Received 15 September 2023, accepted 14 October 2023, date of publication 23 October 2023, date of current version 2 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3326847

RESEARCH ARTICLE

Explainable Time Series Tree: An Explainable Top-Down Time Series Segmentation Framework

VITOR DE CASTRO SILVA[1], **BRUNO BOGAZ ZARPELÃO**[1], **ERIC MEDVET**[2], **AND SYLVIO BARBON JR.**[2]

[1]Department of Computer Science, State University of Londrina, Londrina, Paraná 86057-970, Brazil

[2]Department of Engineering and Architecture, University of Trieste, 34100 Trieste, Italy

Corresponding author: Sylvio Barbon Jr. (sylvio.barbonjunior@units.it)

This work was supported in part by Fundação Araucária de Apoio ao Desenvolvimento Científico e Tecnológico do Estado do Paraná, Brazil (Novos Arranjos de Pesquisa e Inovação (NAPI)-Norte Centro de Inteligência Artificial (CIA)-Agro), under Grant 137/2021.

ABSTRACT A wide range of Machine Learning algorithms can model time series to address classification, forecasting, and clustering problems. However, time series may exhibit characteristics that complicate these tasks, such as repeating patterns and seasonal variations. Time series segmentation could be a solution to these problems, but current approaches need to be improved. Most of them employ linear regression to solve problems such as detecting changes in a series' behaviour, bypassing tools specifically designed for these challenges, such as change detectors. Moreover, explainability is seldom taken into account during time series segmentation. To automatically identify different time series patterns using appropriate techniques while leveraging explainability, we proposed the eXplainable Time Series Tree (XTSTree). XTSTree divides a time series into a binary tree, hierarchically splitting it according to a criterion based on change detectors, ideally finding a cutting point that creates the two most different sub-series. The segmentation process continues until it reaches a stopping condition, which relies on a stationarity test that assesses whether the series has a sufficiently homogeneous behaviour. Based on well-behaved segments, XTSTree paves the way for a more comprehensive pattern explanation and also supports the application of explainable approaches. Our experimental study applied XTSTree on several real-life time series to isolate the series' different behaviours. To evaluate the effectiveness of our method, we used Symbolic Regression to find the best representation of the time series and its splits using algebraic expressions, comparing the differences before and after XTSTree. We show an improvement in terms of expression complexity, improving the model accuracy compared to the original time series.

INDEX TERMS Concept drift, time series, meta-learning, time series segmentation, symbolic regression.

I. INTRODUCTION

Temporal data is a common type of data that has been increasingly used in several tasks from different areas, such as e-health [1], IoT (Internet of Things) systems [2], and autonomous vehicles [3]. The most common way to represent temporal data for modelling and prediction is through the use of time series. Several studies have already demonstrated that time series classification and prediction can be used to build relevant applications. In [4], a model for human behaviour is created and subsequently used to indicate and suggest actions for other humans. Another study [2] uses data from a blood pressure monitor to predict blood pressure levels and point out which routine changes caused the most amount of change in blood pressure.

One problem these techniques face is the change in data behaviour through time [5], which often happens due to the nature of data, such as a temperature drop during a rain period or a heart rate rise during stress. Often defined as *concept drift*, these changes require retraining models along the time series, or assuring models are suited for every behaviour present. Moreover, these behaviours can repeat over time,

The associate editor coordinating the review of this manuscript and approving it for publication was Jad Nasreddine.

such as in seasons over a year or Internet traffic over a week. Identifying areas with different behaviours can ease time series modelling since different behaviours can be treated individually. One way to identify similar behaviours in a time series is through time series segmentation [6].

Time series segmentation is the process of obtaining segments of a time series that either have similar behaviour or behave differently from the rest of the series [6]. There are three main types of segmentation algorithms: sliding window algorithms, which create and repeatedly increase the size of a window from the start of the series until it reaches an error threshold, repeating the process with a new window at that point [7]; bottom-up algorithms, which partition the time series into several small pieces and merge the least influential ones until a threshold is met [8]; and top-down algorithms, which find the best cutting point in a series, and then repeat the process for each sub-series that falls above a user-defined threshold [9].

Proposed top-down algorithms often derive the error value and stopping condition from a linear regression model [9]. Furthermore, the best cutting point is either tied to the linear regression or is found through an exhaustive search of possible partitions [10]. In the former, the position where the series deviates the most from the expected value given by the linear regression model is chosen as a split point. This process is similar to detecting a concept drift, a problem for which there are existing tools, such as change detectors [11].

In this paper, we present the eXplainable Time Series Tree, or XTSTree, a top-down segmentation algorithm that uses a statistical test as a stopping criterion for the segmentation process, and a change detector to find the cutting position. Unlike other top-down segmentation methods, our approach uses a change detector to identify the best cutting positions, instead of relying on an error function based on linear regression. The primary goal of the XTSTree framework is to provide a tree-based structure that enables in-depth comprehension of the patterns within a time series in an explainable manner. By representing each leaf as a contiguous pattern, XTSTree facilitates intuitive modelling and analysis of individual segments. This tree-based approach ensures interpretability through the hierarchy of cuts, allowing users to understand and interpret the diverse range of patterns captured within the time series. Additionally, XTSTree is scalable due to the binary tree's low memory cost and the change detector's efficiency, making it suitable for processing large and complex datasets, and since it treats the sub-series as individual time series, it allows for the integration of advanced techniques and algorithms, further enhancing its accuracy and versatility in real-world time series analysis scenarios. In this study, we created an implementation using the Page-Hinkley change detector (PH) [11] and the Augmented Dickey-Fuller test (ADF) [12].

To test the effectiveness of XTSTree, we segmented several different weather-related time series using both the original XTSTree and other splitting methods and stopping criteria as baselines. Afterwards, we used an implementation of Symbolic Regression (SR) using Genetic Programming (GP) to create algebraic formulas that describe the series' behaviour as a whole and segmented. Finally, we compared formula complexity, prediction errors, and the number of cuts made.

The manuscript is organised as follows. Section II presents definitions of concepts related to explainability, time series segmentation, and tools used by the methodology. Section III discusses related work in the field of time series segmentation. Section IV introduces the proposed approach. Section IV-A contains a toy experiment explaining the step-by-step process for the segmentation of a synthetic time series and the formulas obtained from SR before and after segmentation. Section V presents the experimental evaluation and analysis. Finally, Section VI makes the concluding remarks.

II. BACKGROUND

In this section, we explain concepts relevant to understanding the segmentation method and stopping criterion implemented by XTSTree, as well as the evaluation method for its effectiveness.

A. EXPLAINABILITY AND INTERPRETABILITY

Nauta et al. [13] define the concept of an explanation in the context of explainable AI as "a presentation of (aspects of) the reasoning, functioning and/or behaviour of a machine learning model in human-understandable terms". In the context of this definition, reasoning is the process behind a decision, functioning is how data is stored and manipulated inside a model, and behaviour is how the model works from an outside view, such as what input created a given output.

The quality of explanations given by an explained model can be defined as a balanced combination of interpretability and completeness [14]. Interpretability can be defined as the description of the components and results of a model in a meaningful way to humans. Completeness is an accurate definition of how the model works that enables a predictable sequence of actions and results. Good explanations achieve a good balance of interpretability and completeness.

A machine learning model can be explained through a range of different explanation methods [13]. The explanations created can be represented by different representation types, such as decision trees, visual representations, and natural language. A model can also be intrinsically interpretable, providing explanations through its functioning and results.

B. TIME SERIES SEGMENTATION

Time series segmentation is a form of data pre-processing for time series data mining [6]. Given a time series X such that $X = [x_1, x_2 \ldots x_n], x_i \in R$, the segmentation process aims to find $k \in N$ distinct points such that $k_0 = 0, k_K = N$ and $F(X) \geq F(X_{[k_i,k_{i+1}]}), i \in [1 \ldots K]$, where F is a modelling technique that outputs a score or error value. There are several different implementations of time series segmentation in the literature, the majority of them being based on Piecewise Linear Representation (PLR), which is the process of reducing a time series to a sequence of lines [6], using mostly Linear Regression as a modelling technique. There are three main types of segmentation algorithms: sliding window, bottom-up, and top-down [10].

Sliding window algorithms work by setting an anchor at the start of the time series and increasing the size of a window towards the end of the series until a user-specified error threshold for a prediction model is met, at which point the window becomes a segment, the stopping point becomes the anchor, and another window is created [10]. The bottom-up algorithm starts by dividing a series into several small segments. Then, the adjacent segments that create the smallest increase in error when together are merged. This process is repeated until either a specified number of segments is obtained, or a maximum approximation error for the complete series is met [6]. The top-down algorithm initially handles the complete series as a segment. If the segment produces an error value greater than a user-specified threshold, a cut that results in the most different segments is made, and the error value is calculated for both segments. This process is repeated until no cuts are made [10].

C. PAGE-HINKLEY CHANGE DETECTOR

The Page-Hinkley test is a change detector, an algorithm to detect concept drifts, for univariate time series. It continuously observes values from a time series and compares them to a moving average, storing the accumulated difference [11]. When the accumulated difference exceeds a predefined threshold value, an alert is raised, signalling a change in the series' behaviour. It has three main parameters: α, used as a forgetting factor for the accumulated difference; δ, used to disregard changes smaller than it, and *threshold*, representing the minimum accumulated difference needed for an alert.

D. AUGMENTED DICKEY-FULLER TEST

The Augmented Dickey-Fuller test is a unit root test that is primarily used to determine whether a time series is stationary or non-stationary [12]. The null hypothesis is that a unit root is present, meaning it is non-stationary. A statistic test is calculated and compared to critical values computed as specified in [15] to decide if the null hypothesis should be rejected. Then, if the p-value obtained is significant and the statistic test result is smaller than the corresponding critical value, the null hypothesis is rejected and the time series is stationary.

E. SYMBOLIC REGRESSION

Symbolic Regression is a form of regression model that creates algebraic expressions fitted to a given numerical dataset [16]. Values can then be passed through the formula to create predictions based on the values used for the regression. In the context of time series, this means finding a mathematical model that can accurately predict future values of a time series based on its past values. One of the most common methods to find the best expressions for the formula is by using GP to create a syntax tree that represents the formula [17]. The syntax trees are created in a process similar to natural selection, where trees with the worst performances are ignored and the best trees of the previous generation create the trees for the next. GP is both effective in terms of accuracy [18] and interpretability due to being able to produce smaller formulas that are easier for humans to understand [19].

III. RELATED WORK

There are several different segmentation algorithms in the literature. The majority of works create divisions by finding the least amount of linear regressions that need to be used to represent a time series such that every regression has an error below a user-specified threshold. These works can be mostly divided into those that use top-down algorithms, those that use bottom-up, and those that use sliding windows. Martí et al. [9] propose a top-down segmentation algorithm that applies linear regression on the time series and performs a statistical test to evaluate if the linear regression and the time series are statistically equal. If they are different, then a division is made at the point with the greatest residual error, and the whole process is repeated on the created sub-series. While achieving good results and reducing computation time, Martí et al. use linear regression to find the segmentation point instead of a change detector.

Li et al. [20] uses a bottom-up algorithm to create segments and then filters redundant segments using the R^2 statistic of linear regression, achieving either better performance or comparable performance using fewer segments. Once again, segmentation points are decided based on the error of linear regression instead of change detection.

Wee and Nayak [21] implement a sliding-window algorithm to reduce a time series to a simpler representation. Elements of the time series are added to the window until the window's mean is bigger than the next element, at which point a new window is created. Instead of updating the mean whenever adding a new element to the window, it is updated periodically according to a user-defined value to speed up the reduction process at the expense of accuracy. Although a good reduction is achieved and an algorithm similar to Page-Hinkley is used to find the segment position, comparing the next time series value to the window mean can lead to interference from values at the start of a big window. Keogh et al. [10] use a combination of sliding window and bottom-up algorithms to create some form of simplified representation by dividing the time series and creating simple representations of each segment.

Fillon and Bartoli [22] propose a new approach for symbolic regression based on GP called Hyper-Volume Error Separation (HVES). This approach consists of running a preliminary GP model and locating discontinuity boundaries by analysing the errors given by the model. These discontinuity boundaries are then used to partition the dataset into two different partitions, where the previous process is recursively applied until either a user-specified threshold is met or no boundaries are found. Tests made on synthetic time series

Algorithm 1 Finds Cut Positions for Given Series

```
1  Function find_splits(series) begin
       Data: series: Series to cut
       Result: XTSTree: Binary tree containing cut positions
2      $XTSTree \leftarrow Node_{root}$
3      $XTSTree.root \leftarrow find\_recursive\_splits(series, depth = 0)$
           /* recursively creates the tree and
           returns the root                              */
4      return XTSTree
5  end
```

showed an improvement in final formula accuracy and overall runtime.

Another less common class of segmentation algorithms is viewing segmentation as a classification problem. The segments of a time series can be divided by classifying stretches into different classes, resulting in a segmented time series. These algorithms often use neural networks, such as in [23], where a Temporal Convolution Network is used to create a latent representation of the time series, which is then sent to a clustering and classification layer. The segments are represented by the clusterization, which then are classified into the same or different classes.

Segmentation algorithms in general resort to some form of linear regression or error-based heuristic to find the best segmentation point, essentially employing a detector for a change in behaviour of the time series. These solutions may not be the best ones, since there are more robust change detector algorithms. Moreover, explainability is seldom approached in these studies, and although there are studies on explainability for time series algorithms and machine learning in general [13], [14], [24], these are more present in classification problems, and rarely seen in the segmentation field.

IV. PROPOSED APPROACH

XTSTree is an algorithm for top-down segmentation of time series that also aims to provide insight into the segmentation process. The time series' segmentation is recursive: first deciding where to split the data using a change detector, referred to as the splitting step; and then deciding when to stop splitting by using a stationarity-based stopping criterion. The cutting position is the point where a change was detected with the highest possible threshold value for the change detector, such that only the point with the most amount of change was detected. To stop the recursion, a stationarity test is applied to the series, and the recursion stops when the series is sufficiently stationary.

A binary tree is used to store the cutting points, which can then be retrieved in an ordered manner to create the segments. We chose a binary tree data structure because it treats each sub-series independently for the splitting step. The binary tree also provides explainability in a similar way to a decision tree. Due to how cuts are made, divisions made closer to the root of the tree signal a more drastic change in the behaviour of the time series, and divisions made closer to the leaves represent less drastic changes in behaviour that are still deemed relevant by the stopping criterion.

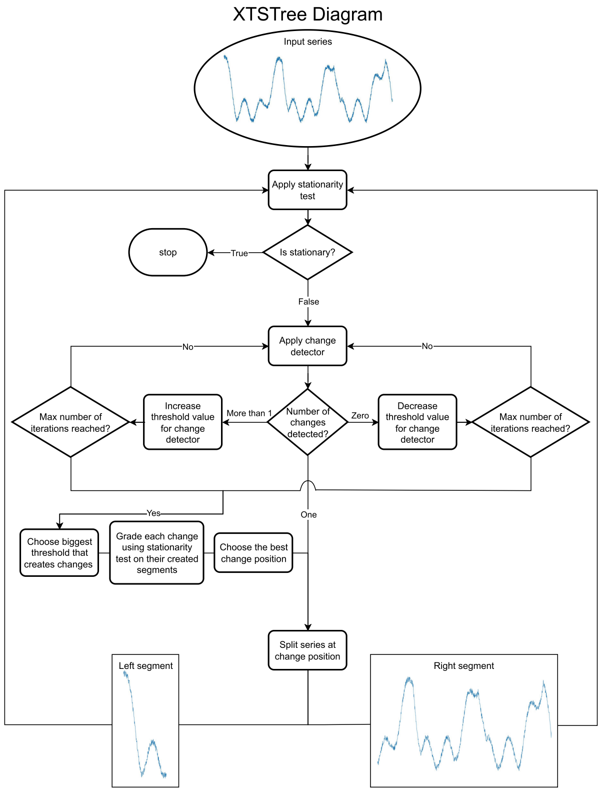

XTSTree Diagram
Input series
Apply stationarity test
Is stationary?
True
stop
False
Apply change detector
No
No
Max number of iterations reached?
Increase threshold value for change detector
More than 1
Number of changes detected?
Zero
Decrease threshold value for change detector
Max number of iterations reached?
Yes
One
Choose biggest threshold that creates changes
Grade each change using stationarity test on their created segments
Choose the best change position
Split series at change position
Left segment
Right segment

FIGURE 1. XTSTree's flowchart.

Algorithm 2 Recursive Function to Decide If Cuts Should Be Made

```
1  Function find_recursive_splits(series) begin
       Data: series: Series to cut
       Result: Node: Root node containing cut position and children
               nodes
2      if should stop cutting then return null ;
       /* find best cut according to cut method
          */
3      $cut\_position \leftarrow find\_cut(series)$
4      if $cut\_position = \varnothing$ then return $\varnothing$ ;
5      $node \leftarrow newNode(cut\_position)$
6      $node.right\_child \leftarrow find\_recursive\_splits(series[0 : cut\_position])$
7      $node.left\_child \leftarrow find\_recursive\_splits(series[cut\_position : series.length])$
8      return node
9  end
```

Algorithm 1, and 2 show the recursive segmentation process and creation of the binary tree. Algorithm 1 starts the recursive call of Algorithm 2. Algorithm 2 checks the stopping condition, and if it is not met, finds the best cutting point, stores it in a new node in the tree, and repeats the described process on the sub-series before and after the cutting point. Figure 1 shows a flowchart of the XTSTree segmentation process.

For this work, we implemented two stopping conditions to validate the results of our proposed tree. The first one uses a max depth for the tree, effectively creating a fully balanced binary tree as long as there are cuts to be found by the segmentation step. This acts as our baseline for the stopping condition since we can set multiple depths as stopping points for testing purposes. The other stopping condition we explored uses the stationarity detection method Augmented Dickey-Fuller (ADF) Test to determine if a series is stationary, meaning its values are dependent on the instant they are observed. We use a version of the ADF test that does not take the series' seasonality into account, meaning that if a series is said to be stationary, it either is indeed stationary, or it is non-stationary but the ADF test deems it as stationary due to a seasonal pattern. In our implementation, we assume that if a series is stationary before a cut and becomes non-stationary after a cut, it has a seasonal pattern that is divided by the splitting step. Therefore, we stop the segmentation process when a series is deemed non-stationary, and the score provided is the difference between the approximate p-value given by the test (computed as in [15]) and the p-value threshold provided on the XTSTree instantiation.

While the depth stopping condition serves as a fixed baseline, the ADF stopping condition aims to prevent unnecessary cuts by deeming that the series is non-stationary and consequently has a distinct enough behaviour from the rest of the series. Moreover, we show that the ADF stopping conditions can maintain a stable improvement in accuracy as the length of the series grows, making it scalable.

Algorithm 3 describes the ADF stopping condition. If the series has the minimum length required by the ADF test, the test values for the series are calculated. Then, the difference between the specified stop value and the calculated p-value is returned as the test score, and the segmentation stops if the score is positive.

We suggest an implementation for the *find_cut* method shown in Algorithm 2 that uses the Page-Hinkley change detector to find the point with the most significant change in the series' behaviour. To achieve this, the threshold value for the detector is chosen such that only one change is detected for the whole series. The method finds the best Page-Hinkley threshold value by performing a binary search, reducing the threshold value if no cuts are found on the series, and increasing it if more than one cut is found. The cutting process stops either after a threshold that produces only one cut is found, or after a specified number of iterations have passed. If the cutting process stopped due to exceeding the iteration limit, the threshold that created the smallest number of cuts is chosen as the best. The best candidate for the cut position is the one among cuts created by the best threshold that has the smallest combined score given by the stopping condition on the sub-series' created by the cut. If there is not a threshold that produces at least one cut, then the series is not cut at all. For control purposes, the periodic cut and random cut methods were also implemented for the *find_cut* method, where the series is cut in half and randomly, respectively.

Algorithm 3 Augmented Dickey-Fuller Test Stopping Condition Function

```
1 Function adf_stop_condition(series; stop_value) begin
      Data: series: Series to be checked
      Input: stop_value: Minimum critical value required to stop
             cuts
      Result: test_confidence: The more positive the value, the more
              confidence that the series is non-stationary
2     if series is too small then return 0 ;
3     $test\_confidence \leftarrow adfuller(series) - stop\_value$
4     return test_confidence
5 end
```

The three splitting methods address different biases. Periodic cuts produce predictable sub-series of the same size, whereas random cuts may create sub-series that are bigger or smaller than needed, and may even remove seasonality by chance. The Page-Hinkley method creates splits that balance the reduction of complexity with the need to provide explanations, since only the change detectors provide a meaningful hierarchy of cuts, thus contributing to the explainability as a XTSTree feature. It also refrains from creating more segments than needed, since it chooses the point with the most amount of change in behaviour for cutting.

Algorithm 4 describes the Page-Hinkley splitting method. The Page-Hinkley detector is executed over the full series until either it is completely analysed or more than one cut is found. If only one cut is found, the splitting step ends, otherwise, the binary search process starts. If no cuts are found, the threshold value decreases, and if more than one cut is found, the threshold increases. After the maximum number of iterations, the cuts created by the biggest threshold value are used to create pairs of sub-series for each cut, and the scores given by the stopping condition for each sub-series in each pair are added. The chosen cut is the one that creates the greatest total value.

A. XTSTREE EXAMPLE

To better illustrate the proposed solution, we present an application example for a synthetic time series that highlights the performance difference of SR before and after XTSTree is used. In this example, as well as throughout the rest of the paper, SR is leveraged as a powerful tool to enhance explainability. Symbolic regression provides a concise mathematical representation of a numerical sequence, thereby offering improved interpretability. This formal mathematical transcription not only gives a clear understanding of the underlying patterns but also facilitates a more comprehensive explanation of the time series data. The series used in the example, presented in Figure 2, is a combination of several sinusoidal-like series with added noise, as shown below, $f(x)$, as shown at the bottom of the next page.

The first step of our example is to apply Genetic Programming for SR over the whole series, without any segmentation, to extract a formula that can represent the

Algorithm 4 Page-Hinkley Segmentation Method

```
1  Function find_page_hinkley_cut(series; starting_threshold, max_number_iterations, delta) begin
       Data: series: Series to be cut
       Input: starting_threshold: Starting value for the page-hinkley threshold
       Input: max_number_iterations: Maximum number of iterations before deciding on the cut
       Input: delta: Delta parameter for the Page-Hinkley change detector
       Result: cut_position: Position that yields the best cut
2      $min\_threshold \leftarrow 0$
3      $max\_threshold \leftarrow -1$
4      $threshold \leftarrow starting\_threshold$
5      for iteration in max_number_iterations do
6          $pagehinkley \leftarrow newPageHinkley(threshold, delta)$
7          initiate cuts as empty lists
8          for element in series do
9              update pagehinkley with the element
10             if drift detected by pagehinkley then adds index of element to cuts ;
11         end
12         if cuts.length = 1 then return cut[0] index ;
13         if cuts.length > 1 then
14             $min\_threshold \leftarrow threshold$
15             if max_threshold is negative then
16                 $threshold \leftarrow threshold + threshold/2$
17             else
18                 $threshold \leftarrow (max_threshold - threshold)/2$
19             end
20         end
21         if cuts.length = 0 then
22             $max\_threshold \leftarrow threshold$
23             $threshold \leftarrow (threshold - min\_threshold)/2$
24         end
25     end
26     if no cuts found after max_number_iterations then return null ;
27     if cuts.length > 1 aftermax_number_iterations then
28         $pagehinkley \leftarrow newPageHinkley(min\_threshold, delta)$
29         initiate cuts as empty lists for element in series do
30             update pagehinkley with the element
31             if drift detected by pagehinkley then adds index of element to cuts ;
32         end
33         $final\_cut \leftarrow first\_cut\_in\_cuts$
34         $max\_stat \leftarrow stop\_function(series[0 : cut]) + stop\_function(series[cut : series.length])$
35         if stat > max_stat then
36             $max\_stat \leftarrow stat$
37             $final\_cut \leftarrow cut$
38         end
39     end
40     return final_cut
41 end
```

entire series. Figure 3 shows SR results along with the actual series. The presented model was the one that yielded the best accuracy. As can be seen, the formula found can barely describe the series' behaviour. The formula should ideally

$$f(x) = \begin{cases} 5 - 0.05x + \sin x/20 + \epsilon, & \text{if } x \leq 100 \\ \sin x/20 + \epsilon, & \text{if } x \geq 100 \text{ and } x \leq 300 \\ 0.03x - 10 + \sin x/20 + \epsilon, & \text{if } x \geq 300 \text{ and } x \leq 450 \\ 50 - 0.1x + \sin x/20 + \epsilon, & \text{if } x \geq 450 \text{ and } x \leq 500 \\ \sin x/20 + \epsilon, & \text{if } x \geq 500 \text{ and } x \leq 700 \\ 0.03x - 23.3 + \sin x/20 + \epsilon, & \text{if } x \geq 700 \text{ and } x \leq 850 \\ 90 - 0.1x + \sin x/20 + \epsilon, & \text{if } x \geq 850 \text{ and } x \leq 900 \\ \sin x/20 + \epsilon, & \text{if } x \geq 900 \text{ and } x \leq 1100 \\ 0.03x - 36.7 + \sin x/20 + \epsilon, & \text{if } x \geq 1100 \text{ and } x \leq 1250 \\ 130 - 0.1x + \sin x/20 + \epsilon, & \text{if } x \geq 1250 \text{ and } x \leq 1300 \end{cases}, \epsilon \in \{-0.2, 0.2\}$$

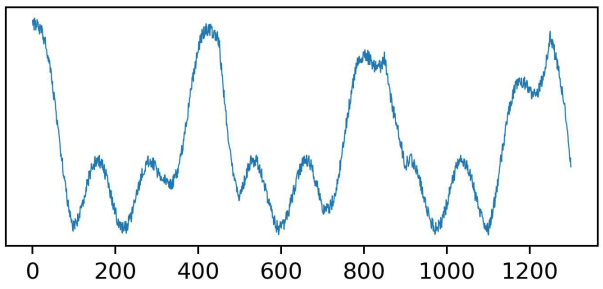


FIGURE 2. Time series used in the example.

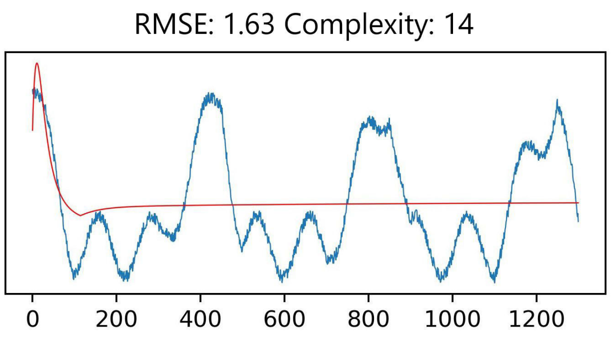


FIGURE 3. Symbolic Regression applied using accuracy for formula selection. RSME was computed for the formula against the complete series. Complexity is a combination of number of operators and how nested they are in the formula, as computed by PySR [25].

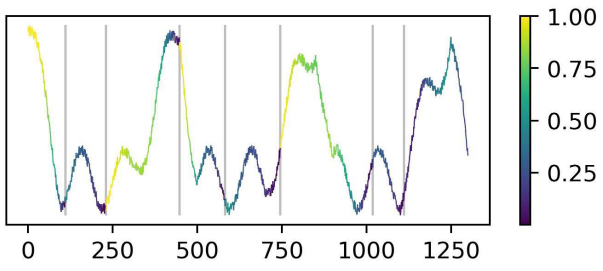


FIGURE 4. Time series used in the example after cuts performed by XTSTree. The colour scale ranges from 0 to 1.

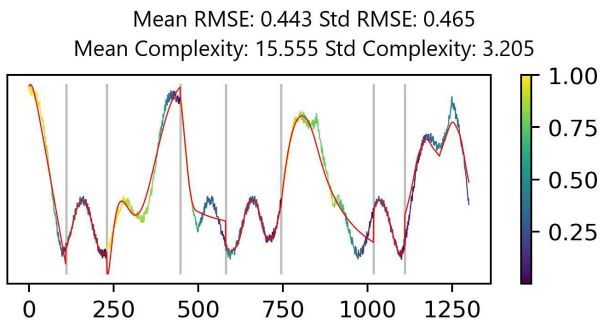


FIGURE 5. Symbolic Regression applied on the resulting leaves. The colour scale ranges from 0 to 1. RSME was computed for each formula against the corresponding sub-series. Complexity is a combination of the number of operators and how nested they are in each formula, as computed by PySR [25]. In both metrics, the mean and standard deviation were taken for each leaf.

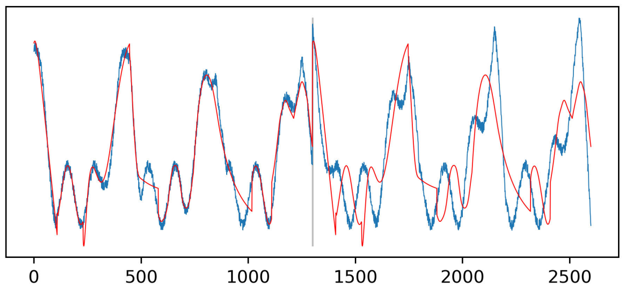


FIGURE 6. Formulas applied on the extended series used in the example.

reproduce not only the more prominent periodic behaviour seen every 400 steps, but also the smaller repetitions every 100 steps. Aiming to get better results from SR, we now employ XTSTree to cut the series, with ADF-test as the stopping condition and Page-Hinkley as the cut method. The outcome is presented in Figure 4.

XTSTree subdivided the original time series into 8 sub-series, which have a way simpler behaviour. The colours used in the series help us understand the reasoning behind the cuts. The darker the series' colour, the more it has deviated from the initial behaviour according to the Page-Hinkley change detector, and when it reaches the threshold, the series is cut. With the series now divided into stationary segments, SR can be applied to each of them. Figure 5 depicts these results.

Employing SR on the stationary segments yielded a set of formulas that describe the series significantly better than before, as can be seen in Figure 5. Moreover, if the initial series was actually part of a seasonal time series, such as in temperature or energy consumption over a week, the extracted formulas could be applied to future time steps, as seen in Figure 6. In this figure, the same pattern from the initial time series is repeated but using a different seed for random components and concatenating both series. Then, the formulas given by the leaf models were reapplied for the newly created series. As shown in Figure 6, the values obtained from the formulas are visually well-adjusted to the new series.

Figure 7 provides an overview of the generated XTSTree, illustrating the segmented series and associated formulas. The tree structure follows a binary tree formation with six depth levels. At the root level, the complete time series is depicted. Moving to the first layer of depth, we observe the split generated by the most divisive sample, resulting in two nodes without any leaves. This indicates a point where a change in the time series pattern occurs but does not present leaves capable of representing the expected level of simplification. Each split was designed to avoid complex formulas by merging segments with completely different behaviours.

In the second layer, we can observe three leaves and a node. Each leaf displays an optimised formula obtained through SR. The corresponding nodes represent large windows from the

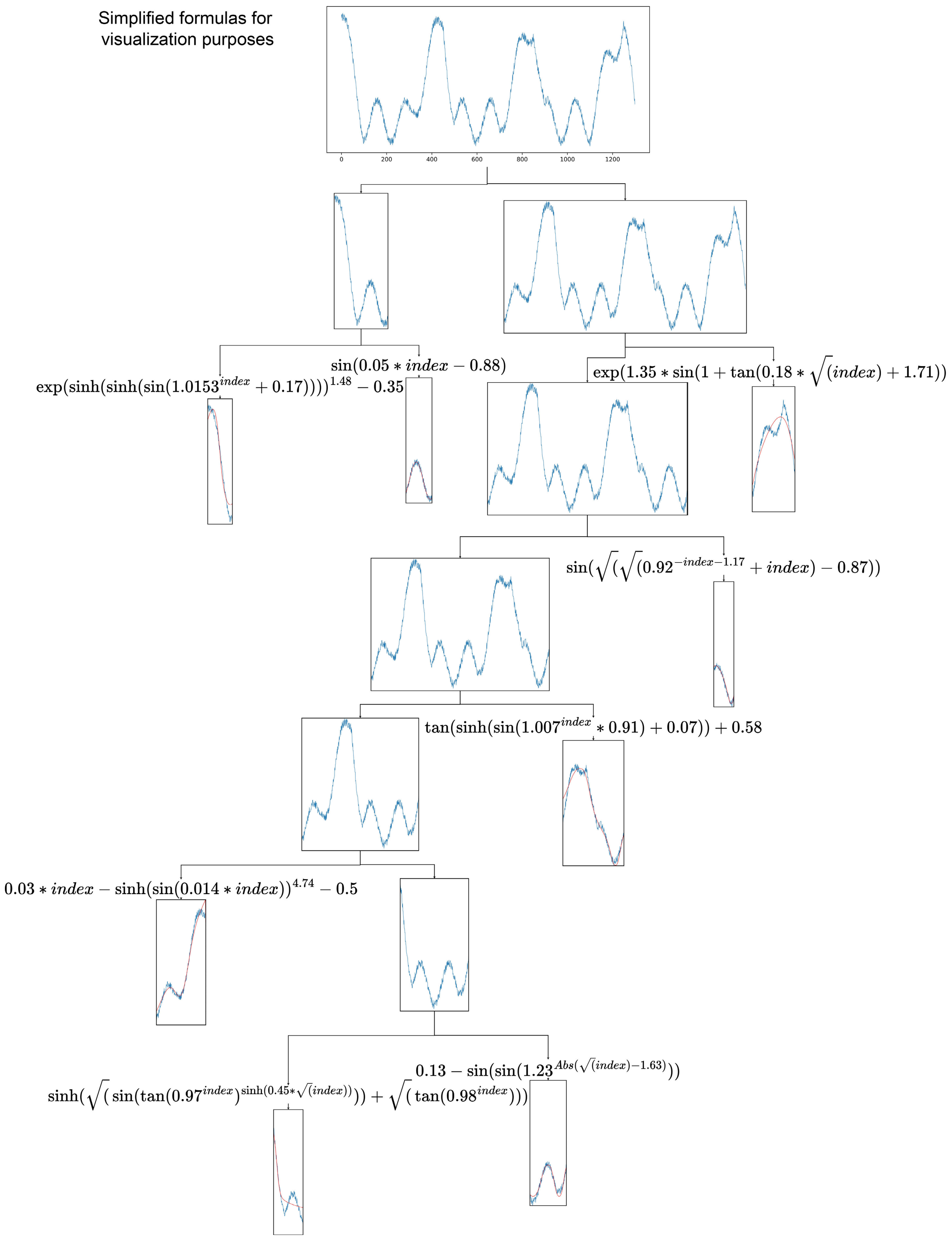

Simplified formulas for visualization purposes
0
200
400
600
800
1000
1200
$\sin(0.05 * index - 0.88)$
$\exp(\sinh(\sinh(\sin(1.0153^{index} + 0.17))))^{1.48} - 0.35$
$\exp(1.35 * \sin(1 + \tan(0.18 * \sqrt{(index)} + 1.71))$
$\sin(\sqrt{(\sqrt{(0.92^{-index-1.17}} + index) - 0.87))}$
$\tan(\sinh(\sin(1.007^{index} * 0.91) + 0.07)) + 0.58$
$0.03 * index - \sinh(\sin(0.014 * index))^{4.74} - 0.5$
$0.13 - \sin(\sin(1.23^{Abs(\sqrt{(index)}-1.63)}))$
$\sinh(\sqrt{(\sin(\tan(0.97^{index})^{\sinh(0.45*\sqrt{(index)})}))} + \sqrt{(\tan(0.98^{index}))})$

FIGURE 7. Representation of the XTSTree obtained from a synthetic time series using ADF and Page-Hinkley.

original time series, where it is not possible to produce the expected level of explanation according to the experimental setup. Layers 3 to 5 illustrate a sequence of multiple splits identifying small segments represented by simple formulas based on sine and hyperbolic sine functions. These formulas typically consist of less than five trigonometric operators.

As we delve deeper into the layers, more complex formulations become apparent, but they remain less complex than the complete formula. XTSTree effectively represents each segment as a simple formula. Moreover, the XTSTree's structure can be utilised to assess the time series' complexity, serving as a meta-feature in the observed domain.

V. EXPERIMENTAL STUDY

We conducted experimental studies on the capabilities of XTSTree from two perspectives: accuracy increase and simplification of expressions by reducing original complexity.

We carried out the experiments using real data that consists of humidity data collected by the Rural Development Institute of Paraná (IDR-PR), Brazil. Readings were collected by a sensor every fifteen minutes starting from January 2015 up to November 2021, totalling 96 samples per day. These readings were then split into segments of 5, 10, 15, and 20 days, resulting in 747 univariate time series of four different lengths (480, 960, 1440, and 1920 readings).

These time series were chosen because they fit the expected scenario for XTSTree: seasonal time series that change over time and demand explanations, preferably formal ones, about their behaviour. Table I briefly describes these datasets.

TABLE I. Meta-data of the time series used in the experiments.

Days	Length	# Time Series
5	480	358
10	960	179
15	1440	120
20	1920	90

Six different versions of XTSTree were tested by combining three splitting methods (Page-Hinkley, Periodic, and Random) and two stopping criteria (ADF test and depth).

To compare the quality of the six XTSTree versions and showcase their potential for explainability, we used the SR algorithm presented in [25]. First, we modelled the original time series with SR using the timestamp as a feature and the value as the prediction target. Then the series is segmented and the signals on the XTSTree leaves are modelled using the same parameters as before. In our experiments, we utilised the PySR [25] and Julia [26] implementations of SR. We selected both because they are high-performance platforms for implementing SR algorithms, which can be particularly useful for time series data.

The hyperparameters used for the Page-Hinkley splitting method were *delta*: 0.005, *alpha*: 0.999, max number of iterations: 100. The other splitting methods require no parameters. We used a stop value of 0.5 for the *ADF* stopping

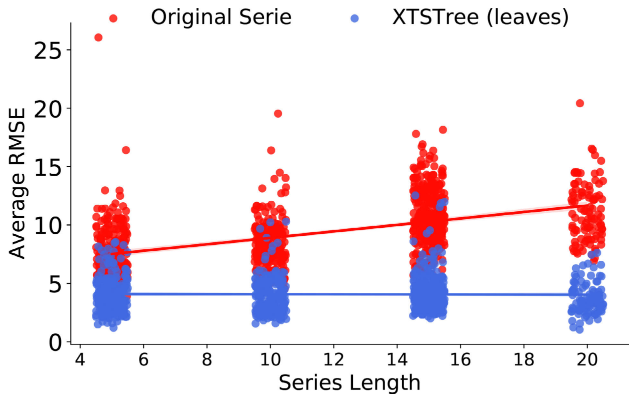


FIGURE 8. RMSE by series' length in days, Symbolic Regression on XTSTree leaves using ADF test as stopping condition. Red dots represent formulas fitted to the original series while blue dots represent average RMSE for formulas fitted on the XTSTree leaves.

criterion and a depth value of 3 for the *Depth* stopping criterion. Moreover, we used the following parameters for PySR: 20 populations; population size of 40; 10 iterations; model selection 'accuracy' and 'best'. The XTSTree implementation can be accessed in this public repository.[1]

A. ACCURACY IMPROVEMENT

The length of a time series is evidently an important factor when trying to create a model for it. As a time series grows in length, so does its complexity. However, identifying the proper segmentation point in a series to best improve its representation is a tricky task. To assess the XTSTree's capacity to improve model accuracy, we compared the error obtained for the whole signal (original series) and the mean of the models' error obtained from the signal of each leaf. We used Root Mean Square Error (RMSE) to measure the error, and it is defined by the formula:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{n}}.$$

where y_t is the value of Y at time t, $\hat{y}_t$ is the predicted value of Y at time t and n is Y's length.

Figure 8 shows the RMSE according to the series' length. The red and blue dots represent the RMSE for SR models applied on the whole series and for models applied on the XTSTree with ADF as the stopping criterion, respectively. The lines represent the average RMSE for both methods. The results show the contribution of the ADF test as the stopping criterion, improving the XTSTree's scalability. As presented in Fig. 8, the average RMSE for models applied using XTSTree remains constant regardless of the series length, whereas the ones that were applied on the whole series exhibit increased error as the series length increases.

To highlight the significance of ADF as a stopping criterion, we carried out further experiments using different stopping criteria. A fixed depth of three levels was also used

[1] github.com/BobVitorBob/XTSTree

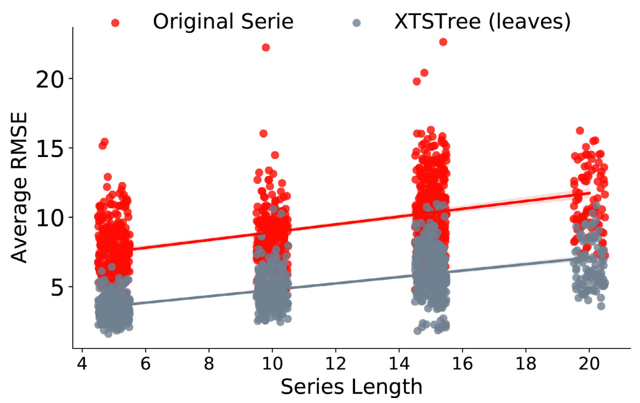


FIGURE 9. RMSE by series' length in days, Symbolic Regression on XTSTree leaves using a fixed depth of 3 as stopping condition. Red dots represent formulas fitted to the original series while grey dots represent average RMSE for formulas fitted on the XTSTree leaves.

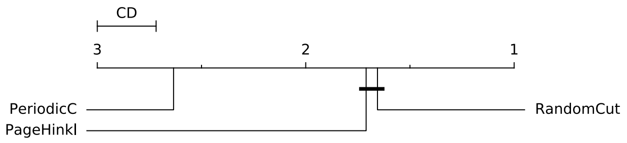


FIGURE 10. Critical Distance diagram based on the Nemenyi post hoc test using CD of 0.282 considering the RMSE of three splitting methods (Periodic, Random, and PH) with 140 paired time series.

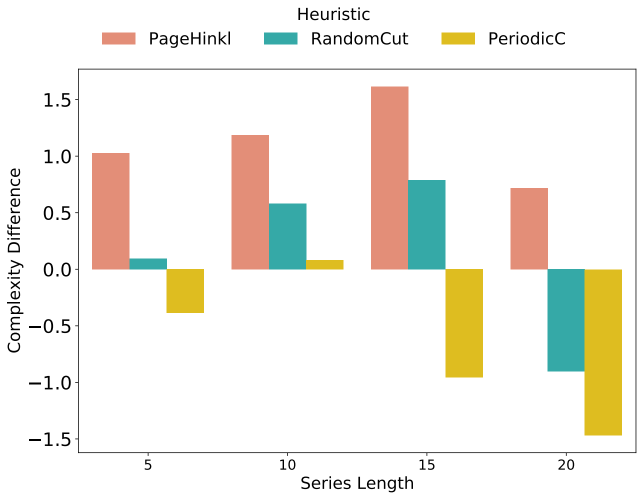


FIGURE 11. Average complexity reduction using three splitting methods (PH, Random, and Periodic) with ADF stopping criterion.

as the stopping criterion for XTSTree. Figure 9 presents the same graph as Figure 8, but with grey dots representing models relying on XTSTrees with depth as the stopping criterion. Although XTSTree obtained more accurate models in comparison to the original time series, both results show a similar error-increasing pattern proportional to the original signal length. This demonstrates the benefits of using the ADF test as the stopping criterion to improve the accuracy and scalability of XTSTree.

As mentioned before, we evaluated different splitting methods for XTSTree. To determine if there are any significant differences between using Periodic, Random, or Page-Hinkley as the splitting method, we conducted a statistical analysis based on the non-parametric Friedman test over 140 time series. We used the post hoc Nemenyi test to infer which differences are statistically significant. To find statistical superiority in terms of RMSE reduction, we compared the three splitting methods using ADF as the stopping criteria. Figure 10 shows the Nemenyi post hoc test results on the obtained error (RMSE), where we can observe that Random and PH obtained the most accurate models, with no significant differences between them, and the Periodic splitting methods performed the worst.

It is important to mention that the number of splits and the depth obtained from the Random splitting method were higher than Page-Hinkley's, which means that even though Page-Hinkley and Random are equal in terms of accuracy improvement, Page-Hinkley is better because it produces fewer leaves, which means fewer segments to model using SR.

B. COMPLEXITY REDUCTION ANALYSIS

We performed an analysis of complexity reduction based on default PySR hyperparameters for inducing SR. We define complexity reduction as the difference between the complexity of the formula, as calculated by [25], for the original series and the average complexity of the leaves' formulas. The XTSTrees used had all three different splitting methods and used ADF as the stopping criterion. Figure 11 shows a positive difference when using PH as a splitting method for all series lengths. It is important to mention that the highest reduction of complexity was observed for series with 15 days of signal acquisition. The random splitting method was able to deliver slight improvements with 5-day signals and remarkable results with 10 and 15-day. However, when processing 20 days, the random selection of a splitting point led to more complex equations. The periodic splitting method presented a slight improvement with 10 days, but for other signal lengths, it resulted in more complex representations than the whole signal.

To verify the statistical validity of the complexity reduction, we followed the same statistical test that was previously employed (Friedman and Nemenyi post hoc test). Figure 12 shows the Nemenyi post hoc test results on the obtained complexity analysis. In the figure, it is possible to observe that all results were statistically different, with PH being the most promising method for reducing complexity, followed by Random and Periodic, respectively

We considered that interpretability could also be evaluated in terms of the XTSTree number of splits, i.e., the number of recommended segmentations on a time series. Using the same experimental setup, we observed that the Periodic splitting method generates fewer segments driven by the hyperparameters employed. Conversely, Random as a splitting method generates more cuts without reaching the improvements in

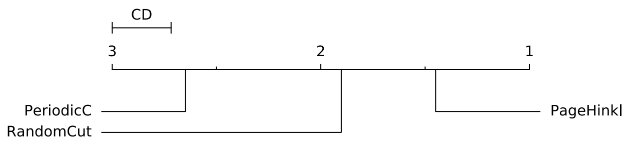


FIGURE 12. Critical Distance diagram using CD of 0.282 based on the Nemenyi post hoc test considering the difference of complexity between XTSTree and original series using three splitting methods (Periodic, Random, and PH) with 140 paired time series.

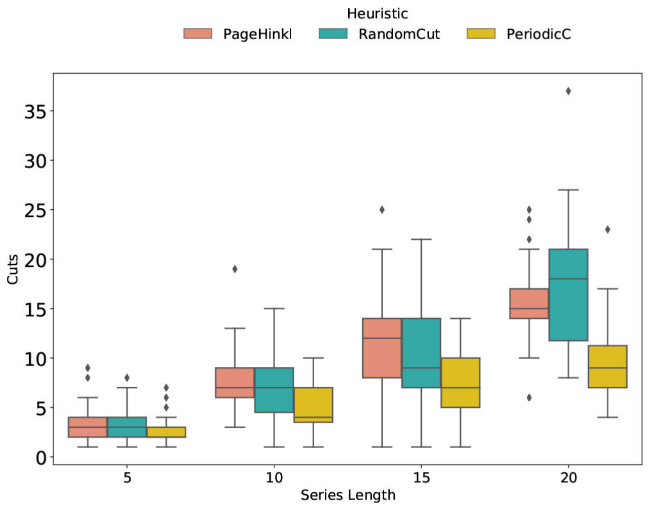


FIGURE 13. Number of segments (cuts) with various time series lengths for different splitting methods.

terms of the reduction of complexity provided by PH. Finally, the PH splitting method proved to deliver the best trade-off, improving the accuracy and reducing complexity when using XTSTree to process a time series. Figure 13 presents the relation between the number of segments (cuts) with various time series lengths with the different splitting methods.

C. COMPUTATION TIME COMPARISONS

To study how XTSTree affects computation time, we investigated several scenarios and observed that the impact is mainly related to the final size of the tree. The time cost of using XTSTree was negligible during our experiments when compared to the time spent on SR, hovering around less than 1 s. When using the max depth stopping criterion, there was a small increase in the time cost of modelling the original series. However, when growing a XTSTree using the ADF stopping criterion, deeper trees are created, more splits are processed, and more time is spent on modelling. Figure 14 shows the difference obtained using ADF and Max Depth of 3 levels. Experiments were performed on a computer with an Intel Core i5-1035G1 processor and 16 GB of RAM.

Figure 15 shows the Nemenyi post hoc test results on the obtained time. We investigated the influence of the different splitting methods. Due to the nature of the Periodic method, it was the fastest one to create the cuts, and since it creates fewer leaves, it was also the fastest to apply SR. The Random method provided deep trees, but the computational cost of this criterion is very low. Finally, PH provided average depth trees but is the slowest to create splits. The statistical evaluation

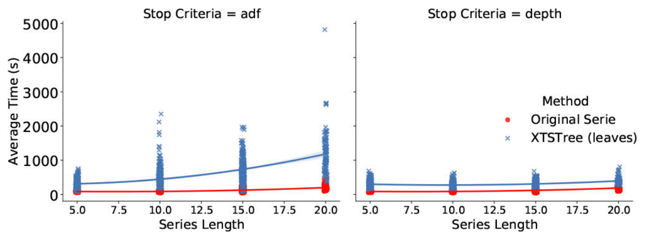


FIGURE 14. Average time consumption by length of series in days with ADF and Depth as stopping criterion.

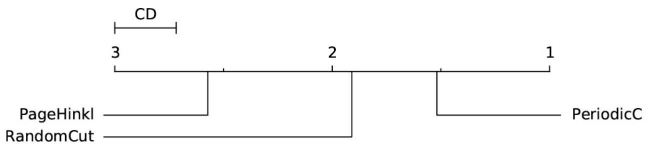


FIGURE 15. Critical Distance diagram using CD of 0.282 based on the Nemenyi post hoc test considering the time of three splitting methods (PeriodicC, RandomCut, and PageHinkl) with 140 paired time series.

based on the Friedman and Nemenyi post hoc tests shows that there are no statistically similar methods, with Periodic being the fastest to create splits, and Page-Hinkley the slowest.

VI. CONCLUSION

In this paper, we proposed an algorithm for top-down segmentation that leverages explainability in the form of the XTSTree. XTSTree combines a segmentation method based on change detectors with a stopping condition that uses a stationarity test to cut a time series into several sub-series with similar behaviour. The use of a change detector and a stationarity test allows XTSTree to approach the time series segmentation problem while leveraging the explainability of the cuts. We implemented two other segmentation methods, as well as a one-stop condition, for control purposes. Performance tests using SR were performed on several time series, and XTSTree managed to improve formula accuracy by a considerable amount. Among the several combinations of segmentation methods and stopping conditions, the ADF stopping condition proved to be the most adaptable to the series' length. The Page-Hinkley segmentation method also performed better than other methods, having a statistically lower error and a more consistent number of cuts and mean leaf error while achieving a level of explainability through the hierarchy of cuts.

In future work, other segmentation methods and stopping conditions can be implemented, and a deeper exploration of explainability can be done. There are also tree optimisations that can be done through the tree structure used to reduce the number of cuts and cluster sub-series created after segmentation.

REFERENCES

[1] T. V. Sushmitha, C. P. Deepika, R. Uppara, and R. N. Sai, "Vehicle trajectory prediction using non-linear input-output time series neural network," in *Proc. Int. Conf. Power Electron. Appl. Technol. Present Energy Scenario (PETPES)*, Aug. 2019, pp. 1–5.

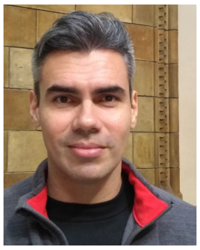
[2] P.-H. Chiang and S. Dey, "Personalized effect of health behavior on blood pressure: Machine learning based prediction and recommendation," in *Proc. IEEE 20th Int. Conf. E-Health Netw., Appl. Services (Healthcom)*, Sep. 2018, pp. 1–6.

[3] P. Ni, C. Zhang, and Y. Ji, "A hybrid method for short-term sensor data forecasting in Internet of Things," in *Proc. 11th Int. Conf. Fuzzy Syst. Knowl. Discovery (FSKD)*, Aug. 2014, pp. 369–373.
[4] K. Doki, K. Hashimoto, S. Doki, S. Okuma, and T. Ohtsuka, "Estimation of next behavior and its timing based on human behavior model with time series signal," in *Proc. Comput. Intell. Control Autom. (CICA)*, Apr. 2011, pp. 102–107.
[5] R. Anderson, Y. S. Koh, G. Dobbie, and A. Bifet, "Recurring concept meta-learning for evolving data streams," *Expert Syst. Appl.*, vol. 138, Dec. 2019, Art. no. 112832. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417419305342
[6] M. Lovric, M. Milanovic, and M. Stamenkovic, "Algorithmic methods for segmentation of time series: An overview," *J. Contemp. Econ. Bus. Issues*, vol. 1, pp. 31–53, Jan. 2014.
[7] C. Wang and X. S. Wang, "Supporting content-based searches on time series via approximation," in *Proc. 12th Int. Conf. Sci. Statistica Database Manage.*, 2000, pp. 69–81.
[8] J. Hunter and N. Mcintosh, "Knowledge-based event detection in complex time series data," in *Proc. Joint Eur. Conf. Artif. Intell. Med. Med. Decis. Making*, vol. 1620. Aalborg, Denmark, Jun. 1999, pp. 271–280.
[9] L. Marti, N. Sanchez-Pi, J. Molina, and A. C. Garcia, "YASA: Yet another time series segmentation algorithm for anomaly detection in big data problems," in *Proc. 9th Int. Conf. Hybrid Artif. Intell. Syst.*, 2014, pp. 697–708.
[10] E. Keogh, S. Chu, and M. Pazzani, "Segmenting time series: A survey and novel approach," in *Data Mining in Time Series Databases*, vol. 57. World Scientific Publishing Company, Mar. 2003.
[11] R. Sebastiao and J. M. Fernandes, "Supporting the page-Hinkley test with empirical mode decomposition for change detection," in *Proc. Int. Symp. Methodologies Intell. Syst.*, 2017, pp. 492–498.
[12] D. A. Dickey and W. A. Fuller, "Distribution of the estimators for autoregressive time series with a unit root," *J. Amer. Stat. Assoc.*, vol. 74, no. 366, p. 427, Jun. 1979.
[13] M. Nauta, J. Trienes, S. Pathak, E. Nguyen, M. Peters, Y. Schmitt, J. Schlötterer, M. van Keulen, and C. Seifert, "From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable AI," 2022, *arXiv:2201.08164*.
[14] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *Proc. 5th Int. Conf. Data Sci. Adv. Anal.*, 2019, pp. 80–89.
[15] J. Mackinnon, "Critical values for cointegration tests," in *Long-Run Economic Relationships*. USA: Oxford University Press, Feb. 1990.
[16] Y. Jin, W. Fu, J. Kang, J. Guo, and J. Guo, "Bayesian symbolic regression," 2019, *arXiv:1910.08892*.
[17] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, vol. 1. Cambridge, MA, USA: MIT Press, 1992.
[18] W. La Cava, P. Orzechowski, B. Burlacu, F. O. de França, M. Virgolin, Y. Jin, M. Kommenda, and J. H. Moore, "Contemporary symbolic regression methods and their relative performance," 2021, *arXiv:2107.14351*.
[19] M. Virgolin, A. De Lorenzo, F. Randone, E. Medvet, and M. Wahde, "Model learning with personalized interpretability estimation (ML-PIE)," in *Proc. Genetic Evol. Comput. Conf. Companion*, New York, NY, USA, Jul. 2021, pp. 1355–1364, doi: 10.1145/3449726.3463166.
[20] G. Li, J. Wang, X. Jia, and Z. Yang, "A new piecewise linear representation method based on the R-squared statistic," in *Proc. 3rd Int. Conf. Mach. Learn., Big Data Bus. Intell. (MLBDBI)*, 2021, pp. 515–519.
[21] C. K. Wee and R. Nayak, "Alternate approach to time series reduction," in *Proc. Int. Conf. Soft-computing Netw. Secur. (ICSNS)*, Feb. 2018, pp. 1–4.
[22] C. Fillon and A. Bartoli, "Symbolic regression of discontinuous and multivariate functions by hyper-volume error separation (HVES)," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 23–30.
[23] H. Min and J.-G. Lee, "Temporal convolutional network-based time-series segmentation," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2023, pp. 269–276.
[24] T. Rojat, R. Puget, D. Filliat, J. D. Ser, R. Gelin, and N. Díaz-Rodríguez, "Explainable artificial intelligence (XAI) on TimeSeries data: A survey," 2021, *arXiv:2104.00950*.
[25] M. Cranmer. (May 2023). *Interpretable Machine Learning for Science With PySR & SymbolicRegression.jl*. [Online]. Available: https://github.com/MilesCranmer/pysr_paper
[26] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Rev.*, vol. 59, no. 1, pp. 65–98, Jan. 2017.

VITOR DE CASTRO SILVA received the B.Sc. degree in computer science from the State University of Londrina, Brazil, in 2021, where he is currently pursuing the M.Sc. degree in computer science. His research interests include machine learning and data science, with current emphasis in time series.

BRUNO BOGAZ ZARPELÃO received the B.Sc. degree in computer science from the State University of Londrina (UEL), Brazil, and the Ph.D. degree in electrical engineering from the University of Campinas, Brazil. He is currently an Associate Professor with the Computer Science Department, UEL, where he joined in 2012. From March 2018 to February 2019, he was a Visiting Postdoctoral Researcher with the City, University of London. His research interests include data science in cybersecurity, intrusion detection, and the Internet of Things.

ERIC MEDVET received the degree (cum laude) in electronic engineering and the Ph.D. degree in computer engineering from the University of Trieste, Italy, in 2004 and 2008, respectively. He is currently an Associate Professor of computer engineering with the University of Trieste, the Director of the Evolutionary Robotics and Artificial Life Laboratory, and the Co-Director of the Machine Learning Laboratory. His research interests include evolutionary robotics, artificial life, evolutionary computation, and applications of machine learning.



SYLVIO BARBON JR. received the B.Sc. degree in computer science and the M.Sc. degree in computational physics from the University of São Paulo, in 2005 and 2007, respectively, and the degree in computational engineering and the Ph.D. degree in computational physics from IFSC/USP, in 2008 and 2011, respectively. He is an Associate Professor with the Department of Engineering and Architecture, University of Trieste (UNITS), Italy. He is also the Co-Director of the Machine Learning Laboratory. Prior to this, he led a research group dedicated to the study of machine learning with the Computer Science Department, State University of Londrina (UEL), Brazil, from 2012 to 2021. His research interests include computer vision, pattern recognition, and machine learning, with a current emphasis on meta-learning, stream mining, and process mining.

Open Access funding provided by 'Università degli Studi di Trieste' within the CRUI CARE Agreement