From Local to Global Explainability in Time Series Classification: Segmentation-Based SHAP and Hierarchical Explanations

Iro Skandali

Master Thesis

— ♦ —

Ioannina, March 2026

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

From Local to Global Explainability in Time Series Classification: Segmentation-Based SHAP and Hierarchical Explanations

A Thesis

submitted to the designated

by the Assembly

of the Department of Computer Science and Engineering

Examination Committee

by

Iro Skandali

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN DATA AND COMPUTER SYSTEMS ENGINEERING

WITH SPECIALIZATION

IN DATA SCIENCE AND ENGINEERING

University of Ioannina

School of Engineering

Ioannina 2026

Examining Committee:

- **Evaggelia Pitoura**, Professor, Department of Computer Science and Engineering, University of Ioannina (Advisor)
- **Aristidis Likas**, Professor, Department of Computer Science and Engineering, University of Ioannina
- **Panayiotis Tsaparas**, Associate Professor, Department of Computer Science and Engineering, University of Ioannina

Dedication

This thesis is dedicated to my parents, Ourania and Ermis, and my beloved siblings, Elisavet and Paris, for their unconditional love; and to Vasilis, for his endless support and for being my strength throughout it all.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Evaggelia Pitoura, for her invaluable guidance, encouragement, and support throughout this research. Her expertise, feedback, patience, and trust were essential to the completion of this thesis, and her constructive suggestions consistently helped me improve both the direction and the quality of my work.

I would also like to sincerely thank the PhD candidate, Christos Fragkathoulas, for his valuable help and collaboration.

I am especially grateful to Vasilis for his constant support and belief in me—particularly during difficult moments, when he believed in me even more than I did.

Last but not least, I would like to thank my family and friends for their unwavering support, understanding, and encouragement throughout this journey.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Iro Skandali, M.Sc. in Data and Computer Systems Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, 2026.
From Local to Global Explainability in Time Series Classification: Segmentation-Based SHAP and Hierarchical Explanations.
Advisor: Evaggelia Pitoura,Professor.

In this thesis, we study the problem of explaining univariate time series classification in diverse application domains, such as healthcare and finance. We consider both local explanations for individual instances and global explanations that summarize important regions at the dataset or class level.

The challenge in time series explanation lies in temporal dependencies between values. Thus, treating each time point as an independent feature, or aggregating time points using windows of arbitrary length, would lead to a loss of temporal structure. This issue is addressed through segmentation, which finds meaningful segments. On top of these segments, we apply our SHAP-based method to estimate the contribution of each segment. The explanation output returns scores/SHAP values per segment, where the sign indicates whether a segment supports or contradicts the predicted class. For better granularity, we propose a variant of our method that provides a hierarchical explanation, *i.e.*, within the highest-contributing segments, which subsegments are most influential.

Moreover, we extend our method with two algorithms for global explanations: either by aggregating local explanations, or by treating the whole dataset as a multivariate instance for the segmentation step.

Finally, we evaluate against baselines in terms of faithfulness (how much removing the top-ranked segments affects the classifier's output), robustness (how stable the explanations remain under noise perturbations), and runtime (computational cost).

Εκτεταμενη Περιληψη

Ηρώ Σκανδάλη, Δ.Μ.Σ. στη Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, 2026.

Από τις Τοπικές στις Συνολικές Εξηγήσεις για Ταξινόμηση Χρονοσειρών: SHAP με Τμηματοποίηση και Ιεραρχικές Εξηγήσεις.

Επιβλέπων: Πιτουρά Ευαγγελία, Καθηγήτρια.

Στην παρούσα διπλωματική εργασία μελετάται το πρόβλημα της εξηγησιμότητας στην ταξινόμηση μονοδιάστατων χρονοσειρών σε ποικίλα πεδία εφαρμογής, όπως η υγεία και τα οικονομικά.Στόχος είναι να παραχθούν εξηγήσεις που βοηθούν στην κατανόηση της απόφασης ενός ταξινομητή, τόσο σε επίπεδο μεμονωμένου δείγματος (τοπικές εξηγήσεις) όσο και σε πιο συνολικό επίπεδο που συνοψίζει ποια χρονικά τμήματα είναι γενικά σημαντικά σε ένα σύνολο δεδομένων ή σε μια συγκεκριμένη κλάση (global εξηγήσεις).

Η εξηγησιμότητα στις χρονοσειρές παρουσιάζει ιδιαίτερη δυσκολία λόγω των ισχυρών χρονικών εξαρτήσεων μεταξύ των τιμών. Σε αντίθεση με στατικά δεδομένα, όπου συχνά είναι λογικό να αντιμετωπίζονται τα χαρακτηριστικά ως ανεξάρτητα, στις χρονοσειρές η πληροφορία κωδικοποιείται σε διαδοχικές ακολουθίες και σε χρονικά μοτίβα. Επομένως, αν κάθε χρονικό σημείο θεωρηθεί ως ανεξάρτητο χαρακτηριστικό ή αν τα χρονικά σημεία ομαδοποιηθούν σε παράθυρα αυθαίρετου μήκους, είναι πιθανό να χαθεί κρίσιμη δομή του σήματος, όπως η μορφή, η διάρκεια και το μοτίβο των μεταβολών στο χρόνο. Για να αντιμετωπιστεί αυτό το πρόβλημα, η εργασία αξιοποιεί τμηματοποίηση (segmentation), δηλαδή διαδικασία που εντοπίζει "νοηματικά" και συνεκτικά χρονικά τμήματα τα οποία αντανακλούν καλύτερα τη φυσική δομή της χρονοσειράς.

Πάνω στα τμήματα που προκύπτουν από την τμηματοποίηση εφαρμόζεται μια μέθοδος βασισμένη στο SHAP, με σκοπό την εκτίμηση της συνεισφοράς κάθε segment

στην τελική πρόβλεψη του μοντέλου. Αντί να παράγονται αποδόσεις σημασίας ανά χρονικό σημείο, η μέθοδος αποδίδει βαθμολογίες/τιμές SHAP ανά segment. Με αυτόν τον τρόπο, η εξήγηση γίνεται πιο ερμηνεύσιμη, καθώς αναφέρεται σε συνεκτικές χρονικές περιοχές. Επιπλέον, το πρόσημο της τιμής SHAP επιτρέπει μια άμεση ερμηνεία: θετικές τιμές υποδεικνύουν ότι το συγκεκριμένο segment ενισχύει την πρόβλεψη της κλάσης που επιλέγει το μοντέλο, ενώ αρνητικές τιμές υποδεικνύουν ότι λειτουργεί ανταγωνιστικά ή "αντικρούει" την πρόβλεψη.

Για μεγαλύτερη λεπτομέρεια στην ερμηνεία, προτείνεται μια παραλλαγή της μεθόδου που παρέχει ιεραρχική εξήγηση. Η κεντρική ιδέα είναι ότι ένα segment μπορεί να είναι συνολικά σημαντικό, αλλά η πραγματική "πηγή" της σημαντικότητας να εντοπίζεται σε ένα μικρότερο υποτμήμα του. Η ιεραρχική εξήγηση εστιάζει στα segments με τη μεγαλύτερη συνεισφορά και εξετάζει περαιτέρω ποια υπο-segments μέσα σε αυτά είναι τα πιο επιδραστικά. Έτσι, η εξήγηση αποκτά καλύτερη χωρική/χρονική ανάλυση χωρίς να χάνει τη συνοχή που προσφέρει η τμηματοποίηση.

Πέρα από τις τοπικές εξηγήσεις, η εργασία επεκτείνει τη μεθοδολογία και σε global εξηγήσεις μέσω δύο εναλλακτικών αλγορίθμων. Στην πρώτη προσέγγιση, οι global εξηγήσεις προκύπτουν από συνάθροιση (aggregation) των τοπικών εξηγήσεων σε επίπεδο συνόλου δεδομένων ή κλάσης, ώστε να αναδειχθούν χρονικές περιοχές που επαναλαμβανόμενα εμφανίζονται ως σημαντικές. Στη δεύτερη προσέγγιση, το σύνολο δεδομένων αντιμετωπίζεται ως μία πολυδιάστατη (multivariate) "οντότητα" ειδικά στο βήμα της τμηματοποίησης, ώστε να εξαχθούν κοινά segments που αντανακλούν συνολικές δομές και όχι μεμονωμένες ιδιαιτερότητες ενός δείγματος. Με αυτόν τον τρόπο επιδιώκεται μια πιο συμπαγής και σταθερή απεικόνιση της σημαντικότητας σε επίπεδο dataset ή κλάσης.

Τέλος, η προτεινόμενη μεθοδολογία αξιολογείται πειραματικά σε σύγκριση με baseline προσεγγίσεις χρησιμοποιώντας τρεις βασικούς άξονες αξιολόγησης. Πρώτον, εξετάζεται η *faithfulness*, δηλαδή κατά πόσο οι εξηγήσεις αντανακλούν πραγματικά τη συμπεριφορά του μοντέλου: αυτό μετριέται μέσω του πόσο μεταβάλλεται η έξοδος/πιθανότητα του ταξινομητή όταν αφαιρούνται ή αντικαθίστανται τα κορυφαία (top-ranked) segments που η μέθοδος θεωρεί σημαντικά. Δεύτερον, αξιολογείται η *robustness*, δηλαδή η σταθερότητα των εξηγήσεων όταν το σήμα υφίσταται μικρές διαταραχές (noise perturbations): μια robust μέθοδος θα πρέπει να διατηρεί παρόμοια κατάταξη/σημαντικότητα segments υπό ήπιες αλλαγές στα δεδομένα. Τρίτον, αναλύεται ο υπολογιστικός χρόνος (runtime), ώστε να αποτυπωθεί το πρα-

κτικό κόστος της παραγωγής εξηγήσεων και η δυνατότητα εφαρμογής της μεθόδου σε ρεαλιστικά σενάρια.

Συνολικά, η εργασία προτείνει ένα segmentation-based πλαίσιο εξηγήσεων τύπου SHAP για χρονοσειρές, το οποίο υποστηρίζει τόσο τοπική όσο και συνολική ερμηνεία, ενώ η ιεραρχική εκδοχή ενισχύει περαιτέρω τη λεπτομέρεια της εξήγησης στα σημαντικότερα χρονικά τμήματα. Η πειραματική αξιολόγηση, βασισμένη στα κριτήρια *faithfulness* και *robustness*, καθώς και στη μελέτη του χρόνου εκτέλεσης, προσφέρει μια ολοκληρωμένη εικόνα της αποτελεσματικότητας και της πρακτικής χρησιμότητας της προσέγγισης.

Chapter 1

Introduction

1.1 Scope and Objectives

Time series data are widespread in real-world applications, particularly in domains such as finance [1], healthcare (e.g., ECG recordings)[2], and sensor-driven systems (e.g., industrial monitoring and IoT streams)[3]. In recent years, machine learning has achieved highly accurate performance on time series classification tasks, enabling reliable automated decision-making at scale[4]. Despite this progress, these models are often treated as black boxes: they provide predictions that a human cannot inspect and trust. This limitation is especially critical in high-stakes settings, where decisions may affect safety, cost, or clinical outcomes.

Consequently, explainability methods have become essential for understanding what patterns a model uses and how these patterns drive its decisions. Explaining time series models is challenging because temporal values are not independent: meaningful patterns often extend across multiple time points and depend on their ordering and context[5]. Explanations that operate at the level of individual time points can therefore be difficult to interpret and may fail to reflect coherent temporal structure. On the other hand, explanations based on fixed-length windows can be too restrictive, potentially merging distinct events or splitting meaningful ones arbitrar-

ily. Similar trade-offs arise in segmentation-based approaches when the user must pre-specify the number of changepoints.

Motivated by these challenges, this thesis investigates segment-based explanations for univariate time series classification, with a focus on identifying informative temporal regions without requiring the user to manually set the window length or the number of changepoints. Building on these regions, we develop attribution methods that quantify how different parts of the signal support or contradict a prediction, and we study both instance-level (local) and dataset-/class-level (global) perspectives. We validate the proposed approach empirically, examining how well the explanations reflect model behavior, how stable they are under perturbations, and their computational cost.

The objectives of this thesis are:

(i) design a SHAP-based attribution approach over segments for univariate time series classification,

(ii) detect the most influential sub-segments within top-ranked regions using a hierarchical refinement mechanism,

(iii) develop global explanation techniques that summarize important temporal regions at the dataset, class, or cluster level, and

(iv) evaluate the proposed methods in terms of faithfulness (the effect of removing top-ranked segments on the classifier's output), robustness (the stability of explanations under noise perturbations), and runtime (computational cost).

1.2 Thesis Structure

This chapter introduces the problem of explainability in univariate time series classification and presents the scope and objectives of the thesis. The remainder of this thesis is organized as follows. Chapter 2 reviews the necessary background and related work on explainability for time series classification. Chapter 3 formulates the problem setting and clarifies what the output of an explanation method represents. Chapter 4 presents the proposed method, including the segmentation procedure and the hierarchical refinement strategy. Chapter 5 introduces the proposed global explanation methods, which summarize important temporal regions at the dataset, class, or

cluster level. Chapter 6 describes the experimental setup, including the datasets, the model under explanation, the baselines, and the evaluation metrics for both local and global explanations. Chapter 7 reports the experimental results and provides an analysis of the findings. Finally, Chapter 8 concludes the thesis and discusses limitations and directions for future work.

Chapter 2

Background and Related Work

2.1 Background

In this section, we introduce the formal notation for univariate and multivariate time series and summarize the main families of explanation methods: surrogate-based, SHAP-based, and counterfactual-based explanations.

2.1.1 Definition of a Time Series Dataset

We consider a time series dataset $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ of N instances. Each instance

$$x^{(i)} \in \mathbb{R}^{d \times T}, \tag{2.1}$$

is a sequence of T ordered time steps, where d denotes the number of features (channels) and T denotes the number of time steps. Specifically,

$$x^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \ldots, x_T^{(i)}\right], \tag{2.2}$$

where each column $x_t^{(i)} \in \mathbb{R}^d$ is the feature vector at time step $t \in \{1, \ldots, T\}$. Let

$$x_j^{(i)}(t) \in \mathbb{R} \tag{2.3}$$

Table 2.1: Matrix representation of a univariate time series instance $x^{(i)} \in \mathbb{R}^{1\times T}$.

	t_1	t_2	$\cdots$	t_T
Feature	$x^{(i)}(t_1)$	$x^{(i)}(t_2)$	$\cdots$	$x^{(i)}(t_T)$

Table 2.2: Matrix representation of a multivariate time series instance $x^{(i)} \in \mathbb{R}^{d\times T}$.

	t_1	t_2	$\cdots$	t_T
Feature 1	$x_1^{(i)}(t_1)$	$x_1^{(i)}(t_2)$	$\cdots$	$x_1^{(i)}(t_T)$
Feature 2	$x_2^{(i)}(t_1)$	$x_2^{(i)}(t_2)$	$\cdots$	$x_2^{(i)}(t_T)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Feature d	$x_d^{(i)}(t_1)$	$x_d^{(i)}(t_2)$	$\cdots$	$x_d^{(i)}(t_T)$

denote the value of feature $j \in \{1, \ldots, d\}$ at time t for instance i.

This general formulation can represent both univariate and multivariate time series, depending on d:

- When $d = 1$, the instance is **univariate**, i.e., a single feature (channel) observed over time.
- When $d > 1$, the instance is **multivariate**, i.e., multiple features (channels) observed jointly over time.

Classification Models: Time series classification relies on a prediction function

$$f : \mathbb{R}^{d\times T} \to [0, 1], \tag{2.4}$$

which maps a time series instance $x^{(i)} \in \mathbb{R}^{d\times T}$ to the predicted probability of the positive class,

$$f\big(x^{(i)}\big) = \mathbb{P}\big(y = 1 \mid x^{(i)}\big)\,. \tag{2.5}$$

In the binary setting, the predicted label is obtained by thresholding (typically at 0.5):

$$\hat{y}^{(i)} = \begin{cases} 1, & \text{if } f\big(x^{(i)}\big) \geq 0.5, \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

2.2 Explanation Types

Three types of explanations are considered : surrogate-based, SHAP-based, and counterfactual-based.These are general families of explanation methods, independent of the type of data; time-series-specific adaptations are discussed in the next section.

- **Surrogate-based**: Mini local models trained to approximate the behavior of the black-box model around a specific input.
- **SHAP-based**: Coming from cooperative game theory, SHAP explains predictions by attributing contributions to each feature.
- **Counterfactual-based**: Counterfactual explanations identify minimal changes to the input that flip the model's prediction.

2.2.1 Surrogate-Based

LIME: Local Interpretable Model-agnostic Explanations

The LIME [6] (Local Interpretable Model-agnostic Explanations) method belongs to the category of surrogate-based approaches and aims to generate interpretable explanations for the predictions of black-box models. Its key principle is that, while the overall behavior of a complex model may be difficult to interpret, the model's behavior in the local neighborhood of a specific instance can often be adequately approximated by a simpler, interpretable model.

Method Overview

1.Input Representation

Given an input $x \in \mathbb{R}^d$, LIME defines an interpretable representation $x' \in \{0,1\}^{d'}$.

This refers to human-understandable features (e.g., presence/absence of words in a text, activation/deactivation of superpixels in images).

2.Perturbation Generation

A set of perturbations is generated around x' by randomly masking subsets of its active components.

Each perturbation $z' \in \{0,1\}^{d'}$ is mapped back to a corresponding instance $z \in \mathbb{R}^d$ in the original feature space.

The black-box model f is then evaluated on z, producing the target value $f(z)$.

3.Dataset Construction

This process generates a new dataset:

$$Z = \{(z_i', f(z_i), \pi_x(z_i))\}_{i=1}^{N}$$

Here, $\pi_x(z)$ is a proximity kernel that assigns higher weights to samples closer to the original instance x.

4.Objective Function

On the generated dataset, an interpretable surrogate model $g \in \mathcal{G}$ (typically a sparse linear model) is trained.

The optimization objective is:

$$\xi(x) = \arg\min_{g \in \mathcal{G}} \ \mathcal{L}(f, g, \pi_x) + \Omega(g),$$

where:

- $\mathcal{L}(f, g, \pi_x)$: measures the local mismatch between the surrogate and the black-box model.
- $\Omega(g)$: a complexity penalty that enforces interpretability (e.g., restricting the number of non-zero coefficients).

Training Procedure (K-LASSO)

In practice, the coefficients are estimated through a two-step procedure:

1. **Feature Selection**: Select the top K features using LASSO.
2. **Weight Estimation**: Estimate their coefficients via weighted least squares.

Outcome

The result is a locally faithful and sparse linear explanation, which presents the user with a short list of interpretable features (words, segments, superpixels) and their corresponding weights.

2.2.2 SHAP-based

SHAP (SHapley Additive exPlanations)

Lundberg and Lee [7] introduce SHAP as a unified framework for interpreting individual predictions of complex models. Their key idea is to view an explanation as an

explanation model g that locally approximates the original model f around a specific input x, using a simplified binary representation $x' \in \{0,1\}^M$ and an instance-specific mapping $x = h_x(x')$ [7]. They define the class of *additive feature attribution methods* as linear explanation models of the form

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i,$$

where ϕ_i is the attribution assigned to simplified feature i [7]. The paper shows that within this class there is a *unique* set of attributions that satisfies three desirable properties: *local accuracy* ($f(x) = g(x')$), *missingness* ($x'_i = 0 \Rightarrow \phi_i = 0$), and *consistency* (if a feature's marginal contribution never decreases across models, its attribution cannot decrease) [7]. This unique solution corresponds to Shapley values from cooperative game theory, and SHAP values can be interpreted as explaining how to move from the *base value* $\mathbb{E}[f(Z)]$ to the prediction $f(x)$ by distributing the difference across features. Because exact computation is expensive, the paper also derives practical estimators, including *KernelSHAP*, which recovers Shapley values via weighted linear regression with a specific Shapley kernel.

2.2.3 Counterfactual-based

DiCE: Diverse Counterfactual Explanations

Mothilal et al. [8] propose DiCE, a framework for generating *multiple* counterfactual explanations for a given instance x and a (binary) classifier f. Their key point is that actionable counterfactual explanations should be both *feasible* (close to x and respecting constraints) and *diverse* (offering different ways to change the outcome).

Given a desired target class y, DiCE generates a set of k counterfactuals $\{c_1, \ldots, c_k\}$ by solving a joint optimisation problem that balances validity, proximity, and diversity:

$$\{c_1, \ldots, c_k\} \;=\; \arg\min_{c_1,\ldots,c_k} \frac{1}{k}\sum_{i=1}^{k} \ell\big(f(c_i), y\big) + \lambda_1 \frac{1}{k}\sum_{i=1}^{k} \mathrm{dist}(c_i, x) - \lambda_2\, \mathrm{dpp\_diversity}(c_1, \ldots, c_k),$$

where $\ell(\cdot)$ encourages counterfactuals to reach the desired outcome, $\mathrm{dist}(\cdot,\cdot)$ encourages small changes from x, and diversity is modelled using a determinantal point process (DPP):

$$\mathrm{dpp\_diversity} \;=\; \det(K), \qquad K_{ij} = \frac{1}{1 + \mathrm{dist}(c_i, c_j)}.$$

DiCE supports user-specified constraints (e.g., immutable features or feasible ranges) and is optimised via gradient-based methods for differentiable models, producing a diverse set of counterfactual examples that can be inspected as example-based explanations.

2.2.4 Surrogate-based explanations for time series

LEFTIST

LEFTIST (Local Explainer For TIme Series classificaTion) [9] is a model-agnostic local explainer for time-series classification, designed to explain the prediction of an arbitrary black-box classifier at the level of an individual instance. The method adapts the local-surrogate logic of LIME to the time-series setting by treating an arbitrary segmentation of the input series as the interpretable representation. More specifically, a time series t is decomposed into m segments and encoded as a binary mask over segment presence/absence; local neighbors are then generated by flipping mask entries and mapping them back to the original input space through a reconstruction function h_t, where removed segments are replaced using linear interpolation, a constant baseline, or a random background segment drawn from a background set. On top of these perturbed samples, LEFTIST fits a local additive surrogate model g_t, whose coefficients quantify the contribution of each segment to the classifier's decision. In their experimental evaluation, the authors report that LEFTIST achieves high local fidelity with respect to both interpretable and complex classifiers, while Random Background emerges as the most effective transformation overall.

LIMESegment

Sivill and Flach [5] propose *LIMESegment*, a time-series adaptation of LIME for time series classification. They argue that applying LIME to time series requires addressing three practical questions: (i) what a meaningful interpretable representation of a time series should be, (ii) how to perturb a time series in a realistic way, and (iii) how to define a local neighbourhood around a time series.

Given a univariate time series $x \in \mathbb{R}^T$ and a black-box classifier $f : \mathbb{R}^T \to \mathbb{R}$, LIMESegment first constructs an interpretable representation by segmenting x into T' "super segments" (via their segmentation method *NNSegment*) and representing

the instance as a binary vector $\sigma(x) \in \{0,1\}^{T'}$, where each entry indicates whether a segment is active. To generate perturbed samples, LIMESegment draws random binary masks $\sigma(z)$ (using a Bernoulli sampler) and maps them back to perturbed time series z by replacing inactive segments with *realistic background content* (via their *Realistic Background Perturbation* method, based on extracting persistent frequency content using STFT and inverting it back to the time domain).

To enforce locality, LIMESegment weights perturbed samples using an exponential kernel applied to the (z-normalised) Dynamic Time Warping (DTW) distance between the original series x and each perturbed series z. Finally, it fits a weighted linear surrogate model (ridge regression) on the binary masks $\sigma(z)$ to approximate f locally; the surrogate coefficients are interpreted as importances of the corresponding super segments, producing a segment-level explanation for the prediction on x.

2.2.5 SHAP-based explanations for time series

TimeSHAP

TimeSHAP [10] is a post-hoc, model-agnostic explainer for recurrent models that extends KernelSHAP to sequential inputs by perturbing the entire sequence. The motivation is that applying KernelSHAP by perturbing only the current input vector can ignore the effect of perturbations on the recurrent hidden state, and may therefore miss the contribution of past events. TimeSHAP addresses this by using sequence-wide perturbations and producing Shapley-style attributions at the *event level* (time steps) and the *feature level* (channels), with an optional *cell level* (event–feature pairs).

Given an input sequence $X \in \mathbb{R}^{d \times T}$, TimeSHAP fits a local linear explanation model $g(z) = w_0 + \sum_{i=1}^{m} w_i z_i$ that approximates $f(h_X(z))$, where $z \in \{0,1\}^m$ is a coalition vector and $w_0 = f(h_X(0))$ is the base score. To generate perturbed sequences, TimeSHAP replaces removed components with an uninformative background matrix B (constructed from average feature values). For feature-wise explanations ($m = d$), perturbations toggle entire rows across time: $h_X^f(z) = D_z X + (I - D_z)B$. For event-wise explanations ($m = T$), perturbations toggle entire columns: $h_X^e(z) = X D_z + B(I - D_z)$. Switching between h_X^f and h_X^e is the only implementation change between feature and event explanations.

Because the number of temporal coalitions grows exponentially with the sequence length, TimeSHAP proposes a temporal-coalition pruning procedure that groups

older, low-importance events into a single coalition using a tolerance threshold η, trading off granularity in the distant past for a substantial runtime reduction. For cell-level explanations, TimeSHAP further reduces complexity by grouping cells and focusing on the intersection of the most relevant events and features (selected using an attribution threshold θ), while aggregating the remaining cells into a small number of coalitions.

WindowSHAP

WindowSHAP [11] is a model-agnostic SHAP-based framework for explaining time-series classifiers that was proposed to address the limitations of applying standard SHAP formulations directly to temporal data, particularly the high computational cost and the dependence among adjacent time points. Instead of estimating Shapley values for every variable–time-step pair, WindowSHAP partitions the temporal dimension into windows and computes contributions at the window level; point-wise attributions are then recovered by uniformly distributing each window's Shapley value across its constituent time steps while preserving local accuracy. The framework includes three variants: Stationary WindowSHAP, which uses fixed non-overlapping windows; Sliding WindowSHAP, which evaluates overlapping windows through repeated inside/outside partitions and aggregates the resulting scores; and Dynamic WindowSHAP, which iteratively refines the segmentation by splitting windows with high contribution until a stopping criterion based on a Shapley threshold or a maximum number of windows is reached.

TSHAP: Fast and Exact SHAP for Time Series

Nguyen and Ifrim [12] propose *TSHAP*, a SHAP-based attribution method for interpreting black-box time series classification and regression models. The motivation is that applying SHAP directly at the level of individual time steps quickly becomes expensive for moderately long sequences, and the choice of background data used to emulate "missingness" can strongly affect the resulting attributions. The paper focuses on *univariate* time series.

TSHAP reduces the attribution problem by aggregating consecutive time steps using a sliding window. For a window w (and its complement $\bar{w}$), time steps inside w are treated as one aggregated feature and time steps outside w as a second aggregated

feature, which yields a two-feature tabular view of the original time series. Missingness is emulated by combining the explained sample x with a background time series $\bar{x}$ to form hybrid inputs that swap values inside or outside the window. This construction enables an *exact* Shapley value for each window to be computed using four model evaluations:

$$\phi(w) = \tfrac{1}{2}\Big(f(x) - f(\{\bar{x}_w, x_{\bar{w}}\}) + f(\{x_w, \bar{x}_{\bar{w}}\}) - f(\bar{x})\Big).$$

Window attributions are averaged across background samples, and time-step attributions are obtained by distributing each window score uniformly across its time steps and averaging over all windows that cover a given index. To reduce runtime, TSHAP computes window attributions at a stride and interpolates the scores for the windows in between.

TSHAP has two variants. *TSHAP-Window* returns the full time-step attribution profile, while *TSHAP-ROI* post-processes window attributions to identify contiguous regions of interest (ROIs) by thresholding and grouping consecutive relevant windows, then assigns an ROI attribution uniformly to the time steps inside the region. Finally, the authors analyze background selection and propose choosing background data X_b so that the expected model output on the background matches the reference value: $\mathbb{E}[f(X_b)] \approx r$ for regression and $\mathbb{E}[f(X_b)] \approx 0.5$ for binary classification when explaining predicted probabilities.

2.2.6 Counterfactual-based Explanations for time series

Glacier: Guided Locally Constrained Counterfactuals for Time Series

Glacier [13] is a model-agnostic method for generating *counterfactual explanations* for *univariate* time series classification. Given a trained (deep) classifier $f(\cdot)$ and an input time series X that receives an undesired prediction, Glacier aims to produce a modified time series X' such that the prediction crosses the decision threshold, while keeping the changes to the original signal small. A key idea of Glacier is to support *temporal constraints* that guide where modifications should (or should not) occur, improving reliability and efficiency of the counterfactual search.

Glacier generates counterfactuals using gradient-based optimization (Adam) either directly in the original input space (*NoAE*) or in a learned latent space via an auto-encoder (encode–optimize–decode). The optimization balances a prediction-margin term (encouraging validity, i.e., crossing the decision boundary) and a prox-

imity/constraint term (encouraging small and guided changes), combined with a weighted loss. The constraints are encoded by a binary vector $c \in \{0,1\}^n$, where $c_i = 0$ means changing time point i is *favoured* and $c_i = 1$ means changes are *discouraged*. Glacier considers two constraint types: *example-specific* constraints (derived for the particular instance, e.g., from a local explainer such as LIMESegment) and *global* constraints (derived from the classifier, e.g., via interval importance computed with a permutation-style procedure). The paper evaluates Glacier on UCR datasets and reports improvements in proximity and compactness, while maintaining competitive validity compared to baseline counterfactual methods.

CoMTE: Counterfactual Explanations for Multivariate Time Series

CoMTE [14] is a counterfactual explanation method for *multivariate* time series classifiers treated as black boxes. The goal is to explain a prediction for a test sample $x_{\text{test}} \in \mathbb{R}^{m \times t}$ by identifying a *small number of channels* (metrics) that, if changed, would flip the prediction to a target class c.

CoMTE constructs counterfactuals by *substituting entire time series channels* from a single *distractor* sample x_{dist} (a training instance from the target class). Let $A \in \{0,1\}^{m \times m}$ be a binary diagonal matrix indicating which channels are swapped. The modified sample is

$$x' = (I_m - A)\, x_{\text{test}} + A\, x_{\text{dist}}.$$

The method searches for x_{dist} and A that (i) increase the target-class probability $f_c(x')$ and (ii) keep the number of substitutions small, using an objective of the form

$$(1 - f_c(x'))^2 + \lambda \|A\|_1,$$

and the paper notes that the resulting optimization problem is NP-hard. To obtain practical explanations, CoMTE uses heuristics: it first selects a small set of distractor candidates as nearest neighbors of x_{test} among correctly classified training samples of class c, and then searches for a good substitution set A either with a sequential greedy procedure (adding the channel that yields the largest improvement at each step) or with a faster derivative-free optimizer (random-restart hill climbing) until a target probability threshold is reached. The final explanation is the resulting counterfactual x' together with the (typically small) set of substituted channels.

Chapter 3

Problem Formulation

In this chapter, we formulate the problem of explaining time series and clarify what the output of an explanation method represents. First, we formalize missingness via a background/baseline and explain why such a background is necessary for time-series explanations. Second, we define how time points can be grouped into features, either through segmentation or fixed windows. We then show the strong impact that the chosen granularity has on the resulting explanations—whether it is controlled by the number of changepoints or by the window length. This motivates our method, which follows a CP-free segmentation algorithm.

3.1 Problem Definition and Notation

We consider a univariate time series

$$x = (x_1, \ldots, x_T) \in \mathbb{R}^T, \tag{3.1}$$

where x_t denotes the measurement at time step t and T is the series length. Let

$$f : \mathbb{R}^T \to [0, 1] \tag{3.2}$$

be a trained black-box probabilistic classifier that maps a time series x to a predicted probability (e.g., $f(x) = \mathbb{P}(y = 1 \mid x)$). The goal of local attribution methods is to produce an attribution vector

$$\phi(x) = (\phi_1, \ldots, \phi_T), \tag{3.3}$$

or, more commonly in time-series settings, attributions over groups of features (segments/windows), which are subsequently distributed back to the individual time points.

3.2 Segmentation granularity: change points vs window length

A key question in segmentation-based explainability for time series is: what is an appropriate way to split a time series into meaningful segments without losing the temporal structure, while ensuring that the segments remain semantically meaningful? As in images, where we typically do not want to explain an output "pixel-by-pixel" [6] in time series an explanation at the level of individual time points is not very useful and does not capture salient properties. Similarly, for SHAP-based methods, treating each time point as an independent feature is computationally infeasible[11]. Therefore, we must transform x into a representation based on segments [5].

In prior work, Guillemé et al. [9], Neves et al. [2], and WindowSHAP [11] all adopt fixed-length windows as the segmentation strategy, *i.e.*, they perform segmentation by using arbitrarily defined fixed-length windows.

Other methods use sliding windows [12, 11]. Alternatively, some approaches [5] obtain segments from segmentation algorithms that require specifying the number of changepoints in advance, which imposes a predefined number of segments.

Segmenting in the time domain assumes that neighboring observations have a similar impact on the model's predictions. Intuitively, we would like the "segments" of a time series to capture homogeneous regions of behavior [5, 11]. However, with arbitrary windows, we may (i) "split" a homogeneous region into multiple segments, or (ii) create a segment that contains conflicting behaviors.

When the way we define feature groups changes (e.g., window length w, stride, the number of segments), the explanation also changes. That is, the same model and the same time series can produce substantially different importance maps simply because the grouping has changed.

3.3 Missingness-Background in time series

In tabular settings, "removing" a feature often means setting that variable as missing and estimating its effect on the model output. For example, in LIME for tabular data [6], an "absent" feature corresponds to disabling the corresponding concept, and the original input is filled accordingly: (1) Text (bag-of-words representation): if a word is absent (0), then the feature value is 0 for that word (i.e., "the word does not exist"). (2) Images (super-pixels representation): if a super-pixel is absent (0), then the corresponding patch in the original image is grayed out.

LIME does this by "turning off" concepts. To apply this intuition to time-series *segments*, we must specify what it means to "delete" information from a time series [5]. However, in most time-series explainers, the model f does not accept truly missing values at each time point; therefore, "missingness" is implemented indirectly by replacing the "absent" time points (or segments) with values from a baseline/background. Perturbation approaches for image data include replacing a super-pixel with a constant value, injecting noise, or blurring the image. These techniques can also be applied to time series, e.g., replacing segments with mean-valued segments or replacing segments with randomly selected authentic segments from the original dataset. In some cases, applying blur, zero, and random perturbations can result in visually unrealistic samples. Moreover, using an all-zero background can be problematic in time series, because "0" often does not correspond to the absence of information and may instead be a highly informative signal.

For SHAP/Shapley-based explainers, the core idea is that we explain a difference in prediction relative to a reference induced by the background: in the general SHAP framework, attributions explain the difference between $f(x)$ and a "base value" linked to the expected prediction under the background distribution [7]. In time-series SHAP applications, it has been observed empirically that different background choices can drastically change the explanation [12]. Therefore, "missingness" in time series is not

a neutral step; it is a design choice (the background) that directly affects the resulting explanation.

3.4 Empirical Sensitivity Analysis of Segmentation Hyperparameters

We studied the same problem in two different families of explainers: surrogate-based methods, where a local surrogate is trained on perturbations defined over segments, and SHAP-based methods with sliding windows. In both cases, we observed that changing how segments are defined (i.e., using a different number of changepoints or a different window length) changes the quality of the explanation. We evaluated two representative methods—one per explainer family—in our experimental study, using the same classifier for both methods. Specifically, for LIMESegment, we varied the number of changepoints, which in turn changes the number of produced segments. For T-SHAP, we varied the sliding-window length. We evaluated both methods using a commonly used explanation-quality metric, faithfulness. Faithfulness is measured slightly differently for the two methods.

For LIMESegment, faithfulness measures how much the probability of the originally predicted class drops when we perturb the most important segment (according to LIMESegment). The perturbation is performed by reversing the segment with the largest weight, *i.e.*, reversing the time points within that segment. For each test instance x_i, we obtain an explanation consisting of: (i) weights w (one per segment) and (ii) the corresponding segments. We then identify the top segment, *i.e.*, the segment with the largest absolute weight. Next, we perturb only that segment by reversing its time points.

We compute the model probabilities:

$$original\_predictions[i] = p(y \mid x_i), \qquad reversed\_predictions[i] = p(y \mid x_i^{\text{rev}}). \tag{3.4}$$

We define the target as the originally predicted class:

$$c^* = \arg\max \big(original\_predictions[i]\big). \tag{3.5}$$

We then measure the drop in the probability of this class:

$$d_i = \big|\, p(c^* \mid x_i) - p(c^* \mid x_i^{\text{rev}}) \,\big|. \tag{3.6}$$

Finally, we report the average over all samples:

$$faithfulness = \frac{1}{N}\sum_{i=1}^{N} d_i. \tag{3.7}$$

A larger value means that perturbing the top segment changes the model's confidence substantially, indicating a more faithful explanation (i.e., the segment indeed matters for the decision). A smaller value means that the perturbation has little effect on the prediction, suggesting that the top segment may not be truly critical (or that the perturbation is not sufficiently strong).

For T-SHAP, the comparison between the new prediction (on the perturbed time series) and the original prediction (on the original time series) indicates the faithfulness of the attributions. In our experiments, where f outputs probabilities, faithfulness is computed as follows: x^p is the resulting time series after perturbing the positively attributed part, and x^n results from perturbing the negatively attributed part. The perturbation is performed by substituting the top 10% most relevant data with zero. Faithfulness is then defined as

$$faithfulness = \frac{1}{|X|}\sum_{x\in X}\big(f(x) - f(x^p) + f(x^n) - f(x)\big) = \frac{1}{|X|}\sum_{x\in X}\big(f(x^n) - f(x^p)\big). \tag{3.8}$$

Tables 3.1 and 3.2 show that sensitivity to hyperparameters can lead to different segmentations and, consequently, to different importance attributions. In practice, there is no clear rule for selecting either the number of changepoints or the window length, and the user must choose these parameters each time. Figures 3.2 and 3.1 are particularly interesting, as they show that different numbers of changepoints in the LIMESegment explanation lead to different top-1 segments with no overlapping time points.

Table 3.1: Faithfulness of LIMESegment under different numbers of changepoints (cp) across datasets.

Dataset	cp	**Faithfulness**
Chinatown	2	0.257591
Chinatown	5	0.305359
Chinatown	7	0.371165
Chinatown	15	0.004218
Coffee	3	0.203771
Coffee	5	0.072215
Coffee	7	0.039778
Coffee	15	0.001110
ECG200	2	0.312392
ECG200	4	0.191015
ECG200	8	0.027388
ECG200	10	0.015902
GunPointMaleVersusFemale	3	0.024970
GunPointMaleVersusFemale	6	0.016695
GunPointMaleVersusFemale	8	0.009640
GunPointMaleVersusFemale	10	0.000581
Strawberry	5	0.150733
Strawberry	7	0.251388
Strawberry	10	0.294760
Strawberry	15	0.233272

Table 3.2: Faithfulness of T-SHAP under different numbers of window length across datasets.

Dataset	frac	win-len	T-SHAP-Window	ROI
Coffee	0.01	3	-0.0237	0.0162
Coffee	0.05	14	0.2689	0.1857
Coffee	0.10	29	0.3385	0.2283
Coffee	0.20	57	0.2250	0.2524
ECG200	0.01	2	0.1210	0.1646
ECG200	0.05	5	0.1532	0.1551
ECG200	0.10	10	0.1137	0.0735
ECG200	0.20	19	0.0940	0.0585
Chinatown	0.01	2	0.1790	0.0038
Chinatown	0.05	2	0.1790	0.0038
Chinatown	0.10	2	0.1790	0.0038
Chinatown	0.20	5	-0.0177	-0.0137
GunPointMaleVersusFemale	0.01	2	0.2771	-0.1485
GunPointMaleVersusFemale	0.05	8	0.2308	-0.0932
GunPointMaleVersusFemale	0.10	15	0.1778	-0.1267
GunPointMaleVersusFemale	0.20	30	0.0264	-0.0238
Strawberry	0.01	2	0.0459	0.2152
Strawberry	0.05	12	0.2981	0.2473
Strawberry	0.10	24	0.3451	0.2849
Strawberry	0.20	47	0.4226	0.2996

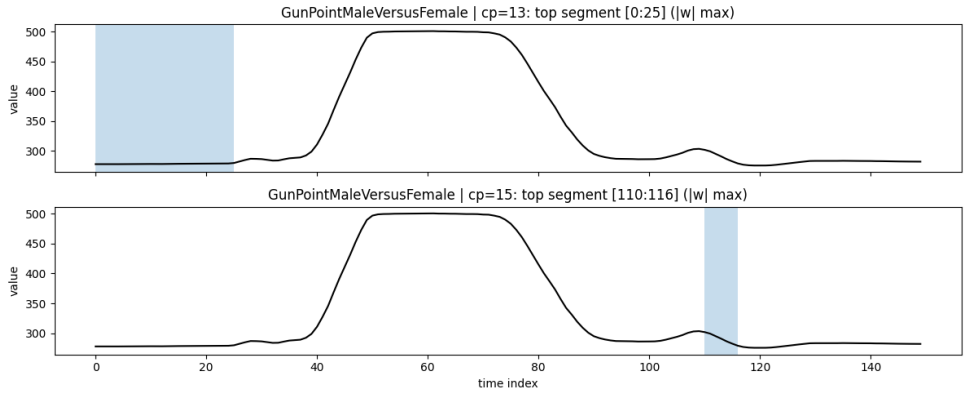

Figure 3.1: GunPointMaleVersusFemale-LIMESegment Explanation

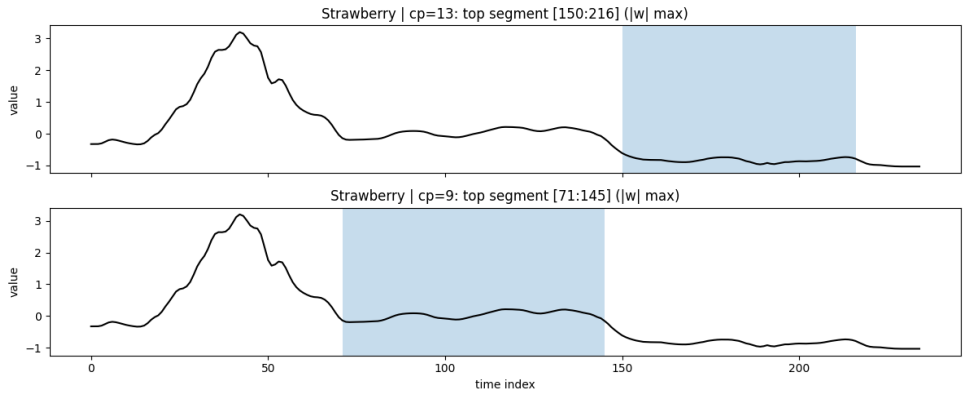

Figure 3.2: Strawberry-LIMESegment Explanation

3.5 Motivation for the Proposed Method

The motivation of our method is to remove the user-defined choice of the changepoint count, because the explanation is highly sensitive to the number of changepoints: a different changepoint count yields different segments and, consequently, a different explanation. To address this issue, our method adopts a segmentation strategy that does not require specifying the number of changepoints. Instead, segmentation is controlled via a penalty term, and the number of changepoints is determined by the optimization. As a result, the interpretable representation becomes less dependent on arbitrary parameter choices.

Chapter 4

Proposed Method

4.1 Preliminaries

We focus on univariate time series. An input instance is an ordered sequence of measurements over time,

$$x = (x_1, x_2, \ldots, x_T) \in \mathbb{R}^T,$$

where x_t denotes the measurement at time step t, and T is the series length.

Given a trained binary probabilistic classifier

$$f : \mathbb{R}^T \to [0, 1], \tag{4.1}$$

which maps a time series x to a predicted probability.Throughout this thesis, the default prediction to be explained is the probability of the positive class,

$$f(x) = \mathbb{P}(y = \text{pos} \mid x) \in [0, 1].$$

Under this choice, attribution values are interpreted as signed contributions to the positive-class probability: positive attributions increase $\mathbb{P}(y = \text{pos} \mid x)$, whereas

negative attributions decrease it, relative to a background. Another option is to explain the probability of the predicted class,

$$f(x) = \mathbb{P}\big(y = \hat{y}(x) \mid x\big) \in [0, 1],$$

so that explanations address the question "why did the model predict this label with this confidence?"

Eplanation output. Given a model f, an instance x, a background b, and a segmentation S into M segments. In our experiments, we define the background (baseline) as the mean time series of the training set. The background (defined in Section 3.3) is computed point-wise as

$$b = \frac{1}{N} \sum_{i=1}^{N} X_{\text{train}}[i, t],$$

where N is the number of instances, i.e., for each time index t we take the average of the values across all training samples. The method returns segment-level attributions

$$\Phi(x; S) = (\phi_1, \ldots, \phi_M) \in \mathbb{R}^M,$$

where $\phi_k \in \mathbb{R}$ quantifies the contribution of segment k to the explained output $f(x)$ relative to the background b. The detailed computation is provided in Section 4.3.

4.2 Segmentation with PELT

In this section, we introduce the segmentation step of our approach and explain how we partition each time series into contiguous segments using PELT. These segments form the feature units used by the explanation method in the following steps.

From time points we move to segments, where segments are treated as the players/features. Suppose we have an instance

$$X_i = (x_1, x_2, x_3, \ldots, x_T).$$

After segmentation, we obtain M segments such that

$$S = \{(s_k, e_k)\}_{k=1}^{M}.$$

Specifically, the segmentation method we use is PELT [15].

We consider the problem of detecting multiple changepoints in large data sets. PELT is a method for finding the minimum of such cost functions and hence the optimal number and location of changepoints that has a computational cost which, under mild conditions, is linear in the number of observations. The pruned Exact Linear Time (PELT) method has a computational cost that is linear in the number of data points. It involves a pruning step in the dynamic programming algorithm. This pruning reduces the computational cost of the method, but does not affect the exactness of the resulting segmentation. It can be applied to find changepoints under a range of statistical criteria such as penalised likelihood, quasi-likelihood [16] and cumulative sum of squares [17, 18].

Changepoint detection can, be considered to be the identification of points within a data set where the statistical properties change. More formally, let us assume we have an ordered sequence of data,

$$y_{1:n} = (y_1, \ldots, y_n).$$

Our model will have a number of changepoints, m, together with their positions,

$$\tau_{1:m} = (\tau_1, \ldots, \tau_m).$$

Each changepoint position is an integer between 1 and $n-1$ inclusive. We define $\tau_0 = 0$ and $\tau_{m+1} = n$ and assume that the changepoints are ordered such that $\tau_i < \tau_j$ if, and only if, $i < j$. Consequently the m changepoints will split the data into $m+1$ segments, with the i-th segment containing

$$y_{(\tau_{i-1}+1):\tau_i}.$$

One commonly used approach to identify multiple changepoints is to minimise:

$$\sum_{i=1}^{m+1} C\big(y_{(\tau_{i-1}+1):\tau_i}\big) + \beta f(m). \tag{4.2}$$

Here C is a cost function for a segment and $\beta f(m)$ is a penalty to guard against overfitting. Twice the negative log-likelihood is a commonly used cost function in the changepoint literature, although other cost functions such as quadratic loss and cumulative sums are also used, or those based on both the segment log-likelihood and the length of the segment. Turning to choice of penalty, in practice by far the most common choice is one which is linear in the number of changepoints, i.e.

$$\beta f(m) = \beta m.$$

Examples of such penalties include Akaike's Information Criterion [19] ($\beta = 2p$) and Schwarz Information Criterion (SIC, also known as BIC, [20]) ($\beta = p \log n$), where p is the number of additional parameters introduced by adding a changepoint. The PELT method is designed for such linear cost functions.

The optimal partitioning method [21] and [22] propose a search method that aims to minimise

$$\sum_{i=1}^{m+1} C\big(y_{(\tau_{i-1}+1):\tau_i}\big) + \beta. \tag{4.3}$$

This is equivalent to 4.2 where $f(m) = m$.

Following [22] the optimal partitioning (OP) method begins by first conditioning on the last point of change. It then relates the optimal value of the cost function to the cost for the optimal partition of the data prior to the last changepoint plus the cost for the segment from the last changepoint to the end of the data. More formally, let $F(s)$ denote the minimisation from 4.3 for data $y_{1:s}$ and

$$T_s = \{\tau : 0 = \tau_0 < \tau_1 < \cdots < \tau_m < \tau_{m+1} = s\}$$

be the set of possible vectors of changepoints for such data. Finally set $F(0) = -\beta$. It therefore follows that:

$$\begin{aligned} F(s) &= \min_{\tau \in T_s} \left\{ \sum_{i=1}^{m+1} C\big(y_{(\tau_{i-1}+1):\tau_i}\big) + \beta m \right\} \\ &= \min_{t} \left\{ \min_{\tau \in T_t} \left(\sum_{i=1}^{m} C\big(y_{(\tau_{i-1}+1):\tau_i}\big) + \beta \right) + C\big(y_{(t+1):n}\big) + \beta \right\} \\ &= \min_{t} \left\{ F(t) + C\big(y_{(t+1):n}\big) + \beta \right\}. \end{aligned} \tag{4.4}$$

This provides a recursion which gives the minimal cost for data $y_{1:s}$ in terms of the minimal cost for data $y_{1:t}$ for $t < s$. This recursion can be solved in turn for $s = 1, 2, \ldots, n$. The cost of solving the recursion for time s is linear in s, so the overall computational cost of finding $F(n)$ is quadratic in n. Steps for implementing the OP method are given in Algorithm 4.1.

PELT's computational cost can be linear in n whilst retaining an exact minimisation of 4.3. This exact and efficient computation is achieved via a combination of optimal partitioning and pruning.

Algorithm 4.1 Optimal Partitioning (OP)

1: **Input**: A set of data of the form $(y_1, y_2, \ldots, y_n)$ where $y_i \in \mathbb{R}$.
2: A measure of fit $C(\cdot)$ dependent on the data.
3: A penalty constant β which does not depend on the number or location of changepoints.
4: **Initialise**: Let $n =$ length of data and set $F(0) = -\beta$, $cp(0) =$ NULL.
5: **for** $\tau^* = 1, \ldots, n$ **do**
6: Calculate $F(\tau^*) = \min_{0 \leq \tau < \tau^*} F(\tau) + C\big(y_{(\tau+1):\tau^*}\big) + \beta$.
7: Let $\tau = \arg\min_{0 \leq \tau < \tau^*} F(\tau) + C\big(y_{(\tau+1):\tau^*}\big) + \beta$.
8: Set $cp(\tau^*) = (cp(\tau), \tau)$.
9: **end for**
10: **Output**: the change points recorded in $cp(n)$.

A Pruned Exact Linear Time Method

According to [15], pruning can improve the computational efficiency of the OP method while still ensuring that the global minimum of the cost function 4.3 is found. The essence of pruning in this context is to remove those values of τ which can never be minima from the minimisation performed at each iteration in 4.2 of Algorithm 4.1.

The following theorem gives a simple condition under which we can do such pruning.

Theorem 4.1 ([15]). *We assume that when introducing a changepoint into a sequence of observations, the cost, C, of the sequence reduces. More formally, we assume there exists a constant K such that for all $t < s < T$,*

$$C\big(y_{(t+1):s}\big) + C\big(y_{(s+1):T}\big) + K \leq C\big(y_{(t+1):T}\big) \tag{4.5}$$

Then if

$$F(t) + C\big(y_{(t+1):s}\big) + K \geq F(s) \tag{4.6}$$

then $\tau^* < t$ can never be the optimal last changepoint prior to T, where

$$F(s) = \min_{\tau} \left[\sum_{i=1}^{m+1} C\big(y_{(\tau_{i-1}+1):\tau_i}\big) + \beta f(m) \right].$$

holds, at a future time $T > s$, t can never be the optimal last changepoint prior to T.

The intuition behind this result is that if 4.6 holds then for any $T > s$ the best segmentation with the most recent changepoint prior to T being at s will be better than

any which has this most recent changepoint at t. Note that almost all cost functions used in practice satisfy assumption 4.5. The condition imposed in Theorem 4.1 for a candidate changepoint, t, to be discarded from future consideration is important as it removes computations that are not relevant for obtaining the final set of changepoints. This condition can be easily implemented into the OP method and the pseudo-code is given in Algorithm 4.2. This shows that at each step in the method the candidate changepoints satisfying the condition are noted and removed from the next iteration. The computational cost of this method will be linear in the number of observations, as a result we call this the Pruned Exact Linear Time (PELT) method.

Algorithm 4.2 Pruned Exact Linear Time (PELT) Method

Input: A set of data of the form $(y_1, y_2, \ldots, y_n)$ where $y_i \in \mathbb{R}$.
    A measure of fit $C(\cdot)$ dependent on the data.
    A penalty constant β which does not depend on the number or location of changepoints.
    A constant K that satisfies equation 4.5.
Initialise: Let $n =$ length of data and set $F(0) = -\beta$, $cp(0) =$ NULL, $R_1 = \{0\}$.
for $\tau^* = 1, \ldots, n$ **do**
    Calculate $F(\tau^*) = \min_{\tau \in R_{\tau^*}} F(\tau) + C\big(y_{(\tau+1):\tau^*}\big) + \beta$.
    Let $\tau_1 = \arg\min_{\tau \in R_{\tau^*}} F(\tau) + C\big(y_{(\tau+1):\tau^*}\big) + \beta$.
    Set $cp(\tau^*) = [cp(\tau_1), \tau_1]$.
    Set $R_{\tau^*+1} = \{\tau \in R_{\tau^*} \cup \{\tau^*\} : F(\tau) + C(y_{\tau+1:\tau^*}) + K \leq F(\tau^*)\}$.
end for
Output: the change points recorded in $cp(n)$.

4.3 Exact Segment SHAP Computation

Given a trained classifier f and a time series instance x that has been partitioned into segments S, our objective is to determine how much each segment contributes to the prediction $f(x)$.

1) What problem it solves. We aim to explain the prediction of a model f for a time series x. However, if we apply the classical SHAP formulation at the time-

point level, we obtain T features and the computational cost becomes exponential. Therefore, borrowing the key idea from TSHAP [12], which uses a sliding window to aggregate time steps, we group time steps within the window into one feature and those outside into another feature. As a result, we define a two-feature game for each segment ("exact 2-feature per segment vs. rest").

2) Exact SHAP for a two-feature game. In general, the Shapley value for a feature A is the average of its marginal contribution across all permutations/coalitions. For two features $\{A, B\}$, the possible coalitions are:

$$\emptyset,\ \{A\},\ \{B\},\ \{A, B\}.$$

Accordingly, the Shapley value for A is:

$$\phi_A = \frac{1}{2}\big(f(\{A\}) - f(\emptyset)\big) + \frac{1}{2}\big(f(\{A, B\}) - f(\{B\})\big),$$

where $f(S)$ is the coalition "value", i.e., the model output when the features in S are kept "active" and the remaining features are replaced by the baseline/background.

Here, "feature A present" means that within the segment we take values from x, while "feature A absent" means that within the segment we take values from the baseline b. Similarly, the same notion applies to the "rest" feature B.

Thus, we have $f(x)$ as the original output, and we evaluate $f(\cdot)$ for the required baseline-masked inputs. Concretely:

- inside the segment: we either take x or b,
- outside the segment: we either take x or b,

so that we can compute the two-feature exact SHAP term. In this notation, the resulting formula can be written as:

$$\phi_A = \frac{1}{2}\big(f(x) - f(x_B \cup b_A)\big) + \frac{1}{2}\big(f(x_A \cup b_B) - f(b)\big).$$

Therefore, the method returns a single attribution value ϕ_A for the entire segment, i.e., one scalar per segment.

Mapping segment attributions to time points. If we want attributions at the time-point level, we apply a simple mapping: we distribute the segment attribution uniformly across all time points of that segment.

Algorithm 4.3 SegSHAP

Inputs: model f, instance x, baseline b, penalty β, cost function $C(\cdot)$.

Output: $\Phi = (\phi_1, \ldots, \phi_M)$.

Compute segmentation $S = \{(s_k, e_k)\}_{k=1}^{M}$ via PELT using cost $C(\cdot)$ and penalty β.

for $k = 1$ to M **do**

    Compute ϕ_k via the two-player exact SHAP game (segment s_k vs. rest).

end for

return $\Phi = (\phi_1, \ldots, \phi_M)$.

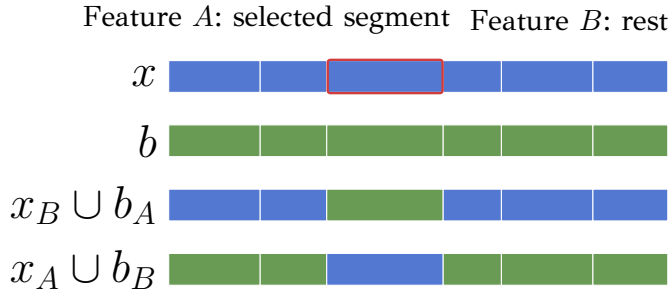

Figure 4.1: Segment-based illustration of the two-feature formulation for exact segment SHAP. Feature A corresponds to the selected segment, and Feature B to the rest of the timeline.

Complexity. For each instance, the two-feature formulation requires a constant number of model evaluations per segment, i.e., $\mathcal{O}(M)$ model calls for M segments (with $f(x)$ and $f(b)$ computed once and reused across segments). In contrast, classical exact SHAP over T time points is exponential in T, while KernelSHAP replaces the exponential cost with a sampling-based approximation that typically requires many model evaluations (scaling with the number of samples and the number of features), which becomes expensive for long time series.

Additivity Considerations. For a given segment s, we define a two-player cooperative game in which player A corresponds to the segment s (kept as in the original instance), and player B corresponds to the rest of the time series (denoted as s^-). In this setting, the Shapley values $\phi(s)$ and $\phi(s^-)$ satisfy the local accuracy [7] property for that specific two-player game:

$$\phi(s) + \phi(s^-) = f(x) - f(b),$$

where b is the baseline (background) and $f(b)$ acts as the reference value.

This mirrors the additive explanation model view of SHAP, where for a fixed set of players/features the explanation takes the form

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i,$$

and local accuracy requires the explanation to match the model output for the explained instance, i.e.,

$$f(x) = \phi_0 + \sum_{i=1}^{M} \phi_i,$$

with ϕ_0 commonly corresponding to the baseline prediction (e.g., $f(b)$).

However, in the proposed two-player-per-segment computation, the crucial difference is that the "rest" player is redefined for every segment, and therefore the underlying cooperative game is not the same across segments. Concretely, if we have four segments s_1, s_2, s_3, s_4, then:

- for $j = 1$, the players are $\{s_1, R_1\}$ with $R_1 = s_2 \cup s_3 \cup s_4$,
- for $j = 2$, the players are $\{s_2, R_2\}$ with $R_2 = s_1 \cup s_3 \cup s_4$,
- and similarly for $j = 3, 4$.

As a result, each attribution $\phi(s_j)$ measures how much that specific subsegment contributes to $f(x)$ relative to the baseline under a different definition of the opposing player R_j. In other words, the attributions $\phi(s_1), \ldots, \phi(s_M)$ are computed from different two-player games, rather than from a single M-player game in which the players are $\{s_1, \ldots, s_M\}$.

This has two direct implications. First, while local accuracy holds within each two-player game (i.e., $\phi(s_j) + \phi(R_j) = f(x) - f(b)$), there is no theoretical guarantee that the segment attributions obtained from different two-player games form a single coherent SHAP decomposition over all segments; in particular, there is no guarantee that

$$\sum_{j=1}^{M} \phi(s_j) = f(x) - f(b),$$

because the $\phi(s_j)$ do not originate from the same cooperative game.

Second, this formulation cannot explicitly capture joint effects arising from combinations of multiple segments (e.g., when two segments together drive the decision),

since each computation contrasts one segment against the aggregated remainder. Consequently, the approach provides an exact Shapley attribution for the corresponding two-player game, but it should be interpreted as a per-segment relevance score rather than an exact M-player SHAP decomposition; whether the sum of segment attributions approximates the prediction difference can be assessed empirically.

4.4 Refinement via Iterative Split

After the initial segmentation (global PELT with penalty β_{global}), some segments may be too large. A very large segment loses detail, and the granularity is reduced because the attribution ϕ is distributed uniformly across the entire segment, which spreads the importance and makes the explanation more generic. Moreover, a large segment may hide internal structure. Therefore, we apply a refinement step that is length-based.

We identify segments whose length exceeds a maximum threshold L_{max}, which depends on the length of each dataset. Concretely, we refine any segment with length

$$L > p \cdot T,$$

where p is a percentage of T. For each such segment, we run PELT again within the segment, using a smaller local penalty

$$\beta_{\text{local}} = \frac{\beta_{\text{global}}}{\gamma},$$

so as to allow more changepoints locally. This is done using a list of candidate γ values.

We perform this procedure iteratively (in rounds) until one of the following stopping conditions is met:

- there are no remaining "too long" segments, or
- no split occurs in a round, or
- we reach max rounds.

For each segment, we select the first γ that actually splits the segment, and we keep that split. If no γ produces a split, the segment is left unchanged. We also stop early if, in a given round, no segment was split.

Why we chose this algorithm. We did not want to simply decrease the penalty everywhere, because this would over-segment the entire signal and significantly increase the computational cost. Instead, we wanted a targeted procedure that corrects only excessively large segments, while keeping the remaining segments unchanged. In this way, we cut only the overly large regions and obtain more detailed explanations.

A practical issue in many explanation methods is that one must manually specify "how detailed" the explanation should be, either by setting the number of changepoints K or by choosing a window length. As we have shown, these choices can strongly affect the results. With our approach, this issue is largely removed: the initial segmentation is selected automatically using PELT + BIC, and the iterative split then breaks only the overly large segments when necessary. Thus, the granularity is driven by the data rather than manual tuning.

How we avoid over-segmentation. We do not split everything—we split only the excessively large segments. Therefore, we do not aim to produce "more and more" changepoints, we only correct under-segmentation.

Each split is supported by PELT itself. We run PELT on the sub-signal and keep the split only if it returns more than one sub-segment (i.e., if the cost model and penalty support it). If no γ returns a split, the segment remains unchanged. This acts as a brake against over-segmentation.

There is early stopping. If no split occurs in a round, we stop. Therefore, there is no mechanism that forces additional segments to be created.

Algorithm 4.4 Iterative Length-Based Refinement

Inputs: $x \in \mathbb{R}^T$, β_{global}, $C(\cdot)$, `min_size`, p, $\Gamma = (\gamma_1, \ldots, \gamma_J)$, `max_rounds`.
Output: final segmentation S.
$L_{\max} \leftarrow \lceil pT \rceil$; $S \leftarrow \texttt{PELT}(x; \beta_{\text{global}}, C, \texttt{min\_size})$.
for $r = 1$ **to** `max_rounds` **do**
  `changed` $\leftarrow$ `False`.
  for all segments $(s, e) \in S$ with $e - s > L_{\max}$ **do**
    Find first $\gamma \in \Gamma$ such that

$$S_{\text{sub}} = \texttt{PELT}(x_{s:e}; \beta_{\text{global}}/\gamma, C, \texttt{min\_size}) \quad \text{and} \quad |S_{\text{sub}}| > 1.$$

    If found, replace (s, e) with shifted S_{sub}; set `changed` $\leftarrow$ `True`.
  end for
  if not `changed` **then**
    break.
  end if
end for
return S.

Algorithm 4.5 Refinement with IterativeSplit

Input: $x \in \mathbb{R}^T$, β_{global} (BIC), min length m, $L_{\max} = \lceil pT \rceil$, $\Gamma = (\gamma_1, \ldots, \gamma_J)$, max rounds R
Output: Final segmentation S
Step 1 (Global): $S \leftarrow \texttt{PELT}(x; \beta_{\text{global}}, C, m)$
Step 2 (Iterate): Repeat for up to R rounds
  Mark segments $(s, e) \in S$ with length $e - s > L_{\max}$ as *long*
Step 3 (Local refinement):
  For each long segment (s, e), try $\gamma \in \Gamma$ (in order)
    $\beta_{\text{local}} \leftarrow \beta_{\text{global}}/\gamma$
    $S_{\text{sub}} \leftarrow \texttt{PELT}(x[s:e]; \beta_{\text{local}}, C, m)$
    If $|S_{\text{sub}}| > 1$, replace (s, e) with shifted S_{sub} and stop trying γ
Step 4 (Update): Keep unchanged segments and apply successful splits
Step 5 (Stop): If no split happened in this round, stop early
return S

4.5 Hierarchical Explanation: From Segments to Time Point

Motivation After performing segmentation with IterativeSplit refinement and computing exact two-feature SHAP, we obtain an attribution value ϕ_i for each segment. We can then rank the segments and select the top-k segments based on $|\phi|$. However, if we want point-level importance, the standard step is to distribute each segment's attribution uniformly across its time points, i.e., assign ϕ/L to each point in the segment (where L is the segment length). This uniform spreading implicitly assumes that all time points within the segment are equally important, which reduces granularity and prevents us from identifying where inside the top-k segments the truly important time points lie. Our goal is therefore to localize where the predictive information lies in the time series, but in a hierarchical manner: first coarsely, then more finely.

Concretely, we first apply segmentation (IterativeSplit refinement) to obtain the parent segments, and we compute exact segment SHAP values ϕ_i for all parents. We then sort these parent segments by $|\phi|$ and select the top-k parents for refinement. For each selected parent segment, we split it into child segments by running a local PELT on the corresponding sub-signal with a smaller penalty. Specifically, we use a reduced penalty

$$\beta_{\text{local}} = \beta_{\text{global}}/\gamma$$

and test γ values from a predefined list. If a γ leads to a non-trivial split (i.e., produces $n \geq 2$ child segments), we accept it immediately; if no γ returns any split, we keep the parent segment unchanged. In this way we avoid over-segmentation (i.e., producing an excessive number of children) while still increasing the resolution where it matters.

Next, given this refined segmentation inside the selected parents, we compute Permutation Shapley within each parent. Outside the parent segment, we keep the input fixed as the original instance x. Inside the parent, we start by setting the entire parent interval to the baseline, and then we "turn on" a subset of child segments by replacing their values with the corresponding values from x. Thus, the game is defined over the child segments: the "features" are the children, the coalition determines which children are on (from x) and which are off (from the baseline), and the context outside the parent remains fixed. To estimate SHAP values for the m child segments, we construct a base input where all children are off and compute $f(x_{\text{base}})$. For each permutation, we activate children one-by-one in the permutation order, compute the marginal gain $f_{\text{new}} - f_{\text{prev}}$ at each step, and accumulate it to the corresponding child's

Algorithm 4.6 Hierarchical Segment Attribution

1: **Input**: $x \in \mathbb{R}^T$, baseline b, β_{global}, k, $\Gamma = [\gamma_1, \gamma_2, \ldots, \gamma_J]$, B.

2: **Output**: $A \in \mathbb{R}^T$.

3: **Parents**: run PELT with β_{global} and iterative length refinement $\Rightarrow$ parent segments $S_P = \{(s, e)\}$ covering $[0, T)$.

4: **Parent SHAP**: for each $(s, e) \in S_P$, compute exact 2-feature SHAP $\phi_P(s, e)$ (segment vs. rest).

5: **Top-k**: select $T \subset S_P$ as the k segments with largest $|\phi_P|$.

6: **Local zoom**: for each $(s, e) \in T$, run local PELT on $x_{s:e}$ with

$$\beta_{\text{local}} = \beta_{\text{global}}/\gamma, \qquad \gamma \in \Gamma,$$

and choose the first γ returns a non-trivial split into children

$$S_C(s, e) = \{(s_j, e_j)\}_{j=1}^m.$$

If no split, set $S_C(s, e) = \{(s, e)\}$.

7: **Within-parent Shapley**: if $m > 1$, estimate $\phi_{\text{child}} \in \mathbb{R}^m$ via permutation Shapley over children (outside parent fixed to x; inside parent children on/off vs. b), using B permutations.

8: **Rescale**: enforce attribution conservation:

$$\phi_{\text{child}} \leftarrow \phi_{\text{child}} \cdot \frac{\phi_P(s, e)}{\sum_{j=1}^m \phi_{\text{child},j}} \quad \Rightarrow \quad \sum_{j=1}^m \phi_{\text{child},j} \approx \phi_P(s, e).$$

9: Replace (s, e) by its children with these ϕ_{child}. Non-selected parents keep ϕ_P.

10: **Point attribution**: for each final segment (s, e) with attribution ϕ,

$$A_t \mathrel{+}= \frac{\phi}{e - s}, \qquad t \in [s, e).$$

attribution. Averaging over B_{perm} permutations returns $\phi_{\text{child}} \in \mathbb{R}^m$, i.e., a distribution of attribution within the parent. Importantly, this is a local game inside the parent (not the global segment-vs-rest game).

Finally, we want the children to redistribute the parent attribution while preserving the parent's total mass.We then form the final point-level attribution map by uniformly spreading each final segment's attribution over its time points. Thus, the final output is a point-level attribution map produced via a hierarchical segment-based explanation: coarse at the parent level, and more fine-grained inside the most important parents through child segments.

Remark. The local permutation game inside a parent produces child attributions $\phi_{\text{child}} \in \mathbb{R}^m$ whose sum does not necessarily match the parent attribution ϕ_P. To

preserve the parent's total attribution mass, we apply a multiplicative rescaling:

$$\phi_{\text{child}} \leftarrow \phi_{\text{child}} \cdot \frac{\phi_P}{\sum_j \phi_{\text{child},j}},$$

so that $\sum_j \phi_{\text{child},j} \approx \phi_P$.

Chapter 5

Global Explanations

Motivation Local, instance-level explanations are particularly valuable in time series because they identify which regions of the signal were influential for a given prediction—i.e., which temporal intervals provided evidence in favor of the predicted class and which intervals acted against it. Moreover, they can further localize importance within those regions, revealing which specific time points contributed most and in what direction.

However, in many applications we also require explanations at a higher level of abstraction, such as for an entire dataset, a specific class, or a group of instances (e.g., a cluster). Such global explanations address questions of the form: What is important overall for this dataset (or class/cluster), and which recurring temporal patterns does the model rely on? In this chapter, we focus on global explanations as a means of summarizing the model's overall decision logic using two strategies: (i) aggregating local explanations via a voting mechanism, and (ii) treating the dataset as a multivariate instance to extract global importance directly.

5.1 Global Explanations via Voting

We hypothesize that some instances may share a common segmentation, in the sense that they contain similar regions that the algorithm identifies as segments. This motivates the idea of finding time locations where change points (CPs) occur across many instances simultaneously. To do so, we build a vote histogram that counts how frequently a changepoint appears at each time position. Specifically, the "vote histogram" records how many times an instance "votes" for the presence of a CP at time t.

To account for imperfect alignment, we introduce a tolerance window: each CP does not vote only at its exact index, but also within a neighborhood

$$[cp - window,\ cp + window].$$

Thus, if different instances have CPs in approximately the same region (but not at exactly the same index), the histogram aggregates these votes and amplifies them as a shared changepoint region. This produces a histogram of length T, where higher values indicate "many votes for a CP here".

We then extract global changepoints from this histogram by applying a threshold. We set

$$threshold = \rho \cdot N,$$

meaning that we keep only those time positions supported by at least a fraction ρ of the N instances (e.g., $\rho = 0.6$ corresponds to 60%). The output is a sorted array of global CP indices.

Once we have identified the global changepoints—i.e., changepoint locations supported by at least a fraction ρ of the instances—we automatically obtain the corresponding global segmentation S_G. We then proceed as follows. For each instance $x^{(i)}$ and for each global segment $(s_m, e_m) \in S_G$, we compute the exact two-feature Shapley value (segment vs. rest) using the baseline b. This produces an attribution matrix $\Phi \in \mathbb{R}^{N \times M}$, where each entry $\Phi_{i,m}$ corresponds to the attribution of the m-th global segment for the i-th instance.

To obtain a global attribution score for each global segment at the dataset level, we aggregate across instances by taking the mean of the segment attributions over all instances for that segment. Algorithm 5.1 summarizes the proposed voting-based global segmentation and the Exact-Two segment attribution computation.

Algorithm 5.1 Global Segmentation via Voting + Exact-Two Segment SHAP

1: **Input**: dataset $X \in \mathbb{R}^{N\times 1\times T}$, PELT cost model and global penalty β_{global}, iterative-split parameters, voting window w, threshold ratio $\rho \in (0, 1]$, baseline b.

2: **Output**: global segments S_G; segment SHAP matrix $\Phi \in \mathbb{R}^{N\times M}$; dataset-level segment score $\bar{\phi} \in \mathbb{R}^M$.

3: **Local segmentations.** For each instance $x^{(i)}$ (length T): run PELT with β_{global}, apply iterative length-based refinement, obtain $S^{(i)} = \{(s_\ell, e_\ell)\}$, and convert to changepoints

$$CP^{(i)} = \{\, e_\ell : (s_\ell, e_\ell) \in S^{(i)},\ e_\ell < T \,\}.$$

4: **Voting histogram.** Initialize $h \in \mathbb{R}^T \leftarrow 0$. For each $CP^{(i)}$ and each $c \in CP^{(i)}$: if $w = 0$, increment $h[c]$; else increment $h[t]$ for all

$$t \in \big[\max(1, c - w),\ \min(T - 1, c + w)\big].$$

5: **Select global changepoints.** Set $\texttt{thr} = \rho N$ and define

$$CP_G = \{\, t \in \{1, \ldots, T - 1\} : h[t] \geq \texttt{thr} \,\}.$$

6: **Global segmentation.** Sort $CP_G = \{cp_1 < \cdots < cp_m\}$ and form

$$S_G = \{(0, cp_1), (cp_1, cp_2), \ldots, (cp_m, T)\}, \qquad M = |S_G|.$$

7: **Exact-Two SHAP on global segments.** For each instance $x^{(i)}$ and each $(s_m, e_m) \in S_G$, compute the exact two-feature Shapley value (segment vs. rest) using baseline b and store $\Phi_{i,m} \leftarrow \phi_{i,m}$, yielding $\Phi \in \mathbb{R}^{N\times M}$.

8: **Global segment score + top-k.** For each segment m:

$$\bar{\phi}_m = \frac{1}{N} \sum_{i=1}^{N} |\phi_{i,m}|\,.$$

Rank segments by $\bar{\phi}_m$ and report the top-k.

Class-wise Global Segments via Voting. The algorithm (Algorithm 5.2) is applied class-wise. For each dataset, and for each class separately, we construct global segments via voting using only the instances belonging to that class. In other words, the voting histogram and the resulting global changepoints are computed from class-specific local segmentations. This returns a set of class-specific global segments, on which we then compute Exact-Two segment SHAP (segment vs. rest) for the instances of that class, using the same baseline b.

Algorithm 5.2 Class-wise Global Segments via Voting

1: **Input**: test set $(X_{\text{test}}, y_{\text{test}})$, class of interest c, vote window w, threshold ratio $\rho \in (0, 1]$.

2: **Output**: class-wise global segments $S_G^{(c)}$ over $[0, T)$.

3: **Filter the test set to the class.** Keep only instances with $y = c$:

$$X_c = \{x^{(i)} : y^{(i)} = c\}, \qquad |X_c| = N_c.$$

4: **Define the explained output**:

$$f_c(X) = P(y = c \mid X).$$

5: **Local changepoints (within the class).** For each $x^{(i)} \in X_c$: run local segmentation (IterativeSplit), extract its changepoints $CP^{(i)} \subset \{1, \ldots, T-1\}$.

6: **Voting histogram.** Build $h[t]$ = number of votes for a changepoint around time t. If $w > 0$, each changepoint also votes in a $\pm w$ neighborhood (tolerance window).

7: **Select class-wise global changepoints.** Keep:

$$CP_G^{(c)} = \{\, t : h[t] \geq \rho N_c \,\}.$$

8: **Convert to class-wise global segments.** Sort $CP_G^{(c)} = \{cp_1 < \cdots < cp_m\}$ and form:

$$S_G^{(c)} = \{(0, cp_1), (cp_1, cp_2), \ldots, (cp_m, T)\},$$

which covers $[0, T)$.

9: **Result**: $S_G^{(c)}$ is class-specific global segmentation (computed only from instances of class c).

5.2 Global Explanations: Dataset as a Multivariate Instance

In this section we present the second algorithm for global explanations, based on treating the entire dataset as a single multivariate time series. The main idea is to view a univariate dataset

$$X \in \mathbb{R}^{N \times T} \text{ (Figure 5.1)}$$

as a multivariate signal by constructing

$$Y \in \mathbb{R}^{T \times D} \text{ with } D = N \text{ (Figure 5.2)}$$

where $Y_{t,i} = X_{i,t} = x_t^{(i)}$.

where each column corresponds to one test instance (i.e., one "dimension"), and time runs along the T axis. Equivalently, rows correspond to time steps and columns correspond to instances $i = 1, 2, \ldots, N$, with each column being a time series.

Under this representation, PELT searches for common changepoints that appear simultaneously across many dimensions/instances. Conceptually, PELT assumes that

Figure 5.1: Dataset $X \in \mathbb{R}^{N \times T}$.

$$X \in \mathbb{R}^{N \times T} \qquad \left[\begin{array}{c|cccc} & t_1 & t_2 & \cdots & t_T \\ \hline x^{(1)} & x_1^{(1)} & x_2^{(1)} & \cdots & x_T^{(1)} \\ x^{(2)} & x_1^{(2)} & x_2^{(2)} & \cdots & x_T^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x^{(N)} & x_1^{(N)} & x_2^{(N)} & \cdots & x_T^{(N)} \end{array}\right]$$

Figure 5.2: Dataset-as-multivariate instance: $Y \in \mathbb{R}^{T \times N}$, where $Y_{t,i} = X_{i,t} = x_t^{(i)}$.

$$Y \in \mathbb{R}^{T \times D}, \quad D = N \qquad \left[\begin{array}{c|cccc} & x^{(1)} & x^{(2)} & \cdots & x^{(N)} \\ \hline t_1 & Y_{1,1} & Y_{1,2} & \cdots & Y_{1,N} \\ t_2 & Y_{2,1} & Y_{2,2} & \cdots & Y_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_T & Y_{T,1} & Y_{T,2} & \cdots & Y_{T,N} \end{array}\right]$$

the axis being segmented is time, and that at each time step we observe a vector-valued observation containing the values of all dimensions/features. Therefore, we use the convention rows = time (T) and columns = dimensions/features (D).This is also illustrated in Figure 5.2.

Importantly, if we instead used the layout $N \times T$ (rows = instances, columns = time), then PELT would interpret the axis to be segmented as the instance index, and the features as the time points, which is not the intended use in this setting.

Before changepoint detection, Y is normalized per dimension across time using z-score normalization, so that instances with different scales, variances, or amplitudes contribute comparably to the multivariate PELT cost. In this way, the segmentation is driven less by absolute magnitude differences and more by shared temporal structure. The normalization is used only for changepoint detection. PELT is then applied to the normalized multivariate signal to obtain an initial set of global segments. These segments are subsequently refined through the *IterativeSplit* procedure, following the

same principle as in the univariate case, but operating on multivariate segments.

After this step, we have a list of global segments

$$\{(s_m, e_m)\}_{m=1}^{M}.$$

Next, we compute SHAP in a "segment vs. rest" manner for each instance and each global segment. Specifically, for each instance i and segment m, we compute

$$\Phi_{i,m} \leftarrow \phi_{i,m},$$

where $\phi_{i,m}$ is the Exact-Two feature Shapley value with features:

$$A = [s_m, e_m), \qquad B = \text{rest of the signal.}$$

This returns an attribution matrix

$$\Phi \in \mathbb{R}^{N \times M}.$$

Finally, we can obtain a dataset-level importance score per segment, for example by aggregating across instances as

$$\bar{\phi}_m = \frac{1}{N} \sum_{i=1}^{N} |\Phi_{i,m}|,$$

and then ranking segments by $\bar{\phi}_m$ to report the top-k segments. The procedure described above is summarized in Algorithm 5.3.

Class-wise Global Explanations: Class as a Multivariate Instance. We can also obtain global explanations in a class-wise (or cluster-wise) manner using the multivariate-instance formulation. For each dataset and for each class separately, we keep only the test instances belonging to that class (i.e., a class subset). On this subset, we construct class-specific global segments using multivariate PELT, where each instance is treated as one dimension (Algorithm 5.4).

Starting from

$$X \in \mathbb{R}^{N \times 1 \times T},$$

we form

$$Y \in \mathbb{R}^{T \times D}, \qquad D = N,$$

Algorithm 5.3 Multivariate Global Segmentation (Instances-as-Dimensions) + Exact-Two SegSHAP

1: **Input**: dataset $X \in \mathbb{R}^{N\times 1\times T}$, cost model, β, iterative-split parameters, baseline $b \in \mathbb{R}^{1\times 1\times T}$.

2: **Output**: global segmentation S_G, SHAP matrix $\Phi \in \mathbb{R}^{N\times M}$, segment importances $\bar{\phi} \in \mathbb{R}^{M}$.

3: **Dataset-as-multivariate signal.** Construct

$$Y \in \mathbb{R}^{T\times D}, \qquad D = N, \qquad Y_{t,i} = X_{i,0,t}.$$

Normalize only for changepoint detection:

$$Y^{(c)} = \text{Normalize}(Y).$$

4: **Multivariate PELT.** Run multivariate PELT on $Y^{(c)}$, obtaining global changepoints

$$CP_G = \{cp_1 < \cdots < cp_M\} \subset \{1, \ldots, T-1\}.$$

Convert changepoints to global segments:

$$S_G = \{(0, cp_1), (cp_1, cp_2), \ldots, (cp_M, T)\}, \qquad M = |S_G|.$$

5: **Exact Two SegSHAP.** For each instance i and each global segment m, compute the exact 2-feature Shapley value ("segment vs. rest") using baseline b and store

$$\Phi_{i,m} \leftarrow \phi_{i,m}, \qquad \Phi \in \mathbb{R}^{N\times M}.$$

6: **Dataset-level segment importance.** Define per-segment importance:

$$\bar{\phi}_m = \frac{1}{N}\sum_{i=1}^{N} |\Phi_{i,m}|, \qquad \bar{\phi} \in \mathbb{R}^{M}.$$

7: **return** $S_G,\ \Phi,\ \bar{\phi}$.

so that each instance becomes one "dimension/column". We then run PELT on Y to obtain class-specific global segments. For each instance x_i and each global segment (s_m, e_m), we compute the Exact-Two Shapley value in a "segment vs. rest" manner:

$$\Phi_{i,m} = \phi(x_i;\ \text{segment } m \text{ vs. rest}),$$

using the background X_b as the baseline. This returns an attribution matrix

$$\Phi \in \mathbb{R}^{N_c\times M}.$$

Algorithm 5.4 Class-wise Multivariate Global Segmentation (Instances-as-Dimensions)

1: **Input**: test set $(X_{\text{test}}, y_{\text{test}})$ with $X_{\text{test}} \in \mathbb{R}^{N \times T}$, cost model, class c , penalty β; iterative-split parameters.

2: **Output**: class-specific global segmentation S_G covering $[0, T)$.

3: **Filter to class** c**.** Let $I_c = \{\, i : \; y_{\text{test}}[i] = c \,\}$ and $X_c \leftarrow X_{\text{test}}[I_c] \in \mathbb{R}^{N_c \times 1 \times T}$. Set $D \leftarrow N_c$.

4: **Dataset-as-multivariate signal.** Construct $Y \in \mathbb{R}^{T \times D}$ by

$$Y_{t,j} = X_c[j, 0, t], \qquad t = 0, \ldots, T-1, \quad j = 0, \ldots, D-1.$$

5: **Normalize only for changepoint detection.** Compute

$$Y^{(c)} = \text{Normalize}(Y),$$

and use this normalization *only* for changepoint detection (not for later explanations).

6: **Multivariate PELT.** Run PELT on $Y^{(c)}$ and iterative-split parameters to obtain changepoints

$$CP_G = \{cp_1 < \cdots < cp_M\} \subset \{1, \ldots, T-1\}.$$

Convert changepoints to segments:

$$S_G = \{(0, cp_1), (cp_1, cp_2), \ldots, (cp_M, T)\}.$$

7: **return** S_G .

Chapter 6

Experimental Setup & Evaluation Metrics

In this chapter, we describe the experimental setup and the evaluation methodology used throughout our study. We present the datasets and the predictive model under explanation, as well as the baseline method used for comparison. Finally, we define the evaluation metrics used to assess both local and global explanations.

6.1 Datasets

We used univariate binary classification datasets from the UCR Time Series Classification Archive[23]. Additional dataset information and download links are available via the Time Series Classification repository website.[1] We selected datasets that vary in

[1] https://www.timeseriesclassification.com/dataset.php

time-series length, classification difficulty, and application domain, to cover a diverse range of settings. For each dataset, we used the official train/test splits provided by the UCR archive.The selected datasets, along with a brief description of what each one measures/classifies and its corresponding type, are summarized in Table 6.1.The main characteristics of the selected datasets are reported in Table 6.2.

Table 6.1: UCR binary univariate datasets.

Dataset	**Short description**	**Type**
Coffee	Spectral measurements (food spectrographs) to distinguish Robusta vs Arabica coffee beans.	SPECTRO
Wine	FTIR spectroscopy of wine to distinguish Cabernet Sauvignon vs Shiraz.	SPECTRO
BirdChicken	1D series extracted from image contours (from MPEG-7) to distinguish bird vs chicken.	IMAGE
ECG200	Cardiac electrical activity per heartbeat to distinguish normal vs myocardial infarction.	ECG
Chinatown	Pedestrian traffic time series to predict weekday vs weekend.	TRAFFIC
BeetleFly	1D series extracted from image contours (MPEG-7) to distinguish beetle vs fly.	IMAGE
SonyAIBORobotSurface1	Robot accelerometer (x-axis) readings to recognize surface type: cement vs carpet.	SENSOR
ShapeletSim	Synthetic dataset: one of five embedded "shapes" is injected into noise for classification.	SIMULATED
GunPoint	Hand motion (x-axis centroid) to distinguish Gun-Draw vs Point.	HAR
PowerCons	Household electricity consumption to distinguish warm vs cold season.	DEVICE
ToeSegmentation1	Walking motion-capture signals to distinguish normal vs abnormal gait (x-axis).	MOTION
GunPointAgeSpan	GunPoint variant (2003+2018): classifies Gun vs Point across different "flavors" (age/year of acquisition).	HAR
GunPointMaleVersusFemale	GunPoint variant (2003+2018): classifies Male vs Female.	HAR
ProximalPhalanxOutlineCorrect	Hand image outlines: whether the proximal phalanx outline was extracted correctly vs incorrectly.	IMAGE
Strawberry	FTIR food spectrographs to distinguish authentic strawberry vs non-strawberry/adulterated.	SPECTRO
Earthquakes	Seismic measurements: predict whether a major event (defined as > 5 Richter) is impending vs not.	SEISMIC

Table 6.2: Dataset statistics (length and split sizes) for the UCR binary univariate datasets used in our experiments.

Dataset	T **(Length)**	**Train size**	**Test size**
Coffee	286	28	28
Wine	234	57	54
BirdChicken	512	20	20
ECG200	96	100	100
Chinatown	24	20	345
BeetleFly	512	20	20
SonyAIBORobotSurface1	70	20	601
ShapeletSim	500	20	180
GunPoint	150	50	150
PowerCons	144	180	180
ToeSegmentation1	277	40	228
GunPointAgeSpan	150	135	316
GunPointMaleVersusFemale	150	135	316
ProximalPhalanxOutlineCorrect	80	600	291
Strawberry	235	613	370
Earthquakes	512	322	139

6.2 Model Under Explanation

In our experiments, we explain a model from the ROCKET [24] framework of kernel-based time-series transforms followed by a linear classifier. Specifically, the input time series is first transformed into a feature vector via MiniROCKET (a convolutional kernel transform)[25], and a linear classifier (Logistic Regression) is then trained on the transformed features. This follows the ROCKET approach: transforming time series using convolutional kernels and training a linear classifier on the resulting features.

Specifically, MiniROCKET[25] is a very fast and (almost) deterministic transform for time series classification, which retains the key components of ROCKET and produces features that are subsequently fed to a linear classifier.

The pipeline used in this thesis is:

$$\hat{y} = \mathrm{LR}\big(\mathrm{Scaler}(\mathrm{MiniROCKET}(x))\big).$$

Training protocol & reproducibility: For each dataset, we used the predefined train/test splits provided by the archive. The model is trained exclusively on the train split, and its performance is reported on the independent test split, without additional resampling or cross-validation. All time series are represented in the three-dimensional format required by MiniROCKET [25]:

$$X \in \mathbb{R}^{N \times C \times T}, \tag{6.1}$$

where N is the number of samples, T is the time-series length, and $C = 1$ for univariate data. When necessary, we reshape the data from (N, T) to $(N, 1, T)$ by inserting a channel dimension (without changing the values).

For each dataset, the same classification pipeline was trained: MiniROCKET as the feature transform, followed by standardization/scaling of the resulting features, and Logistic Regression as the final classifier.

Reproducibility. To ensure full reproducibility, all sources of randomness were controlled via fixed random seeds, both at the level of numerical computations (NumPy) and across the individual stages of the pipeline (MiniROCKET transform and Logistic Regression classifier).

Output definition (what we explain). The explainability methods we compare require access to a prediction function $f(\cdot)$ that returns probabilities.

Table 6.3: Test accuracy of the MiniROCKET + Logistic Regression classifier on the UCR binary univariate datasets.

Dataset	**Test Acc.**
Coffee	1.000
Wine	0.852
BirdChicken	0.900
ECG200	0.920
Chinatown	0.983
BeetleFly	0.850
SonyAIBORobotSurface1	0.880
ShapeletSim	1.000
GunPoint	0.993
PowerCons	0.994
ToeSegmentation1	0.965
GunPointAgeSpan	0.975
GunPointMaleVersusFemale	1.000
ProximalPhalanxOutlineCorrect	0.893
Strawberry	0.978
Earthquakes	0.741

Binary classification. In the binary classification setting, the explanation target is defined as the probability of the positive class:

$$f_{\text{proba}}(X) = P(\hat{y} = \text{pos} \mid X),$$

i.e., the probability assigned by the model to the "positive" category (as determined by the label ordering in the trained classifier). This quantity is then used by the explanation methods to measure changes in the prediction under perturbations.

We report test accuracy to provide performance context for the models under explanation.We report test accuracy to provide performance context for the models under explanation. The results are shown in Table 6.3.

Why this model. For the classifier we use MiniROCKET because it is an efficient and effective time-series classifier. Its stable training behavior supports consistent

comparisons across datasets and runs. Finally, its computational efficiency is crucial for explainability experiments, since SHAP-style methods require many repeated model evaluations.

6.3 Methods Compared

As baselines, we included the two official variants of TSHAP[12]: TSHAP-Window and TSHAP-ROI. Both are based on the same core idea: instead of treating each time step as a separate feature, they group consecutive time steps using a sliding window, enabling the computation of exact Shapley values at the "window" level with much lower cost than exact SHAP over all time steps.

TSHAP-Window (sliding-window attributions). TSHAP-Window formulates the explanation as a two feature problem: (i) the time steps inside the window w and (ii) the time steps outside the window $\bar{w}$. For each window position, it computes the exact contribution of the window using a closed-form Shapley expression, with "missingness" implemented by replacing missing parts with a background series. It then assigns a score to each time step by averaging the scores of all windows that contain it. For efficiency, the paper recommends computing window attributions with a stride $s > 1$ and interpolating between window positions.

TSHAP-ROI (Regions of Interest). TSHAP-ROI starts from the window attributions produced by TSHAP-Window and uses them to identify "important regions" (ROIs), rather than diffusing importance across neighboring (potentially irrelevant) time steps. Specifically, it marks a window as "relevant" when $|\phi(w_i)|$ exceeds a threshold (defined in the paper as a fraction of the maximum $|\phi|$), groups consecutive relevant windows, and defines an ROI as the union of the windows in each group. It then assigns a single attribution to the ROI and distributes it uniformly across the time steps within the ROI.

Hyperparameters. Following the experimental protocol in the TSHAP paper, we set the sliding-window length to $0.1T$ (10% of the series length) and used stride $s = 5$. All remaining hyperparameters were kept at their default values in the official implementation (e.g., interpolation enabled and ROI extraction when applicable).

Below we provide a summary of our variants used for comparison.

Our variants:

- **Local variants**:
  - **SegSHAP**: PELT segmentation + Exact-Two Feature SHAP (segment vs. rest) with a fixed baseline.
  - **SegSHAP (Refine)**: SegSHAP + IterativeSplit refinement on overly long segments (reduced local penalty).
  - **SegSHAP (Hierarchical)**: Parent Exact-Two attributions + refine top-k parents into children and compute within-parent local Shapley to localize importance.
- **Global variants**:
  - **SegSHAP-Global (Voting)**: Vote histogram over changepoints $\rightarrow$ shared global segmentation $\rightarrow$ Exact-Two SHAP on global segments.
  - **SegSHAP-Global (Multivariate)**: Dataset as multivariate (instances as dimensions) $\rightarrow$ multivariate PELT global segments $\rightarrow$ Exact-Two SHAP on global segments.

To ensure a fair comparison across methods, we use shared evaluation settings: all explainers operate with the same background/baseline definition and explain the same model output, namely the predicted probability of the positive class $P(\hat{y} = \text{pos} \mid X)$ in the binary setting.

6.4 Local Evaluation Metrics

In this section, we describe how we assess the quality of the produced explanations. After obtaining explanations, a natural question is how we can determine whether they are "good". We address this by focusing on two key criteria: (i) faithfulness—how well the explanations reflect the behavior of the underlying classifier, and (ii) robustness—how stable the explanations remain under small variations in the input space.

6.4.1 Faithfulness

Faithfulness analysis is a common approach for evaluating attribution methods on real-world classification datasets[3]. In our experiments, we adopt the faithfulness protocol used in TSHAP [12], since both TSHAP and our approach produce SHAP-based explanations. The main idea is to perturb the most relevant parts of the time series (as identified by the attributions) to create a new input, and then compare the model's prediction on the perturbed series to its prediction on the original series. This comparison provides an indication of how faithful the attributions are to the classifier's behavior.

Equations (6.2) show how faithfulness is computed, where $f(\cdot)$ outputs a probability. Let x_p denote the time series obtained by perturbing the time steps with the largest positive attributions, and let x_n denote the time series obtained by perturbing the time steps with the largest negative attributions. The perturbation is performed by substituting the top $r = 10\%$ most relevant time points with zero.Alternatively, perturbations can be defined at the segment level by masking the top-1 attributed segment. In addition, instead of zeroing-out, the masked region can be replaced with noise, providing an alternative perturbation choice under the same evaluation principle. Faithfulness is then defined as:

$$\textit{faithfulness} = \frac{1}{|X|}\sum_{x\in X}\Big(f(x) - f(x_p) + f(x_n) - f(x)\Big) = \frac{1}{|X|}\sum_{x\in X}\Big(f(x_n) - f(x_p)\Big). \quad (6.2)$$

This faithfulness measure is a signed perturbation test of the form "remove positive attributions vs. remove negative attributions", using the top-r time points and comparing the resulting predictions. Intuitively, perturbing time steps with positive attributions should reduce the model's confidence (i.e., $f(x)-f(x_p) > 0$), whereas perturbing time steps with negative attributions should increase it (i.e., $f(x_n)-f(x) > 0$). Higher positive faithfulness scores therefore indicate greater attribution fidelity, since the direction of the prediction change aligns with the signs of the attributions. The computation is described in detail in Algorithm 6.1.

Interpretation (what "high" faithfulness means). If the positive attributions indeed identify time points that support the positive prediction, then perturbing those points should reduce the model's confidence in the positive class. If the negative attributions correctly capture time points that act against the positive class (i.e., push

Algorithm 6.1 Faithfulness (Top-r perturbation)

Inputs: dataset $X = \{x^{(i)}\}_{i=1}^{N}$; point-level attributions Φ (per instance); explained model output $f(\cdot)$; fraction r (e.g., 0.1); perturbation type: zero.

Output: faithfulness score.

Split attributions into two parts:

$$\Phi_{\text{pos}}, \Phi_{\text{neg}}$$

for $i = 1$ **to** N **do**

Rank time points by attribution magnitude (separately for $\Phi_{\text{pos}}^{(i)}$ and $\Phi_{\text{neg}}^{(i)}$) and select the top-r% indices.

Construct perturbed inputs:

$$x_p^{(i)} \leftarrow \texttt{Perturb}\Big(x^{(i)},\ \text{top-}r\%\ \text{indices from}\ \Phi_{\text{pos}}^{(i)}\Big),$$

$$x_n^{(i)} \leftarrow \texttt{Perturb}\Big(x^{(i)},\ \text{top-}r\%\ \text{indices from}\ \Phi_{\text{neg}}^{(i)}\Big).$$

Compute predictions:

$$y_p^{(i)} \leftarrow f\Big(x_p^{(i)}\Big), \qquad y_n^{(i)} \leftarrow f\Big(x_n^{(i)}\Big).$$

Instance contribution:

$$\Delta^{(i)} \leftarrow y_n^{(i)} - y_p^{(i)}.$$

end for

return

$$\textit{faithfulness} \leftarrow \frac{1}{N}\sum_{i=1}^{N} \Delta^{(i)}.$$

the prediction downward), then perturbing those points should increase the positive-class probability. Therefore, we expect the faithfulness value to be as large as possible: higher values indicate that the attribution map successfully separates supportive evidence from opposing evidence with respect to the positive class.

6.4.2 Robustness

A natural question then arises: if we add a small amount of random noise to a given instance, do we obtain essentially the same explanation? Ideally, an end-user would want explanations to be robust to small changes in the input space, so that anomalous observations do not influence the resulting explanation. We adopt the robustness definition used in LIMESegment[5], which measures robustness by observing the difference between explanations generated for a time series before and after it is

perturbed with randomly generated noise.

In our setting, robustness is quantified as follows: after adding small Gaussian noise to the input, we measure how stable the "most important" time points identified by SegSHAP remain. Stability is measured using the Jaccard similarity between:

- the top-10% time points from the attribution map on the clean input, and
- the top-10% time points from the attribution map on the noisy input.

We then summarize robustness via a robustness rate, defined as the proportion of trials for which the Jaccard similarity exceeds a threshold τ (i.e., Jaccard $\geq \tau$). The Jaccard similarity measures overlap between two sets and is defined as the ratio of the number of shared elements (intersection) to the total number of elements that appear in at least one of the sets (union):

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

It takes values in $[0, 1]$: a value of 1 means the sets are identical, whereas 0 means they have no common elements. In our robustness evaluation, the sets consist of the indices of the top-r% most important time points for the clean and noisy inputs.

Procedure. The inputs are the test set X_{test}, the explanation model f, $r = 0.10$ (top 10% of T), noise standard deviation , stability threshold τ, number of perturbations per instance n_{perturb}, and a random seed for reproducible noise. For each instance, we first compute the attribution map on the clean input and form the set S_1 containing the top-k indices. We then generate n_{perturb} noisy versions by adding Gaussian noise with mean 0 and standard deviation σ_{noise}, re-run the explanation to obtain a noisy attribution map, and extract the corresponding top-k index set S_2. For each perturbation trial, we compute the Jaccard similarity between S_1 and S_2. If $J(S_1, S_2) \geq \tau$, the explanation is considered stable for that trial. Finally, across all instances and perturbation trials, we report the mean and median Jaccard similarity, as well as the robustness rate defined

$$\text{robustness rate} = \frac{n_{\text{stable}}}{n_{\text{total}}}.$$

We note that segmentation is re-computed each time on the noisy signals. Therefore, the explanation is evaluated end-to-end:

$$\text{noise} \rightarrow \text{segmentation} \rightarrow \text{segment SHAP} \rightarrow \text{point-level attribution}.$$

Algorithm 6.2 Robustness - Top-r Jaccard Similarity

Inputs: X_{test}, model f, $r = 0.10$, $\sigma_{\text{noise}} = 0.03$, $\tau = 0.7$, $n_{\text{perturb}} = 10$, seed.
Initialize empty list $\mathcal{J}$, set stable $\leftarrow 0$, total $\leftarrow 0$.
for each instance $x \in X_{\text{test}}$ **do**
    Compute point-level attributions ϕ.
    Let $S_1 \leftarrow \text{Top}_r(\phi)$ (indices of top-r fraction by $|\phi|$).
    for $j = 1$ **to** n_{perturb} **do**
        Sample noise $\varepsilon \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$ and set $x^{(j)} \leftarrow x + \varepsilon$.
        Compute noisy attributions $\phi^{(j)} \leftarrow f(x^{(j)})$.
        Let $S_2 \leftarrow \text{Top}_r(\phi^{(j)})$.
        $J \leftarrow \dfrac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$, append J to $\mathcal{J}$.
        if $J \geq \tau$ **then**
            stable $\leftarrow$ stable $+ 1$.
        end if
        total $\leftarrow$ total $+ 1$.
    end for
end for
mean jaccard $\leftarrow$ mean($\mathcal{J}$), median jaccard $\leftarrow$ median($\mathcal{J}$).
robustness rate $\leftarrow$ stable/total.
Output: mean jaccard, median jaccard, robustness rate.

6.4.3 Runtime

We report runtime because computing Shapley values is generally a computationally expensive procedure. Since our approach relies on exact Shapley values in a two-feature formulation, measuring runtime allows us to assess whether this exact computation remains sufficiently fast in practice, and therefore whether the method is convenient to use. We measure runtime by separating the cost of segmentation from the cost of explanation, and we report both dataset-level totals and per-instance averages. Segmentation time is measured per instance as the total time required to compute the segmentation, including all rounds of IterativeSplit refinement. Explanation time is also measured per instance as the total time required to compute all segment-level SHAP values for that instance (i.e., all SHAP calls across all segments), including the final uniform redistribution to obtain point-level attributions when applicable.

At the dataset level, we report:

- the sum of segmentation times across all instances,

- the sum of explanation times across all instances,
- the end-to-end time to produce all attributions, including segmentation, explanation, and any overhead.

We also report mean per-instance times by dividing by N.

Overall, this provides three runtime views—segmentation, explanation, and total—reported both as dataset-level totals and as per-instance averages.

6.5 Global Evaluation Metrics

Global explanations indicate which segment is most important for an entire dataset, for a specific class, or for a cluster. A natural question, however, is whether such a global explanation is truly representative. To evaluate this, we adopt the view that the best available explanation for a dataset is the collection of local explanations. Under this assumption, a global explanation can be assessed by how well it approximates the local explanations across instances. Therefore, in the global setting, "faithfulness" measures how faithful the global explanation is to the local explanation behavior, treating the latter as our strongest available reference.

We measure global faithfulness in three ways.

(1) Global top-1 segment deletion. Following the spirit of LIMESegment[5], we define faithfulness as the decrease in the classifier's confidence when removing the most important segment returned by the explanation. Concretely, we identify the top-1 global segment for the entire dataset and perturb (e.g., zero out) that same segment for all test instances, then measure the mean drop in the positive-class probability. If the selected segment is indeed globally important, deleting it should cause a large decrease in prediction confidence across the dataset. We compare this to the corresponding local faithfulness variant, where we identify the top-1 segment separately for each instance and measure the drop induced by deleting that instance-specific segment.The procedure is described in detail in Algorithm 6.3.

(2) Point-level faithfulness. We also compute faithfulness using point-level attributions obtained from a global segmentation. Specifically, we first compute Exact Two segment SHAP values on the global segments, and then convert segment attributions

Algorithm 6.3 Faithfulness for Global Segments (Top-1 Segment Deletion)

Input: test set $X_{\text{test}} = \{x^{(i)}\}_{i=1}^{N}$, global segments $S_G = \{(s_m, e_m)\}_{m=1}^{M}$, baseline b; explained model $f(\cdot)$).

Output: global faithfulness score.

Segment attributions on global segments. For each instance $x^{(i)}$ and each global segment $(s_m, e_m) \in S_G$, compute exact Segment SHAP (segment vs. rest) to obtain $\phi_{i,m}$.

Dataset-level segment importance. Compute per-segment importance as the mean attribution across instances:

$$\bar{\phi}_m \leftarrow \frac{1}{N} \sum_{i=1}^{N} \phi_{i,m}.$$

Select the top-1 global segment. Let

$$m^\star \leftarrow \arg \max_{m \in \{1,\dots,M\}} \bar{\phi}_m, \qquad (s^\star, e^\star) \leftarrow (s_{m^\star}, e_{m^\star}).$$

Perturb the selected segment for all instances.

for $i = 1$ **to** N **do**

  $x_{\text{del}}^{(i)} \leftarrow \text{Perturb}\big(x^{(i)}, [s^\star, e^\star)\big)$.

  $\Delta^{(i)} \leftarrow f\big(x^{(i)}\big) - f\Big(x_{\text{del}}^{(i)}\Big)$.

end for

Faithfulness.

$$\textit{faithfulness} \leftarrow \frac{1}{N} \sum_{i=1}^{N} \left|\Delta^{(i)}\right|.$$

return *faithfulness*.

to a point-level attribution map by uniformly distributing each segment's attribution across its time points. We then apply the standard top-r% point perturbation faithfulness test on these point-level attributions.The procedure is described in detail in Algorithm 6.4.

(3)Most-frequent local top-1 deletion. In addition, we report Most-Frequent Local Top-1 segment deletion as an intermediate baseline between fully local and fully global explanations. Concretely, we identify the segment that most frequently appears as the top-1 local explanation across test instances and delete this same segment for all instances, measuring the mean drop in the positive-class probability. This allows us to compare a "global" explanation obtained by simple aggregation of local winners (majority top-1) against our true global segmentation-based explanation, which is derived by pooling information from all instances. If the global method is effective, deleting its selected segment should be more impactful than deleting the most common local top-1 segment, since the former leverages broader dataset evidence

Algorithm 6.4 Point-perturbation Faithfulness (from Global Segments)

1: **Input**: $X_{\text{test}} = \{x^{(i)}\}_{i=1}^{N}$; global segments $S_G = \{(s_m, e_m)\}_{m=1}^{M}$, baseline b, explained model$f(\cdot)$; fraction r.
2: **Output**: faithfulness score.
3: Use the global segments S_G (same for all instances).
4: **for** $i = 1$ **to** N **do**
5: Compute exact SegmentSHAP on S_G and obtain segment attributions $\{\phi_{i,m}\}_{m=1}^{M}$.
6: Convert to point-level scores by uniform spreading within each segment:

$$\phi_t^{(i)} \leftarrow \frac{\phi_{i,m}}{e_m - s_m} \qquad \forall\, t \in [s_m, e_m).$$

7: Keep only positive scores and perturb the top-r% time points $\rightarrow$ compute $y_p^{(i)} \leftarrow f(x_p^{(i)})$.
8: Keep only negative scores and perturb the top-r% time points $\rightarrow$ compute $y_n^{(i)} \leftarrow f(x_n^{(i)})$.
9: Store $\Delta^{(i)} \leftarrow y_n^{(i)} - y_p^{(i)}$.
10: **end for**
11: **return** *faithfulness* $\leftarrow \frac{1}{N}\sum_{i=1}^{N} \Delta^{(i)}$.

rather than a segment that is top-1 only for a subset of instances.The procedure is described in detail in Algorithm 6.5.

For the faithfulness evaluation of class-wise global explanations, we do not use the entire test set jointly. Instead, for each class c, we restrict to the test instances with label c, construct the global segments using voting only within that class subset, and compute prediction drops with respect to the corresponding class probability $P(\text{class} = c \mid x)$. We then evaluate the different faithfulness modes on this same class-specific subset.

Robustness for global explanations. The robustness idea is the same as in the local setting: we add noise to the input and re-run the explanation. Ideally, an end-user would want explanations to be robust to small changes in the input space. In our case, we add random noise drawn from a normal distribution and then examine whether the global explanation remains stable. We measure robustness in two ways.

(A) Robustness of the top-1 global segment. We compare the top-1 global segment obtained on the clean dataset to the top-1 global segment obtained on the noisy dataset, and report:

- **Exact match rate**: how often the exact same segment is returned.

Algorithm 6.5 Most-Frequent Local Top-1 Segment Deletion Faithfulness

1: **Input**: $X_{\text{test}} = \{x^{(i)}\}_{i=1}^{N}$, baseline b, explained output $f(\cdot)$, local segmentation.

2: **Output**: faithfulness score.

3: For each instance $x^{(i)}$, run local segmentation and compute exact SHAP for each local segment, obtaining a top-1 local segment $(s^{(i)}, e^{(i)})$.

4: Find the most frequent top-1 segment across instances:

$$(s^\star, e^\star) \leftarrow \text{mode}\Big(\{(s^{(i)}, e^{(i)})\}_{i=1}^{N}\Big).$$

5: Perturb (delete) this same segment for all instances and compute prediction drops:

$$\Delta^{(i)} \leftarrow f\Big(x^{(i)}\Big) - f\Big(\text{Perturb}\Big(x^{(i)}, [s^\star, e^\star)\Big)\Big).$$

6: **return** *faithfulness* $\leftarrow \frac{1}{N}\sum_{i=1}^{N} \Delta^{(i)}$.

- **Mean IoU**: the average overlap (intersection-over-union) between the clean top-1 and noisy top-1 segments.
- **IoU** $\geq \tau$ **rate**: how often the overlap exceeds a chosen threshold.

The procedure is described in detail in Algorithm 6.6.

(B) Robustness of the top-10% time points. We convert segment-level SHAP values to point-level scores via uniform spreading within each segment. We then obtain dataset-level point importance, select the top-k points (default $r_{\text{points}} = 0.1$), and measure the Jaccard similarity between the clean and noisy top-k sets. We report:

- **Mean Jaccard**, and
- **Jaccard** $\geq \tau$ **rate** (default $\tau = 0.7$).

The procedure is described in detail in Algorithm 6.7.

Algorithm 6.6 Robustness of the Top-1 Global Segment

Inputs: test set X_{test},number of noisy runs B, σ_{noise}; IoU threshold τ_{iou}, global explanation pipeline.

Outputs: exact match rate, mean IoU, IoU $\geq \tau_{\text{iou}}$ rate.

Run the global pipeline on the clean test set X_{test} and obtain the dataset-level top-1 segment top1_0.

Initialize counters: match $\leftarrow 0$, stable $\leftarrow 0$; list $\mathcal{I} \leftarrow [\,]$.

for $b = 1$ **to** B **do**

    Add Gaussian noise to the test set to obtain $X_{\text{test}}^{(b)}$.

    Re-run the global pipeline on $X_{\text{test}}^{(b)}$ and obtain top1_b.

    Exact match indicator: $m_b \leftarrow \mathbb{I}\big[\text{top1}_b = \text{top1}_0\big]$.

    IoU: $I_b \leftarrow \text{IoU}(\text{top1}_0, \text{top1}_b)$, append I_b to $\mathcal{I}$.

    Stability indicator: $s_b \leftarrow \mathbb{I}[I_b \geq \tau_{\text{iou}}]$.

    match $\leftarrow$ match $+ m_b$; stable $\leftarrow$ stable $+ s_b$.

end for

Exact match rate $\leftarrow$ match$/B$.

Mean IoU $\leftarrow$ mean$(\mathcal{I})$.

IoU $\geq \tau_{\text{iou}}$ **rate** $\leftarrow$ stable$/B$.

Output: exact match rate, mean IoU, IoU $\geq \tau_{\text{iou}}$ rate.

Algorithm 6.7 Robustness of Global Top-10% Time Points

Inputs: test set X_{test}, number of noisy runs B, noise σ_{noise}, threshold τ, global explanation pipeline, $r = 0.10$.

Outputs: mean Jaccard, Jaccard $\geq \tau$ rate.

Run the global explanation pipeline on the clean test set X_{test} to obtain global segments and Segment SHAP attributions.

Convert segment-level attributions to point-level scores by uniform spreading within each segment, and select the top-k time points (top 10% of T) to obtain the clean set S_0.

Define $J(A, B) \leftarrow \dfrac{|A \cap B|}{|A \cup B|}$ (Jaccard similarity)

Initialize list $\mathcal{J} \leftarrow [\,]$ and counter stable $\leftarrow 0$.

for $b = 1$ **to** B **do**

    Add Gaussian noise to X_{test} to obtain $X_{\text{test}}^{(b)}$.

    Re-run the full global pipeline on $X_{\text{test}}^{(b)}$ to obtain global segments and Segment SHAP attributions.

    Convert to point-level scores (uniform spreading) and select the top-k time points to obtain S_b.

    $J_b \leftarrow J(S_0, S_b)$; append J_b to $\mathcal{J}$.

    if $J_b \geq \tau$ **then**

        stable $\leftarrow$ stable $+ 1$.

    end if

end for

Mean Jaccard $\leftarrow$ mean$(\mathcal{J})$.

Jaccard $\geq \tau$ **rate** $\leftarrow$ stable$/B$.

Output: mean Jaccard, Jaccard $\geq \tau$ rate.

We use the same robustness evaluation for both global-explanation approaches. Moreover, instead of measuring robustness on the entire test set jointly, we can compute it separately for each class c, using only the test instances of that class.

For each dataset and for each class c, we form the class subset:

$$X_c = X_{\text{test}}\big[y_{\text{test}} = c\big].$$

We then run the global pipeline on X_c:

- compute global segments via voting using only X_c,
- compute Exact SegmentSHAP (segment vs. rest) for each instance/segment,
- identify the dataset-level top-1 segment (using mean $|\phi|$),
- and identify the dataset-level top-10% time-point set (used by the Jaccard metric).

Next, we perform B noisy runs:

- we add Gaussian noise only to X_c (not to the full test set),
- and each time we re-run the entire pipeline (segments + exact SHAP).

Finally, we measure: (A) top-1 segment stability (exact match + IoU), and (B) top-10% time-point stability (Jaccard), and we report robustness metrics per dataset and per class.

Runtime. A third metric for global explanations is runtime. We report the time required (i) to construct the global segments for each dataset, (ii) to compute the corresponding explanations (i.e., segment SHAP on the global segments), and (iii) the total end-to-end time including both steps.

Chapter 7

Experimental Results

In this chapter, we present the empirical results of our approach and explain how they arise from the design choices in our method. We report faithfulness, robustness, and runtime for both local and global explanations, and compare our variants against the baseline methods. We begin by motivating the refinement step, showing that the initial segmentation can produce overly long segments and therefore limit explanation granularity. We then present local results (faithfulness, robustness, runtime) together with qualitative visualizations of local explanations, followed by the corresponding global results using the same evaluation protocols. Finally, we provide ablation and sensitivity studies to quantify the contribution of key components and the effect of important hyperparameters.

7.1 Motivation for Refinement: Under-segmentation and Segment-Length Statistics.

In this section, we test our hypothesis that the initial global segmentation can produce overly long segments, which reduces explanation granularity due to uniform attribution spreading within each segment. We therefore analyze segment-length statistics under the initial segmentation and then examine how these lengths decrease across Iterative Split refinement rounds.

Round 0 corresponds to the global PELT segmentation, while rounds 1–3 apply iterative refinement. Refinement substantially reduces the median segment length (a) and increases the mean number of segments per instance (b), with diminishing changes after round 2–3.

In Table 7.1, we report the median segment length, maximum segment length, and the mean number of segments from round 0 to round 3. In Figure 7.1, we visualize the same trend across datasets using boxplots.

(a) `median len` across datasets. The y-axis shows the median segment length for each dataset. At $r = 0$, the box is high and exhibits large outliers (points near ~ 100–140), indicating that some datasets contain very long segments. From $r = 1$ to $r = 2$, the median segment length decreases sharply (the box drops towards ~ 5–10), showing that refinement splits overly long segments and increases granularity. From $r = 2$ to $r = 3$, the change is smaller (the boxes are already low), indicating saturation: after 2–3 rounds, additional gains in granularity are limited.

(b) `mean num segments` across datasets. The y-axis shows the mean number of segments per instance for each dataset. At $r = 0$, there are few segments (typically ~ 4–10), corresponding to a coarse partition. From $r = 1$ to $r = 2$ to $r = 3$, the number of segments increases steadily (the box moves upward), with some datasets reaching very high values (outliers > 50), indicating that refinement can unlock a much finer segmentation where there is room to split. Here as well, we observe diminishing returns: the largest increase occurs early, and then the trend stabilizes.

As shown in Table 7.1 and Figure 7.1, the initial global segmentation ($r = 0$) can be under-segmented because the global penalty favors a compact partition and therefore tends to merge multiple local patterns into a single long segment. In the context of segment-based explanations, such long segments reduce interpretability because any segment-level attribution must be distributed across many time points, leading

Table 7.1: Segment-length statistics across IterativeSplit refinement rounds (r0 → r3).

Dataset	**median_len (r0→r3)**	**max_len (r0→r3)**	**mean #segments (r0→r3)**
Coffee	40.0 → 5.0	150.0 → 80.0	5.86 → 33.14
Wine	15.0 → 5.0	125.0 → 75.0	6.24 → 25.89
BirdChicken	40.0 → 10.0	180.0 → 70.0	11.25 → 41.70
ECG200	16.0 → 5.0	76.0 → 40.0	4.18 → 13.37
Chinatown	5.0 → 5.0	10.0 → 5.0	4.99 → 5.00
BeetleFly	30.0 → 10.0	125.0 → 40.0	14.35 → 56.85
SonyAIBORobotSurface1	15.0 → 5.0	65.0 → 25.0	5.18 → 12.31
ShapeletSim	97.5 → 10.0	500.0 → 220.0	2.87 → 38.76
GunPoint	45.0 → 5.0	115.0 → 100.0	3.38 → 10.79
PowerCons	20.0 → 5.0	144.0 → 65.0	4.93 → 16.40
ToeSegmentation1	25.0 → 5.0	140.0 → 70.0	8.95 → 36.17
GunPointAgeSpan	5.0 → 5.0	70.0 → 30.0	21.09 → 28.25
GunPointMaleVersusFemale	5.0 → 5.0	85.0 → 35.0	21.08 → 28.24
ProximalPhalanxOutlineCorrect	15.0 → 5.0	35.0 → 30.0	4.97 → 12.44
Strawberry	20.0 → 5.0	95.0 → 80.0	5.55 → 22.85
Earthquakes	136.0 → 10.0	512.0 → 290.0	2.55 → 39.72

to coarse importance. IterativeSplit addresses this by selectively revisiting only overly long segments and lowering the penalty locally, which allows additional changepoints to be introduced where the signal supports them, while leaving already reasonable segments unchanged. The stabilization observed after $r \approx 2$–3 rounds suggests that further rounds mainly increase segmentation complexity.

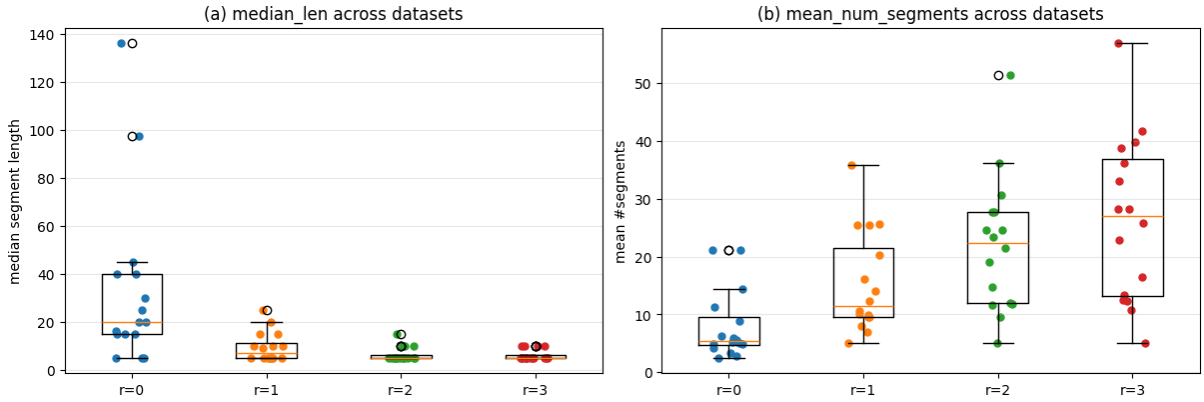

Figure 7.1: granularity vs round.

7.2 Local Explanation Results

In this section, we examine and analyze the results for faithfulness, robustness, and runtime of the local explanations.

7.2.1 Local Faithfulness Results

We evaluate the faithfulness of local explanations by perturbing the top-10% most important time points according to each method and measuring the resulting change in the predicted probability. We compare our method against two TSHAP baselines.

Observed Trends: As shown in Table 7.2 and Figure 7.2, SegSHAP generally improves in faithfulness as refinement rounds are added, with $r = 3$ often returning the strongest performance. A natural explanation is that refinement increases granularity in a targeted way: when the initial segmentation is too coarse, important and non-important regions may be merged into a long segment, and the resulting attribution is then spread over a broad interval. By re-segmenting only overly long segments with a smaller local penalty, IterativeSplit can separate more meaningful temporal patterns, allowing the attribution map to better isolate the regions that truly drive the classifier's output. This effect is particularly visible in datasets where faithfulness increases consistently from round to round, suggesting that each additional refinement step reveals additional structure that is useful for attribution.

At the same time, Table 7.2 also shows non-monotonic behavior on a small number of datasets (e.g., Coffee and Strawberry). Such deviations can arise when refinement introduces boundaries that split a previously coherent predictive region into subseg-

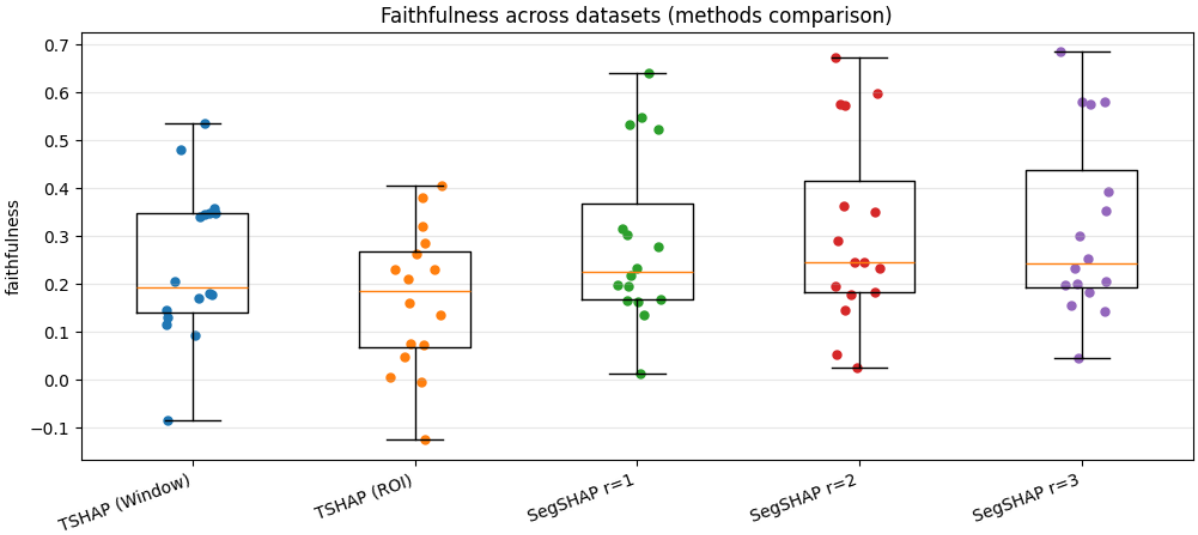

Figure 7.2: Faithfulness distributions across datasets for SegSHAP (rounds $r = 0 \dots 3$) and the TSHAP baselines.

ments whose individual contributions are weaker or whose ranking changes under the top-10% perturbation. In other cases, a finer segmentation can also redistribute attribution mass across more segments, so that the set of top-ranked time points changes in a way that does not always increase the signed perturbation score.

Finally, in Table 7.18 we observe that improvements from $r = 3$ to $r = 4$ occur in only a small subset of datasets. This suggests that, for most datasets, the refinement process has already recovered the main useful structure by $r \approx 2$–3, and additional rounds mainly increase segmentation complexity without commensurate benefit in faithfulness. For this reason, we adopt $r = 3$ as the default refinement depth in the remainder of our experiments.

Table 7.2 reports faithfulness per dataset for SegSHAP with refinement rounds ($r = 0 \dots 3$) and for both baselines (TSHAP-Window and TSHAP-ROI).

Figure 7.2 summarizes the across-dataset picture: the distributions for SegSHAP at $r = 2$ and $r = 3$ are shifted toward higher faithfulness compared to the TSHAP baselines. It also highlights variability across datasets—improvements are not uniform and some outliers remain—but the overall trend favors using refinement rounds.

Dataset	SegSHAP (iterative split) Faithfulness				TSHAP Faithfulness (baseline)	
	r=0	r=1	r=2	r=3	Window	ROI
Coffee	0.1647	0.1956	0.0523	0.1533	0.3385	0.2283
Wine	0.6490	0.6385	0.5974	**0.5748**	0.4803	0.2606
BirdChicken	0.1026	0.1611	0.1824	**0.1959**	0.1301	0.0464
ECG200	0.0781	0.1653	0.1438	**0.1422**	0.1137	0.0735
Chinatown	0.2311	0.2311	0.2311	**0.2311**	0.1790	0.0038
BeetleFly	0.2900	0.5224	0.6714	**0.6848**	0.3568	0.2297
SonyAIBORobotSurface1	0.1270	0.1665	0.1779	**0.1811**	0.1697	0.0725
ShapeletSim	0.4083	0.5329	0.5734	**0.5805**	0.5340	0.3191
GunPoint	-0.0734	0.0114	0.0249	0.0434	0.0918	-0.0056
PowerCons	0.3299	0.3017	0.3504	**0.3526**	0.3462	0.2083
ToeSegmentation1	0.5001	0.5476	0.5715	**0.5807**	0.3466	0.4047
GunPointAgeSpan	0.1655	0.1943	0.1954	**0.1984**	0.1431	0.1339
GunPointMaleVersusFemale	0.1566	0.2160	0.2439	**0.2527**	0.1778	-0.1267
ProximalPhalanxOutlineCorrect	0.4232	0.2761	0.3619	**0.3921**	-0.0850	0.3804
Strawberry	0.3364	0.3152	0.2906	0.2036	0.3451	0.2849
Earthquakes	0.0493	0.1346	0.2451	**0.2995**	0.2045	0.1598

Table 7.2: Faithfulness comparison. We highlight SegSHAP at round $r = 3$ when it outperforms both TSHAP baselines (Window and ROI) on the same dataset.

7.2.2 Hierarchical Refinement Results

Table 7.3 compares the faithfulness of SegSHAP across refinement rounds ($r = 0$–3), the hierarchical variant (*Hier.*), and the two TSHAP baselines. Overall, the hierarchical variant is often slightly better than $r = 3$ (highlighted by the red cells), and in several cases it achieves the best overall score among the baselines and the hierarchical method. In the box plot of Figure 7.3, the distributions of SegSHAP at $r = 3$ and SegSHAP (*Hier.*) are overall shifted toward higher values compared to the TSHAP baselines, indicating improved dataset-level faithfulness.

We observe in Table 7.3 that the hierarchical variant improves faithfulness because, within the most important segment, instead of distributing the attribution uniformly, it redistributes it across the subsegments according to their relative importance. Consequently, faithfulness tends to improve on datasets that already exhibit a pattern of improvement from round to round. We also observe that the hierarchical step is useful primarily when the top-1 parent segment can actually be split. In Table 7.24,

Dataset	SegSHAP Faithfulness					TSHAP Faithfulness (baseline)	
	r=0	r=1	r=2	r=3	Hier.	Window	ROI
Coffee	0.1647	0.1956	0.0523	0.1533	**0.1982**	**0.3385**	0.2283
Wine	0.6490	0.6385	0.5974	0.5748	**0.5128**	0.4803	0.2606
BirdChicken	0.1026	0.1611	0.1824	0.1959	**0.2123**	0.1301	0.0464
ECG200	0.0781	0.1653	**0.1438**	0.1422	0.1355	0.1137	0.0735
Chinatown	0.2311	0.2311	0.2311	0.2311	**0.2311**	0.1790	0.0038
BeetleFly	0.2900	0.5224	0.6714	0.6848	**0.6915**	0.3568	0.2297
SonyAIBORobotSurface1	0.1270	0.1665	0.1779	**0.1811**	0.1797	0.1697	0.0725
ShapeletSim	0.4083	0.5329	0.5734	0.5805	**0.5870**	0.5340	0.3191
GunPoint	-0.0734	0.0114	0.0249	0.0434	0.0435	**0.0918**	-0.0056
PowerCons	0.3299	0.3017	**0.3504**	0.3526	0.3488	0.3462	0.2083
ToeSegmentation1	0.5001	0.5476	0.5715	0.5807	**0.5830**	0.3466	0.4047
GunPointAgeSpan	0.1655	0.1943	0.1954	0.1984	**0.2005**	0.1431	0.1339
GunPointMaleVersusFemale	0.1566	0.2160	0.2439	0.2527	**0.2538**	0.1778	-0.1267
ProximalPhalanxOutlineCorrect	0.4232	0.2761	0.3619	0.3921	**0.3947**	-0.0850	0.3804
Strawberry	0.3364	0.3152	0.2906	0.2036	0.2037	**0.3451**	0.2849
Earthquakes	0.0493	0.1346	0.2451	0.2995	**0.3158**	0.2045	0.1598

Table 7.3: Faithfulness comparison. Red background marks the hierarchical variant when it improves over $r = 3$. Bold indicates the best score among TSHAP Window, TSHAP ROI, Hierarchical.

we see that a split is not obtained for all datasets.

In some cases, no split occurs because even with a smaller local penalty β, the local PELT run does not produce any additional changepoints, and the parent segment remains unchanged. Moreover, this behavior differs across datasets, since the top-1 segments have different characteristic lengths and structures depending on the dataset.

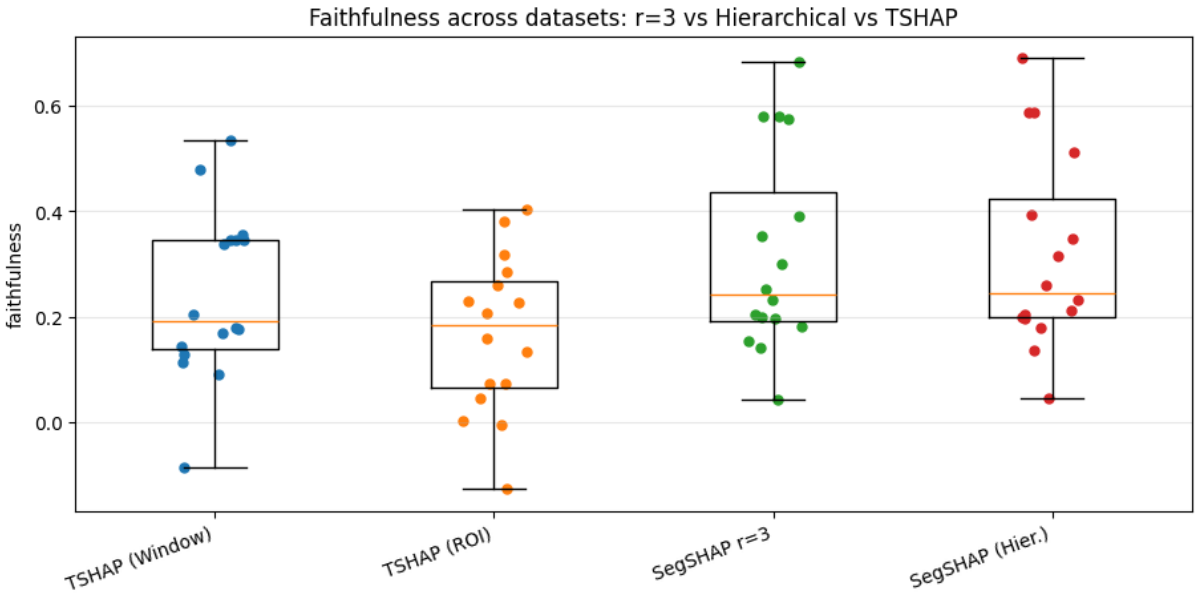

Figure 7.3: Faithfulness distributions across datasets for SegSHAP (rounds $r = 0 \dots 3$) and the TSHAP baselines.

7.2.3 Local Robustness Results

We evaluate robustness by adding Gaussian noise ($\sigma = 0.03$) to each test instance, recomputing the explanation, and measuring the Jaccard overlap between the clean and noisy top-10% time-point sets. We report both the mean Jaccard similarity and the fraction of perturbations exceeding a threshold $\tau = 0.7$, using the same perturbation settings for TSHAP and SegSHAP.

More precisely, for each instance and each noisy explanation, we compute

$$J = \text{Jaccard}(S_0, S_1),$$

where S_0 and S_1 are the top-r% index sets for the clean and noisy attributions, respectively, and we check whether $J \geq \tau$. The robustness rate is then defined as

$$\text{rate} = \frac{\#\{J \geq \tau\}}{\text{total noisy explanations}},$$

i.e., the proportion of perturbation trials for which the explanation is considered stable.

From the results in Table 7.4, we observe that TSHAP (Window) is almost always the most stable method (highest robustness rate), while SegSHAP typically ranks second and outperforms TSHAP (ROI) in nearly all cases. The box plot in Figure 7.4 shows that TSHAP (Window) achieves the highest overall stability across datasets,

Dataset	SegSHAP	TSHAP Window	TSHAP ROI
Coffee	0.579	**0.925**	0.268
Wine	0.046	**0.065**	0.002
BirdChicken	0.380	**0.530**	0.330
ECG200	0.828	**0.964**	0.578
Chinatown	0.988	**0.999**	0.970
BeetleFly	0.455	**0.800**	0.460
SonyAIBORobotSurface1	0.687	**0.924**	0.525
ShapeletSim	0.754	**0.946**	0.898
GunPoint	**0.251**	0.189	0.086
PowerCons	0.852	**0.991**	0.321
ToeSegmentation1	0.848	**0.959**	0.714
GunPointAgeSpan	0.922	**0.998**	0.947
GunPointMaleVersusFemale	0.961	**1.000**	0.966
ProximalPhalanxOutlineCorrect	0.251	**0.343**	0.198
Strawberry	0.226	**0.450**	0.107
Earthquakes	0.586	**0.806**	0.483

Table 7.4: Robustness comparison (Top-10% Jaccard).

whereas TSHAP (ROI) is generally the least stable. SegSHAP lies in between, with robustness rates that are typically higher than ROI and closer to the Window baseline.

TSHAP (Window) is more stable (Table 7.4) because, although the instance changes slightly under noise, the window definition remains the same. In contrast, in our method the noise can change not only the SHAP values computed on a fixed segmentation, but also the segmentation itself, since segmentation is re-computed each time on the noisy signal.

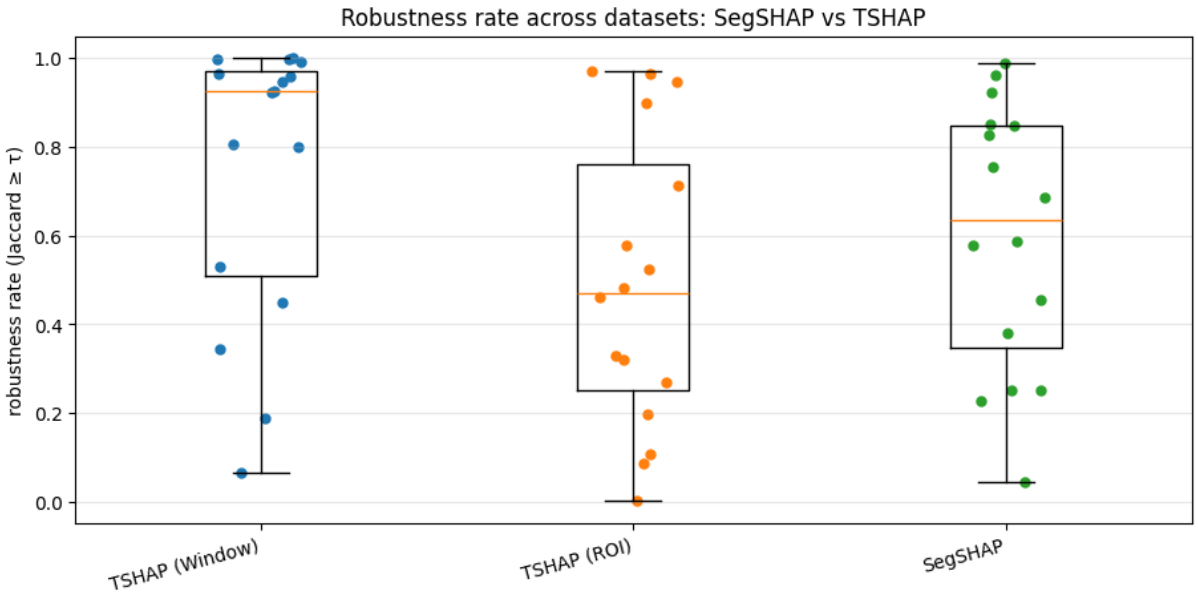

Figure 7.4: Robustness distributions across datasets for SegSHAP (rounds $r = 3$) and the TSHAP baselines.

7.2.4 Local Runtime Results

We measure the time required to produce explanations for the entire dataset. Table 7.5 shows that all methods run in practical time across datasets. TSHAP (Window) is typically the fastest variant, but SegSHAP at $r = 3$ is close in runtime and is almost always faster than TSHAP (ROI). Figure 7.5 summarizes the total runtime per dataset.

SegSHAP is generally slower than TSHAP (Table 7.5) because it first performs segmentation (PELT + iterative splits) and then computes exact segment SHAP on the resulting segments. In contrast, TSHAP (Window) passes once over each window position, whereas our refinement typically requires additional time until it either successfully refines long segments or stops at round $r = 3$ in any case. Nevertheless, the differences are not large, and our method remains computationally efficient in practice. The hierarchical variant incurs extra overhead because it performs an additional local segmentation and attribution step inside the top parent segment, which naturally increases runtime.

Dataset	TSHAP (Window)	TSHAP (ROI)	SegSHAP r=3 (total)	SegSHAP Hier. (total)
Coffee	**1.56**	5.08	4.66	5.77
Wine	**7.10**	9.80	8.06	10.94
BirdChicken	10.14	12.05	**4.90**	7.33
ECG200	**3.25**	4.20	4.53	6.18
Chinatown	**2.43**	5.57	3.61	5.52
BeetleFly	11.01	11.59	**5.29**	10.27
SonyAIBORobotSurface1	**4.09**	6.88	21.08	26.49
ShapeletSim	**43.76**	73.19	47.86	86.64
GunPoint	**7.83**	11.06	9.33	12.62
PowerCons	**9.56**	13.60	11.38	14.81
ToeSegmentation1	**39.05**	43.02	35.60	42.67
GunPointAgeSpan	**19.64**	21.96	22.78	27.83
GunPointMaleVersusFemale	**18.87**	21.56	22.06	27.79
ProximalPhalanxOutlineCorrect	**6.77**	9.20	11.50	15.07
Strawberry	**46.24**	53.26	46.80	66.07
Earthquakes	68.72	52.90	**38.73**	66.45

Table 7.5: Total runtime (seconds). TSHAP reports Window-only and ROI(run) totals. SegSHAP reports end-to-end total time for $r = 3$ and for the hierarchical variant.

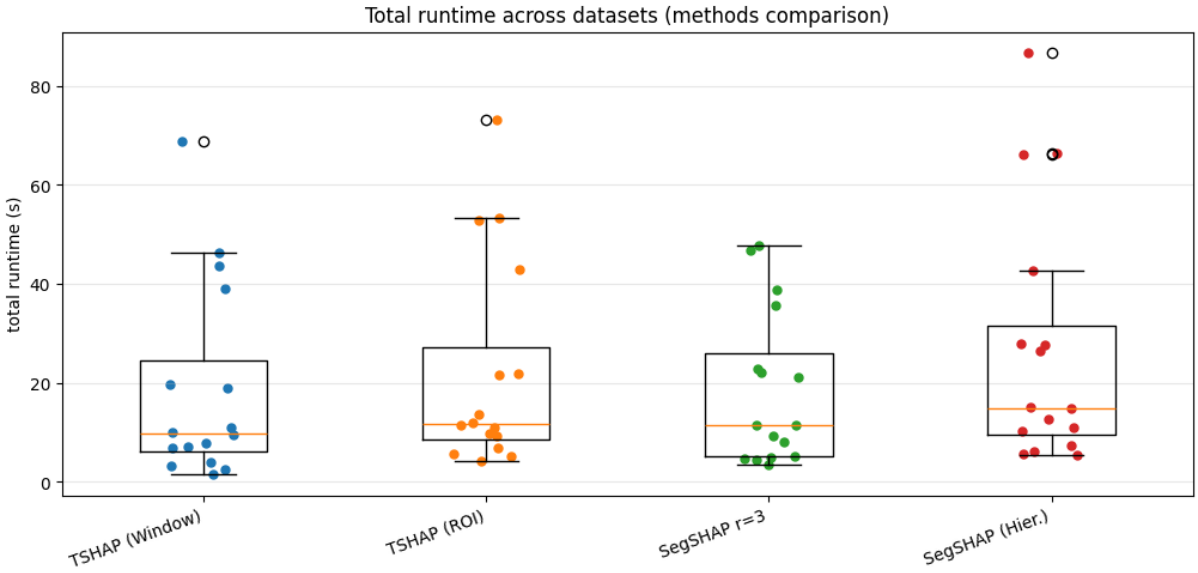

Figure 7.5: Total runtime across datasets.

7.2.5 Local Explanation Visualizations Examples

In Figures 7.6 and 7.8, we illustrate what an explanation looks like for a single instance. The plots show the segments identified by the segmentation step, together with the attribution assigned to each segment. Color indicates whether a segment supports the predicted class (red) or opposes it (blue), while color intensity reflects the strength (magnitude) of the attribution. In Figures 7.7 and 7.9, we show how attribution is redistributed within the top-1 segment through the hierarchical refinement procedure.

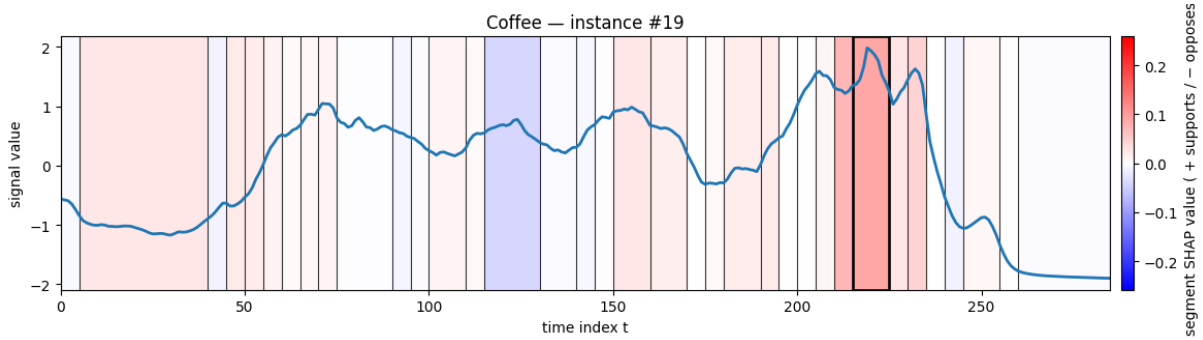

Figure 7.6: Local Explanation.

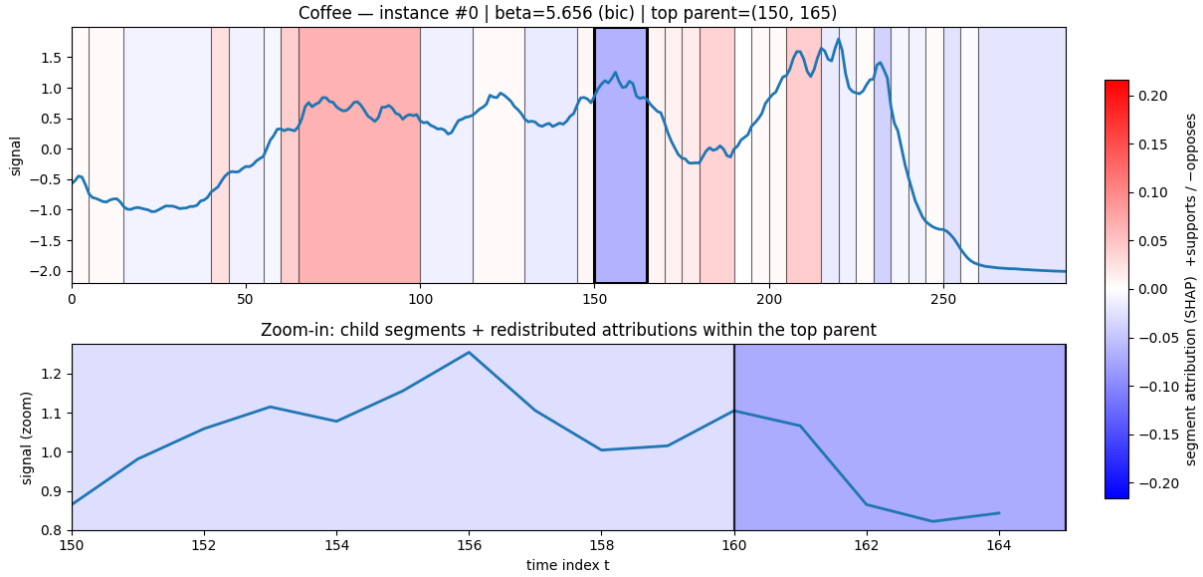

Figure 7.7: Hierarchical Explanation.

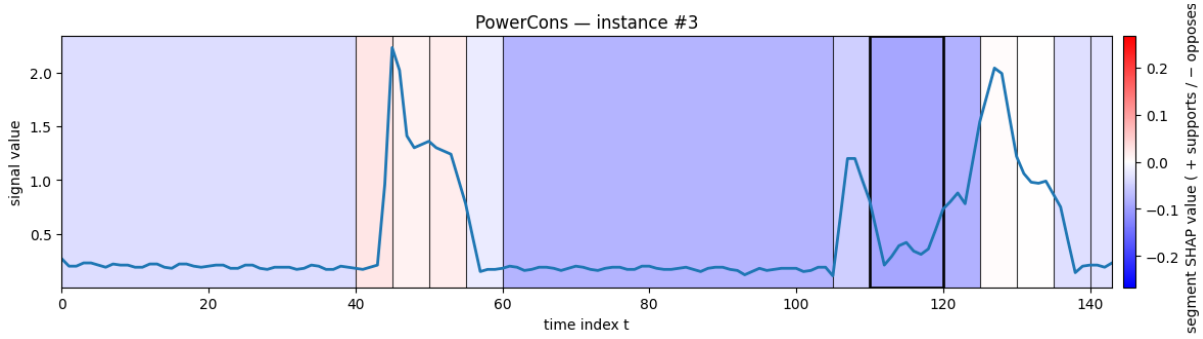

Figure 7.8: Local Explanation.

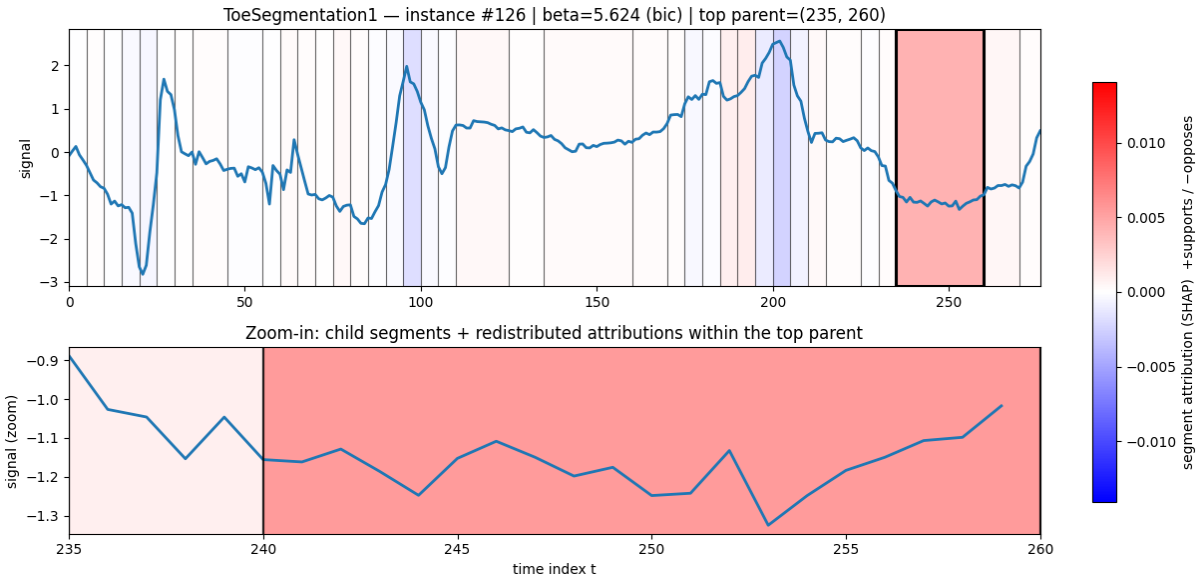

Figure 7.9: Hierarcical Explanation.

7.3 Global Results

In this section, we examine and analyze the results for faithfulness, robustness, and runtime of the global explanations.

7.3.1 Global Faithfulness Results

In this subsection, we report the faithfulness results for our two global explanation variants (Voting and Multivariate). We evaluate faithfulness using two settings:

- **Deletion-based faithfulness**: if we remove (zero out) the most important region, the model's predicted probability should decrease accordingly.
- **Point-perturbation faithfulness**: if we perturb the top-r most important time points, the model output should change in the expected direction.

Table 7.6 presents the top-10% point-perturbation faithfulness scores. Cells highlighted in red indicate datasets where the local method outperforms both global variants, while green highlights which of the two global methods performs best for each dataset. The corresponding distribution across datasets is summarized in Figure 7.10.

Table 7.7 reports the top-1 segment deletion faithfulness results, together with the corresponding boxplot in Figure 7.11.

Point-perturb faithfulness: The key point is that the comparison is fair because, for all methods, we perturb exactly the same fraction of time points (10% of T). Therefore, no method "wins" simply by applying a larger perturbation, the only difference is which points each method considers important. Comparing voting vs multivariate, the multivariate variant returns better point-level faithfulness. The local method is better in most cases (Table 7.6) because it is tailored to each instance.

We also observe that the multivariate global approach is often better than the voting global approach. The reason is that, in the multivariate setting, we detect changepoints on a "multivariate" signal where each instance is treated as a dimension, thus, changes that are consistent across many samples are retained and this returns more stable boundaries. In contrast, voting relies on aggregation of local boundaries, and if there is variability in the local changepoints it may struggle to identify global boundaries, this is consistent with the FAILED cases, where it could not find changepoints supported by 50% of the instances.

In some datasets we observe negative values (e.g., GunPoint, ProximalPhalanxOutlineCorrect). This occurs because the metric computes the mean difference between perturbing "negative" and "positive" contributions. If the attribution signs are misaligned (e.g., a region that is truly supportive is labeled as negative by the explanation), then perturbing the "negative" set may reduce the probability more than perturbing the "positive" set, resulting in a negative mean score.

In the local setting, each instance returns a different top-10% set. For some instances, these selected points can be unrelated to each other and spread across distant time locations, which can reduce the drop when averaging across many instances. The local top-10% for a given instance may be distributed across many distant time points, which can make the perturbation less effective in magnitude than perturbing the top-10% time points induced by a global segmentation, since those points tend to be more localized and closer together.

Deletion Top-1:As expected, local per-instance top-1 deletion is typically the most impactful, because the removed top-1 segment is selected separately for each instance, and the segmentation is computed specifically for that instance. Between the two global methods, the multivariate variant achieves higher faithfulness in most cases, although performance remains dataset-dependent. Overall, for segment-deletion faithfulness, the strongest global approach is the Multivariate Global variant, because it derives global segments directly from the structure of the whole dataset: it treats each instance as a dimension and searches for changepoints that are consistent across many instances.

Dataset	Voting G2	Multivariate G2	Local top-10% perturb (r=3)
Coffee	0.050	0.200	0.153
Wine	0.295	0.462	0.575
BirdChicken	0.055	0.179	0.196
ECG200	0.147	0.195	0.142
Chinatown	0.042	0.231	0.231
BeetleFly	0.128	0.496	0.685
SonyAIBORobotSurface1	0.257	0.188	0.181
ShapeletSim	FAILED	0.607	0.581
GunPoint	-0.066	-0.053	0.043
PowerCons	0.326	0.362	0.353
ToeSegmentation1	0.562	0.547	0.581
GunPointAgeSpan	0.255	0.202	0.198
GunPointMaleVersusFemale	0.238	0.265	0.253
ProximalPhalanxOutlineCorrect	-0.179	-0.455	0.392
Strawberry	0.119	0.163	0.204
Earthquakes	FAILED	0.209	0.300

Table 7.6: Point-perturb faithfulness comparison (top-10%, zero). Green highlights the better global between Voting and Multivariate. Red highlights Local when it outperforms both global.

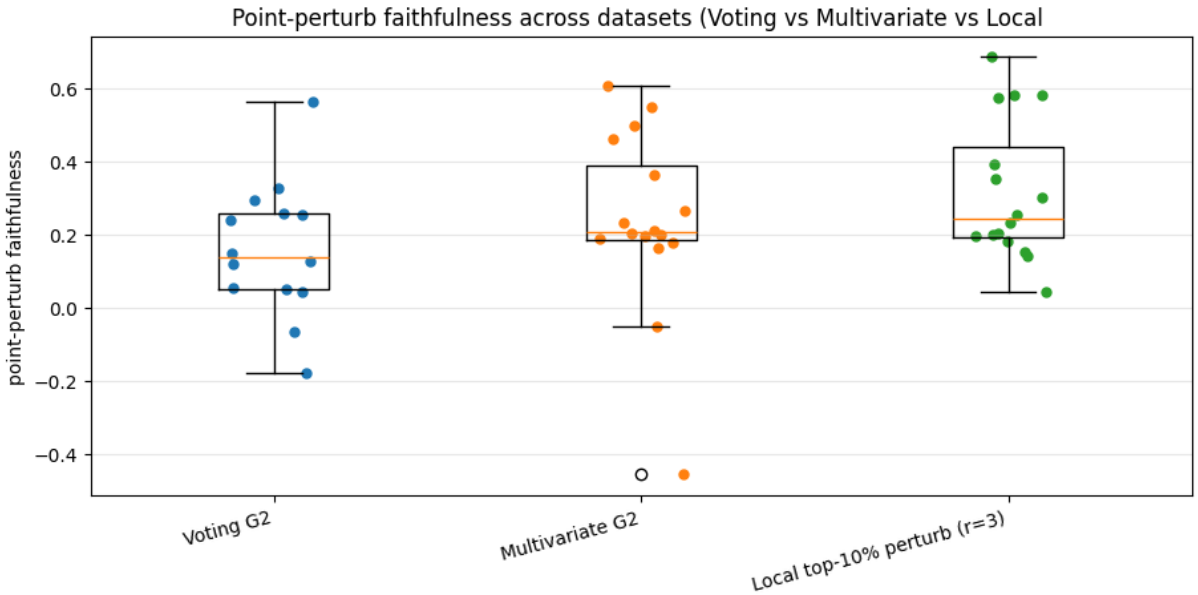

Figure 7.10: Point-perturb faithfulness across datasets.

Dataset	Local		Voting	Multivariate
	Per-inst top1	Most-freq top1	Global top1	Global top1
Coffee	0.138	0.381	0.100	0.381
Wine	0.554	0.410	0.410	0.410
BirdChicken	0.055	0.013	0.129	0.027
ECG200	0.083	0.063	0.042	0.063
Chinatown	0.286	0.290	0.289	0.290
BeetleFly	0.148	0.067	0.386	0.061
SonyAIBORobotSurface1	0.162	0.118	0.071	0.118
ShapeletSim	0.487	0.041	FAILED	0.056
GunPoint	0.049	0.004	0.387	0.054
PowerCons	0.137	0.071	0.037	0.071
ToeSegmentation1	0.174	0.042	0.029	0.028
GunPointAgeSpan	0.077	0.041	0.035	0.036
GunPointMaleVersusFemale	0.359	0.341	0.129	0.341
ProximalPhalanxOutlineCorrect	0.233	0.263	0.271	0.242
Strawberry	0.194	0.052	0.049	0.052
Earthquakes	0.111	0.037	FAILED	0.033

Table 7.7: Top 1 segment faithfulness metrics. Green highlights the best score among {Voting Global top1, Multivariate Global top1, Most-freq Local top1}. Red highlights Local Per-inst top1 when it outperforms all other available columns.

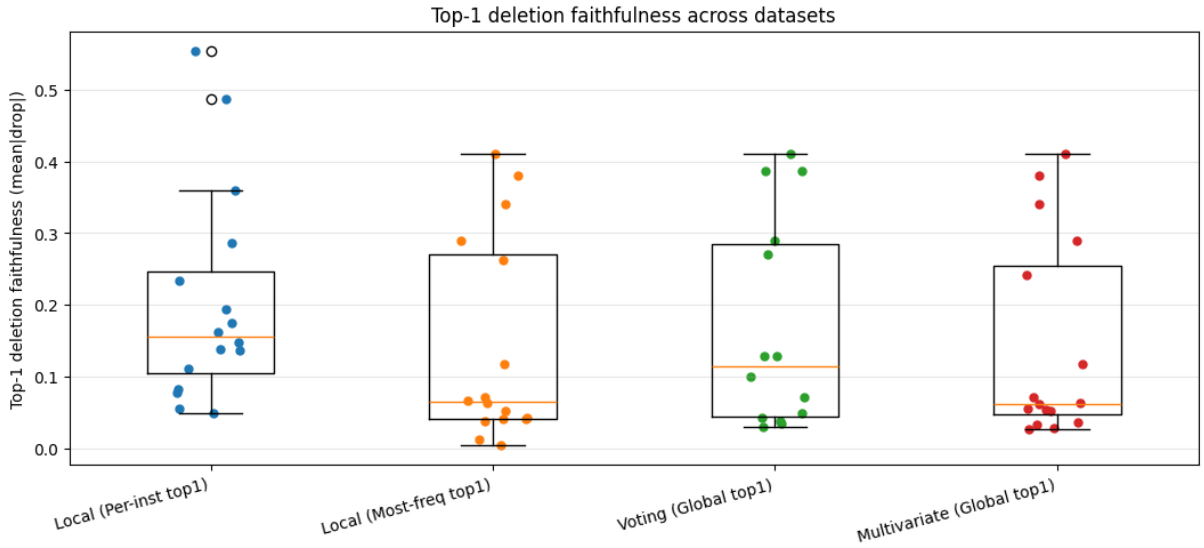

Figure 7.11: Top-1 deletion faithfulness across datasets.

7.3.2 Global Class-wise Faithfulness

Tables 7.8 and 7.9 report the class-wise faithfulness results for the two global variants (Voting and Multivariate), using both the top-10% point perturbation metric and the top-1 segment deletion metric. The green cells indicate, for each dataset and class, which of the two global methods achieves the higher score.

In Tables 7.8 and 7.9, we observe that faithfulness scores can differ substantially, not only between the two methods (Voting and Multivariate), but also between the two classes within the same dataset. This class-wise asymmetry is expected in time-series classification, since the two classes may exhibit different patterns and temporal structure, and in some cases these patterns may be easier or harder to identify.

Dataset	Voting		Multivariate	
	class 0	class 1	class 0	class 1
Coffee	-0.061	0.025	0.105	0.511
Wine	0.184	0.253	0.379	0.530
BirdChicken	0.151	0.003	0.362	0.006
ECG200	0.282	0.099	0.355	0.105
Chinatown	0.430	-0.101	0.915	-0.027
BeetleFly	0.057	0.548	0.214	0.730
SonyAIBORobotSurface1	0.414	0.032	0.309	0.027
ShapeletSim	FAILED	FAILED	0.682	0.535
GunPoint	-0.190	-0.002	-0.103	-0.003
PowerCons	0.007	0.620	0.005	0.718
ToeSegmentation1	0.906	0.196	0.888	0.170
GunPointAgeSpan	0.190	0.556	0.216	0.187
GunPointMaleVersusFemale	0.002	0.533	0.000	0.559
ProximalPhalanxOutlineCorrect	0.017	0.296	-0.345	-0.582
Strawberry	0.205	0.023	0.394	0.217
Earthquakes	FAILED	0.081	0.199	0.311

Table 7.8: Class-wise faithfulness scores (top-10% point perturbation, zero). Green marks the larger value between Voting and Multivariate for the corresponding class.

Dataset	Voting		Multivariate	
	class 0	class 1	class 0	class 1
Coffee	0.003	0.003	0.003	0.407
Wine	0.080	0.741	0.080	0.741
BirdChicken	0.127	0.014	0.080	0.009
ECG200	0.069	0.039	0.069	0.029
Chinatown	0.953	0.028	0.982	0.028
BeetleFly	0.004	0.417	0.017	0.120
SonyAIBORobotSurface1	0.069	0.009	0.172	0.016
ShapeletSim	FAILED	FAILED	0.110	0.030
GunPoint	0.007	0.005	0.006	0.007
PowerCons	0.001	0.028	0.002	0.215
ToeSegmentation1	0.042	0.017	0.048	0.021
GunPointAgeSpan	0.014	0.043	0.035	0.062
GunPointMaleVersusFemale	0.002	0.083	0.002	0.751
ProximalPhalanxOutlineCorrect	0.747	0.476	0.165	0.046
Strawberry	0.115	0.747	0.114	0.021
Earthquakes	FAILED	0.868	0.034	0.078

Table 7.9: Class-wise top-1 deletion. Green marks the larger value between Voting and Multivariate for the corresponding class.

7.3.3 Global Top-1 Segments Summary

We observe in Tables 7.10 and 7.11 that, in some datasets, the top-1 segments for the two classes overlap in time points, and in some cases the global top-1 segments also agree. This indicates that something "important" happens in the time series at these time points—important for both classes, but in a different way (e.g., one class may rely on the presence of a pattern, while the other relies on its absence or on a different manifestation of the same event). In other datasets, we observe different top-1 segments per class, which is also reasonable, since each class may rely on different regions to make the decision.

Moreover, Voting and Multivariate sometimes produce very similar top-1 segments (Table 7.12), for example in *Strawberry*, *GunPointAgeSpan*, and *GunPointMaleVersusFemale*; in these cases, the two methods agree, so this interval may indeed be important for the dataset overall. There are also cases where the two methods identify different top-1 segments. Voting is based on aggregation from local boundaries (and can therefore be affected by variability in the local segmentations). Multivariate relies on changepoints that are consistent "horizontally" across many instances. Therefore, when heterogeneity is high, it is reasonable that Voting "pulls" toward frequently occurring boundaries, whereas Multivariate prefers segments that are more consistent overall.

Dataset	Global top-1		Most-freq local top-1			Class A global top-1		Class B global top-1	
	seg	len	seg	len	supp.	seg	len	seg	len
Coffee	(226,234)	8	(220,235)	15	6/28	(227,233)	6	(172,188)	16
Wine	(126,144)	18	(125,145)	20	24/54	(37,48)	11	(127,143)	16
BirdChicken	(376,469)	93	(395,405)	10	2/20	(377,413)	36	(0,43)	43
ECG200	(41,44)	3	(40,45)	5	16/100	(41,43)	2	(82,96)	14
Chinatown	(0,4)	4	(0,5)	5	310/343	(0,3)	3	(0,3)	3
BeetleFly	(231,494)	263	(245,280)	35	1/20	(407,493)	86	(22,108)	86
SonyAIBORobotSurface1	(11,14)	3	(10,15)	5	130/601	(51,53)	2	(36,38)	2
ShapeletSim	FAILED	–	(290,305)	15	4/180	FAILED	–	FAILED	–
GunPoint	(0,44)	44	(95,100)	5	19/150	(96,98)	2	(0,48)	48
PowerCons	(116,119)	3	(110,115)	5	42/180	(111,113)	2	(117,119)	2
ToeSegmentation1	(56,69)	13	(195,210)	15	10/228	(52,68)	16	(202,208)	6
GunPointAgeSpan	(131,134)	3	(15,20)	5	28/316	(46,48)	2	(137,139)	2
GunPointMaleVersusFemale	(61,64)	3	(60,65)	5	49/316	(61,63)	2	(62,64)	2
ProximalPhalanxOutlineCorrect	(51,80)	29	(50,80)	30	169/291	(52,80)	28	(52,68)	16
Strawberry	(96,109)	13	(95,110)	15	35/370	(97,108)	11	(157,203)	46
Earthquakes	FAILED	–	(20,45)	25	4/139	FAILED	–	(0,493)	493

Table 7.10: Voting segments summary.Dataset-level Global top-1 segment, the Most-frequent local top-1, and class-wise Global top-1 segments.

Dataset	Global top-1		Most-freq local top-1			Class A global top-1		Class B global top-1	
	seg	len	seg	len	supp.	seg	len	seg	len
Coffee	(220,235)	15	(220,235)	15	6/28	(225,240)	15	(225,235)	10
Wine	(125,145)	20	(125,145)	20	24/54	(35,45)	10	(125,135)	10
BirdChicken	(385,405)	20	(395,405)	10	2/20	(390,420)	30	(385,405)	20
ECG200	(40,45)	5	(40,45)	5	16/100	(35,40)	5	(40,45)	5
Chinatown	(0,5)	5	(0,5)	5	310/343	(0,5)	5	(0,5)	5
BeetleFly	(295,315)	20	(245,280)	35	1/20	(455,475)	20	(295,315)	20
SonyAIBORobotSurface1	(10,15)	5	(10,15)	5	130/601	(10,15)	5	(35,40)	5
ShapeletSim	(150,170)	20	(290,305)	15	4/180	(480,500)	20	(380,395)	15
GunPoint	(15,20)	5	(95,100)	5	19/150	(90,95)	5	(15,20)	5
PowerCons	(110,115)	5	(110,115)	5	42/180	(110,115)	5	(115,120)	5
ToeSegmentation1	(200,210)	10	(195,210)	15	10/228	(180,190)	10	(200,210)	10
GunPointAgeSpan	(130,135)	5	(15,20)	5	28/316	(45,50)	5	(130,135)	5
GunPointMaleVersusFemale	(60,65)	5	(60,65)	5	49/316	(55,60)	5	(65,70)	5
ProximalPhalanxOutlineCorrect	(65,70)	5	(50,80)	30	169/291	(40,45)	5	(65,70)	5
Strawberry	(95,110)	15	(95,110)	15	35/370	(95,110)	15	(60,70)	10
Earthquakes	(25,50)	25	(20,45)	25	4/139	(25,45)	20	(150,180)	30

Table 7.11: Multivariate segments summary.Dataset-level Global top-1 segment, the Most-frequent local top-1, and class-wise Global top-1 segments.

Dataset	Voting top-1		Multivariate top-1	
	seg	len	seg	len
Coffee	(226,234)	8	(220,235)	15
Wine	(126,144)	18	(125,145)	20
BirdChicken	(376,469)	93	(385,405)	20
ECG200	(41,44)	3	(40,45)	5
Chinatown	(0,4)	4	(0,5)	5
BeetleFly	(231,494)	263	(295,315)	20
SonyAIBORobotSurface1	(11,14)	3	(10,15)	5
ShapeletSim	FAILED	--	(150,170)	20
GunPoint	(0,44)	44	(15,20)	5
PowerCons	(116,119)	3	(110,115)	5
ToeSegmentation1	(56,69)	13	(200,210)	10
GunPointAgeSpan	(131,134)	3	(130,135)	5
GunPointMaleVersusFemale	(61,64)	3	(60,65)	5
ProximalPhalanxOutlineCorrect	(51,80)	29	(65,70)	5
Strawberry	(96,109)	13	(95,110)	15
Earthquakes	FAILED	--	(25,50)	25

Table 7.12: Global top-1 segment (interval and length) under Voting and Multivariate segmentation.

7.3.4 Global Robustness Results

Robustness is measured under Gaussian noise by (i) the exact match rate of the global top-1 segment and (ii) the mean Jaccard overlap of the top-10% most important time points across perturbation runs. The results are reported in Table 7.13.

Table 7.13 shows that robustness of global explanations is dataset-dependent: on several datasets the Multivariate variant exhibits higher stability (especially in the Top-10% mean Jaccard), while on others Voting is comparable or better.

We observe that some datasets remain highly stable (with large robustness scores), whereas others exhibit much lower stability. For the Voting variant, this behavior may be due to the fact that it needs to recompute the local explanations and then build the histogram; these intermediate steps may be more sensitive to noise. Conversely, when the Multivariate variant performs poorly, this may be because the injected noise directly affects the segmentation step.

Dataset	Voting		Multivariate	
	Top-1 exact-rate	Top-10% Jacc-mean	Top-1 exact-rate	Top-10% Jacc-mean
Coffee	1.000	0.954	1.000	0.849
Wine	0.000	0.393	0.000	0.195
BirdChicken	0.000	0.330	0.600	0.537
ECG200	1.000	0.982	1.000	0.867
Chinatown	1.000	1.000	1.000	0.600
BeetleFly	0.900	0.607	1.000	0.993
SonyAIBORobotSurface1	1.000	0.975	1.000	1.000
ShapeletSim	FAILED	FAILED	0.800	0.544
GunPoint	0.000	0.073	0.000	0.120
PowerCons	0.100	0.875	1.000	1.000
ToeSegmentation1	0.400	0.822	1.000	0.556
GunPointAgeSpan	0.000	0.071	0.000	0.000
GunPointMaleVersusFemale	1.000	0.967	1.000	1.000
ProximalPhalanxOutlineCorrect	1.000	0.143	1.000	0.455
Strawberry	1.000	0.539	0.000	0.263
Earthquakes	FAILED	FAILED	1.000	1.000

Table 7.13: Robustness comparison. Green highlights the better score between Voting and Multivariate, separately for Top-1 exact-rate and Top-10% Jaccard mean.

7.3.5 Global Class-wise Robustness

We also measure robustness per class to examine whether one class returns less stable explanations and whether stability differs across classes.

Table 7.14 reports class-wise robustness under Gaussian noise: we report the exact match rate of the global top-1 segment across perturbation runs, computed separately for each class (higher is better).

Table 7.15 reports the same setting, but summarizes stability using the mean Jaccard overlap of the top-10% most important time points, higher values indicate more stable salient regions.

Tables 7.14 and 7.15 show that class-wise robustness is strongly dataset- and class-dependent.We observe that when robustness is high, this typically holds for both classes, meaning that the global segments are more stable in those cases. There are also exceptions where one class is more robust than the other, because segmentation may be easier and more stable for that class.

Dataset	Voting		Multivariate	
	class 0	class 1	class 0	class 1
Coffee	0.900	0.900	1.000	1.000
Wine	0.000	0.000	0.000	0.000
BirdChicken	0.100	0.200	0.000	1.000
ECG200	1.000	0.900	1.000	1.000
Chinatown	1.000	1.000	1.000	1.000
BeetleFly	0.900	0.500	1.000	1.000
SonyAIBORobotSurface1	1.000	1.000	1.000	1.000
ShapeletSim	FAILED	FAILED	0.900	0.800
GunPoint	0.000	0.000	1.000	0.000
PowerCons	1.000	0.700	1.000	1.000
ToeSegmentation1	0.500	0.800	1.000	1.000
GunPointAgeSpan	0.200	0.000	0.900	0.000
GunPointMaleVersusFemale	0.000	0.000	1.000	1.000
ProximalPhalanxOutlineCorrect	1.000	1.000	1.000	1.000
Strawberry	1.000	0.000	0.000	1.000
Earthquakes	FAILED	FAILED	1.000	1.000

Table 7.14: Class-wise robustness (Top-1 exact-rate). Green highlights the better score between Voting and Multivariate for each dataset and class.

Dataset	Voting		Multivariate	
	class 0	class 1	class 0	class 1
Coffee	0.858	0.909	0.836	0.811
Wine	0.298	0.425	0.207	0.053
BirdChicken	0.328	0.097	0.007	0.765
ECG200	0.948	1.000	1.000	1.000
Chinatown	1.000	1.000	1.000	1.000
BeetleFly	0.733	0.777	0.625	1.000
SonyAIBORobotSurface1	0.975	0.950	0.911	1.000
ShapeletSim	FAILED	FAILED	0.758	0.812
GunPoint	0.900	0.000	0.500	0.020
PowerCons	0.912	0.853	1.000	1.000
ToeSegmentation1	0.558	0.558	0.545	0.907
GunPointAgeSpan	0.429	0.000	0.800	0.000
GunPointMaleVersusFemale	0.283	0.514	1.000	1.000
ProximalPhalanxOutlineCorrect	0.474	0.143	0.455	0.455
Strawberry	0.291	0.702	0.092	0.688
Earthquakes	FAILED	FAILED	0.903	0.584

Table 7.15: Class-wise robustness (Top-10% Jaccard mean). Green highlights the better score between Voting and Multivariate for each dataset and class.

7.3.6 Runtime

Table 7.17 reports a runtime breakdown for global explanation generation. Across almost all datasets, the multivariate variant is substantially faster than voting in total time and this is mainly due to the time required for segmentation: it is more time-consuming to perform local segmentation and then build the histogram in order to identify and inspect candidate segments.

Dataset	Voting (total time)	Multivariate (total time)
Coffee	16.220	2.120
Wine	29.516	2.276
BirdChicken	17.154	1.843
ECG200	29.552	3.338
Chinatown	7.214	2.450
BeetleFly	1.784	2.013
SonyAIBORobotSurface1	158.357	14.161
ShapeletSim	FAILED	22.411
GunPoint	13.603	9.022
PowerCons	25.778	10.056
ToeSegmentation1	95.169	17.017
GunPointAgeSpan	61.191	20.377
GunPointMaleVersusFemale	62.789	20.126
ProximalPhalanxOutlineCorrect	19.516	7.861
Strawberry	79.319	18.779
Earthquakes	FAILED	13.034

Table 7.16: Total runtime (seconds) for global explanation generation. Green highlights the faster method per dataset.

Dataset	Voting			Multivariate		
	Seg	Expl	Total	Seg	Expl	Total
Coffee	1.775	14.445	16.220	0.048	2.073	2.120
Wine	3.290	26.226	29.516	0.033	2.244	2.276
BirdChicken	5.640	11.514	17.154	0.014	1.829	1.843
ECG200	12.144	17.408	29.552	0.019	3.320	3.338
Chinatown	0.749	6.465	7.214	0.006	2.445	2.450
BeetleFly	0.746	1.038	1.784	0.012	2.001	2.013
SonyAIBORobotSurface1	38.625	119.732	158.357	0.007	14.154	14.161
ShapeletSim	19.194	FAILED	FAILED	0.010	22.401	22.411
GunPoint	5.659	7.944	13.603	0.010	9.012	9.022
PowerCons	5.023	20.755	25.778	0.004	10.052	10.056
ToeSegmentation1	9.603	85.567	95.169	0.016	17.002	17.017
GunPointAgeSpan	2.148	59.042	61.191	0.006	20.370	20.377
GunPointMaleVersusFemale	3.003	59.786	62.789	0.009	20.117	20.126
ProximalPhalanxOutlineCorrect	4.439	15.077	19.516	0.015	7.846	7.861
Strawberry	19.262	60.057	79.319	0.037	18.742	18.779
Earthquakes	16.609	FAILED	FAILED	0.020	13.015	13.034

Table 7.17: Runtime breakdown (seconds) for global explanation generation.

7.3.7 Global Explanation Visualizations Examples

Figures 7.13 and 7.12 illustrate global explanations produced with the multivariate segmentation variant.

In Figure 7.13, the y-axis lists the global segments as intervals (s_m, e_m), while the x-axis shows the dataset-level mean *signed* attribution for each segment. Red bars correspond to segments that, on average, push the model output toward the explained class/output, whereas blue bars indicate segments that contribute in the opposite direction. The bar length reflects the strength of the average effect across all instances.

In Figure 7.12, we plot the dataset mean signal together with the variability across instances, and we overlay the global segmentation along the time axis. The colored background bands indicate the signed contribution of each global segment (red supporting, blue opposing), while their intensity is proportional to the magnitude of the segment's average attribution. This highlights which temporal regions are most influential at the dataset level.

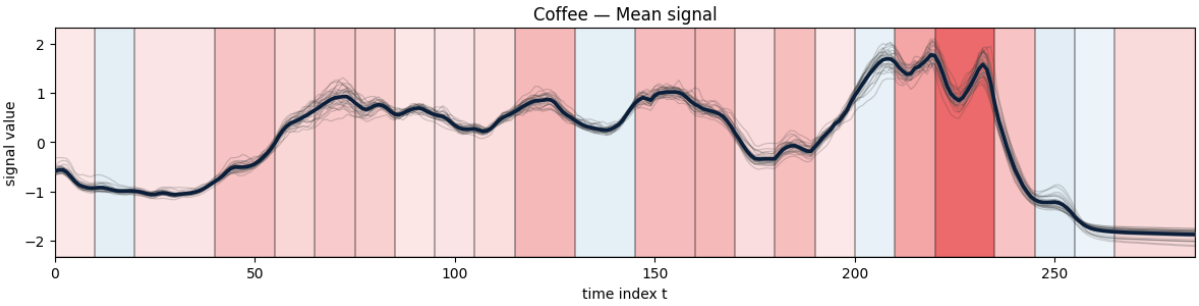

Figure 7.12: Coffee Global Explanation.

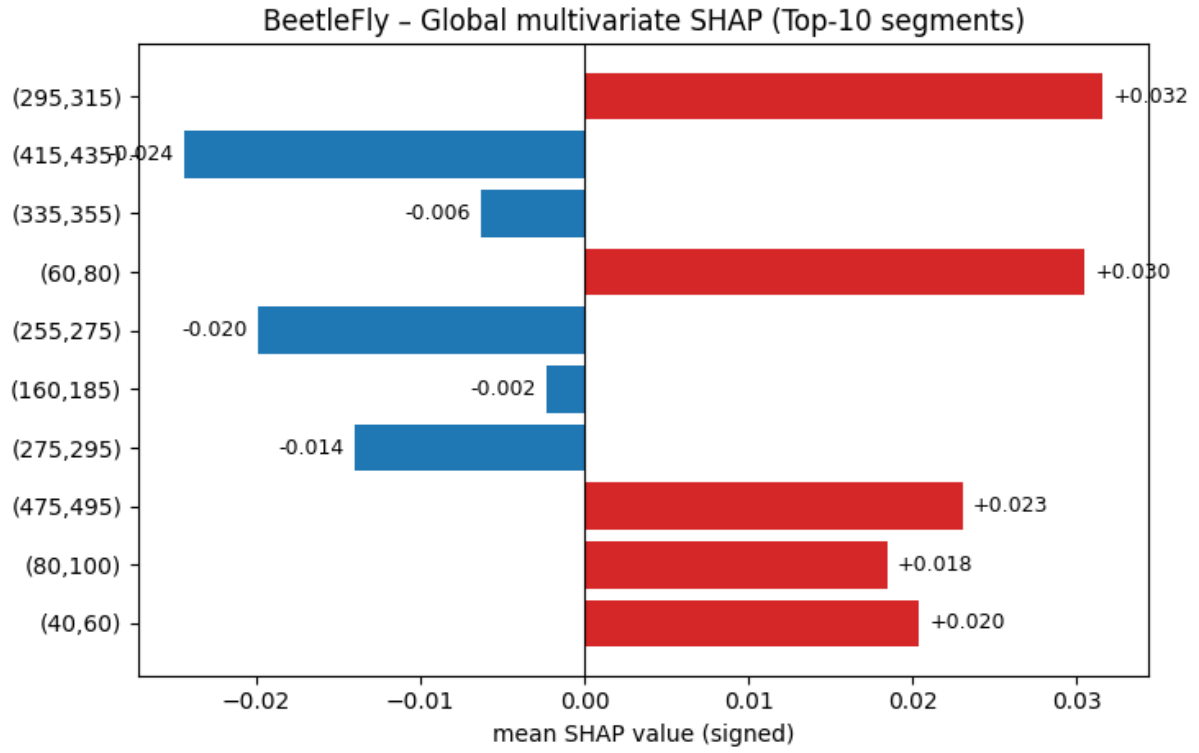

Figure 7.13: Beetlefly Globar Bar.

7.4 Ablation Studies

7.4.1 Number of Refinement Rounds

Table 7.18 reports faithfulness across iterative refinement rounds ($r = 0 \ldots 4$). Red shading highlights rounds where faithfulness increases relative to the previous round. We observe gains from $r = 0$ up to $r = 3$ in the majority of datasets, whereas improvements from $r = 3$ to $r = 4$ occur in substantially fewer cases (only a small subset of datasets). Since improvements level off after $r = 3$, we use $r = 3$ as the default refinement depth for the remainder of our experiments.

Dataset	r0	r1	r2	r3	r4
Coffee	0.1647	0.1956	0.0523	0.1533	0.2080
Wine	0.6490	0.6385	0.5974	0.5748	0.5552
BirdChicken	0.1026	0.1611	0.1824	0.1959	0.1800
ECG200	0.0781	0.1653	0.1438	0.1422	0.1403
Chinatown	0.2311	0.2311	0.2311	0.2311	0.2311
BeetleFly	0.2900	0.5224	0.6714	0.6848	0.6833
SonyAIBORobotSurface1	0.1270	0.1665	0.1779	0.1811	0.1794
ShapeletSim	0.4083	0.5329	0.5734	0.5805	0.5999
GunPoint	-0.0734	0.0114	0.0249	0.0434	0.0428
PowerCons	0.3299	0.3017	0.3504	0.3526	0.3502
ToeSegmentation1	0.5001	0.5476	0.5715	0.5807	0.5809
GunPointAgeSpan	0.1655	0.1943	0.1954	0.1984	0.1988
GunPointMaleVersusFemale	0.1566	0.2160	0.2439	0.2527	0.2526
ProximalPhalanxOutlineCorrect	0.4232	0.2761	0.3619	0.3921	0.3958
Strawberry	0.3364	0.3152	0.2906	0.2036	0.1907
Earthquakes	0.0493	0.1346	0.2451	0.2995	0.3355

Table 7.18: Faithfulness across iterative refinement rounds.

7.4.2 Penalty Selection (BIC vs AIC)

In practice, AIC and BIC produced very similar segment-length statistics (e.g., number of segments, and mean/max segment length) in our experiments. We nevertheless adopt BIC as the default because its penalty includes a $\log n$ term and therefore depends on the sample size n, making it more adaptive to series length than AIC, whose penalty is independent of n. The corresponding statistics are summarized in Table 7.19.

Penalty trade-off (under- vs. over-segmentation). In penalized changepoint detection, the penalty controls a direct trade-off between under-segmentation and over-segmentation: a large penalty favors fewer changepoints (and can merge distinct regimes), whereas a small penalty encourages many changepoints (risking overfit-

Dataset	AIC			BIC		
	mean len	max len	#segs mean	mean len	max len	#segs mean
Coffee	7.44	55.0	38.46	8.63	80.0	33.14
Wine	8.73	75.0	26.80	9.04	75.0	25.89
BirdChicken	10.24	45.0	50.00	12.28	70.0	41.70
ECG200	6.47	35.0	14.83	7.18	40.0	13.37
Chinatown	4.80	5.0	5.00	4.80	5.0	5.00
BeetleFly	8.85	20.0	57.85	9.01	40.0	56.85
SonyAIBORobotSurface1	5.44	20.0	12.87	5.69	25.0	12.31
ShapeletSim	8.77	70.0	57.01	12.90	220.0	38.76
GunPoint	12.04	100.0	12.46	13.90	100.0	10.79
PowerCons	8.07	60.0	17.84	8.78	65.0	16.40
ToeSegmentation1	6.57	55.0	42.19	7.66	70.0	36.17
GunPointAgeSpan	5.19	30.0	28.88	5.31	30.0	28.25
GunPointMaleVersusFemale	5.20	30.0	28.86	5.31	35.0	28.24
ProximalPhalanxOutlineCorrect	6.34	30.0	12.61	6.43	30.0	12.44
Strawberry	8.64	65.0	27.18	10.28	80.0	22.85
Earthquakes	8.96	140.0	57.12	12.89	290.0	39.72

Table 7.19: Segment statistics after round 3 under AIC and BIC penalties.

ting). This corresponds to the standard penalized objective $\sum_i C(\cdot) + \beta f(m)$, where the penalty term guards against overfitting, and linear penalties βm are commonly used in practice, including AIC and BIC.

Finally, although AIC/BIC are typically motivated for likelihood-based segmentation costs, in our implementation, this is consistent with using an l_2 (squared-error) segment cost.

7.4.3 Sensitivity to the Segmentation Cost Model (l2 vs normal)

We use the l_2 segment cost as the default. The `rbf` cost is kernel-based rather than a standard parametric likelihood cost; therefore, its connection to AIC/BIC is less direct. The l_2 cost corresponds to mean squared error (SSE within each segment), whereas normal corresponds to a Gaussian likelihood–based cost. We perform a sensitivity check by replacing the default l_2 segment cost with the normal cost model in PELT, while keeping the rest of the pipeline fixed. The normal cost yields comparable faithfulness on several datasets, but it also leads to negative faithfulness values in a few

Dataset	Faithfulness (normal)	Faithfulness (l_2)
Coffee	0.074	0.153
Wine	0.336	0.575
BirdChicken	0.143	0.196
ECG200	0.195	0.142
Chinatown	0.231	0.231
BeetleFly	0.665	0.685
SonyAIBORobotSurface1	0.189	0.181
ShapeletSim	0.625	0.581
GunPoint	-0.010	0.043
PowerCons	0.353	0.353
ToeSegmentation1	0.588	0.581
GunPointAgeSpan	0.202	0.198
GunPointMaleVersusFemale	0.265	0.253
ProximalPhalanxOutlineCorrect	-0.424	0.392
Strawberry	0.162	0.204
Earthquakes	0.312	0.300

Table 7.20: SegSHAP faithfulness under two segmentation cost models (normal vs. l_2).

cases. We therefore use the l_2 cost as the default choice, since it provides strong overall performance while remaining simple, general, and free of stronger distributional assumptions. The results are reported in Table 7.20.

7.4.4 Refinement length threshold

We study the sensitivity of IterativeSplit to the refinement length threshold, which determines when a segment is considered "too long" and thus becomes a candidate for local re-segmentation. Concretely, we vary the threshold as a fraction of the series length T and refine segments whose length satisfies $L > \rho \cdot T$. This ablation evaluates

Dataset	Length threshold (0.03)	Length threshold (0.05)	Length threshold (0.10)	Length threshold (0.15)
Coffee	0.153	0.160	0.128	0.254
Wine	0.575	0.596	0.551	0.536
BirdChicken	0.196	0.197	0.178	0.091
ECG200	0.142	0.142	0.106	0.060
Chinatown	0.231	0.231	0.231	0.231
BeetleFly	0.685	0.550	0.360	0.304
SonyAIBORobotSurface1	0.181	0.181	0.181	0.171
ShapeletSim	0.581	0.557	0.540	0.531
GunPoint	0.043	0.043	0.068	0.057
PowerCons	0.353	0.353	0.292	0.280
ToeSegmentation1	0.581	0.575	0.541	0.521
GunPointAgeSpan	0.198	0.198	0.170	0.170
GunPointMaleVersusFemale	0.253	0.253	0.225	0.203
ProximalPhalanxOutlineCorrect	0.392	0.392	0.392	0.314
Strawberry	0.204	0.252	0.283	0.323
Earthquakes	0.300	0.278	0.242	0.225

Table 7.21: Faithfulness for different length-threshold settings. Green highlights the best faithfulness per dataset.

how the choice of ρ affects faithfulness. The results are reported in Table 7.21.

We additionally report how often the refinement mechanism is triggered with $\rho = 0.03$ (i.e., the number of segments exceeding the length threshold; Table 7.22) and how often it results in an actual split. This indicates that IterativeSplit does not enforce refinement uniformly, but performs splits only when the local PELT objective supports a non-trivial segmentation.

Dataset	Long segments found	Split success
Coffee	748	583
Wine	938	756
BirdChicken	527	491
ECG200	2392	764
Chinatown	1723	2
BeetleFly	594	589
SonyAIBORobotSurface1	15686	3597
ShapeletSim	2634	2618
GunPoint	1585	1036
PowerCons	2569	1590
ToeSegmentation1	6671	4981
GunPointAgeSpan	2631	2007
GunPointMaleVersusFemale	2620	2005
ProximalPhalanxOutlineCorrect	7153	1591
Strawberry	7089	5001
Earthquakes	2298	2059

Table 7.22: Iterative splitting statistics for SegSHAP (cap_frac=0.03, cost_model=l_2, BIC).

7.4.5 Minimum Segment Length

We study the sensitivity of our method to the minimum segment length parameter, which is a standard parameter of the PELT implementation in `ruptures`. This parameter enforces a hard constraint that no segment shorter than min_size is allowed, thereby controlling how fine-grained the segmentation can become. In this ablation, we vary min_size and report the resulting changes in faithfulness. Table 7.23 shows that varying the minimum segment length parameter min_size over the tested range results in essentially identical faithfulness across all datasets. This indicates that, under our current refinement setting ($\rho = 0.03$ and max_rounds=3), faithfulness is largely insensitive to `min_size`. A plausible explanation is that the refinement step dominates the final segmentation granularity in these experiments.

Dataset	min_size=0.01	min_size=0.03	min_size=0.05	min_size=0.10
Coffee	0.153	0.153	0.153	0.153
Wine	0.575	0.575	0.575	0.575
BirdChicken	0.196	0.196	0.196	0.196
ECG200	0.142	0.142	0.142	0.142
Chinatown	0.231	0.231	0.231	0.231
BeetleFly	0.685	0.685	0.685	0.685
SonyAIBORobotSurface1	0.181	0.181	0.181	0.181
ShapeletSim	0.581	0.581	0.581	0.581
GunPoint	0.043	0.043	0.043	0.043
PowerCons	0.353	0.353	0.353	0.353
ToeSegmentation1	0.581	0.581	0.581	0.581
GunPointAgeSpan	0.198	0.198	0.198	0.198
GunPointMaleVersusFemale	0.253	0.253	0.253	0.253
ProximalPhalanxOutlineCorrect	0.392	0.392	0.392	0.392
Strawberry	0.204	0.204	0.204	0.204
Earthquakes	0.299	0.299	0.299	0.299

Table 7.23: Faithfulness across different min size values.

7.4.6 Hierarchical refinement statistics.

Table 7.24 summarizes how often the hierarchical actually refines the most influential (top-1) segment. For each dataset, we report the number of child subsegments produced by the local re-segmentation and the resulting split fraction (splits/N). Overall, the table shows that the top segment split is applied selectively (often only when the top segment is sufficiently long), while some datasets rarely or never trigger a valid split.

Dataset	Split fraction (splits/N)
Coffee	5/28
Wine	9/54
BirdChicken	12/20
ECG200	5/100
Chinatown	0/343
BeetleFly	16/20
SonyAIBORobotSurface1	7/601
ShapeletSim	180/180
GunPoint	5/150
PowerCons	9/180
ToeSegmentation1	14/228
GunPointAgeSpan	13/316
GunPointMaleVersusFemale	3/316
ProximalPhalanxOutlineCorrect	11/291
Strawberry	71/370
Earthquakes	118/139

Table 7.24: Hierarchical split fraction for top-1 segments (splits/N).

7.4.7 Global-Voting tolerance window w

We study the sensitivity of the voting-based global segmentation to the tolerance window w, which controls how votes are assigned around each detected changepoint. Concretely, a changepoint at position cp contributes votes not only at cp itself, but across the neighborhood

$$[cp - w,\ cp + w].$$

Thus, w determines the alignment tolerance for declaring that multiple instances exhibit a changepoint in the same temporal region, even if their detected changepoints do not appear at the exact same index. We observe in Table 7.25 that as the voting

Table 7.25: Number of global segments as a function of the voting window size.

Dataset	window=0	window=1	window=3
Coffee	27	79	115
Wine	21	61	95
BirdChicken	12	34	196
ECG200	14	40	51
Chinatown	5	13	16
BeetleFly	5	13	159
SonyAIBORobotSurface1	14	40	44
ShapeletSim	1	1	129
GunPoint	9	25	47
PowerCons	20	58	70
ToeSegmentation1	46	136	153
GunPointAgeSpan	30	88	92
GunPointMaleVersusFemale	30	88	92
ProximalPhalanxOutlineCorrect	11	31	38
Strawberry	22	64	82
Earthquakes	1	1	15

window increases, the number of detected global changepoints (and thus the number of global segments) also increases.However, when the tolerance window is large, changepoints that occur at different nearby indices are more likely to be grouped together, reducing the precision of their temporal localization. Therefore, we use `window` $= 1$ as the default setting.

7.4.8 Global Voting Threshold (ρ).

The parameter ρ defines the threshold used to decide how many instances must have a changepoint at a given location for it to be considered a global changepoint. In other words, ρ sets the minimum fraction of instances that must "support" a changepoint at that position. We study how the number of detected global changepoints—and consequently the number of global segments—changes for different values of ρ. We

Table 7.26: Number of global segments for different ρ values (vote_window=1).

Dataset	**ρ=0.4**	**ρ=0.5**	**ρ=0.7**	**ρ=0.9**
Coffee	85	79	58	46
Wine	64	61	61	58
BirdChicken	106	34	1	1
ECG200	49	40	28	16
Chinatown	13	13	13	13
BeetleFly	31	13	10	1
SonyAIBORobotSurface1	40	40	37	22
ShapeletSim	7	1	1	1
GunPoint	31	25	7	1
PowerCons	70	58	25	1
ToeSegmentation1	160	136	1	1
GunPointAgeSpan	88	88	88	76
GunPointMaleVersusFemale	88	88	88	70
ProximalPhalanxOutlineCorrect	37	31	31	28
Strawberry	73	64	37	16
Earthquakes	1	1	1	1

observe in Table 7.26 that as ρ increases, the number of global segments decreases, because an increasingly larger fraction of instances must agree on a changepoint for it to be considered global. We choose $\rho = 0.5$ as the default, since for larger ρ values we often observe that the entire dataset is treated as a single segment, i.e., no common changepoints are detected.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

Explainability in time series remains a challenging problem because information is strongly temporally dependent, and classical XAI methods rely on assumptions that do not transfer directly to the time-series setting. Two critical issues are (i) the choice of background/baseline, *i.e.*, what "absence" of information means when we remove parts of a time series, and (ii) the choice of segmentation, *i.e.*, which temporal intervals we map the explanation onto so that it is both interpretable and faithful to the model.

In this thesis, we showed that moving to segment-level explanations—where segmentation is obtained without requiring a manual specification of the number of changepoints or the window length—substantially improves faithfulness, because replacement is applied over coherent temporal intervals that better correspond to meaningful structures in the signal. Moreover, we showed that when the explanation is constructed hierarchically, the result can become more precise at the level of subsegments: rather than assuming that an "important" segment is uniformly responsible, hierarchical refinement isolates the most critical sub-interval, leading to more targeted and interpretable explanations.

For global explanations, the multivariate approach proved particularly strong in terms of robustness, providing stable explanations at the dataset/class level and a

clearer picture of which temporal patterns systematically influence the model's behavior. Overall, the results support that a unified combination of (a) adaptive segmentation, (b) hierarchical refinement for more accurate attribution at the subsegment level, and (c) global explanations for an overall dataset/class-level view constitutes a practical direction towards more faithful and useful explanations for time series classification.

Despite these positive results, we identify a clear limitation: segmentation can be sensitive to noise, meaning that in noisy versions of the same signal, the segmentation may change and lead to variations in the robustness of the explanation. This highlights that explanation stability depends not only on the explainer itself, but also on the stability of the underlying segmentation.

8.2 Future Work

A first direction for future work is to further extend and evaluate the proposed approach on multivariate datasets. In particular, although the global segmentation stage has already been adapted using multivariate PELT, a more complete multivariate formulation is still needed, including hierarchical refinement. In the multivariate setting, such refinement could be especially useful, since the initial coarse explanation may identify a broader time interval that is important across all features, while a subsequent hierarchical step could further refine this interval into smaller subregions, potentially in a feature-specific way. This would be meaningful because an important event in a multivariate time series may span multiple time points and multiple channels. Thus, a coarse explanation could indicate when an important event occurs, whereas a hierarchical refinement could reveal more precisely in which subintervals and in which dimensions the most critical information is concentrated.

A second, and particularly important, direction is a more systematic study of the background problem. In time series, replacing "missing" intervals with a simple baseline may either remove too much information or introduce artificial patterns. Future work could examine backgrounds that produce realistic but "non-informative" completions, *i.e.*, completions that match the statistical structure of the time series without introducing evidence in favour of a particular class. In this context, generative models that learn the temporal structure of the data and can produce realistic backgrounds

or complete missing intervals could be explored, with the goal that the replacement remains natural without injecting additional class-relevant information.

Finally, evaluation across a broader range of models and architectures would help test whether the conclusions generalize to different types of classifiers. A more extensive experimental study could also relate faithfulness/robustness metrics to domain-expert–driven evaluation criteria, such as whether explanations are understandable and useful to users in realistic settings, and—where ground truth is available—to agreement measures between the highlighted segments and known relevant temporal regions.

Bibliography

[1] M. T. Mohsin and N. B. Nasim, "Explaining the unexplainable: A systematic review of explainable ai in finance," 2025. [Online]. Available: https://arxiv.org/abs/2503.05966

[2] I. Neves, D. Folgado, S. Santos, M. Barandas, A. Campagner, L. Ronzio, F. Cabitza, and H. Gamboa, "Interpretable heartbeat classification using local model-agnostic explanations on ecgs," *Computers in Biology and Medicine*, vol. 133, p. 104393, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482521001876

[3] A. Theissler, F. Spinnato, U. Schlegel, and R. Guidotti, "Explainable AI for time series classification: A review, taxonomy and research directions," *IEEE Access*, vol. 10, pp. 100 700–100 724, 2022. [Online]. Available: https://doi.org/10.1109/ACCESS.2022.3207765

[4] T. Rojat, R. Puget, D. Filliat, J. D. Ser, R. Gelin, and N. D. Rodríguez, "Explainable artificial intelligence (XAI) on timeseries data: A survey," *CoRR*, vol. abs/2104.00950, 2021. [Online]. Available: https://arxiv.org/abs/2104.00950

[5] T. Sivill and P. Flach, "Limesegment: Meaningful, realistic time series explanations," in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Camps-Valls, F. J. R. Ruiz, and I. Valera, Eds., vol. 151. PMLR, 28–30 Mar 2022, pp. 3418–3433. [Online]. Available: https://proceedings.mlr.press/v151/sivill22a.html

[6] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should I trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco,*

CA, USA, August 13-17, 2016, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, Eds. ACM, 2016, pp. 1135–1144. [Online]. Available: https://doi.org/10.1145/2939672.2939778

[7] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," *CoRR*, vol. abs/1705.07874, 2017. [Online]. Available: http://arxiv.org/abs/1705.07874

[8] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, ser. FAT* '20. ACM, Jan. 2020, p. 607–617. [Online]. Available: http://dx.doi.org/10.1145/3351095.3372850

[9] M. Guilleme, V. Masson, L. Roze, and A. Termier, "Agnostic local explanation for time series classification," 11 2019, pp. 432–439.

[10] J. Bento, P. Saleiro, A. F. Cruz, M. A. T. Figueiredo, and P. Bizarro, "Timeshap: Explaining recurrent models through sequence perturbations," *CoRR*, vol. abs/2012.00073, 2020. [Online]. Available: https://arxiv.org/abs/2012.00073

[11] A. Nayebi, S. Tipirneni, C. K. Reddy, B. Foreman, and V. Subbian, "Windowshap: An efficient framework for explaining time-series classifiers based on shapley values," 2023. [Online]. Available: https://arxiv.org/abs/2211.06507

[12] T. L. Nguyen and G. Ifrim, "[tshap: Fast and exact shap for explaining time series classification and regression," 2025.

[13] Z. Wang, I. Samsten, I. Miliou, R. Mochaourab, and P. Papapetrou, "Glacier: guided locally constrained counterfactual explanations for time series classification," *Mach. Learn.*, vol. 113, no. 7, pp. 4639–4669, 2024. [Online]. Available: https://doi.org/10.1007/s10994-023-06502-x

[14] E. Ates, B. Aksar, V. J. Leung, and A. K. Coskun, "Counterfactual explanations for multivariate time series," in *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*. IEEE, May 2021, p. 1–8. [Online]. Available: http://dx.doi.org/10.1109/ICAPAI49758.2021.9462056

[15] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical*

Association, vol. 107, no. 500, p. 1590–1598, Oct. 2012. [Online]. Available: http://dx.doi.org/10.1080/01621459.2012.737745

[16] J. V. Braun, R. K. Braun, and H.-G. Müller, "Multiple changepoint fitting via quasilikelihood, with application to dna sequence segmentation," *Biometrika*, vol. 87, no. 2, pp. 301–314, 2000. [Online]. Available: http://www.jstor.org/stable/2673465

[17] C. Inclan and G. C. Tiao, "Use of cumulative sums of squares for retrospective detection of changes of variance," *Journal of the American Statistical Association*, vol. 89, no. 427, pp. 913–923, 1994.

[18] F. Picard, E. Lebarbier, M. Hoebeke, G. Rigaill, B. Thiam, and S. Robin, "Joint segmentation, calling, and normalization of multiple cgh profiles," *Biostatistics*, vol. 12, no. 3, pp. 413–428, 2011.

[19] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, no. 6, pp. 716–723, 1974.

[20] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[21] Y.-C. Yao, "Estimation of a noisy discrete-time step function: Bayes and empirical bayes approaches," *The Annals of Statistics*, vol. 12, no. 4, pp. 1434–1447, 1984.

[22] B. Jackson, J. D. Sargle, D. Barnes, S. Arabhi, A. Alt, P. Gioumousis, E. Gwin, P. Sangtrakulcharoen, L. Tan, and T. T. Tsai, "An algorithm for optimal partitioning of data on an interval," *IEEE Signal Processing Letters*, vol. 12, no. 2, pp. 105–108, 2005.

[23] H. A. Dau, A. J. Bagnall, K. Kamgar, C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. J. Keogh, "The UCR time series archive," *CoRR*, vol. abs/1810.07758, 2018. [Online]. Available: http://arxiv.org/abs/1810.07758

[24] A. Dempster, F. Petitjean, and G. I. Webb, "ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels," *CoRR*, vol. abs/1910.13051, 2019. [Online]. Available: http://arxiv.org/abs/1910.13051

[25] A. Dempster, D. F. Schmidt, and G. I. Webb, "MINIROCKET: A very fast (almost) deterministic transform for time series classification," *CoRR*, vol. abs/2012.08791, 2020. [Online]. Available: https://arxiv.org/abs/2012.08791

Short Biography

Iro Skandali was born in Ioannina, Greece, in 1999. She completed her undergraduate studies in the Department of Mathematics at the University of Ioannina in 2022, with most of her courses focused on Probability, Statistics, and Operations Research. She subsequently completed her Master's degree in the Department of Computer Science and Engineering at the same university, specializing in Data Science and Engineering, in 2026. Her thesis focused on explainability in time series, with an emphasis on interpreting classification models. Her research interests include data analysis, machine learning, and explainability in machine learning models.