, only the minimum values and the index of these minimum are stored (in the MP and PI respectively). We can have an idea of the relationship of both on Fig. 3.

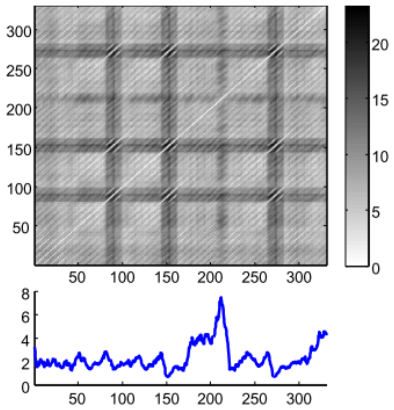


Figure 3: A distance matrix (top), and a matrix profile (bottom). The matrix profile stores only the minimum values of the distance matrix.

This research has already yielded two R packages concerning the MP algorithms from UCR[26]. The first package is called `tsmp`, and a paper has also been published in the R Journal[27] (Journal Impact Factor™, 2020 of 3.984). The second package is called `matrixprofiler` and enhances the first one, using low-level language to improve computational speed. The author has also joined the Matrix Profile Foundation as co-founder together with contributors from Python and Go languages[28,29].

This implementation in R is being used for computing the MP and MP-based algorithms of this thesis.

6.3 The data

The current dataset used is the CinC/Physionet Challenge 2015 public dataset, modified to include only the actual data and the header files in order to be read by the pipeline and is hosted by Zenodo[30] under the same license as Physionet.

The dataset is composed of 750 patients with at least five minutes records. All signals have been resampled (using anti-alias filters) to 12 bit, 250 Hz and have had FIR band-pass (0.05 to 40Hz) and mains notch filters applied to remove noise. Pacemaker and other artifacts are still present on the ECG[6]. Furthermore, this dataset contains at least two ECG derivations and one or more variables like arterial blood pressure, photoplethysmograph readings, and respiration movements.

The *events* we seek to identify are the life-threatening arrhythmias as defined by Physionet in Table 1.

Table 1: The alarms we seek to identify.

Alarm	**Definition**
Asystole	No QRS for at least 4 seconds
Extreme Bradycardia	Heart rate lower than 40 bpm for 5 consecutive beats
Extreme Tachycardia	Heart rate higher than 140 bpm for 17 consecutive beats
Ventricular Tachycardia	5 or more ventricular beats with heart rate higher than 100 bpm
Ventricular Flutter/Fibrillation	Fibrillatory, flutter, or oscillatory waveform for at least 4 seconds

The fifth minute is precisely where the alarm has been triggered on the original recording set. To meet the ANSI/AAMI EC13 Cardiac Monitor Standards[31], the onset of the event is within 10 seconds of the alarm (i.e., between 4:50 and 5:00 of the record). That doesn't mean that there are no other arrhythmias before.

For comparison, on Table 2 we collected the score of the five best participants of the challenge[32–36].

Table 2: Challenge Results on real-time data. The scores were multiplied by 100.

Score	Authors
81.39	Filip Plesinger, Petr Klimes, Josef Halamek, Pavel Jurak
79.44	Vignesh Kalidas
79.02	Paula Couto, Ruben Ramalho, Rui Rodrigues
76.11	Sibylle Fallet, Sasan Yazdani, Jean-Marc Vesin
75.55	Christoph Hoog Antink, Steffen Leonhardt

The equation used on this challenge to compute the score of the algorithms is in the Equation 1. This equation is the accuracy formula, with penalization of the false negatives. The reasoning pointed out by the authors[6] is the clinical impact of existing a genuine life-threatening event that was considered unimportant. Accuracy is known to be misleading when there is a high class imbalance[37].

$$Score = \frac{TP + TN}{TP + TN + FP + 5 * FN} \tag{1}$$

Assuming that this is a finite dataset, the pathologic cases (1) $\lim_{TP \to \infty}$ (whenever there is an event, it is positive) or (2) $\lim_{TN \to \infty}$ (whenever there is an event, it is false), cannot happen. This dataset has 292 True alarms and 458 False alarms. Experimentally, this equation yields:

- 0.24 if all guesses are on False class
- 0.28 if random guesses
- 0.39 if all guesses are on True class
- 0.45 if no false positives plus random on True class
- 0.69 if no false negatives plus random on False class

This small experiment (knowing the data in advance) shows that "a single line of code and a few minutes of effort"[38] algorithm could achieve at most a score of 0.39 in this challenge (the last two lines, the algorithm must to be very good on one class).

Nevertheless, this equation will only be useful to allow us to compare the results of this thesis with other algorithms.

6.4 Work structure

6.4.1 Project start

The project started with a literature survey on the databases Scopus, PubMed, Web of Science, and Google Scholar with the following query (the syntax was adapted for each database):

TITLE-ABS-KEY (algorithm OR 'point of care' OR 'signal processing' OR 'computer assisted' OR 'support vector machine' OR 'decision support system*' OR 'neural network*' OR 'automatic interpretation' OR 'machine learning') AND TITLE-ABS-KEY (electrocardiography OR cardiography OR 'electrocardiographic tracing' OR ecg OR electrocardiogram OR cardiogram) AND TITLE-ABS-KEY ('Intensive care unit' OR 'cardiologic care unit' OR 'intensive care center' OR 'cardiologic care center')

The inclusion and exclusion criteria were defined as in Table 3.

Table 3: Literature review criteria.

Inclusion criteria	**Exclusion criteria**
ECG automatic interpretation	Manual interpretation
ECG anomaly detection	Publication older than ten years
ECG context change detection	Do not attempt to identify life-threatening arrhythmias, namely asystole, extreme bradycardia, extreme tachycardia, ventricular tachycardia, and ventricular flutter/fibrillation
Online Stream ECG analysis	No performance measurements reported
Specific diagnosis (like a flutter, hyperkalemia, etc.)	

The survey is being conducted with peer review, all articles on full-text phase were obtained and assessed for the extraction phase, with exception of 5 articles that were not available. The survey

is currently staled on the Data Extraction phase due to external factors.

Fig. 4 shows the flow diagram of the resulting screening using PRISMA format.

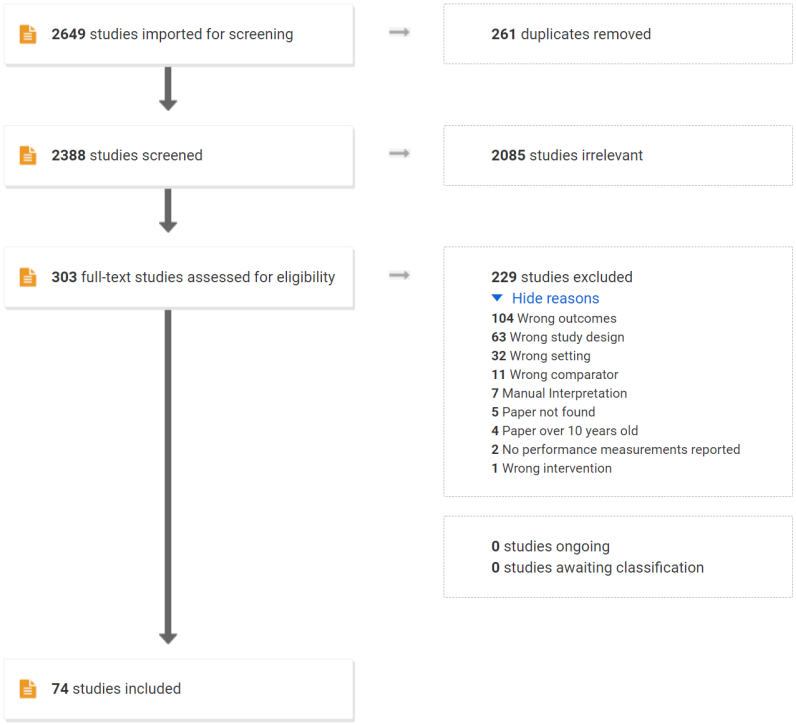


Figure 4: Flowchart of the literature survey.

The peer review is being conducted by the author of this thesis together with another coleague, Dr. Andrew Van Benschoten from the Matrix Profile Foundation[28].

Table 4 shows the Inter-rater Reliability (IRR) of the screening phases, using Cohen's κ statistic. The bottom line shows the estimated accuracy after corrected for possible confounders[39].

The purpose of using Cohen's κ in such review is to allow us to gauge the agreement of both reviewers on the task of selecting the articles according to the goal of the survey. The most naive way to verify this would be simply to measure the overall agreement (the number of articles included and excluded by both, divided by the total number of articles). Nevertheless, this would not take into account the agreement we could expect purely by chance.

However, the κ statistic must be assessed carefully. This topic is beyond the scope of this work therefore it will be explained briefly.

Table 4: Reliability on the literature survey process.

		Title-Abstract (2388 articles)		**Full-Review (303 articles)**	
		Reviewer #2		Reviewer #2	
		Include	Exclude	Include	Exclude
Reviewer #1	Include	185	381	63	58
	Exclude	129	1693	13	169
Cohen's omnibus κ		0.30		0.48	
Maximum possible κ		0.66		0.67	
Std Err for κ		0.02		0.05	
Observed Agreement		79%		77%	
Random Agreement		69%		55%	
Agreement corrected with KappaAcc		**82%**		**85%**	

While it is widely used, the κ statistic is also well criticized. The direct interpretation of its value depends on several assumptions that are often violated. (1) It is assumed that both reviewers have the same level of experience; (2) The "codes" (include, exclude) are identified with same accuracy; (3) The "codes" prevalence are the same; (4) There is no reviewer bias towards one of the choices[40,41].

In addition, the number of "codes" affects the relation between the value of κ and the actual agreement between the reviewers. For example, given equiprobable "codes" and reviewers who are 85% accurate, the value of κ are 0.49, 0.60, 0.66, and 0.69 when number of codes is 2, 3, 5, and 10, respectively[41,42].

In order to take these limitations in account, the agreement between reviewers was calculated using the KappaAcc[39] from Professor Emeritus Roger Bakeman, Georgia State University, which computes the estimated accuracy of simulated reviewers.

6.4.2 RAW data

In order to better understand the data acquisition, it has been acquired a Single Lead Heart Rate Monitor breakout from Sparkfun™[43] using the AD8232[44] microchip from Analog Devices Inc., compatible with Arduino®[45], for an in-house experiment Fig. 5.

The output gives us a RAW signal, as shown in Fig. 6.

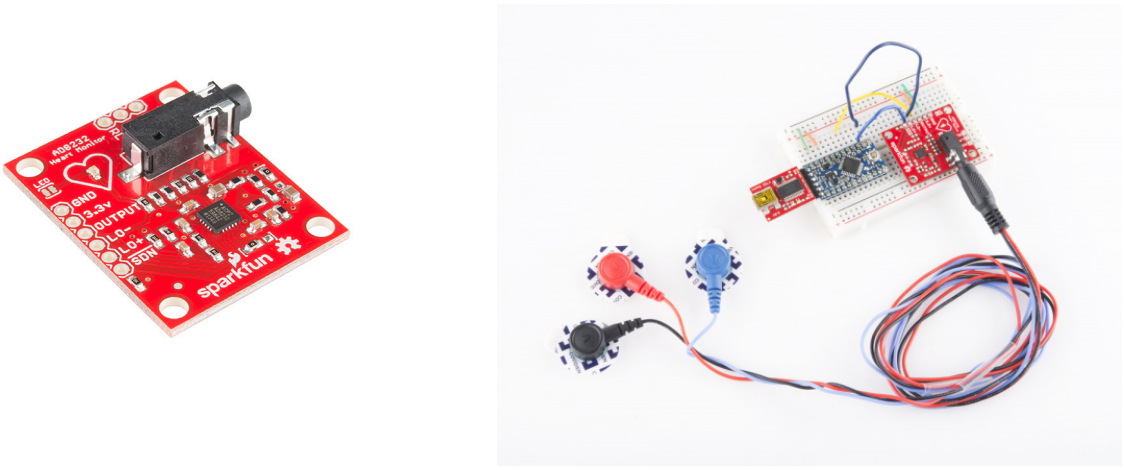


Figure 5: Single Lead Heart Rate Monitor

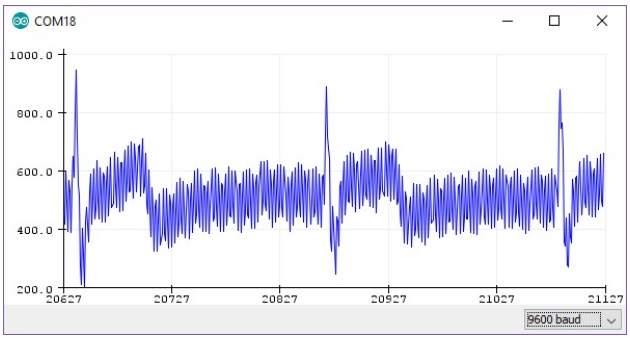


Figure 6: RAW output from Arduino at ~300hz

After applying the same settings as the Physionet database (collecting the data at 500hz, resample to 250hz, pass-filter, and notch filter), the signal is much better, as shown in Fig. 7.

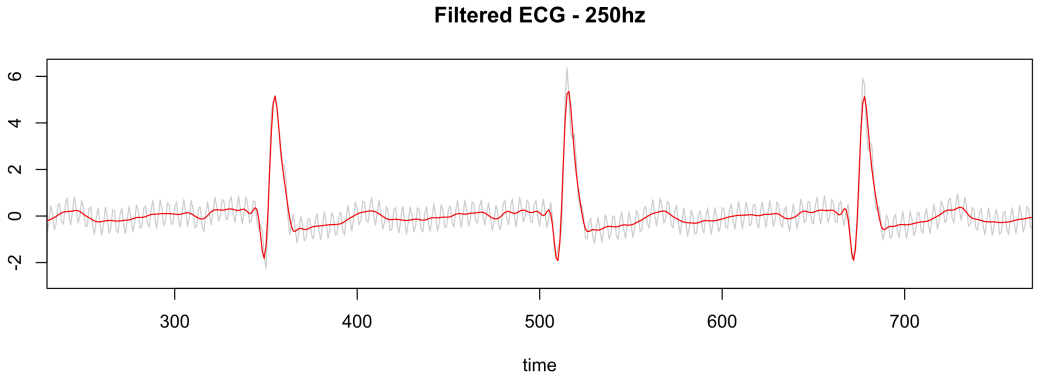


Figure 7: Gray is RAW, Red is filtered

6.4.3 Preparing the data

Usually, data obtained by sensors needs to be "cleaned" for proper evaluation. That is different from the initial filtering process where the purpose is to enhance the signal. Here we are dealing with artifacts, disconnected cables, wandering baselines and others.

Several SQIs (Signal Quality Indexes) are used in the literature[46], some trivial measures as *kurtosis*, *skewness*, median local noise level, other more complex as pcaSQI (the ratio of the sum of the five largest eigenvalues associated with the principal components over the sum of all eigenvalues obtained by principal component analysis applied to the time aligned ECG segments in the window). By experimentation (yet to be validated), a simple formula gives us the "complexity" of the signal and correlates well with the noisy data is shown in Equation 2.

$$\sqrt{\sum_{i=1}^{w}((x_{i+1}-x_i)^2)}, \quad \text{where } w \text{ is the window size} \tag{2}$$

The Fig. 8 shows some SQIs and their relation with the data.

Figure 8: Green line is the "complexity" of the signal

Fig. 9 shows that noisy data (probably patient muscle movements) are marked with a blue point

and thus are ignored by the algorithm.

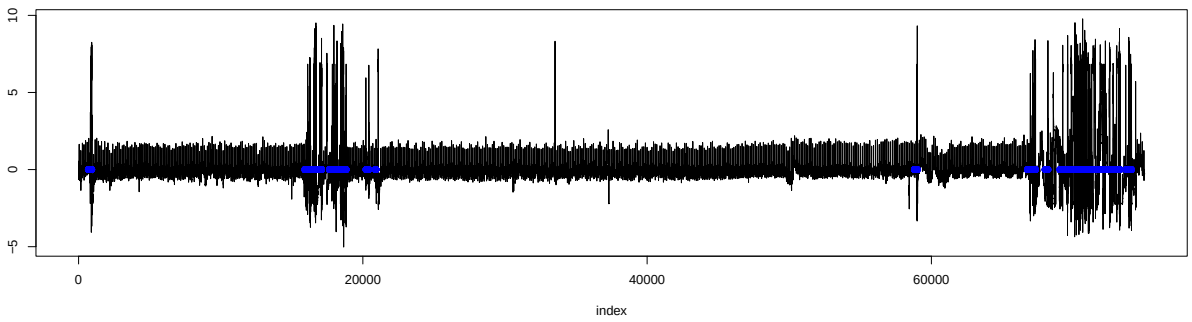


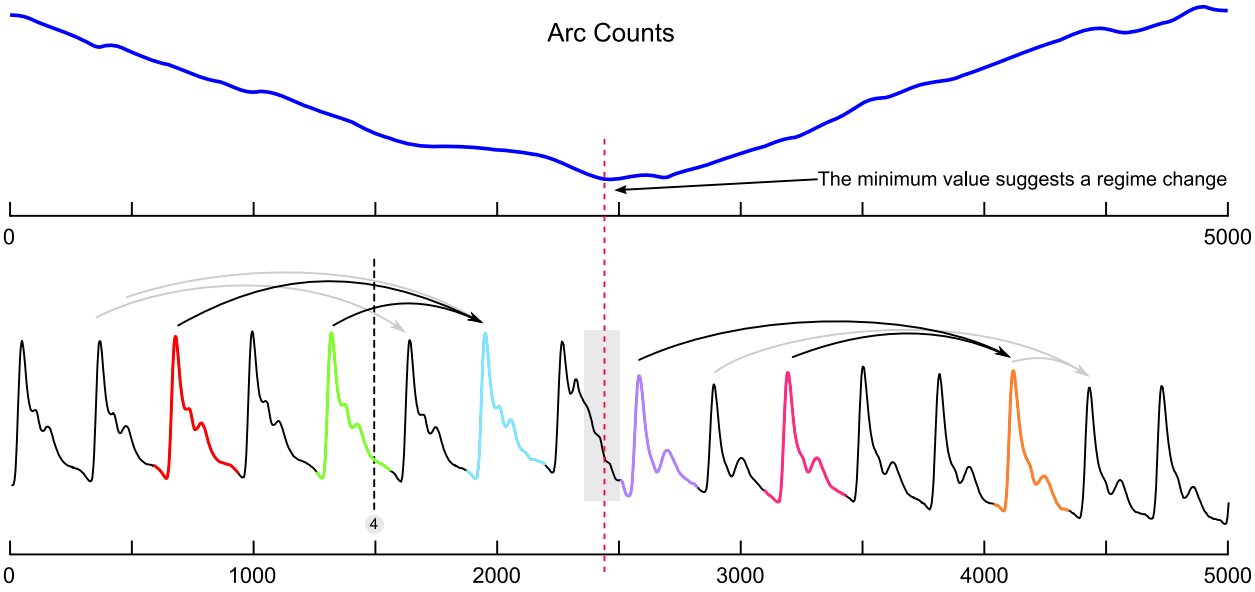
Figure 9: Noisy data marked by the "complexity" filter

Although this step of "cleaning" the data is often used, this step will also be tested if it is really necessary and the performance with and without "cleaning" will be reported.

6.4.4 Detecting regime changes

The regime change approach will be using the *Arc Counts* concept, used on the FLUSS (Fast Low-cost Unipotent Semantic Segmentation) algorithm, as explained by Gharghabi, *et al.*,[47].

The FLUSS (and FLOSS, the on-line version) algorithm is built on top of the Matrix Profile (MP)[22], described on Section 6.2.1. Recalling that the MP and the companion Profile Index (PI) are two vectors holding information about the 1-NN. One can imagine several "arcs" starting from one "index" to another. This algorithm is based on the assumption that between two regimes, the most similar shape (its nearest neighbor) is located on "the same side", so the number of "arcs" decreases when there is a change on the regime, and increases again. As show on Fig. 10. This drop on the *Arc Counts* is a signal that a change on the shape of the signal has happened.



Above: The computed arc counts for this time series.
Below: The arrows represent a shape pointing to its nearest neighbor. The gray area in the center shows the regime changing point.
The dashed line with a number underneath is a toy example, showing that four arcs cross that position.

Figure 10: FLUSS algorithm, using arc counts.

The choice of the FLOSS algorithm (on-line version of FLUSS) is founded on the following arguments:

- **Domain Agnosticism:** the algorithm makes no assumptions about the data as opposed to most available algorithms to date.
- **Streaming:** the algorithm can provide real-time information.
- **Real-World Data Suitability:** the objective is not to *explain* all the data. Therefore, areas marked as "don't know" areas are acceptable.
- **FLOSS is not:** a change point detection algorithm[48]. The interest here is changes in the shapes of a sequence of measurements.

Other algorithms we can cite are based on Hidden Markov Models (HMM) that require at least two parameters to be set by domain experts: cardinality and dimensionality reduction. The most attractive alternative could be the Autoplait[49], which is also domain agnostic and parameter-free. It segments the time series using Minimum Description Length (MDL) and recursively tests if the region is best modeled by one or two HMM. However, Autoplait is designed for batch operation, not streaming, and also requires discrete data. FLOSS was demonstrated to be superior in several datasets in its original paper. In addition, FLOSS is robust to several changes in data like down-sampling, bit depth reduction, baseline wandering, noise, smoothing, and even deleting 3% of the

data and filling with simple interpolation. Finally, the most important, the algorithm is light and suitable for low-power devices.

In the MP domain, it is worth also mentioning other possible algorithm: the Time Series Snippets[50], based on MPdist[51]. The latter measures the distance between two sequences considering how many similar sub-sequences they share, no matter the order of matching. It proved to be a useful measure (not a metric) for meaningfully clustering similar sequences. Time Series Snippets exploits MPdist properties to summarize a dataset extracting the k sequences that represent most of the data. The final result seems to be an alternative for detecting regime changes, but it is not. The purpose of this algorithm is to find which pattern(s) explains most of the dataset. Also, it is not suitable for streaming data. Lastly, MPdist is quite expensive compared to the trivial Euclidean distance.

The regime change detection will be evaluated following the criterias explained on Section 6.5.

6.4.5 Classification of the new regime

The next step towards the objective of this work is to verify if the new regime detected by the previous step is indeed a life-threatening pattern that we should trigger the alarm.

First let's dismiss some apparent solutions: (1) Clustering. It is well understood that we cannot cluster time series subsequences meaningfully with any distance measure, or with any algorithm[52]. The main argument is that in a meaningfull algorithm, the output depends on the input, and this has been proven to not happen in time series subsequence clustering[52]. (2) Anomaly detection. In this work we are not looking for surprises, but for patterns that are known to be life-threatening. (3) Forecasting. We may be tempted to make predictions, but clearly this is not the idea here.

The method of choice is classification. The simplest algorithm could be a `TRUE`/`FALSE` binary classification. Nevertheless, the five life-threatening patterns have well defined characteristics that may seem more plausible to classify the new regime using some kind of ensamble of binary classifiers or a "six-class" classifier (being the sixth class the `FALSE` class).

Since the model doesn't know which life-threatening pattern will be present in the regime (or if it will be a `FALSE` case), the model will need to check for all five `TRUE` cases and if none of these cases are identified, it will classify the regime as `FALSE`.

In order to avoid exceeding processor capacity, an initial set of shapelets[53] can be sufficient to build the `TRUE`/`FALSE` classifier. And to build such set of shapelets, leveraging on the MP, we will use the Contrast Profile[54].

The Contrast Profile (CP) looks for patterns that are at the same time very *similar* to its neighbors in class A while is very *different* from the nearest neighbor from class B. In other words, this means that such pattern represents well class A and may be taken as a "signature" of that class.

In this case we need to compute two MP, one self-join MP using the *positive* class $MP^{(++)}$ (the class that has the signature we want to find) and one AB-join MP using the *positive* and *negative* classes $MP^{(+-)}$. Then we subtract the first $MP^{(++)}$ from the last $MP^{(+-)}$, resulting in the CP. The high values on CP are the locations for the signature candidates we look for (the author of CP calls these segments *Plato's*).

Due to the nature of this approach, the MP's (containing values in Euclidean Distance) are truncated for values above $\sqrt{2w}$, where w is the window size. This because values above this threshold are negatively correlated in the Pearson Correlation space. Finally, we normalize the values by $\sqrt{2w}$. The Equation 3 synthesizes this computation.

$$CP_w = \frac{MP_w^{(+-)} - MP_w^{(++)}}{\sqrt{2w}} \quad \text{where } w \text{ is the window size} \tag{3}$$

For a more complete understanding of the process, Fig. 11 shows a practical example from the original article[54].

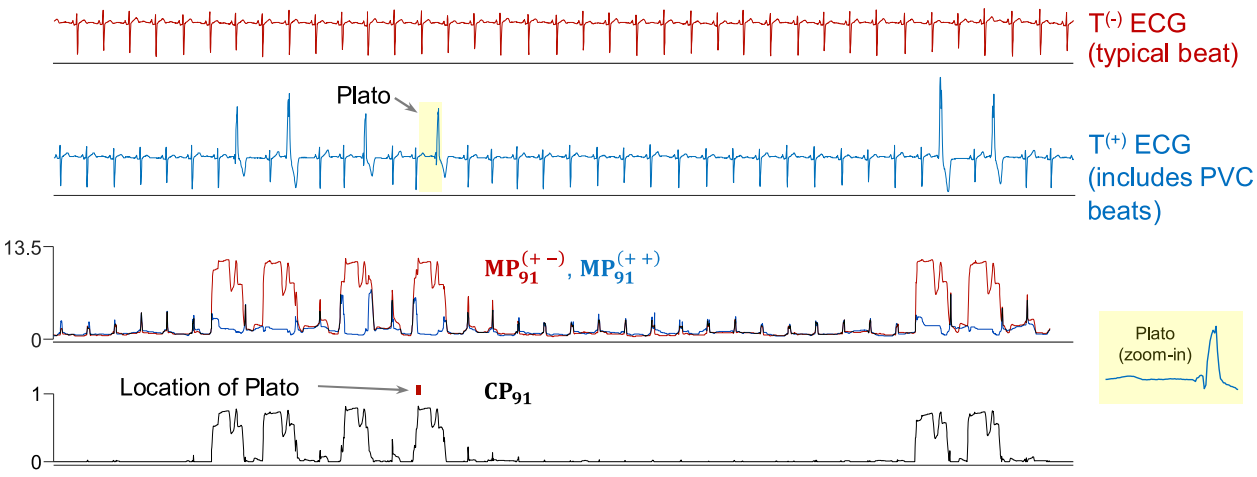


Figure 11: Top to bottom: two weakly-labeled snippets of a larger time series. T(-) contains only normal beats. T(+) also contains PVC (premature ventricular contractions). Next, two Matrix Profiles with window size 91; AB-join is in red and self-join in blue. Bottom, the Contrast Profile showing the highest location.

After extracting candidates for each class signature, a classification algorithm will be fitted and evaluated using the criterias explained on Section 6.5.

6.4.6 Summary of the methodology

In order to summarize the steps taken on this thesis to accomplish the main objective, Fig. 13, Fig. 14 and Fig. 15 show the overview of the processes involved.

First let us introduce the concept of Nested Resampling[55]. It is known that when increasing model complexity, overfitting on the training set becomes more likely to happen[56]. This is an issue that this work has to countermeasure as there are many steps that requires parameter tuning, even for algorithms that are almost parameter-free like the MP.

The rule that must be followed is simple: *do not* evaluate a model on the same resampling split used to perform its own parameter tuning. Using simple cross-validation, the information about the test set "leaks" into the evaluation, which leads to overfitting/overtuning, and gives us an optimistic biased estimative of the performance. Bernd Bischl, 2012[55] describes more deeply these factors, and also gives us a countermeasure for that: (1) from preprocessing the data to model selection use the training set; (2) the test set should be touched once, on the evaluation step; (3) repeat. This guarantees that a "new" separated data is only used *after* the model is trained/tuned.

Fig. 12 shows us this principle. The steps (1) and (2) described above are part of the **Outer resampling**, which in each loop splits the data in two sets: the training set and the test set. The training set is then used in the **Inner resampling** where, for example, the usual cross-validation may be used (creating an *Analysis set* and an *Assessment set*, to avoid conflict of terminology), and the best model/parameters is selected. Then, this best model is evaluated against the unseen test set that was created for this resampling.

The resulting (aggregated) performance of all outer samples gives us a more honest estimative of the expected performance on new data.

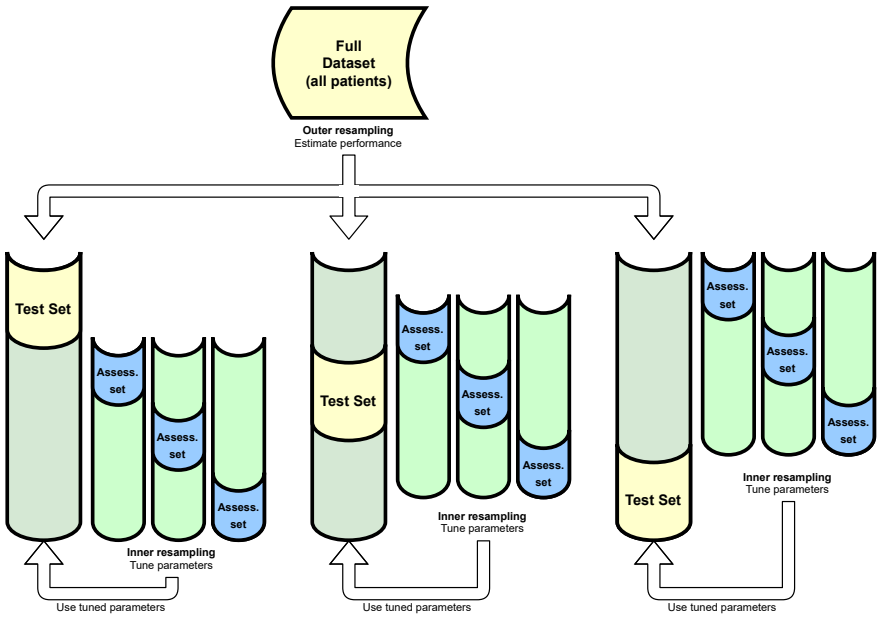


Figure 12: Nested resampling. The full dataset is resampled several times (outer resampling), so each branch has its own Test set (yellow). On each branch, the Training set is used as if it were a full dataset, being resampled again (inner resampling); here the Assessment set (blue) is used to test the learning model and tune parameters. The best model then, is finally evaluated on its own Test set.

After the understanding of the Nested Resampling[55], the following flowcharts can be better interpreted. Fig. 13 starts with the "Full Dataset" that contains all time series from the dataset described on Section 6.3. Each time series represents one file from the database, and represents one patient.

The regime change detection will use subsampling (bootstrapping can lead to substantial bias toward more complex models) in the Outer resampling and cross-validation in the Inner resampling. How the evaluation will be performed and why the use of cross-validation will be explained on Section 6.5.

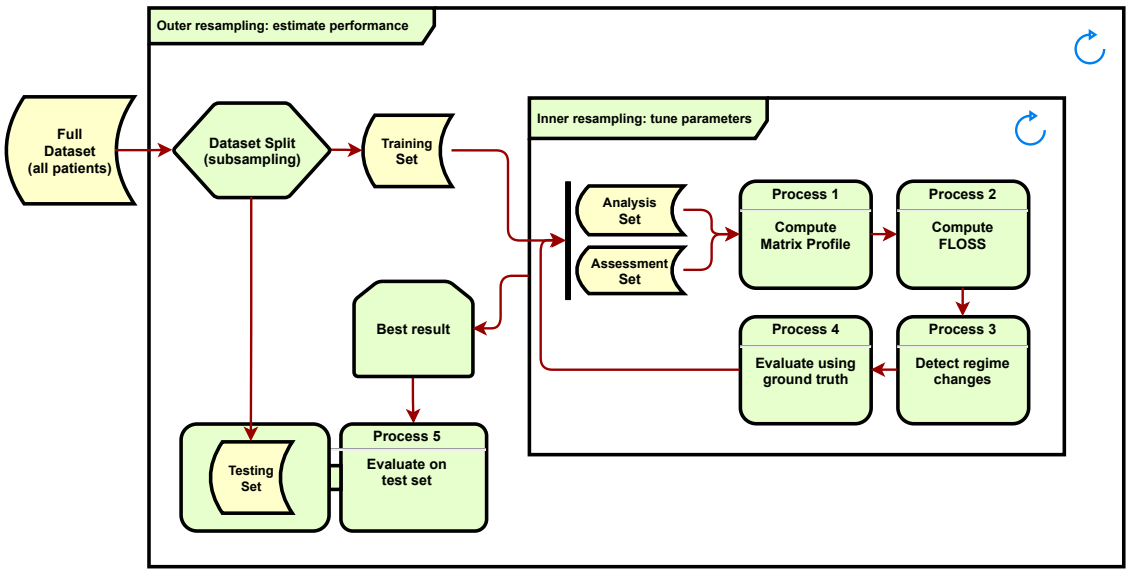


Figure 13: Pipeline for regime change detection. The full dataset (containing several patients) is divided on a Training set and a Test set. The Training set is then resampled in an Analysis set and an Assessment set. The former is used for training/parameter tuning and the latter for assessing the result. The best parameters are then used for evaluation on the Test set. This may be repeated several times.

Fig. 14 shows the processes for training the classification model. First, the last ten seconds of each time series will be identified (the even occurs in this segment). Then the dataset will be grouped by class (type of event) and `TRUE`/`FALSE` (alarm), so the Outer/Inner resampling will produce a Training/Analysis set and Test/Assessment set with similar frequency of the full dataset.

The next step will be to extract shapelet candidates using the Contrast Profile and train the classifier.

This pipeline will use subsampling (for the same reason above) in the Outer resampling and cross-validation in the Inner resampling. How the evaluation will be performed and why the use of cross-validation will be explained on Section 6.5.

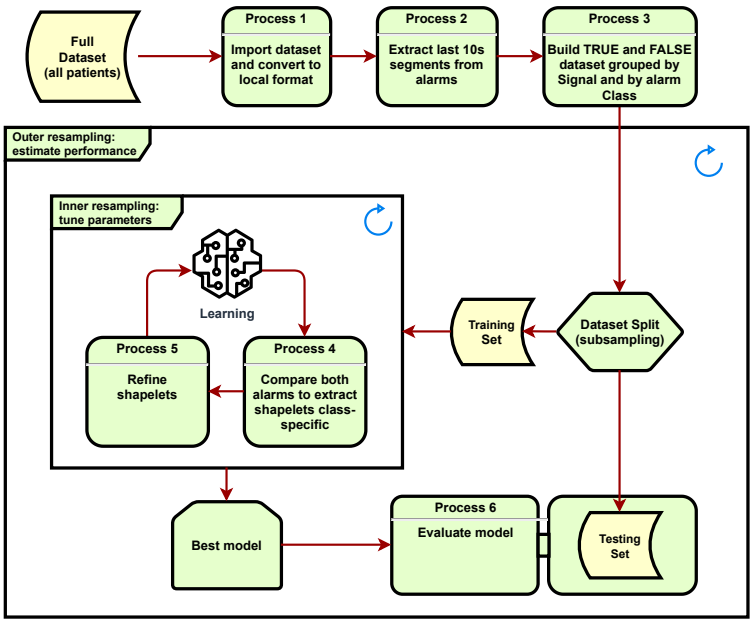


Figure 14: Pipeline for alarm classification. The full dataset (containing several patients) is grouped by class and by TRUE/FALSE alarm. This grouping allows resampling to keep a similar frequency of classes and TRUE/FALSE of the full dataset. Then the full dataset is divided on a Training set and a Test set. The Training set is then resampled in an Analysis set and an Assessment set. The former is used for extracting shapelets, training the model and parameter tuning; the latter for assessing the performance of the model. Finally, the best model is evaluated on the Test set. This may be repeated several times.

Finally, Fig. 15 shows how the final model will be used on the field. In a streaming scenario, the data will be collected and processed in real-time to maintain an up to date Matrix Profile. The FLOSS algorithm will be looking for a regime change. When a regime change is detected, a sample of this new regime will be presented to the trained classifier that will evaluate if this new regime is a life-threatening condition or not.

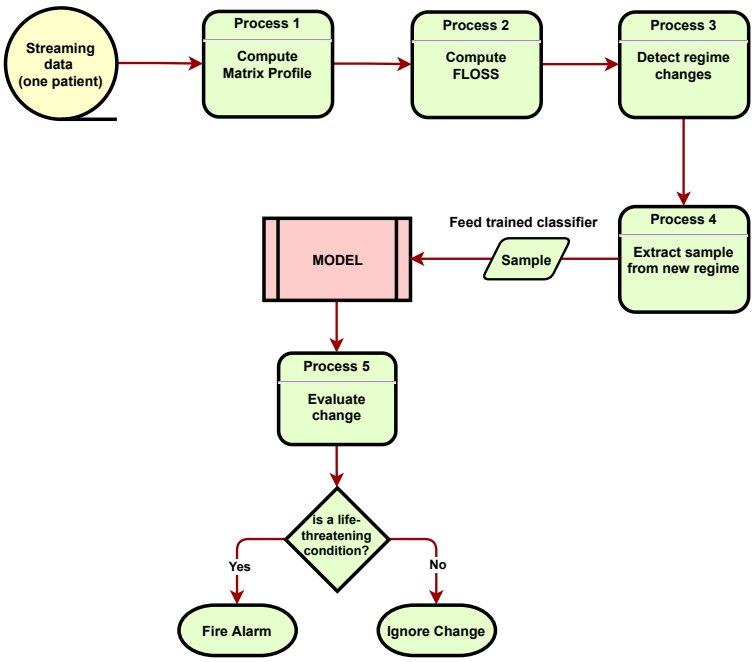


Figure 15: Pipeline of the final process. The streaming data, coming from one patient, is processed to create its Matrix Profile. Then, the FLOSS algorithm is computed for detecting a regime change. When a new regime is detected, a sample of this new regime is analysed by the model and a decision is made. If the new regime is life-threatening, the alarm will be fired.

6.5 Evaluation of the algorithms

The subsampling method used on both algorithms, regime change and classification, will be the Cross Validation, as the learning task will be in batches.

Other options dismissed[55]:

- Leave-One-Out Cross Validation: has better properties for regression than for classification. It has a high variance as an estimator of the mean loss. It also is asymptotically inconsistent and tends to select too complex models. It is demonstrates empirically that 10-fold CV is often superior.
- Bootstrapping: while it has low variance, it may be optimistic biased on more complex models. Also, its resampling method with replacement can leak information into the assessment set.
- Subsampling: is like bootstrapping, but without replacement. The only argument for not choosing it, is that with Cross Validation we make sure all the data is used for analysis and assessment.

6.5.1 Regime change

A detailed discussion about the evaluation process of segmentation algorithms is made by the FLUSS/FLOSS author[47]. Previous researches have used precision/recall or derived measures for performance. The main issue is how to assume that the algorithm was correct? If the ground truth says the change occurred at location 10,000, and the algorithm detects a change at location 10,001, is this a miss?

As pointed out by the author, several independent researchers have suggested a temporal tolerance, that solves one issue, but also has a hard time on penalize any tiny miss beyond this tolerance.

The second issue is a over-penalization of an algorithm in which most of the detections are good, but just one (or a few) is poor.

The author proposes the solution depicted in Fig. 16. It gives 0 as the best score and 1 as the worst. The function sums the distances between the ground truth locations and the locations suggested by the algorithm. The sum is then divided by the length of the time series to normalize the range to [0, 1].

The goal is minimizing this score.

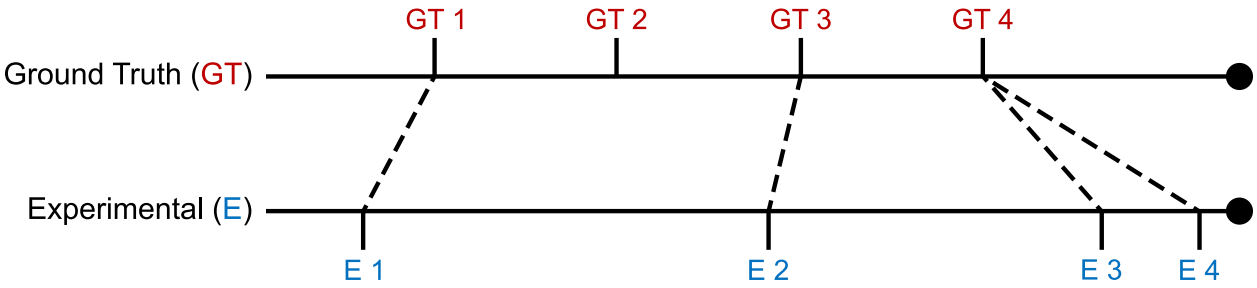


Figure 16: Regime change evaluation. The top line illustrates the ground truth, and the bottom line the locations reported by the algorithm. Note that multiple proposed locations can be mapped to a single ground truth point.

6.5.2 Classification

As described on Section 6.4.5, the model for classification will use a set of shapelets to identify if we have a `TRUE` (life-threatening) regime or a `FALSE` (non life-threatening) regime.

Although the implementation of the final process will be using streaming data, the classification algorithm will work in batches, because it will not be applied on every single data point, but on samples that are extracted when a regime change is detected. During the training phase, the data is also analyzed in batches.

One important factor we must consider is that, on real world, the majority of regime changes will be `FALSE` (i.e., not life-threatening). Thus, a performance measure that is robust to class imbalance is needed if we want to be able to assess the model after it was trained, on the field.

It is well known that the *Accuracy* measure is not reliable for unbalanced data[37,57] as it returns optimistic results for a classifier on the majority class. A description of common measures used on classification is available[37,58]. Here we will focus on three candidate measures that can be used: F-score (well discussed on[58]), Matthew's Correlation Coefficient (MCC)[59] and κ_m statistic[60].

The F-score (let's abbreviate to F_1 as this is the more common setting), is widely used on *information retrieval*, where the classes are usually classified as "relevant" and "irrelevant", and combines the *recall* (also known as sensitivity) and the *precision* (the positive predicted value). *Recall* assess how well the algorithm retrieves relevant examples among the (usually few) relevant items in dataset, while *precision* assess the proportion of indeed relevant items are contained in the retrieved examples. It ranges from [0, 1]. It ignores completely the irrelevant items that were not retrieved (usually this set contain lots of items). In classification tasks, its main weakness is not evaluate the True Negatives, and if the classifier get biased towards the `TRUE` class (increasing the False Positives significantly), this score actually gets better. During the model optimization step it is important to detect such behaviour. In addition, the F_1 score, in the classification task, doesn't inform how well the model is performing compared to a random classification of a *majority voter* (a classifier that only votes on the larger class). The F_1 score is defined on Equation 4.

$$F_1 score = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{4}$$

The MCC is a good alternative to the F_1 when we do care about the True Negatives (still both were considered to "provide more realistic estimates of real-world model performance"[61]). It is a method to compute the *Pearson product-moment correlation coefficient*[62] between the actual and predicted values. It ranges from [-1, 1]. The MCC is the only binary classification rate that only gives a high score if the binary classifier was able to correctly classify the majority of the positive and negative instances[58]. One may argue that Cohen's κ has the same behavior, but there are two main differences (1) MCC is *undefined* in the case of a *majority voter* while Cohen's κ doesn't discriminates this case from the random classifier (κ is zero for both cases) (2) It is proven that in an special case when the classifier is increasing the False Negatives, Cohen's κ doesn't get worse as spected, MCC doesn't have this issue[62]. MCC is defined on Equation 5.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \tag{5}$$

The κ_m statistic[60] is a measure that takes in account not the *random classifier* but the *majority voter*. It was introduced by Bifet *et al.*[60] for being used in online settings, where the class balance may change over time. It is defined on Equation 6, where p_0 is the observed accuracy and p_m is the accuracy of the majority voter. The score ranges from $(-\infty, 1]$, theoretically, but in practice you see negative numbers if the classifier is performing worse than the majority voter and positive numbers if performing better than the majority number, until the maximum of 1, when the classifier is optimal.

$$\kappa_m = \frac{p_0 - p_m}{1 - p_m} \tag{6}$$

In the inner resampling (model training/tuning), the classification will be binary; in our case, the data is expected to be slightly unbalanced (60% false alarms). On this task, we have to pay attention to two results of the confusion matrix: False Positives (FP) and False Negatives (FN). We expect that the Regime Change step will be under pressure of false alarms, and this system's end user may be affected by the cry wolf effect. The FN, besides its importance in a life-threatening situation, it is expected that such a situation, if happens, will keep triggering the classifier, increasing the chance of firing the alarm. Thus, we will rank the models based on the FP.

Then we have two options (1) False Discovery Rate ($FDR = \frac{FP}{TP+FP}$) and (2) False Positive Rate ($FPR = \frac{FP}{FP+TN}$). We will use the terms Precision (1-FDR) and Specificity (1-FPR), respectively, as they are more common in the literature. Following the assumptions above, it is important that when the classifier predicts a `TRUE` class, it is indeed a `TRUE` class. Thus we will rank the models based on Precision.

In the outer resampling, the κ_m will give us important information about whether the model performs better than the majority voter. The MCC will give us information about the True Negatives and whether the model performs better than a random classifier. The model's overall performance will be reported using the median and interquartile range of the MCC and κ_m.

6.5.3 Full model (streaming setting)

For the final assessment, the best and the average model of the previous pipelines will be assembled and tested using the whole original dataset.

The algorithm will be tested in each of the five life-threatening event split individually, in order to evaluate its strengths and weakness.

For more transparency, the whole confusion matrix will be reported, as well as the MCC, κ_m, and the FLOSS evaluation.

7 Current results

7.1 Regime change detection

As we have seen previously, the FLOSS algorithm is built on top of the Matrix Profile (MP). Thus, we have proposed several parameters that may or not impact the FLOSS prediction performance.

The variables for building the MP are:

- **`mp_threshold`**: the minimum similarity value to be considered for 1-NN.
- **`time_constraint`**: the maximum distance to look for the nearest neighbor.
- **`window_size`**: the default parameter always used to build an MP.

Later, the FLOSS algorithm also has a parameter that needs tuning to optimize the prediction:

- **`regime_threshold`**: the threshold below which a regime change is considered.
- **`regime_landmark`**: the point in time where the regime threshold is applied.

Using the `tidymodels` framework, we performed a basic grid search on all these parameters.

Fig. 17 shows the workflow using Nested resamplig as described on Section 6.4.6. Fig. 18 shows an example of the regime change detection pipeline. The graph on top shows the ECG streaming; the blue line marks the ten seconds before the original alarm was fired; the red line marks the time constraint used on the example; the dark red line marks the limit for taking a decision in this case of Asystole; The blue horizontal line represents the size of the sliding window. The graph on the middle shows the Arc counts as seen by the algorithm (with the corrected distribution); the red line marks the current minimum value and its index; the blue horizontal line shows the minimum value seen until then. The graph on the bottom shows the computed Arc counts (raw) and the red line is the theoretical distribution used for correction.

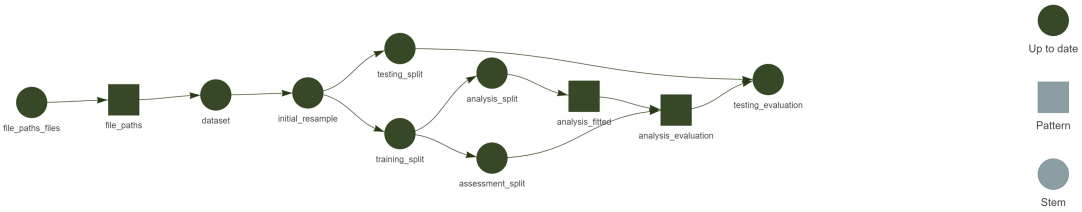


Figure 17: FLOSS pipeline.

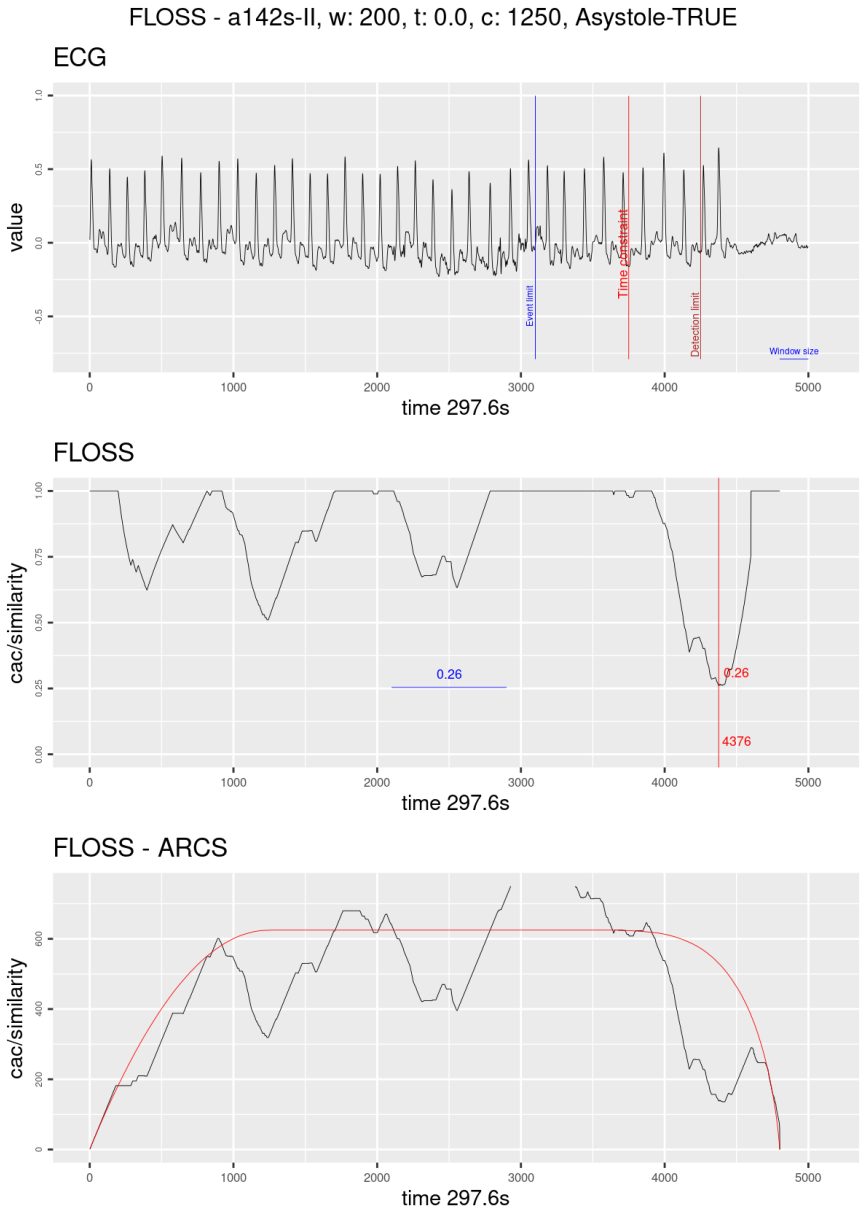


Figure 18: Regime change detection example. The graph on top shows the ECG streaming; the blue line marks the ten seconds before the original alarm was fired; the red line marks the time constraint of 1250; the dark red line marks the limit for taking a decision in this case of Asystole the blue horizontal line represents the size of the sliding window. The graph on the middle shows the Arc counts as seen by the algorithm (with the corrected distribution); the red line marks the current minimum value and its index; the blue horizontal line shows the minimum value seen until then. The graph on the bottom shows the computed Arc counts (raw) and the red line is the theoretical distribution used for correction.

The dataset used for working with the Regime Change algorithm was the "Paroxysmal Atrial Fibrillation Events Detection from Dynamic ECG Recordings: The 4th China Physiological Signal Challenge 2021" hosted by Zenodo[63] under the same license as Physionet.

The selected records were those that contain paroxysmal atrial fibrillation events, a total of 229

records. The records were split in a proportion of 3/4 for the training set (inner resampling) and 1/4 for the test set (outer resampling). The inner resampling was performed using a 5-fold cross-validation, which accounts for 137 records for fitting the models and 92 records for assessing them in the inner resampling.

The following parameters were used:

- The MP parameters were explored using the following values:
  - `mp_threshold`: 0.0 to 0.9, by 0.1 steps;
  - `time_constraint`: 0, 800 and 1500;
  - `window_size`: 25 to 350, by 25 steps;
- The FLOSS parameters were explored using the following values:
  - `regime_threshold`: 0.05 to 0.90, by 0.05 steps;
  - `regime_landmark`: 1 to 10, by 0.5 steps.

7.1.1 Parameters analysis

The above process was an example of parameter tuning seeking the best model for a given set of parameters. It used a nested cross-validation procedure that aims to find the best combination of parameters and avoid overfitting.

While this process is powerful and robust, it does not show us the importance of each parameter. At least one parameter has been introduced by reasoning about the problem (`mp_threshold`), but how important it (and other parameters) is for predicting regime changes?

For example, the process above took 4 days, 20 hours, and 15 minutes to complete the grid search using an Intel(R) Xeon(R) Silver 4210R @ 2.40 GHz server. Notice that about 133 different combinations of parameters were tested on computing the MP (not FLOSS, the `regime_threshold`), 5 folds, 2 times each. That sums up about 35.2 x 10^9 all-pairs Euclidean distances computed on less than 5 days (on CPU, not GPU). Not bad.

Another side note on the above process, it is not a "release" environment, so we must consider lots of overhead in computation and memory usage that must be taken into account during these five days of grid search. Thus, much time can be saved if we know what parameters are essential for the problem.

In order to check the effect of the parameters on the model, we need to compute the *importance* of each parameter.

Wei *et al.* published a comprehensive review on variable importance analysis[64].

Our case is not a typical case of variable importance analysis, where a set of *features* are tested against an *outcome*. Instead, we have to proxy our analysis by using as *outcome* the FLOSS performance score and as *features* (or *predictors*) the tuning parameters that lead to that score.

That is accomplished by fitting a model using the tuning parameters to predict the FLOSS score and then applying the techniques to compute the importance of each parameter.

For this matter, a Bayesian Additive Regression Trees (BART) model was chosen after an experimental trial with a set of regression models (including glmnet, gbm, mlp) and for its inherent characteristics, which allows being used for model-free variable selection[65]. The best BART model was selected using 10-fold cross-validation repeated 3 times, having great predictive power with an RMSE around 0.2 and an R^2 around 0.99. With this fitted model, we could evaluate each parameter's importance.

7.1.2 Interactions

Before starting the parameter importance analysis, we need to consider the parameter interactions since this is usually the weak spot of the analysis techniques.

The first BART model was fitted using the following parameters:

$$\begin{aligned} E(score) = \alpha &+ time\_constraint \\ &+ mp\_threshold + window\_size \\ &+ regime\_threshold + regime\_landmark \end{aligned} \tag{7}$$

After checking the interactions, this is the refitted model:

$$\begin{aligned} E(score) = \alpha &+ time\_constraint \\ &+ mp\_threshold + window\_size \\ &+ regime\_threshold + regime\_landmark \\ &+ (regime\_threshold \times regime\_landmark) \\ &+ (mp\_threshold \times regime\_landmark) \\ &+ (mp\_threshold \times window\_size) \end{aligned} \tag{8}$$

Fig. 19 shows the variable interaction strength between pairs of variables. That allows us to verify if there are any significant interactions between the variables. Using the information from

the first model fit, equation Equation 7, we see that `regime_threshold` interacts strongly with `regime_landmark`. This interaction was already expected, and we see that even after refitting the model, equation Equation 8, this interaction is still strong.

This is not a problem *per se* but a signal we must be aware of when exploring the parameters.

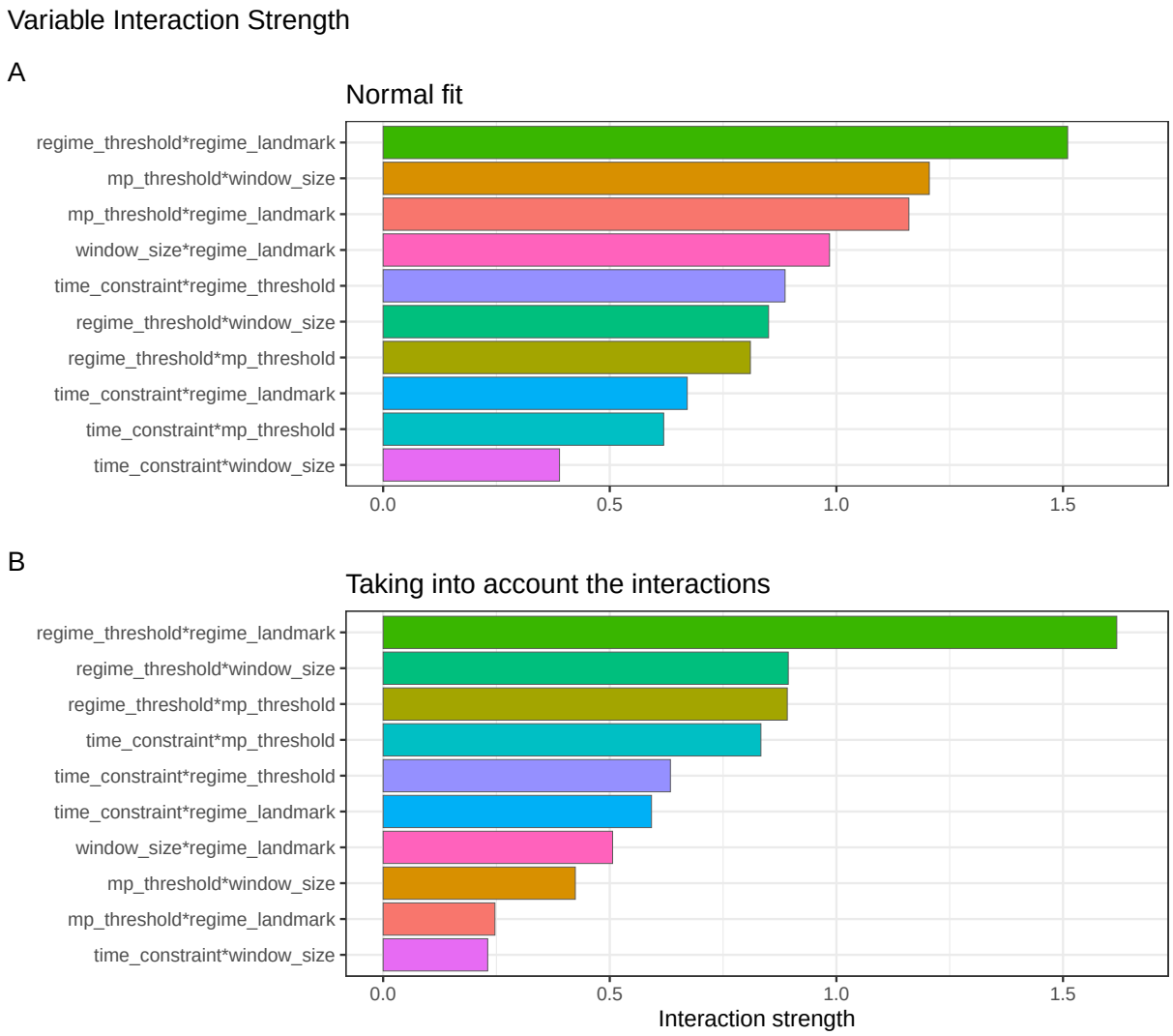


Figure 19: Variable interactions strength using feature importance ranking measure (FIRM) approach[66]. A) Shows strong interaction between `regime_threshold` and `regime_landmark`, `mp_threshold` and `window_size`, `mp_threshold` and `regime_landmark`. B) Refitting the model with these interactions taken into account, the strength is substantially reduced, except for the first, showing that indeed there is a strong correlation between those variables.

7.1.3 Importance

After evaluating the interactions, we then can perform the analysis of the variable importance. The goal is to understand how the FLOSS score behaves when we change the parameters.

Here is a brief overview of the different techniques:

Feature Importance Ranking Measure (FIRM)

The FIRM is a variance-based method. This implementation uses the ICE curves to quantify each feature effect which is more robust than partial dependance plots (PDP)[67].

It is also helpful to inspect the ICE curves to uncover some heterogeneous relationships with the outcome[68].

Advantages:

- Has a causal interpretation (for the model, not for the real world)
- ICE curves can uncover heterogeneous relationships

Disadvantages:

- The method does not take into account interactions.

Permutation

The Permutation method was introduced by Breiman in 2001[69] for Random Forest, and the implementation used here is a model-agnostic version introduced by Fisher *et al.* in 2019[70]. A feature is "unimportant" if shuffling its values leaves the model error unchanged, assuming that the model has ignored the feature for the prediction.

Advantages:

- Easy interpretation: the importance is the increase in model error when the feature's information is destroyed.
- No interactions: the interaction effects are also destroyed by permuting the feature values.

Disadvantages:

- It is linked to the model error: not a disadvantage *per se*, but may lead to misinterpretation if the goal is to understand how the output varies, regardless of the model's performance. For example, if we want to measure the robustness of the model when someone tampers the features, we want to know the *model variance* explained by the features. Model variance (explained by the features) and feature importance correlate strongly when the model generalizes well (it is not overfitting).

- Correlations: If features are correlated, the permutation feature importance can be biased by unrealistic data instances. Thus we need to be careful if there are strong correlations between features.

SHAP

The SHAP feature importance[71] is an alternative to permutation feature importance. The difference between both is that Permutation feature importance is based on the decrease in model performance, while SHAP is based on the magnitude of feature attributions.

Advantages:

- It is not linked to the model error: as the underlying concept of SHAP is the Shapley value, the value attributed to each feature is related to its contribution to the output value. If a feature is important, its addition will significantly affect the output.

Disadvantages:

- Computer time: Shapley value is a computationally expensive method and usually is computed using Montecarlo simulations.
- The Shapley value can be misinterpreted: The Shapley value of a feature value **is not** the difference of the predicted value after removing the feature from the model training. The interpretation of the Shapley value is: "Given the current set of feature values, the contribution of a feature value to the difference between the actual prediction and the mean prediction is the estimated Shapley value"[68].
- Correlations: As with other permutation methods, the SHAP feature importance can be biased by unrealistic data instances when features are correlated.

7.1.4 Importance analysis

Using the three techniques simultaneously allows a broad comparison of the model behavior[67]. All three methods are model-agnostic (separates interpretation from the model), but as we have seen, each method has its advantages and disadvantages[68].

Fig. 20 then shows the variable importance using three methods: Feature Importance Ranking Measure (FIRM) using Individual Conditional Expectation (ICE), Permutation-based, and Shapley Additive explanations (SHAP). The first line of this figure shows an interesting result that probably comes from the main disadvantage of the FIRM method: *the method does not take into account*

interactions. We see that FIRM is the only one that disagrees with the other two methods, giving much importance to `window_size`.

In the second line, taking into account the interactions, we see that all methods somewhat agree with each other, accentuating the importance of `regime_threshold`, which makes sense as it is the most evident parameter we need to set to determine if the *Arc Counts* are low enough to indicate a regime change.

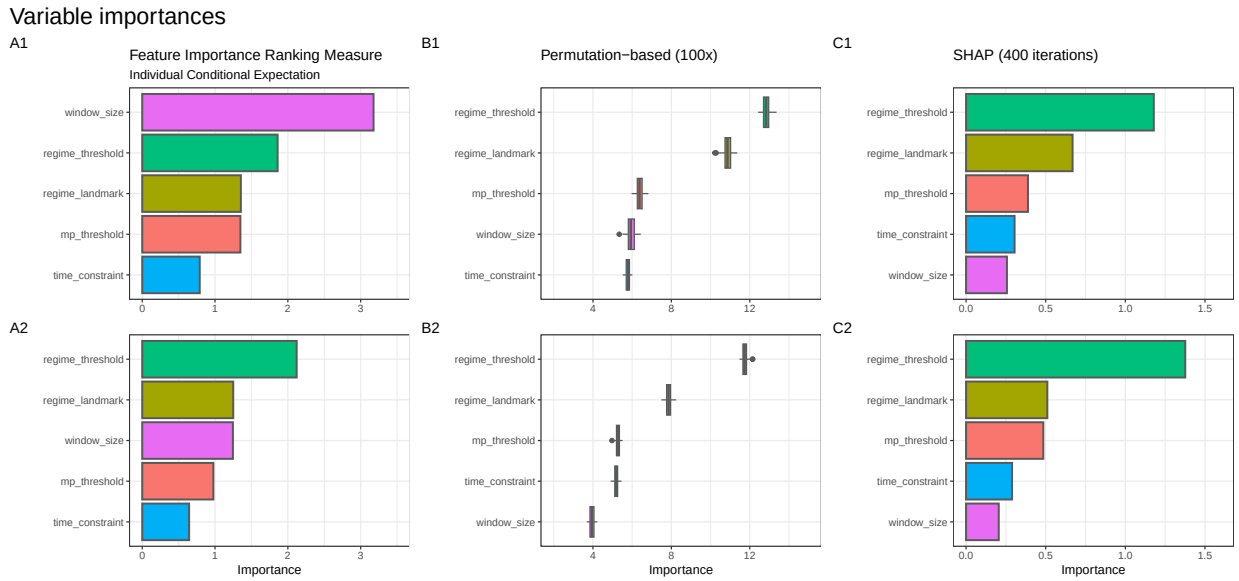


Figure 20: Variables importances using three different methods. A) Feature Importance Ranking Measure using ICE curves. B) Permutation method. C) SHAP (400 iterations). Line 1 refers to the original fit, and line 2 to the re-fit, taking into account the interactions between variables (Fig. 19).

Fig. 21 and Fig. 22 show the effect of each feature on the FLOSS score. The more evident difference is the shape of the effect of `time_constraint` that initially suggested better results with larger values. However, removing the interactions seems to be a flat line.

Based on Fig. 20 and Fig. 22 we can infer that:

- **`regime_threshold`**: is the most important feature, has an optimal value to be set, and since the high interaction with the `regime_landmark`, both must be tuned simultaneously. In this setting, high thresholds significantly impact the score, probably due to an increase in false positives starting on >0.65 the overall impact is mostly negative.

- **`regime_landmark`**: is not as important as the `regime_threshold`, but since there is a high interaction, it must not be underestimated. It is known that the *Arc Counts* have more uncertainty as we approach the margin of the streaming, and this becomes evident looking

at how the score is negatively affected for values below 3.5s.

- `window_size`: has a near zero impact on the score when correctly set. Nevertheless, for higher window values, the score is negatively affected. This high value probably depends on the data domain. In this setting, the model is being tuned towards the changes from atrial fibrillation/non-fibrillation; thus, the "shape of interest" is small compared to the whole heartbeat waveform. Window sizes smaller than 150 are more suitable in this case. As Beyer *et al.* noted, "as dimensionality increases, the distance to the nearest data point approaches the distance to the farthest data point"[72], which means that the bigger the window size, the smaller will be the contrast between different regimes.

- `mp_threshold`: has a fair impact on the score, but primarily by *not using it*. We start to see a negative impact on the score with values above 0.60, while a constant positive impact with lower values.

- `time_constraint`: is a parameter that must be interpreted cautiously. The 0 (zero) value means **no constraint**, which is equivalent to the size of the FLOSS history buffer (in our setting, 5000). We can see that this parameter's impact throughout the possible values is constantly near zero.

In short, for the MP computation, the parameter that is worth tuning is the `window_size`, while for the FLOSS computation, both `regime_threshold` (mainly) and `regime_landmark` shall be tuned.

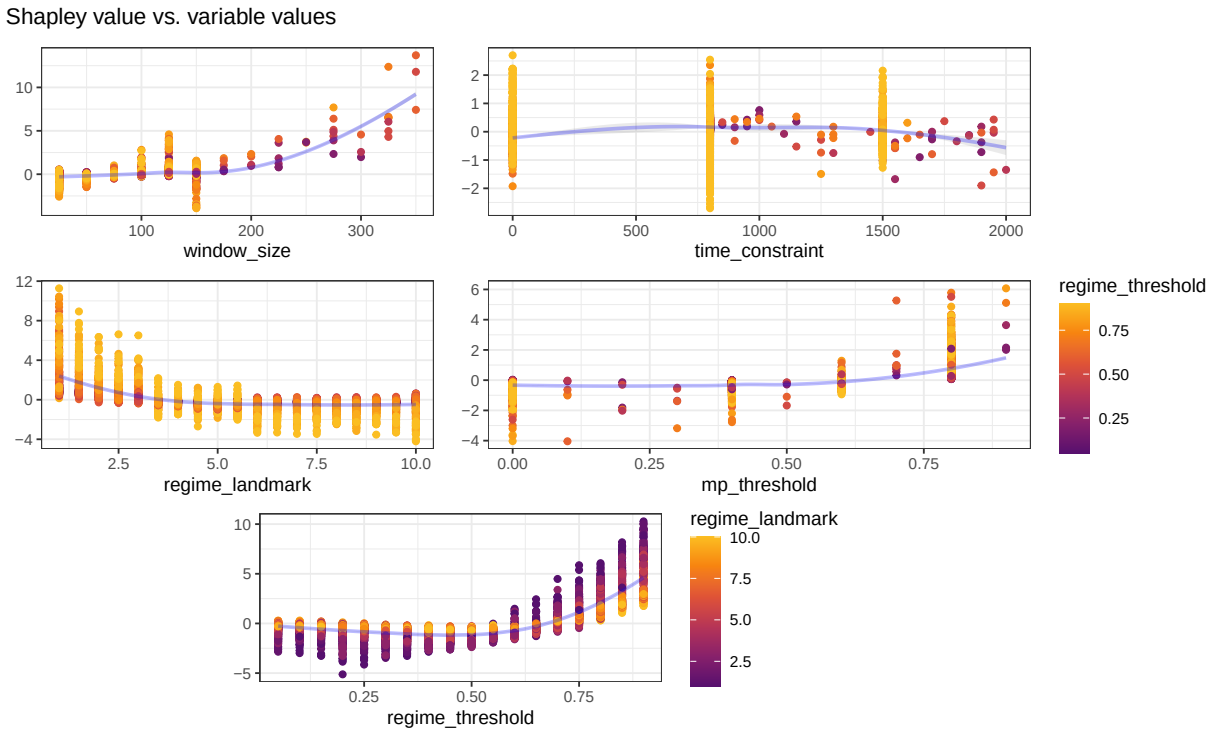


Figure 21: This shows the effect each variable has on the FLOSS score. This plot doesn't take into account the variable interactions.

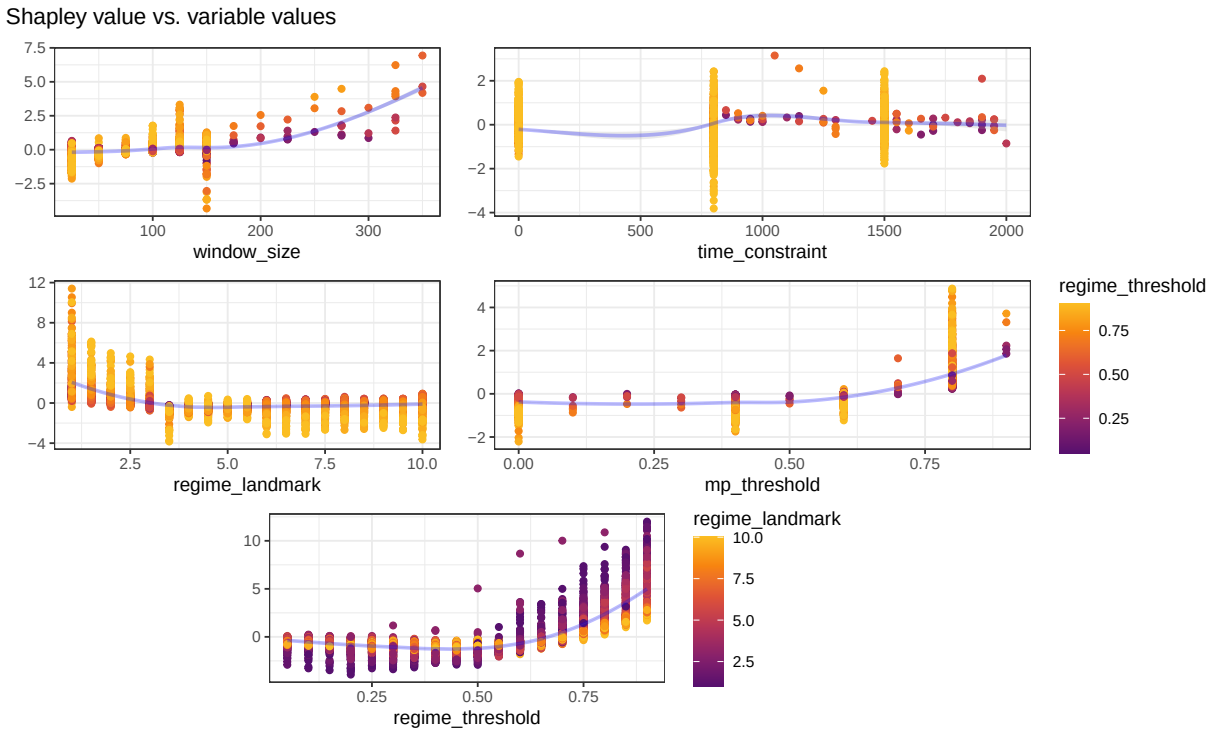


Figure 22: This shows the effect each variable has on the FLOSS score, taking into account the interactions.

According to the FLOSS paper[47], the `window_size` is indeed a feature that can be tuned; nevertheless, the results appear to be similar in a reasonably wide range of window sizes, up to a limit, consistent with our findings.

7.1.5 Visualizing the predictions

At this point, the grid search tested a total of 23,389 models with resulting (individual) scores from 0.0002 to 1669.83 (Q25: 0.9838, Q50: 1.8093, Q75: 3.3890).

By recording

First, we will visualize how the models (in general) performed throughout the individual recordings.

Fig. 23 shows a violin plot of equal areas clipped to the minimum value. The blue color indicates the recordings with a small IQR (interquartile range) of model scores. We see on the left half 10% of the recordings with the worst minimum score, and on the right half, 10% of the recordings with the best minimum score.

Next, we will visualize some of these predictions to understand why some recordings were difficult to segment. For us to have a simple baseline: a recording with just one regime change, and the model predicts exactly one regime change, but far from the truth, the score will be roughly 1.

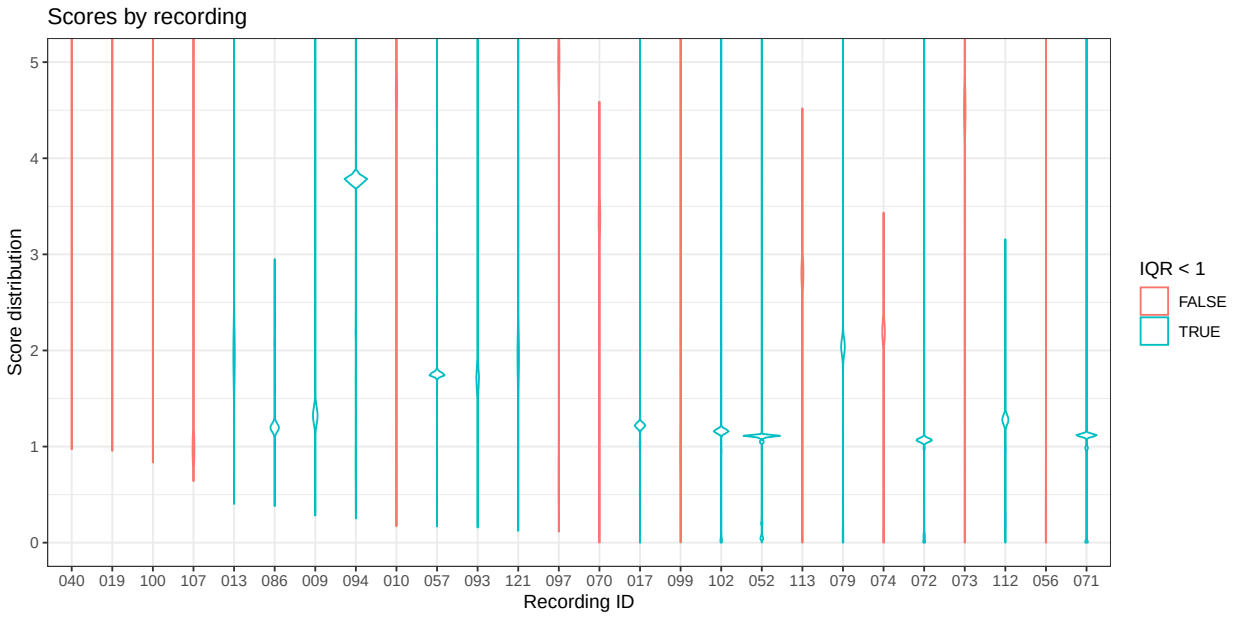


Figure 23: Violin plot showing the distribution of the FLOSS score achieved by all tested models by recording. The left half shows the recordings that were difficult to predict (10% overall), whereas the right half shows the recordings that at least one model could achieve a good prediction (10% overall). The recordings are sorted (left-right) by the minimum (best) score achieved in descending order, and ties are sorted by the median of all recording scores. The blue color highlights recordings where models had an IQR variability of less than one. As a simple example, a recording with just one regime change, and the model predicts exactly one change, far from the truth, the score will be roughly 1.

Fig. 24 shows the best effort in predicting the most complex recordings. One information not declared before is that if the model does not predict any change, it will put a mark on the zero position. On the other side, the truth markers positioned at the beginning and the end of the recording were removed, as these locations lack information and do not represent a streaming setting.

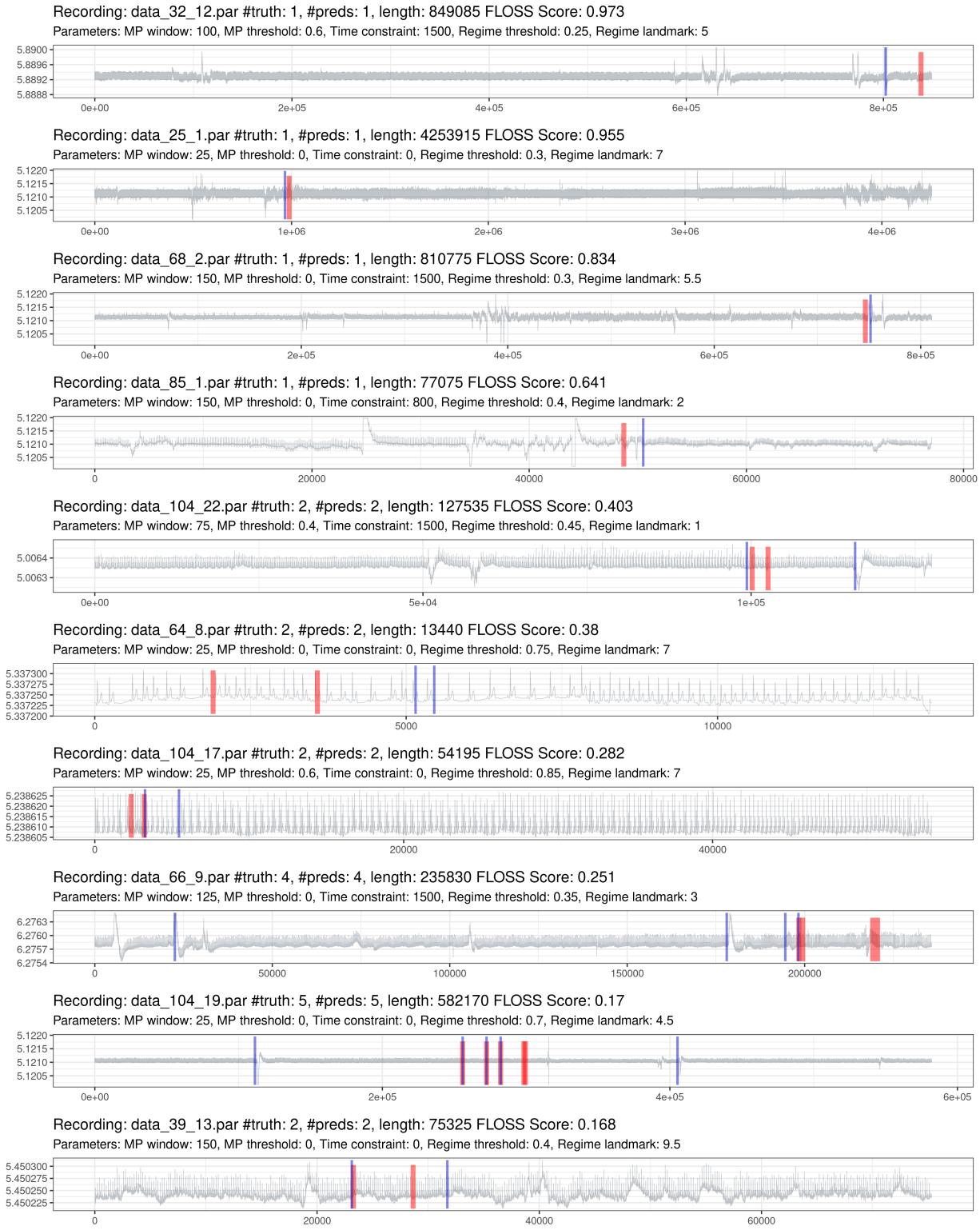


Figure 24: Prediction of the worst 10% of recordings (red is the truth, blue are the predictions).

Fig. 25 shows the best performances of the best recordings. Notice that there are recordings

with a significant duration and few regime changes, making it hard for a "trivial model" to predict randomly.

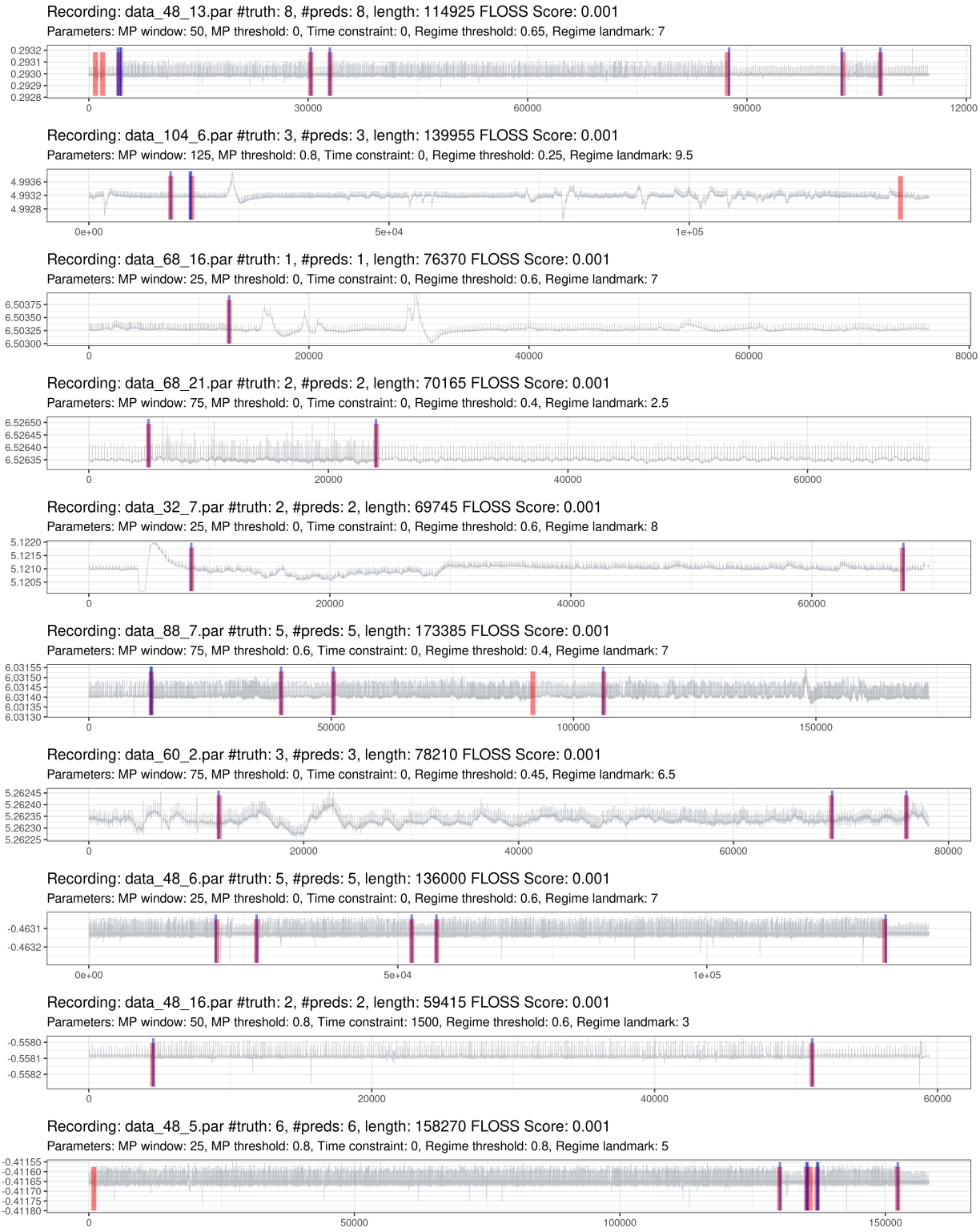


Figure 25: Prediction of the best 10% of recordings (red is the truth, blue are the predictions).

By model

Fig. 26 shows the distribution of the FLOSS score of the 10% worst (left side) and 10% best models across the recordings (right side). The bluish color highlights the models with SD below 3 and IQR below 1.

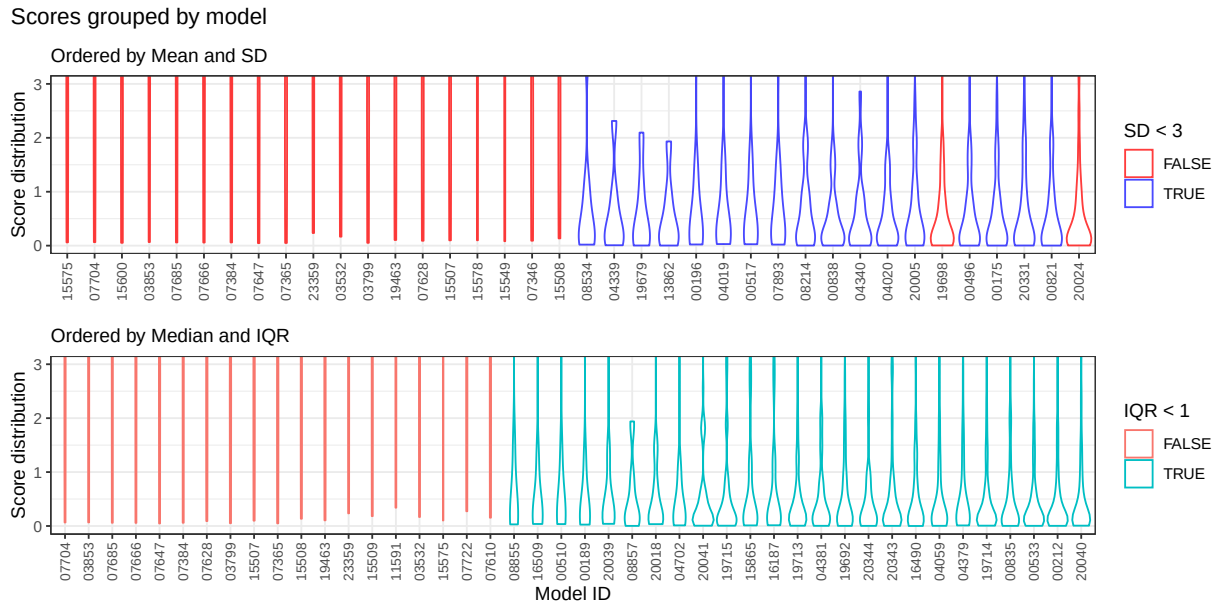


Figure 26: Violin plot showing the distribution of the FLOSS score achieved by all tested models during the inner ressample. The left half shows the models with the worst performances (10% overall), whereas the right half shows the models with the best performances (10% overall). The models are sorted (left-right) by the mean score (top) and by the median (below). Ties are sorted by the SD and IQR, respectively. The bluish colors highlights models with an SD below 3 and IQR below 1.

Fig. 27 the performance of the six best models. They are ordered from left to right, from the worst record to the best record. The top model is the one with the lowest mean across the scores. The blue line indicates the mean score, and the red line the median score. The scores above 3 are squished in the plot and colored according to the scale in the legend.

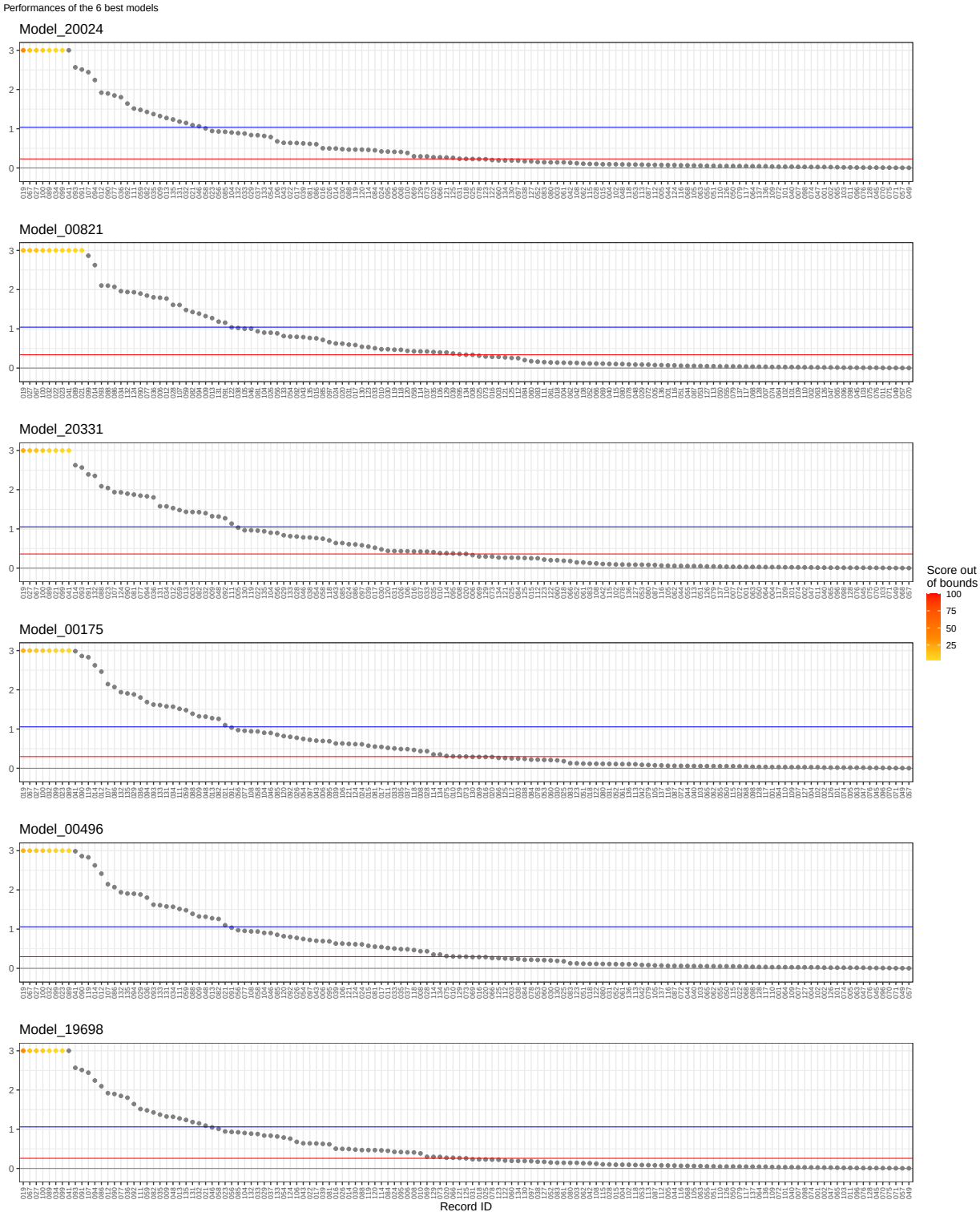


Figure 27: Performances of the best 6 models across all inner resample of recordings. The recordings are ordered by score, from the worst to the best. Each plot shows one model, starting from the best one. The red line indicates the median score of the model. The blue line indicates the mean score of the model. The gray line limits the zero-score region. The plot is limited on the "y" axis, and the scores above this limit are shown in color.

The Holdout

Finally, Table 5 shows a summary of the best five models across all the inner resample (cross-validation). The column `mean` shows the average score, and column `std_err` shows the standard error of the mean. The column `holdout` shows the final score of this model on the holdout set (outer resample).

Table 5: Summary of the five best models. The `mean` shows the inner resample average score. The `holdout` shows the final score of the model on the holdout set (outer resample).

window_size	regime_threshold	regime_landmark	mean	std_err	holdout
150	0.45	9.0	1.08	0.49	**0.66**
150	0.35	5.0	1.10	0.57	**0.80**
100	0.50	9.5	1.10	0.48	**0.62**
125	0.45	7.5	1.11	0.55	**0.57**
125	0.45	7.0	1.11	0.54	**0.61**

7.2 Classification

As described in Section 6.4.5, the classification algorithm is based on the Contrast Profile (CP)[54]. The CP allows us to detect sequences that at the same time very *similar* to its neighbors in class A while is very *different* from the nearest neighbor from class B.

The variables parameters we can tune are the following:

- **`shapelet_size`**: that we will use interchangeably with the term *window_size* as is defines the size of the rolling window used to compute the CP. It was used a series of shapelet sizes with exponential distribution, resulting on `20` values from `21` to `401`.
- **`top_k`**: how many shapelets we will select on each CP, being the first one the shapelet with the highest constrast value. It was chosen the default value of `10`.
- **`max_shapelets`**: the maximum number of shapelets we will allow on selecting the shapelet set. It was set as `20` to allow more freedom on the selection of shapelets.
- **`max_redundance`**: the maximum number of "redundance" we will allow on selecting the shapelet set. The redundance means more than one shapelet can classify correctly the same observation. It was set as `10`, to also allow more freedom on the selection of shapelets.

Fig. 28 shows the workflow using Nested resamplig as described on Section 6.4.6.

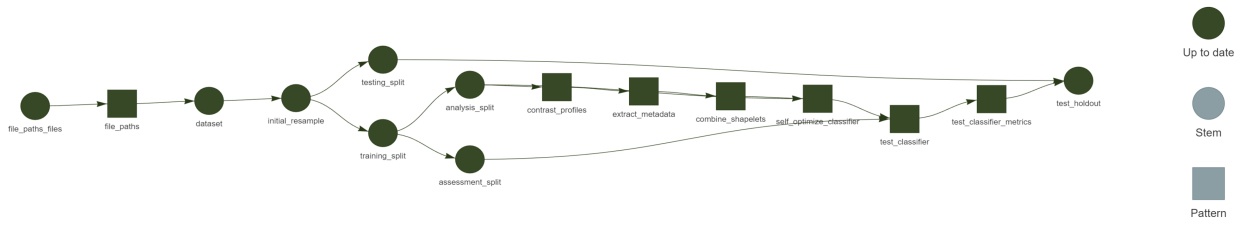


Figure 28: Classification pipeline.

The dataset used for working Classification algorithm was the CinC/Physionet Challenge 2015 was about "Reducing False Arrhythmia Alarms in the ICU[6].

The selected records were those that contain ventricular tachycardia. The last 10 seconds (at 250hz) of all records were selected and grouped as `TRUE` alarm and `FALSE` alarm. A total of 331 records were used, being 245 `FALSE` alarms, and 86 `TRUE` alarms.

The records were split in a proportion of 3/4 (248) for the training set (inner resampling) and 1/4 (83) for the test set (outer resampling). The proportions of `TRUE` and `FALSE` alarms were similar to the original dataset: 184 `FALSE` alarms and 64 `TRUE` alarms in the training set, and 61 `FALSE` alarms and 22 `TRUE` alarms in the test set. The inner resampling was performed using a 5-fold cross-validation.

In order to compute the Contrast Profile (CP), on each fold, the `TRUE` alarms were concatenated in one single time-series with a small gap of 300 observation of random noise in order to isolate each alarm. The same was done for the `FALSE` alarms.

The following steps were performed for each fold:

1. The CP was computed with several shapelet sizes (from 21 to 401), and top 10 best shapelet candidates were stored based on the CP values.
2. Every shapelet candidate was evaluated for the hability to classify as `TRUE` or `FALSE` alarm along all the concatenated time-series.

    a. First, the distance profile of the shapelet was computed against the `FALSE` time-series, and threshold was set in order to not detect any `FALSE` alarm as `TRUE` alarm.

    b. Second, the distance profile of the shapelet was computed against the `TRUE` time-series, and using the threshold computed in the previous step, the number of `TRUE` alarms detected was recorded and called the "coverage" of the shapelet.

3. With this information, a Beam Search was performed in order to select the best set of shapelets

that maximize the coverage and minimize the number of shapelets. The set of shapelets may contain shapelets of different sizes.

4. The confusion matrix of these sets of shapelets was computed. Other metrics were also computed, such as the F1-score, the accuracy, the precision, the specificity, the MCC, the κ_m and the Kappa.

An example of candidates for ventricular tachycardia is presented on Fig. 29.

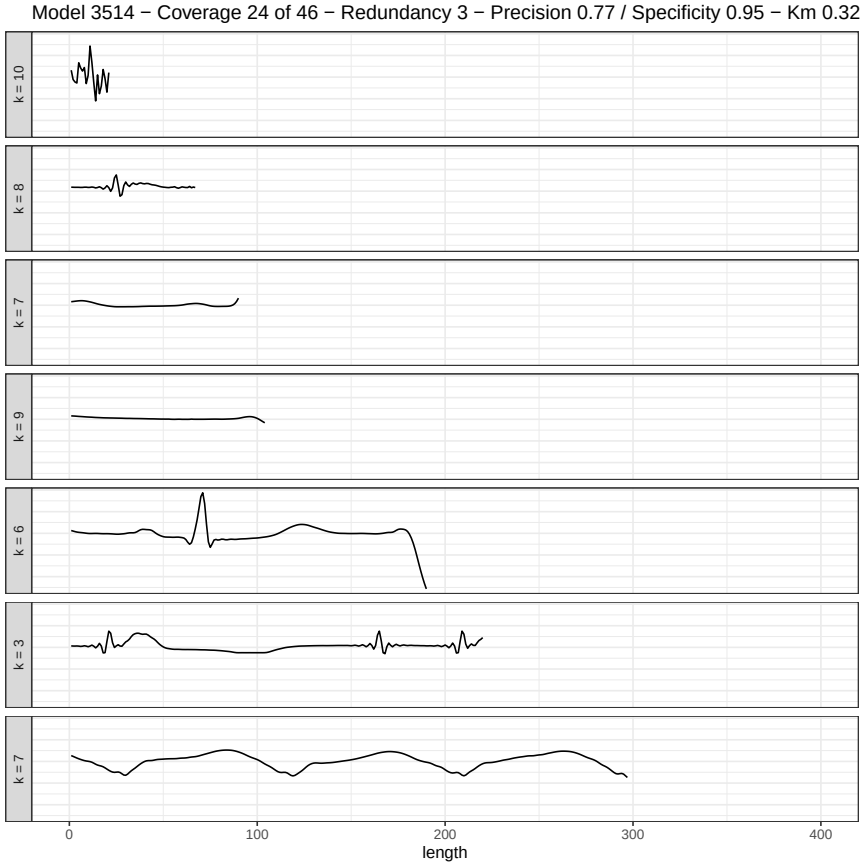


Figure 29: Set of shapelets for the classification task. Above shows the model number, the coverage (the proportion of true alarms that were detected) and redundancy during the analysis_split (inner resample), The Precision, the Specificity, and the Km during the testing_split (outer resample).

After the Inner Resampling is done, the best sets of shapelets are selected and evaluated on the Test Set without retraining a new Contrast Profile. Thus assessing the generalization of the shapelet set on new data.

The criteria to select the best sets of shapelets was described on Section 6.5.2 being the Precision the ranking criteria. It was also required that the set being present on more than one fold and in both repetitions. Also, the sets of shapelets that had a negative κ_m were discarded.

The following results were obtained:

Table 6: Performance of the best five sets of shapelet on the inner resample.

tp	fp	tn	fn	precision	recall	specificity	accuracy	f1	mcc	km	kappa
11	1	30	7	0.92	0.61	0.97	0.84	0.73	0.65	0.56	0.62
11	1	30	7	0.92	0.61	0.97	0.84	0.73	0.65	0.56	0.62
10	1	30	8	0.91	0.56	0.97	0.82	0.69	0.60	0.50	0.57
13	2	29	5	0.87	0.72	0.94	0.86	0.79	0.69	0.61	0.68
13	3	28	5	0.81	0.72	0.90	0.84	0.76	0.64	0.56	0.64

Table 7: Performance of the best five sets of shapelet on outer resample.

tp	fp	tn	fn	precision	recall	specificity	accuracy	f1	mcc	km	kappa
9	3	58	13	0.75	0.41	0.95	0.81	0.53	0.45	0.27	0.42
10	1	60	12	0.91	0.45	0.98	0.84	0.61	0.57	0.41	0.52
8	5	56	14	0.62	0.36	0.92	0.77	0.46	0.34	0.14	0.32
10	3	58	12	0.77	0.45	0.95	0.82	0.57	0.49	0.32	0.47
11	5	56	11	0.69	0.50	0.92	0.81	0.58	0.47	0.27	0.46

Table 8: The aggregated performance of the best five sets of shapelet on the outer resample. Median, Q25 and Q75

precision	recall	specificity	accuracy	f1_micro	f1_macro	mcc	km	kappa
0.75	0.46	0.95	0.81	0.72	0.55	0.47	0.27	0.46
0.69	0.41	0.92	0.81	0.72	0.55	0.45	0.27	0.42
0.77	0.46	0.95	0.82	0.72	0.55	0.49	0.32	0.47

7.3 Feasibility trial

A side-project called "false.alarm.io" has been derived from this work (an unfortunate mix of "false.alarm" and "PlatformIO"[73], the IDE chosen to interface the panoply of embedded systems we can experiment with). The current results of this side-project are very enlightening and show that the final algorithm can indeed be used in small hardware. Further data will be available in the future.

A brief mentioning, linking back to the objectives of this work, an initial trial was done using an ESP32 MCU (Fig. 30) in order to be sure if such small device can handle the task.

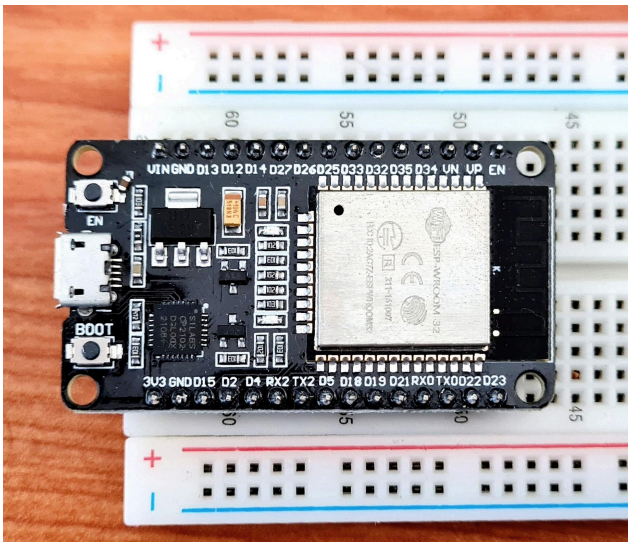


Figure 30: ESP32 MCU

Current results show that such device has enough computation power to handle the task in real-time using just one of its two microprocessors. The main limitation seen in advance is the on-chip SRAM that must be well managed.

8 Scientific contributions

8.1 Matrix Profile

Since the first paper presenting this new concept[22], lots of investigations were made to speed up its computation. It is notable how all computations are not dependent on the *rolling window size* as previous works not using Matrix Profile. Aside from this, we can see that the first STAMP[22] algorithm has the time complexity of $O(n^2 logn)$ while STOMP[74] $O(n^2)$ (a significant improvement), but STOMP lacks the "any-time" property. Later SCRIMP[75] solves this problem keeping the same time complexity of $O(n^2)$. Here we are in the "exact" algorithms domain and we will not extend the scope for conciseness.

The main issue with the algorithms above is the dependency on a fast Fourier transform (FFT) library. FFT has been extensively optimized and architecture/CPU bounded to exploit the most of speed. Also, padding data to some power of 2 happens to increase the efficiency of the algorithm. We can argue that time complexity doesn't mean "faster" when we can exploit low-level instructions.

In our case, using FFT in a low-power device is overkilling. For example, a quick search over the internet gives us a hint that computing FFT on a 4096 data in an ESP32 takes about 21ms (~47 computations in 1 second). This means ~79 seconds for computing all FFT's (~3797) required for STAMP using a window of 300. Currently, we can compute a full matrix of 5k data in about 9 seconds in an ESP32 MCU (Fig. 30), and keep updating it as fast as 1 min of data (at 250hz) in just 6 seconds.

Recent works using *exact* algorithms are using an unpublished algorithm called **MPX**, which computes the Matrix Profile using cross-correlation methods ending up faster and is easily portable.

On computing the Matrix Profile: the contribution of this work on this area is adding the *Online* capability to MPX, which means we can update the Matrix Profile as new data comes in.

On extending the Matrix Profile: the contribution of this work on this area is the use of an unexplored constraint that we could apply on building the Matrix Profile we are calling *Similarity Threshold* (ST). The original work outputs the similarity values in Euclidean Distance (ED) values, while MPX naturally outputs the values in Pearson's correlation coefficients (CC). Both ED and CC are interchangeable using the Equation 9. However, we may argue that it is easier to compare values that do not depend on the window size during an exploratory phase. MPX happens to naturally return values in CC, saving a few more computation time. The ST is an interesting factor that we can use, especially when detecting pattern changes during time. The FLOSS algorithm relies on counting references between indexes in the time series. ST can help remove "noise" from these references since only similar patterns above a certain threshold are referenced, and changes have more impact on these counts. The best ST threshold is still to be determined.

$$CC = 1 - \frac{ED}{(2 \times WindowSize)} \tag{9}$$

8.2 Regime change detection

In the original paper, in chapter 3.5, the authors of FLOSS wisely introduce the **temporal constraint**, which improves the sensitivity of regime change detection on situations where a regime may alternate in short periods of time.

Nevertheless, the authors declare the correction curve typically used on the algorithm as "simply a uniform distribution", but this is not an accurate statement. the *Arc Counts* of newly incoming data are truncated by the same amount of temporal constraint. This prevents completely the

detection of a regime change in the last 10 seconds as this thesis requires.

The main contribution of this work on this area is overcoming this issue by computing the theoretical distribution using the temporal constraint parameters beforehand. as shown in Fig. 31. That gives us enough data to evaluate a regime change accurately utilizing a minimum of $2 \times WindowSize$ datapoints.

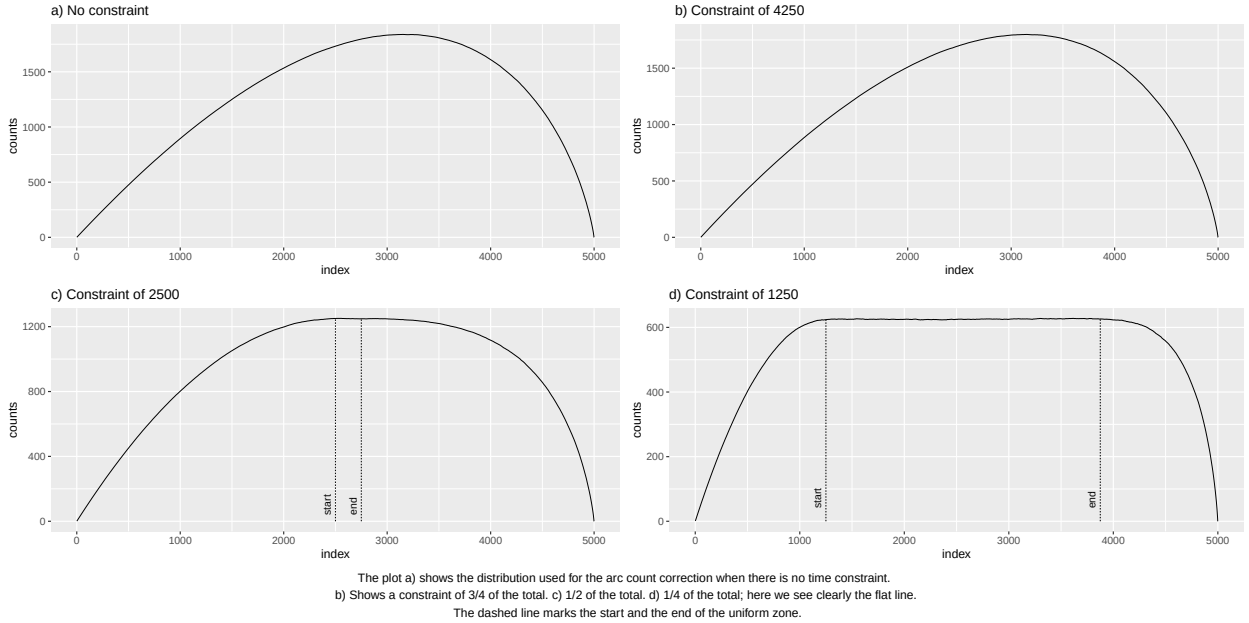


Figure 31: 1D-IAC distributions for earlier temporal constraint (on Matrix Profile)

9 Scientific outcomes

This research has already yielded two R packages concerning the MP algorithms from UCR[26]. The first package is called `tsmp`, and a paper has also been published in the R Journal[27] (Journal Impact Factor™, 2020 of 3.984). The second package is called `matrixprofiler` and enhances the first one, using low-level language to improve computational speed. The author has also joined the Matrix Profile Foundation as co-founder together with contributors from Python and Go languages[28,29]. The benchmarks of the R implementation are available online[76].

Additionally to the above publication and the publication of the ongoing literature survey it is planned to be published two articles about this thesis subject. The first regarding the application of the FLOSS algorithm on real-time ECG showing its potential on using it on low-power devices. The second regarding the use of combined shapelets for relevant ECG patterns identification.

10 Expected results and outcomes

At the end, this thesis will provide a framework for identify life-threatening conditions using biological streaming data on devices with low CPU and low memory specifications. We expect to achieve a high quality model on identifying these pathological conditions, maintaining its robustness in presence of noise and artifacts seen on real-world applications.

11 Research team

- Thesis Author: Francisco Bischoff
- Supervisor: Professor Pedro Pereira Rodrigues
- Co-supervisor: Professor Eamonn Keogh (UCR, Riverside)

References

1. Donchin Y, Seagull FJ. The hostile environment of the intensive care unit. *Current Opinion in Critical Care*. 2002;8(4):316-320. doi:10.1097/00075198-200208000-00008
2. Sendelbach S, Funk M. Alarm Fatigue. *AACN Advanced Critical Care*. 2013;24(4):378-386. doi:10.4037/nci.0b013e3182a903f9
3. The joint commission. Published 2021. Accessed April 8, 2021. https://www.jointcommission.org
4. Joint Commission. Sentinel event alert - Medical device alarm safety in hospitals. 2013;(50):1-3.
5. The joint commission - national patient safety goals. Published 2021. Accessed April 8, 2021. https://www.jointcommission.org/standards/national-patient-safety-goals/hospital-national-patient-safety-goals/
6. Clifford GD, Silva I, Moody B, et al. The PhysioNet/computing in cardiology challenge 2015: Reducing false arrhythmia alarms in the ICU. In: *Computing in Cardiology.*; 2015. doi:10.1109/cic.2015.7408639
7. Lawless ST. Crying wolf: False alarms in a pediatric intensive care unit. *Critical care medicine*. 1994;22(6):981-985.

8. Chambrin MC. Alarms in the intensive care unit: How can the number of false alarms be reduced? *Critical care (London, England)*. 2001;5(4):184-188. doi:10.1186/cc1021

9. Parthasarathy S, Tobin MJ. Sleep in the intensive care unit. *Intensive Care Medicine*. 2004;30(2):197-206. doi:10.1007/s00134-003-2030-6

10. Research compendium. Published 2019. Accessed April 8, 2021. https://research-compendium.science

11. Wilkinson MD, Dumontier M, Aalbersberg IjJ, et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*. 2016;3(1). doi:10.1038/sdata.2016.18

12. The CodeMeta project. Published 2017. Accessed January 10, 2022. https://codemeta.github.io/

13. Landau W, Landau W, Warkentin MT, et al. *Ropensci/Targets, Dynamic Function-Oriented 'Make'-Like Declarative Workflows*. Zenodo; 2021. doi:10.5281/ZENODO.4062936

14. Franzbischoff/false.alarm: Reproducible reports. Published 2021. Accessed April 8, 2021. https://franzbischoff.github.io/false.alarm

15. Blischak JD, Carbonetto P, Stephens M. Creating and sharing reproducible research code the workflowr way [version 1; peer review: 3 approved]. *F1000Research*. 2019;8(1749). doi:10.12688/f1000research.20843.1

16. Kuhn M. Building predictive models in r using the caret package. *Journal of Statistical Software, Articles*. 2008;28(5):1-26. doi:10.18637/jss.v028.i05

17. Kuhn M, Wickham H. *Tidymodels: A Collection of Packages for Modeling and Machine Learning Using Tidyverse Principles.*; 2020. https://www.tidymodels.org

18. Thompson J. On not using tidymodels. Published October 2020. Accessed January 5, 2022. https://staffblogs.le.ac.uk/teachingr/2020/10/05/on-not-using-tidymodels/

19. Bischoff F. GitHub false.alarm repository. Accessed July 14, 2021. https://github.com/franzbischoff/false.alarm

20. GitHub Actions. Accessed July 14, 2021. https://github.com/features/actions

21. Zenhub roadmap. Accessed January 27, 2022. https://app.zenhub.com/workspaces/phd-thesis-5eb2ce34f5f30b3aed0a35af/roadmap

22. Yeh C-CM, Zhu Y, Ulanova L, et al. Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In: *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE; 2016:1317-1322. doi:10.1109/ICDM.2016.0179

23. De Paepe D, Vanden Hautte S, Steenwinckel B, et al. A generalized matrix profile framework with support for contextual series analysis. *Engineering Applications of Artificial Intelligence*. 2020;90(January):103487. doi:10.1016/j.engappai.2020.103487

24. Feremans L, Vercruyssen V, Cule B, Meert W, Goethals B. Pattern-Based Anomaly Detection in Mixed-Type Time Series. In: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol 11906 LNAI.; 2020:240-256. doi:10.1007/978-3-030-46150-8_15

25. Lin J, Keogh E, Wei L, Lonardi S. Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery*. 2007;15(2):107-144. doi:10.1007/s10618-007-0064-z

26. UCR Matrix Profile Page. Accessed January 27, 2022. https://www.cs.ucr.edu/~eamonn/MatrixProfile.html

27. Bischoff F, Rodrigues PP. tsmp: An R Package for Time Series with Matrix Profile. *The R Journal*. 2020;12(1):76-86. doi:10.32614/RJ-2020-021

28. Matrix Profile Foundation. Accessed January 27, 2022. https://matrixprofile.org/

29. Van Benschoten A, Ouyang A, Bischoff F, Marrs T. MPA: A novel cross-language API for time series analysis. *Journal of Open Source Software*. 2020;5(49):2179. doi:10.21105/joss.02179

30. Reducing False Arrhythmia Alarms in the ICU - The PhysioNet Computing in Cardiology Challenge 2015. Published online March 24, 2021. doi:10.5281/zenodo.4634013

31. Association for the Advancement of Medical Instrumentation. *Cardiac monitors, heart rate meters, and alarms*. Association for the Advancement of Medical Instrumentation; 2002.

32. Plesinger F, Klimes P, Halamek J, Jurak P. False alarms in intensive care unit monitors: Detection of life-threatening arrhythmias using elementary algebra, descriptive statistics and fuzzy logic. In: IEEE; 2015. doi:10.1109/cic.2015.7408641

33. Kalidas V, Tamil LS. Enhancing accuracy of arrhythmia classification by combining logical and machine learning techniques. In: IEEE; 2015. doi:10.1109/cic.2015.7411015

34. Couto P, Ramalho R, Rodrigues R. Suppression of false arrhythmia alarms using ECG and pulsatile waveforms. In: IEEE; 2015. doi:10.1109/cic.2015.7411019

35. Fallet S, Yazdani S, Vesin J-M. A multimodal approach to reduce false arrhythmia alarms in the intensive care unit. In: IEEE; 2015. doi:10.1109/cic.2015.7408640

36. Hoog Antink C, Leonhardt S. Reducing false arrhythmia alarms using robust interval estimation and machine learning. In: IEEE; 2015. doi:10.1109/cic.2015.7408642

37. Akosa JS. Predictive accuracy: A misleading performance measure for highly imbalanced data. *SAS Global Forum*. 2017;942:1-12.

38. Wu R, Keogh E. Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress. *IEEE Transactions on Knowledge and Data Engineering*. Published online September 2021. doi:10.1109/TKDE.2021.3112126

39. Bakeman R, Quera V. *Sequential Analysis and Observational Methods for the Behavioral Sciences*. Cambridge University Press; 2011:1-183. doi:10.1017/CBO9781139017343

40. Sim J, Wright CC. The Kappa Statistic in Reliability Studies: Use, Interpretation, and Sample Size Requirements. *Physical Therapy*. 2005;85(3):257-268. doi:10.1093/ptj/85.3.257

41. Bakeman R, McArthur D, Quera V, Robinson BF. Detecting sequential patterns and determining their reliability with fallible observers. *Psychological Methods*. 1997;2(4):357-370. doi:10.1037/1082-989X.2.4.357

42. Morgan S. *Research Methodology and Statistical Methods*.; 2019:300.

43. SparkFun Electronics. AD8232 single lead heart rate monitor. Published 2014. Accessed July 14, 2021. https://www.sparkfun.com/products/12650

44. Analog Devices. AD8232 Single-Lead, Heart Rate Monitor Front End. Published online 2014. Accessed July 14, 2021. https://www.analog.com/media/en/technical-documentation/data-sheets/ad8232.pdf

45. Arduino. Arduino. Published 2008. Accessed July 14, 2021. https://www.arduino.cc/

46. Eerikainen LM, Vanschoren J, Rooijakkers MJ, Vullings R, Aarts RM. Decreasing the false alarm rate of arrhythmias in intensive care using a machine learning approach. In: IEEE; 2015. doi:10.1109/cic.2015.7408644

47. Gharghabi S, Yeh C-CM, Ding Y, et al. Domain agnostic online semantic segmentation for multi-dimensional time series. *Data Mining and Knowledge Discovery*. 2018;33(1):96-130. doi:10.1007/s10618-018-0589-3

48. Aminikhanghahi S, Cook DJ. A survey of methods for time series change point detection. *Knowledge and Information Systems*. 2016;51(2):339-367. doi:10.1007/s10115-016-0987-z

49. Matsubara Y, Sakurai Y, Faloutsos C. AutoPlait: Automatic mining of co-evolving time sequences. *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Published online 2014:193-204. doi:10.1145/2588555.2588556

50. Imani S, Madrid F, Ding W, Crouter S, Keogh E. Matrix profile XIII : Time series snippets : A new primitive for time series data mining. In: *2018 IEEE International Conference on Data Mining (ICDM)*.; 2018.

51. Gharghabi S, Imani S, Bagnall A, Darvishzadeh A, Keogh E. Matrix profile XII: MPdist: A novel time series distance measure to allow data mining in more challenging scenarios. In: IEEE; 2018:965-970. doi:10.1109/ICDM.2018.00119

52. Keogh E, Lin J. Clustering of time-series subsequences is meaningless: implications for previous and future research. *Knowledge and Information Systems*. 2005;8(2):154-177. doi:10.1007/s10115-004-0172-7

53. Rakthanmanon T, Keogh E. Fast shapelets: A scalable algorithm for discovering time series shapelets. In: *Proceedings of the 2013 SIAM International Conference on Data Mining*. Society for Industrial; Applied Mathematics; 2013:668-676. doi:10.1137/1.9781611972832.74

54. Mercer R, Alaee S, Abdoli A, Singh S, Murillo A, Keogh E. Matrix profile XXIII: Contrast profile: A novel time series primitive that allows real world classification. In: *2021 IEEE International Conference on Data Mining (ICDM)*.; 2021:1240-1245. doi:10.1109/ICDM51629.2021.00151

55. Bischl B, Mersmann O, Trautmann H, Weihs C. Resampling Methods for Meta-Model Validation with Recommendations for Evolutionary Computation. *Evolutionary Computation*. 2012;20(2):249-275. doi:10.1162/EVCO_a_00069

56. Hastie T, Tibshirani R, Friedman J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer; 2009. doi:10.1007/978-0-387-84858-7

57. Bekkar M, Djemaa HK, Alitouche TA. Evaluation Measures for Models Assessment over Imbalanced Data Sets. *Journal of Information Engineering and Applications*. 2013;3(10):27-38. http://www.iiste.org/Journals/index.php/JIEA/article/view/7633

58. Chicco D, Jurman G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*. 2020;21(1):6. doi:10.1186/s12864-019-6413-7

59. Matthews BW. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*. 1975;405(2):442-451. doi:10.1016/0005-2795(75)90109-9

60. Bifet A, de Francisci Morales G, Read J, Holmes G, Pfahringer B. Efficient Online Evaluation of Big Data Stream Classifiers. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Vol 2015-Augus. ACM; 2015:59-68. doi:10.1145/2783258.2783372

61. Dubey A, Tarar S. Evaluation of approximate rank-order clustering using matthews correlation coefficient. *International Journal of Engineering and Advanced Technology*. 2018;8(2):106-113.

62. Delgado R, Tibau X-A. Why Cohen's Kappa should be avoided as performance measure in classification. Gu Q, ed. *PLOS ONE*. 2019;14(9):e0222916. doi:10.1371/journal.pone.0222916

63. Paroxysmal Atrial Fibrillation Events Detection from Dynamic ECG Recordings - The 4th China Physiological Signal Challenge 2021. Published online May 2021. doi:10.5281/zenodo.6879232

64. Wei P, Lu Z, Song J. Variable importance analysis: A comprehensive review. *Reliability Engineering & System Safety*. 2015;142:399-432. doi:10.1016/j.ress.2015.05.018

65. Chipman HA, George EI, McCulloch RE. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*. 2010;4(1):266-298. doi:10.1214/09-AOAS285

66. Greenwell BM, Boehmke BC, McCarthy AJ. A simple and effective model-based variable importance measure. Published online 2018. doi:10.48550/arxiv.1805.04755

67. Greenwell BM, Boehmke BC. Variable Importance Plots-An Introduction to the vip Package. *R Journal*. 2020;12(1):343-366. doi:10.32614/rj-2020-013

68. Molnar C. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. 2nd ed.; 2022:329. https://christophm.github.io/interpretable-ml-book

69. Breiman L. Random Forests. *Machine Learning*. 2001;45(1):5-32. doi:10.1023/A:1010933404324

70. Fisher A, Rudin C, Dominici F. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *Journal of machine learning research : JMLR*. 2019;20(Vi). http://arxiv.org/abs/1801.01489 http://www.ncbi.nlm.nih.gov/pubmed/34335110 http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC8323609

71. Lundberg SM, Lee SI. A unified approach to interpreting model predictions. In: *Advances in Neural Information Processing Systems*. Vol 2017-Decem.; 2017:4766-4775. https://arxiv.org/abs/1705.07874

72. Beyer K, Goldstein J, Ramakrishnan R, Shaft U. When Is "Nearest Neighbor" Meaningful? In: *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol 1540.; 1999:217-235. doi:10.1007/3-540-49257-7_15

73. PlatformIO, a professional collaborative platform for embedded development. Accessed January 5, 2022. https://platformio.org/

74. Zhu Y, Zimmerman Z, Senobari NS, et al. 2016 IEEE 16th international conference on data mining (ICDM). In: IEEE; 2016. doi:10.1109/icdm.2016.0085

75. Zhu Y, Yeh C-CM, Zimmerman Z, Kamgar K, Keogh E. 2018 IEEE international conference on data mining (ICDM). In: IEEE; 2018. doi:10.1109/icdm.2018.00099

76. Bischoff F. RPubs - MatrixProfileR - benchmarks. Published 2021. Accessed January 12, 2022. https://rpubs.com/franzbischoff/matrixprofiler