VNU Journal of Science: Comp. Science & Com. Eng., Vol. 41, No. 1 (2025) 30–53

VNU Journal of Science:
Computer Science and Communication Engineering

Journal homepage: http://www.jcsce.vnu.edu.vn/index.php/jcsce

Original Article

Local Explanation Methods for Time Series Classification Models*

Viet Nguyen[1,2], Phuc Vu[1,2], Thanh Le[1,2], Bac Le[1,2*]

[1] *Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam*
[2] *Vietnam National University, Ho Chi Minh City, Vietnam*

Received 06 October 2024
Revised 13 November 2024; Accepted 12 March 2025

Abstract: This paper focuses on researching and proposing perturbation-based model-agnostic methods to explain time series classification models. The main objective of this study is to explain the predictions of the model, or in other words, to give reasons why it classifies a time series into a particular label in a set of labels. In this work, we aim to provide the reliability of the decision and the importance of features in the model. Moreover, in real-world time series, variations in the speed or scale of a particular action can determine the class, so modifying this type of feature leads to arbitrary explanations of the time series. To achieve the set objectives, we provide two methods, each with its own strategies and advantages: the LIME-based method and the SHAP method, with the novelty of using them in combination with data perturbation techniques, especially the ones that affect the above-mentioned characteristics of the time series.

Keywords: Time series classification, model-agnostic explanation, time series transformation, robustness, shapley value.

1. Introduction

Nowadays, machine learning is increasingly being applied in many fields, including research problems and solving practical problems, and time series data analysis is no exception. This is an important issue with many applications in finance, medicine, engineering, etc. In time series data analysis, Time Series Classification (TSC) is a popular problem, intending to predict the label for a given time series. For example, TSC can be used to suggest decisions about buying or selling stocks, diagnose heart disease, or classify traffic information. However, the increasing complexity of machine learning models, especially TSC models, leads to a decrease in transparency in the model decision process. This makes it difficult to learn the expected underlying model of the image,

*Corresponding author.
E-mail address: lhbac@fit.hcmus.edu.vn

https://doi.org/10.25073/2588-1086/vnucsce.3765

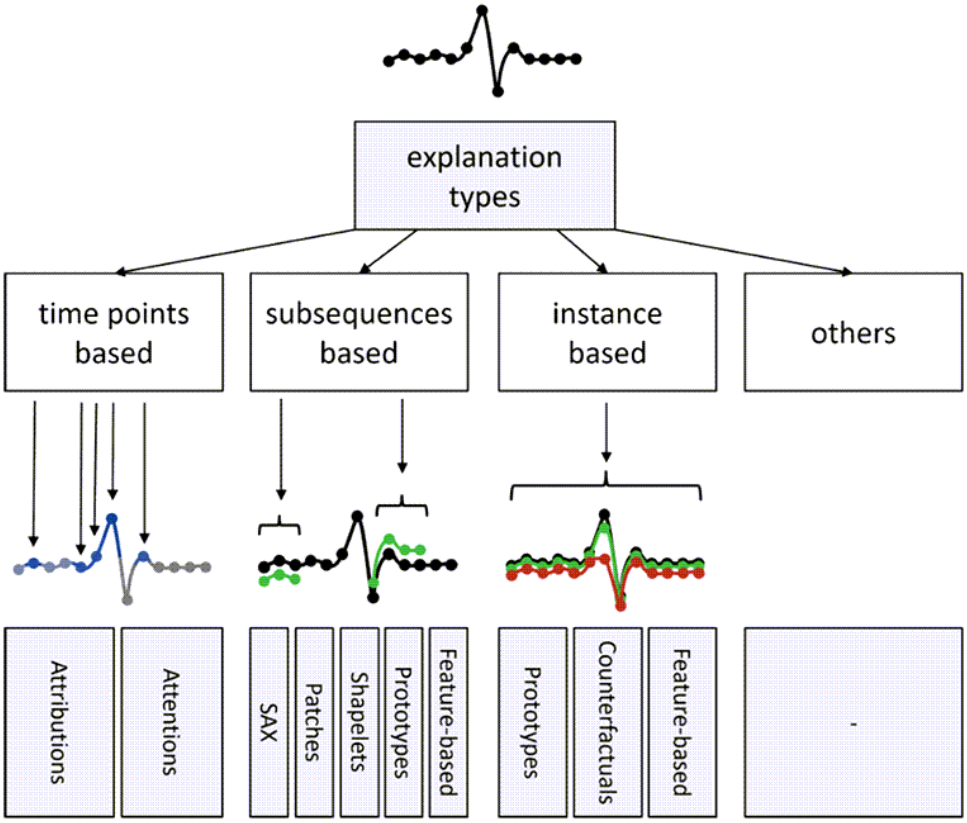


Figure 1. Classification of TSC Explanation Methods [1].

limiting the creativity of the idea and the application of the model in practice.

Explainable Artificial Intelligence (XAI) is an emerging research field that is being vigorously developed to try to solve the problem of transparency lacking in the above-mentioned machine learning models. More specifically, for the problems related to time series data, explaining the model's decisions is not simple, because the time series data often have special correlations with each other, creating a more complex structure to have specific independent data. Therefore, explaining the decisions of the TSC model becomes more difficult, especially in terms of semantic explanation.

TSC explanation methods are often classified based on the level of explanation, depending on the part of the time series used to illustrate the reasons for the model's decisions. In other words, the classification of TSC explanation methods is based on detecting anomalies in the time series. Accordingly, there are three groups of explanation methods based on the detection of these anomalies (Figure 1). Time point-based explanations refer to specific time points in the time series, subsequences-based explanations refer to subsequences of the time series, and instance-based explanations apply the entire time series as an explanation. Explanations are classified as "Other" if they cannot be tied to any of the previous explanation types. Furthermore, for each category of XAI methods for TS, we can further distinguish the algorithmic strategies applied by the considered methods to provide

explanations. In addition, explanation methods can also be classified based on their scope, which can be local explanations or global explanations. The following are existing research works on local TSC explanation methods that have been studied and listed in Table 1.

Among the mentioned methods, the family of time point-based TSC explanation methods provides a very powerful tool in explaining why a TSC model makes a certain prediction by analyzing the regions where time points have a large influence on the model's decision in the input time series, especially the Attributions explanation type. Therefore, we mainly focus on researching and proposing TSC explanation methods belonging to this category, especially LIME for TS [2] and related methods, including SHAP [3] (an improved version of LIME).

The basic LIME method provides explanations by locally approximating the classifier with an interpretable model. To achieve this, a synthetic neighborhood is generated around a given instance by perturbing it, and then, the explanation is obtained by learning an interpretable model using the neighbors (see [4] for more details). In the popular LIME application method for time series data - LEFTIST [5], time series perturbations to generate neighborhoods are identified through transformations applied to one or more intervals (subsequences) of the time series, rather than to isolated points. These transformations include replacing the given interval(s) with another interval, such as a random noise interval, linear interpolation between the first and last points of the interval, or intervals extracted from other series in the training set. However, the main drawback of these methods is that the neighborhoods generated with the transformations are not realistic. For example, if a time series represents the electricity consumption of a country (as in the ItalyPowerDemand dataset from the UCR archive [6]), replacing a certain time interval of the series with random noise does not generate a realistic neighborhood. In particular, the generated neighborhood is semantically meaningless, since a random noise interval cannot be interpreted from the perspective of electricity consumption.

Our proposal with the LIME-based method aims to provide a local model-independent explanation for TSC with the novelty of creating more realistic neighbors for the time series. In this work, we consider four transformations based on [7–9]: warping, scaling, noise, and slicing. For these transformations, the explanations provided by our method have one interpretation: an interval is important because if a certain transformation is applied to this interval, the prediction will change. In addition, we also propose a further improved method using Shapley value combined with LIME to specifically emphasize the influence of each part of the time series on the model's decision, showing whether that influence supports or opposes the model to classify the series into a specific class.

A primary contribution of this work is the elucidation of not only the relevant regions of a given time series, as TSC explanation methods do, but also the specific causal impact of these regions on the classifier's decision. Our methods reveal that a particular region is significant because applying a specific transformation to it is likely to alter the classifier's output, assigning the transformed time series to a different class. This information can be highly valuable in various real-world applications.

The rest of the work is organized as follows: in Section 2, the related work for the TSC explanation is introduced, including the proposed transformations for time series. Two explanation methods are thoroughly described in Section 3, while the experimentation is presented in Section 4. Finally, the main conclusions are drawn in Section 5.

Table 1. Some existing local explanations methods for TSC and their characteristics

Name	Explanation Type	Explanation Method	Model-dependence	Scope
MTEX-CNN [10]	Time points	Attributions	Intrinsic (DNN)	Local
CAM [11]	Time points	Attributions	Intrinsic (DNN)	Local
LEFTIST [5]	Time points	Attributions	Agnostic	Local
TimeXplain [3]	Time points	Attributions	Agnostic	Local
LIME TS [2]	Time points	Attributions	Agnostic	Local
Dynamic Masks [12]	Time points	Attentions	Intrinsic (DNN)	Local
SAX-VSM [13]	Subsequences	SAX	Intrinsic	Local
AI-PR-CNN [14]	Subsequences	Shapelets	Intrinsic (DNN)	Local
LASTS [15]	Subsequences	Shapelets	Agnostic	Local
CEM [16]	Instance	Counter-factuals	Intrinsic (DNN)	Local
Tweaking RSF [17]	Instance	Counter-factuals	Intrinsic	Local
Native-Guides [18]	Instance	Counter-factuals	Intrinsic (K-NN)	Local

2. Related Work

2.1. LIME Method

LIME (Local Interpretable Model-Agnostic Explanations) is a popular model explanation method used to explain the predictions of machine learning models, especially classification ones. LIME works by creating a simple local explanatory model around the data point of interest, making it easy for users to understand the reasoning behind the model's predictions. This method approximates the original complex model into a linear model based on the neighborhood data around the data point to be explained, in order to determine the role of each important feature in that point.

There are many approaches to applying LIME to time series data, but any proposed method must address the following condition: *Firstly*, a time series must be represented as a vector according to Definition 1. *Secondly*, there must be suitable ways to generate perturbed sequences in the neighborhood of x. *Finally*, the neighborhood of x must be determined and a method for calculating the distances from neighboring sequences to x must be proposed.

Theorem 1. *A time series* $x = \{t_1, t_2, ..., t_m\} \in \mathbb{R}^{m \times d}$ *is an ordered set of* m *real-valued observations (or time steps), with* d *dimension.*

2.2. Time Series Transformations

The purpose of using time series transformations is to provide tools for generating perturbed time series (or "neighborhoods") to serve as training grounds for explanatory models. Time series can be analyzed in both the time domain and the frequency domain, so transformations can be defined in both of these spaces. However, in TSC and more specifically in the explainability of TSC, working in the time domain is much more common. Indeed, there are very few problems in constructing explanations for TSC where the series is represented in the frequency domain, one of which is Mujkanovic's work on the TimeXplain method [3]. The main reason is that the transformations in the time domain can be applied to specific time intervals more intuitively and interpretably. Therefore, the transformations proposed in this paper will be defined in the time domain.

There are many methods for performing time-domain series transformations, for example

bootstrapping [19–21], or Generative Adversarial Network (GAN) [22]. The main goal of these methods is to generate the time series that are similar to a given time series (with the same probability distribution). These methods do not allow us to fully control the characteristics of the generated time series, which means that we cannot directly specify the exact properties of the perturbations we want to generate. Thus, if these transformations are applied, the cost of initializing suitable neighboring series may increase due to "trial and error". To provide a more efficient approach, time series neighborhoods will be generated using a set of locally controllable transformations of the series features, proposed in the work of A. Abanda [23], including: warp, scale, noise, and slice.

2.2.1. Warp Transformation

This transformation is a distortion of a time series along the time axis (x-axis), resulting in a compression or expansion (depending on the warp level) of the series' values within a given interval. Time series with time intervals warped at different warp levels appear frequently in real-world problems, for example, in the GunPoint dataset (from UCR repository [6]) context. Figure 2 shows two time series from this dataset, one from each class. The interval in the [40, 60] interval of the Gun class can be considered a warped version of that interval in the Point class (or vice versa). In particular, in the Gun class time series, the value t_i increases more rapidly (in the interval $[t_{40}, t_{60}]$) than in the Point class time series. In this case, this interval is considered as a distinguishing mark for the warp transform, because if we compress/expand it, the classifier may change its class prediction.

To synthetically create warped versions of the reference time series $T = (t_1, ..., t_i, ..., t_l)$, given an interval $[s, e]$ where the Warp transformation is applied (so that $0 < s < e < l$) and the degree of warping k_w, the warped version of T is determined by the vector $T' = (t'_1, ..., t'_i, ..., t'_{s+(e-s)k_w+(l-e)})$, with:

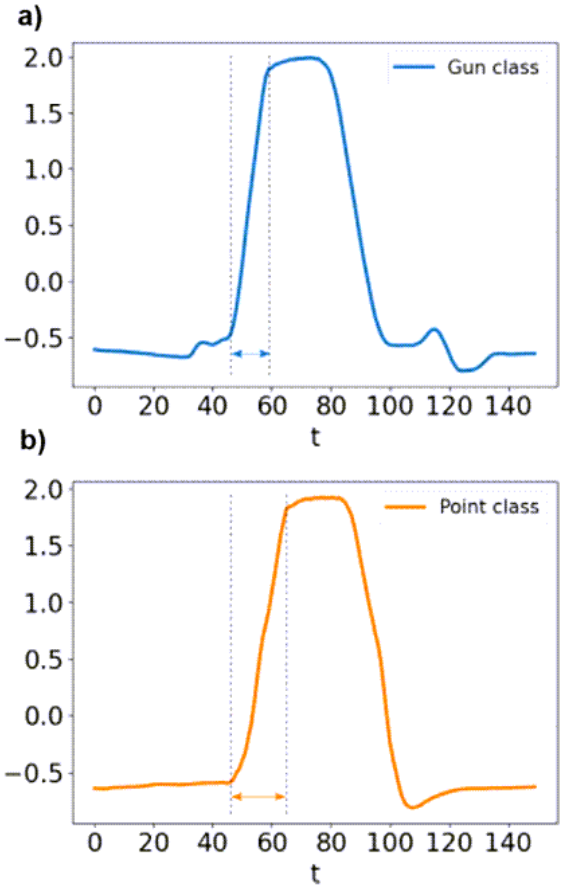


Figure 2. Two time series labeled Gun (Figure a) and Point (Figure b) in the GunPoint dataset [23].

$$t'_i = \begin{cases} t_i, & i \le s \\ (w_1 t_p + w_2 t_q)/k_w, & s \le i \le s + k_w(e-s) \\ t_{e+i-(s+k_w(e-s))}, & i \ge s + k_w(e-s) \end{cases} \quad (1)$$

where $t_p = max_{j=1,...,l}[jk_w] < i$ and $t_q = min_{j=1,...,l}[jk_w] \ge i$. The [ ] notation refers to the nearest integer and the weights are determined by $w_1 = k_w - (i - p)$ and $w_2 = k_w - (q - i)$.

Note that this transformation performs a compression of ($k_w < 1$) or a stretch of ($k_w > 1$) of the reference time series, so the transformed

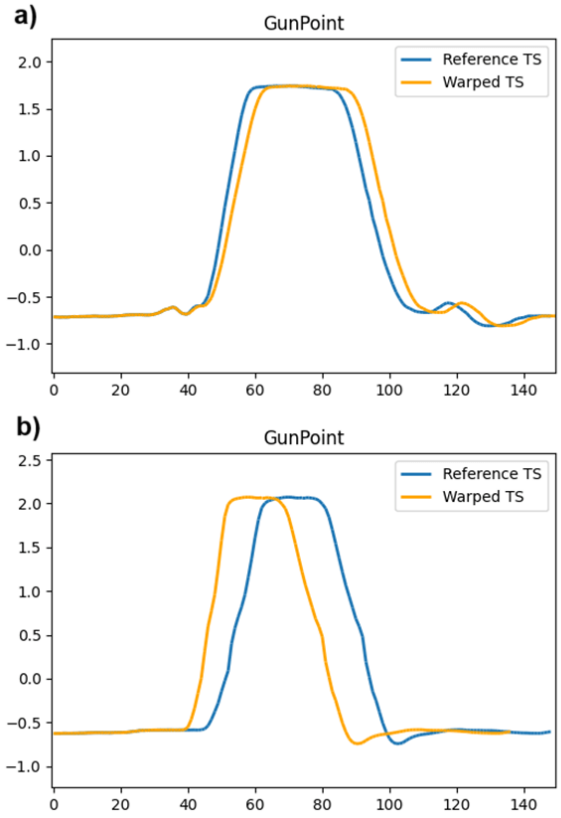


Figure 3. Two examples of applying the Warp transform to time series in the GunPoint dataset.

series will be shorter or larger than the reference series. Figure 3 shows two examples of two time series in the GunPoint dataset that are subjected to a warp transformation with $k_w < 1$ (for the series labeled Gun) and $k_w > 1$ (for the series labeled Point).

2.2.2. Scale Transformation

This transformation is a deformation of the y-axis series. It produces a shift up or down (depending on the scale) of the series values over a given interval. The idea behind applying this transformation is that for time series that are similar in shape, changing the y-axis values by a given scale can change their labels, as in the Coffee dataset from the UCR archive [6]. In this dataset, the change in spectral values of two coffees over a period of time determines their labels. The time series representing the spectral values of these two coffees have very similar shapes but differ in magnitude over some intervals. Thus, the Scale transformation over those intervals is considered to have an impact on the prediction (see the example in Figure 4). The proposed scaling transformation of the reference time series $T = (_t1, ..., t_i, ..., t_l)$ is as follows: given an interval over which the transformation is applied [s,e] (such that $0 < s < e < l$) and a scaling level k_s, the transformed version of the series is: $T' = (t'_1, ..., t'_i, ..., t'_l)$ with:

$$t'_i = \begin{cases} t_i k_s, & s \leq i \leq e \\ t_i, & i < s \text{ or } i > e \end{cases} \quad (2)$$

2.2.3. Noise Transformation

Noise transformation involves adding noise to a sequence in a given interval. Similar to the Scale transformation, for the Coffee dataset, a noise transformation in an interval can cause the model to predict a sequence into a different class and thus, the noise transformation in a given interval is considered to influence the classification decision of the TSC model (e.g. Figure 5). Given a reference time series $T = (t_1, ..., t_i, ..., t_l)$, an interval in which the Noise transformation is applied $[s, e]$ (such that $0 < s < e < l$) with noise level kn, the noise-transformed version $T' = (t'_1, ..., t'_i, ..., t'_l)$ of sequence T is defined by:

$$t'_i = \begin{cases} t_i + \mathcal{N}(0, \frac{A \times k_n}{100}), & s \leq i \leq e \\ t_i, & i < s \text{ or } i > e \end{cases} \quad (3)$$

with $A = |max(T) - min(T)|$ being the amplitude of the series. As the formula 3 shows, the added noise is Gaussian noise $\mathcal{N}(\mu, \sigma)$, with $\mu = 0$ and σ depending on the amplitude of the time series and the noise level k_n. In this way, for example, with $k_n = 5$, the standard deviation is set to 5% of the amplitude of the series.

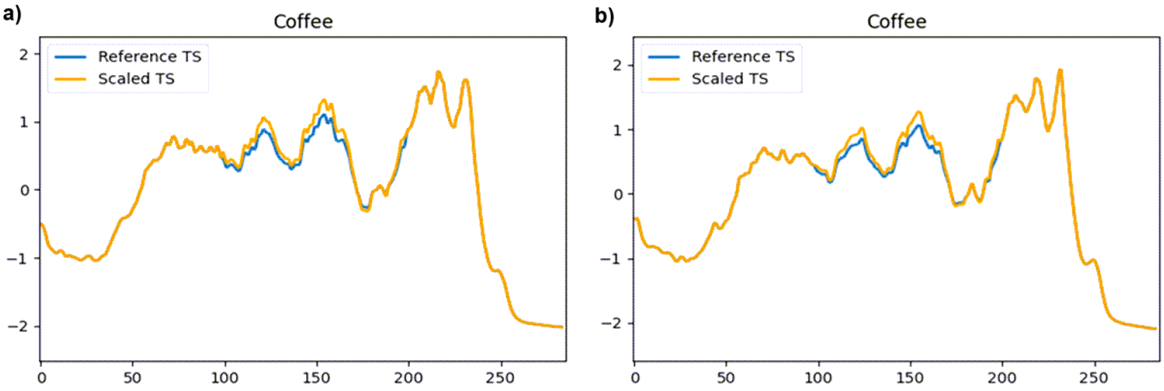


Figure 4. Example of applying Scale transformation to time series of Arabica (Figure a) and Robusta (Figure b) classes in Coffee dataset.

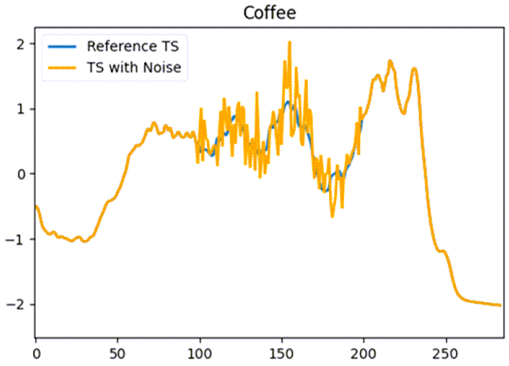


Figure 5. An example of applying the Noise transform to the Coffee dataset.

2.2.4. *Slice Transformation*

The slice transformation selects a sub-series of the time series and shifts it toward the beginning of the reference time series. Using again 2 samples in the GunPoint dataset, it can be seen that both time series have very similar shapes, but the time series from the Gun class can be considered a truncated version of the time series from the Point class. That is, if the recording of the time series from the Point class had started a little later, the classifier could have classified it as a Gun class (see Figure 6). In this way, the chosen interval for truncation and in particular its position in the reference time series are considered to influence the prediction.

Given a reference time series $T = (t_1, ..., t_i, ..., t_l)$, the truncated version of T by an interval $[s, e]$ (with $0 \leq s < e \leq l$) is $T' = (t'_1, ..., t'_i, ..., t'_{e-s})$, with:

$$t'_i = t_{s+i}, i = 1, ..., e - s \tag{4}$$

In summary, these transformations can be applied to any time series from any type of TSC problem, but it is particularly interesting for problems where the transformations are considered meaningful and semantically interpretable.

2.3. *SHAP Values*

SHAP Values provide a consistent, objective way to explain how each feature influences a model's predictions, based on game theory. Positive SHAP values indicate a positive impact on prediction, while negative values indicate a negative impact. The value's magnitude reflects its level of influence.

SHAP Values Fomula

The formula for calculating SHAP values for a feature i in a machine learning model. This is a mathematical and fair way to measure the contribution of each feature to the model's predictions.

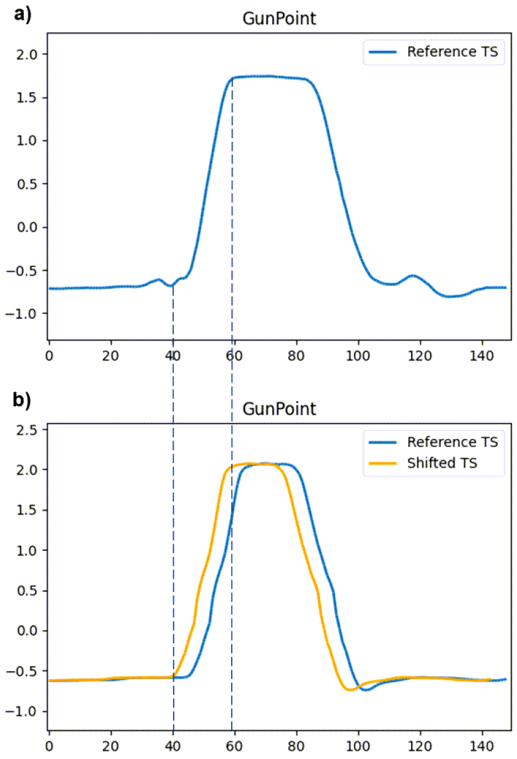


Figure 6. An example of a case where a time series of class Point (Figure b, blue line) is truncated at the first part will become a series labeled Gun (Figure b, yellow line) because of its similarity to another series also of class Gun (Figure a, blue line).

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(M - |S| - 1)!}{M!} [f_x(S \cup \{i\}) - f_x(S)] \tag{5}$$

In this formula, ϕ_i is the SHAP value for feature i, indicating its contribution. $S \subseteq N \setminus i$ are subsets excluding i, and $\frac{|S|!(M-|S|-1)!}{M!}$ is the weight for i's contribution. $[f_x(S \cup i) - f_x(S)]$ shows the prediction change with and without i in S.

In summary, this formula calculates the SHAP value ϕ_i by summing feature i's marginal contributions across all subsets S of features (excluding i), each weighted to ensure fair attribution.

Algorithm 1: Kernel SHAP Algorithm

Inputs: A predictive model f, a data point x with M features.
Outputs: SHAP values ϕ_i for each feature i.
begin
  Create all 2^M subsets (coalitions) of the features;
  for $k = 1, ..., 2^M$ **do**
    Create a new data point $h_x(z'^{(k)})$ by keeping the values of the features in $z'^{(k)}$ from x and replacing the remaining features with baseline values;
    Get the prediction $f(h_x(z'^{(k)}))$ from model f;
  end
  for $k = 1, ..., 2^M$ **do**
    Calculate the weight:

$$\pi_x(z'^{(k)}) = \frac{M - 1}{\binom{M}{|z'^{(k)}|} \cdot |z'^{(k)}| \cdot (M - |z'^{(k)}|)} \tag{6}$$

  end
  Find $\phi_0, \phi_1, ..., \phi_M$ to minimize the loss function:

$$L(f, \pi_x) = \sum_{k=1}^{2^M} \Big| f(h_x(z'^{(k)})) - \Big(\phi_0 + \sum_{i=1}^{M} \phi_i z_i'^{(k)}\Big)\Big|^2 \pi_x(z'^{(k)}) \tag{7}$$

  return SHAP values $\phi_1, \phi_2, ..., \phi_M$ for each feature;
end

2.4. KernelSHAP

Kernel SHAP is an agnostic method of approximating SHAP values by combining LIME and Shapley. The way to apply KernelSHAP is described in Algorithm 1.

In Algorithm 1, $\pi_x(z'^{(k)})$ is the weight of the subset $z'^{(k)}$, $|z'^{(k)}|$ is the number of features in the subset $z'^{(k)}$, and $z_i'^{(k)}$ is the value of feature i at $z'^{(k)}$. Besides, ϕ_0 is the SHAP value when no

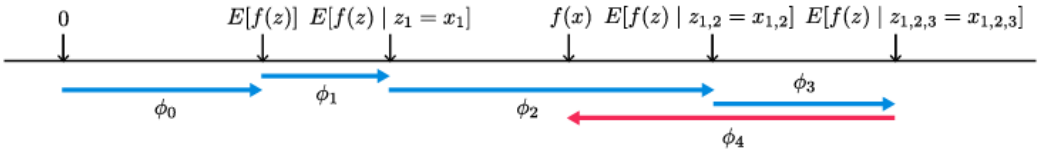


Figure 7. SHAP values for each feature and their respective contributions to the model's prediction.

features are involved, and ϕ_i is the SHAP value at feature i.

The SHAP values ϕ_i quantify each feature's individual impact on the prediction, with ϕ_0 representing the baseline prediction when no features are included. The goal of the algorithm is to minimize the loss function $L(f, \pi_x)$, which measures the difference between the model's actual prediction and the SHAP explanation model across all subsets, weighted by $\pi_x(z'^{(k)})$. By minimizing this loss, Kernel SHAP provides a fair and interpretable distribution of feature importance in line with Shapley value principles.

3. Time Series Classification Explanation Methods

3.1. LIME-based Explanation Method for TSC

Given an input of a time series in a labeled dataset, a classification model, and one of the time series transformations mentioned, this method provides an explanation at two levels: the high-level explanation describes the robustness of the prediction for the transformation; whereas, the low-level explanation shows the influence of each feature in the series on the prediction.

3.1.1. High-level Explanation

The high-level explanation can be summarized in three steps: generating neighboring time series, labeling them, and then computing an index that assesses the robustness in the model's classification decision (see Figure 8):

Algorithm 2: Randomized unwrap the circle

Inputs: Parameters of Beta prime distribution α, β.
Outputs: Random interval in [0,1].
begin
  Sample x randomly generated from BetaPrime(α, β);
  Sample u uniformly on $[-x, 1]$;
  return $[0, 1] \cap [u, u + x]$;
end

Neighbour generation: Generate a neighborhood time series by selecting a random time interval, which is a subsequence of the reference series, and applying one of the four mentioned transformations to this interval; while the features outside the interval are kept unchanged. Thus, it is necessary to randomly select the time interval to be transformed so that all time indexes of the series have the same probability of occurring in that interval. To satisfy this condition, a method called *"Randomized unwrap the circle method"* [24] is used, which treats the randomly generated time intervals as points on the circumference of a circle (described in Algorithm 2).

Neighbour labeling: In this step, the original model classifies the neighboring sequences (noise sequences) generated by the transformation. Some transformations involve distortions in the time axis (x-axis) of the time series, resulting in transformed sequences that are longer or shorter

than the reference time series. However, as Chang Wei Tan et al.'s study [8] has shown, many TSC models are not suitable for handling time series of different lengths, which leaves us with two options: use an additional preprocessing step to make the length of all the perturbed sequences equal, or limit the scope of the method to using TSC models that can handle time series of different lengths. The first option involves adding information to the shorter series or deleting information from the longer series, while the second option makes use of all the available data without making any changes, but this approach is somewhat limited for quantitative evaluation because the number of suitable models is limited. In this paper, to minimize the modification of the time series, the second option was adopted.

Estimation of the Robustness: This method assumes that a prediction is robust (highly certain) to the transformation if all the neighboring time series (generated from that transformation) are classified into the same label as the original reference time series. Hereafter, we call $I = I_{=} \cup I_{\neq}$ the set of randomly generated intervals, where $I_{=}$ refers to the intervals in the neighboring sequences that have the same label as the reference sequence and $I_{\neq}$ contains the intervals of the neighboring sequences that are classified into a different label than the reference sequence. Thus, in this step, the robustness of the prediction for a transformation, R_{transf}, is quantified by measuring the percentage of neighbors that are classified into the same class of the reference time series:

$$R_{transf} = \frac{|I_{=}|}{|I_{\neq}|} \tag{8}$$

where $|I_{=}|$ and $|I_{\neq}|$ are the numbers of elements in the set $I_{=}$ and $I_{\neq}$, respectively. R_{transf} varies in the range $[0, 1]$, where a value of 0 means that the prediction is completely sensitive (has low robustness) to the transformation since all neighbors of the sequence are labeled into a different class. Conversely, a value of 1 means that the prediction is completely robust to the transformation since all considered neighbors are labeled into the same class as the reference sequence.

3.1.2. Low-level Explanation

The low-level explanation consists of calculating the relevance of each region in the time series in the prediction. In this step, two possible situations are considered: a completely robust prediction for a transformation (R_{transf} = 1) or a slightly sensitive prediction for a transformation (R_{transf} ¡ 1). In the first case, the results indicate that the chosen time series transformation does not affect the prediction and thus the explanation is obtained that no time period in the series has a special impact on the prediction for that transformation. In the second case, the low-level explanation is calculated in the following steps (see Fig. 9).

Isolation of intervals of interest: An interval is considered to be influential in the prediction if a transformation applied to this interval changes the classifier's prediction (i.e., the interval belongs to the set $I_{\neq}$). However, in some regions of the time series, there may be intervals belonging to $I_{=}$ and also intervals belonging to $I_{\neq}$. To ensure that a region of the series is influential in the prediction, the term *"intervals of interest"* is introduced, which defines intervals from $I_{\neq}$ that do not have many intervals from $I_{=}$ in their neighborhood.

Specifically, an interval $i_{\neq} \in I_{\neq}$ is considered an intervals of interest if it is predominantly surrounded by intervals from $I_{\neq}$. Based on this idea, the procedure for defining intervals of interest is summarized as follows:

- For each interval $i_{\neq} \in I_{\neq}$, find the K closest intervals in $I = I_{=} \cup I_{\neq}$ using the Hausdorff distance [25]. The calculation of the Hausdorff distance between two intervals is presented in the Appendix 5.

- If more than K/2 of those intervals belong

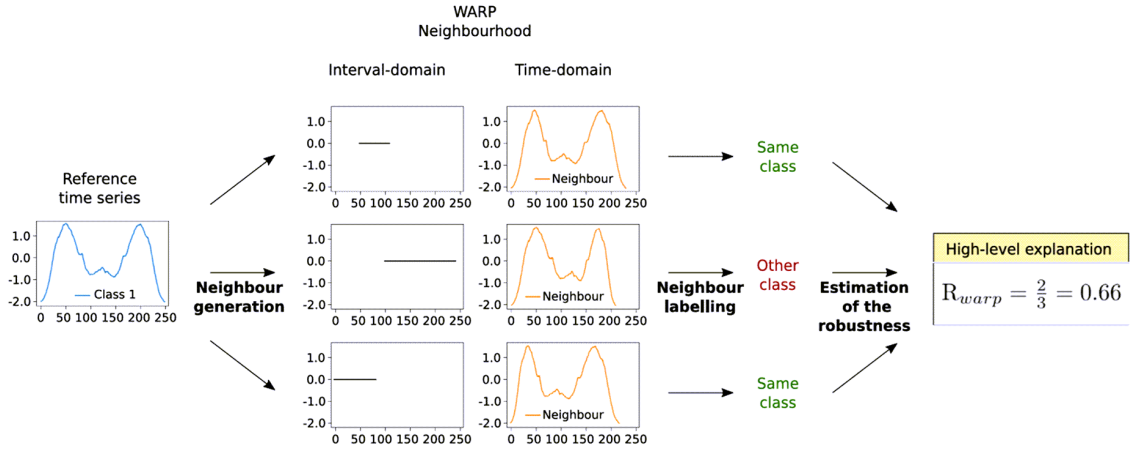


Figure 8. The diagram illustrates a high-level explanation with an example of the Warp transform for a time series in the ArrowHead dataset (from the UCR Repository [6]) with $k_w = 0.7$ [23].

to $I_{\neq}$, then $i_{\neq}$ is considered an intervals of interest. Otherwise, if at least K/2 of those intervals belong to $I_{=}$, then $i_{\neq}$ is not considered an intervals of interest.

Note that the distribution of intervals in the set I between $I_{=}$ and $I_{\neq}$ varies across time series. Thus, using a fixed value of the parameter K for all time series will bias the neighborhood of the reference series toward the predominant interval type ($I_{=}$ and $I_{\neq}$). Therefore, the parameter K is set to the proportion of the number of intervals contained in $I_{\neq}$.

Intersection: This step involves summarizing and presenting information about the intervals of interest in the original time series. The idea is that the more intervals of interest containing time index i, the more influential this index is on the prediction. Thus, given a time series $T = (t_1, ..., t_i, ..., t_l)$, we count the number of intervals of interest containing index i with $i = 1...l$ and these values are stored in a vector $w = (w_1, ..., w_i, ..., w_l)$. Therefore, the low-level explanation can be viewed as a weight vector w, where w_i denotes the influence of time index i on the prediction. The time series is then colored depending on these values: red indicates that these time metrics are contained in many intervals of interest, so the region is important for the prediction, while blue indicates the opposite. Note that the color bar is normalized to $[0, 1]$.

Note that, given a time series, a transformation, and a TSC model, the cost of computing an explanation is dominated by the step of labeling neighboring series. In this step, neighbors are labeled by the corresponding TSC model, so the time taken to perform this process varies significantly depending on the complexity of the model and the length of the time series.

3.2. Explanation Method for TSC Using SHAP

3.2.1. Kernel SHAP for Time Series Data

Although Kernel SHAP can generate model-agnostic explanations, it heavily relies on designing a good mapping for the specific application domain. The pieces of the sample x defined by h_x need to be intuitively understandable to have meaningful impacts. Poorly designed mappings can produce misleading explanations, undermining the idea of explainability. This makes constructing good mappings a significant challenge.

To apply Kernel SHAP to time series data, we need to develop a mapping function that divides

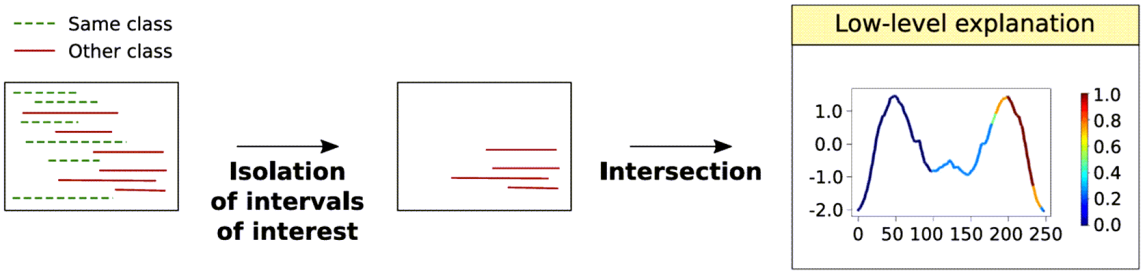


Figure 9. The diagram illustrates the low-level explanation with an example being a time series in the ArrowHead dataset [23].

the time series into fragments and computes the impacts of these fragments on the model. To divide the time series into fragments, we define the input space $I = R^d$ as the space of time series with length d. Each vector $z \in I$ represents a time series, with the value at time $t \in \{1, ..., d\}$ denoted by z_t. For technical time series segmentation, we propose the Time Slice Mapping transformation, which partitions the time series into equal segments and applies value transformations to specific segments to generate perturbed time series.

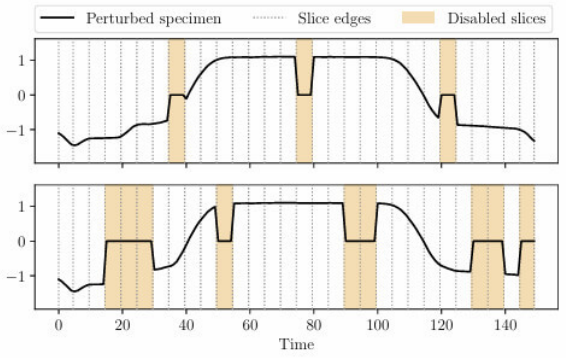


Figure 10. Two different perturbations on the same sample.

3.2.2. *Time Slice Mapping*

This technique segments the sample (x) into (d') equal-length segments along the time axis. Each segment $(i \in \{1, ..., d'\})$ constitutes a segment whose activity is governed by (z'_i). To disable a segment, it is not feasible to simply remove or fill in missing values, as most models cannot handle time series with varying lengths or missing values. Instead, the segment is replaced with the corresponding segment from a second substitute time series $(r \in I)$, effectively masking the original segment.

When the time slice mapping function, tailored to the sample (x), is queried with a binary input (z'), it disables all segments (i) of (x) for which (z'_i) is 0, resulting in a perturbed version of (x), as depicted in Figure 10.

Given a substitute time series $(r \in I)$ and a mapping function $(j : \{1, ..., d\} \to \{1, ..., d'\})$ that assigns each time point (t) to a segment, the mapping function $(h(1)_x : I' \to I)$ generates a perturbed time series for every $(z' \in I')$ according to the following rule:

$$\forall t \in \{1, ..., d\} : (h(1)_x(z'))_t = \begin{cases} x_t, & \text{if } z'_{j(t)} = 1; \\ r_t, & \text{if } z'_{j(t)} = 0. \end{cases} \tag{9}$$

This formula defines how the perturbed time series is created by selectively replacing segments in x with those in r, based on the binary input z'. For each time point t, if $z'_{j(t)} = 1$, the original value x_t is retained; otherwise, the corresponding value r_t from the substitute series is used, effectively masking that segment of x.

In applying Time Slice Mapping to Kernel SHAP, this function allows for generating various perturbations in each step of the SHAP algorithm. Each perturbation corresponds to a subset of features, toggled on or off based on z'. This enables Kernel SHAP to evaluate the impact of each segment on the model's output by observing the predictions with different combinations of active and masked segments, providing a detailed view of feature contributions over time.

4. Experimental Results

4.1. Case of Study

4.1.1. GunPoint Dataset

The dataset consists of 200 data samples (observations) with 50 samples in the training set and 150 samples in the test set, with 151 features, including one dependent variable and the remaining 150 features representing a time series. The dependent variable is considered as the label of the observation (Gun = 1, Point = 2). This data is used in many reputable papers on XAI for time series data.

4.1.2. Coffee Dataset

The dataset consists of 56 instances (observations) with 28 samples in the training set and 28 samples in the testing set, with 286 features representing a time series and a dependent variable representing the label of the sample. The time series represents the process of recording the infrared spectral values of 2 types of Robusta coffee (label 1) and Arabica (label 0) through a food spectrometer in the Coffee discrimination experiment [26].

4.2. Set-up

The experimental program is implemented in Python language, TSC models and some probability distribution functions are used from the libraries `scikit-learn` [27], `sktime` [28] and `scipy` [29]. As mentioned earlier, in this experiment we use the models that handle variable length time series. Therefore, 3 standard and popular TSC models were selected: 1-NN-DTW distance, Shapelet Transform (ST) and, Bag-of-SFA-Symbols (BOSS). We experiment on a system consisting of 16 AMD Ryzen 7 4800H CPUs with 2.9GHz scan frequency and 16GB of RAM.

Because the LIME-based method includes time series transformation process to generate neighbors, experiments were conducted by considering and selecting some parameters of the randomization and transformation. The number of neighboring series is set to 500, while the parameters in Algorithm 2 are set to $\alpha = 8$ and $\beta = 18$, so that the probability that a time index is covered by an interval is 0.3 [24]. In this way, each transformed time index appears in about 150 of the 500 generated neighbors. In the calculation of the intervals of interest, the parameter K is set to $0.1 \times |I_{\neq}|$. Regarding the time series transformation process, the warp and scale levels considered in this experimentation, k_w and k_s, are $\{0.7, 0.8, 0.9, 1.1, 1.2, 1.3\}$, while the considered noise levels, k_n are $\{1, 3, 5, 7, 9\}$. Other sets of parameters could be considered, but we limit our experimentation to the mentioned values; we think that they lead to a reasonable variety in the transformations, while creating realistic neighbours with no extreme modifications in the time series. Each transformation level is independently studied. The slice transformation does not depend on any parameter but, in order to ensure a minimum length in the generated neighbours, we set the minimum interval length to $0.3 \times l$, where l is the length of the series that is the object of study.

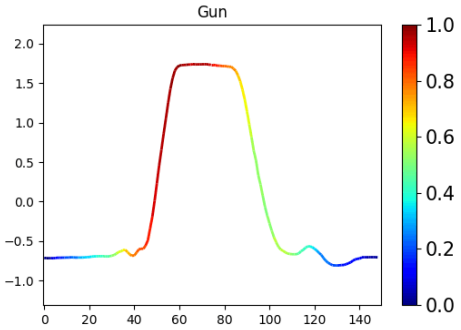
We first train the TSC models with training sets from two datasets, then randomly select time series from the two datasets to predict labels for that series, as well as provide explanations using the proposed methods.

4.3. LIME-based Explanation Method for TSC

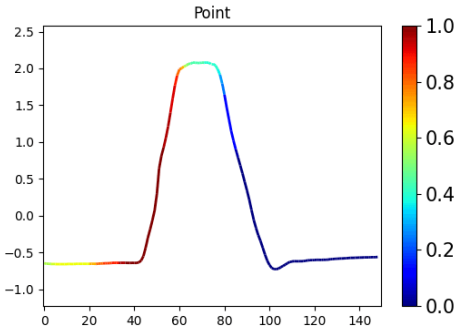
4.3.1. Experimental Results

Select a time series in the GunPoint dataset, use the 1-NN-DTW model to classify through the Warp transformation. With the LIME-based TSC interpretation method, we will get the result including an index showing the confidence of the classification decision for the considered transformation. Besides, a weight vector is given, showing the influence level of each feature in the series on the model's decision. We visualize the weight vector into a saliency map as Figure 11, in which the values of the weights are normalized to values in the range [0,1].

The influence of each feature on the TSC model's decision is specifically shown by a color range from blue to red. Features that have a large influence on the decision will be colored red and features that have a small influence will be colored blue. With the selected data sample in Figure 11, we can see that the red area is concentrated in the time interval [40, 60], indicating that this time interval is the most influential on the model's decision to label this sequence. Indeed, considering the context of the data set, the interval [40, 60] represents the time interval when the experimenter's hand position leaves the initial position, so this interval will have the largest influence on the model's classification decision. In other words, if the speed of this action is changed, there is a high probability that the model's classification decision for this action will also change. We can use this method to derive any transformation as well as any TSC model for data samples, such as the examples in Figures 12, 13 and 14.



(a) Warp (k_w = 0.7) with 1-NN-DTW model, R_{warp} = 0.60.



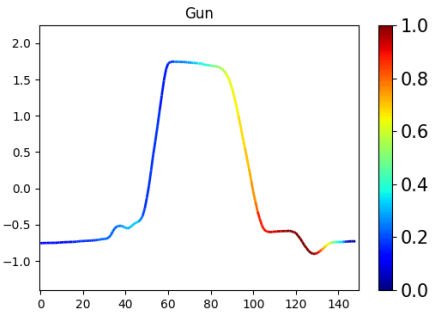
(b) Warp (k_w = 0.7) with 1-NN-DTW model, R_{warp} = 0.64.

Figure 11. Experimental results for data samples labeled Gun (11a) and Point (11b) in the GunPoint dataset with Warp transform.

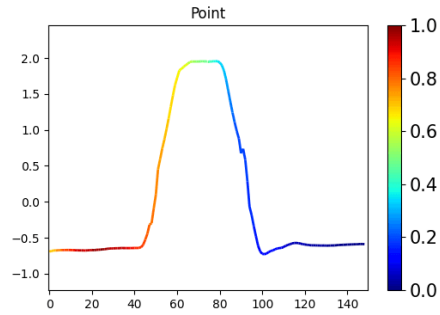
4.3.2. Quantitative Evaluation

In this section, the quantitative evaluation will be based on the method in [2] and [23], with modifications to suit our problem context. The idea behind our evaluation method is to show that perturbations in important regions will change the model's predictions more often than perturbations in unimportant regions.

The important and unimportant regions are determined based on the distribution of values of the computed weight vector w: the most important $p\%$ is the $p\%$ time index with the highest weight in w, and we will call this set of indices $p+$. Similarly, the least important $p\%$ is the $p\%$ time index with the lowest weight in w, and we will call this set $p-$. With X being the test set, let X^{p-} be the set of sequences in which the least important $p\%$ time index is transformed by the chosen transformation. Conversely, X^{p+} is the set in which the most important $p\%$ time

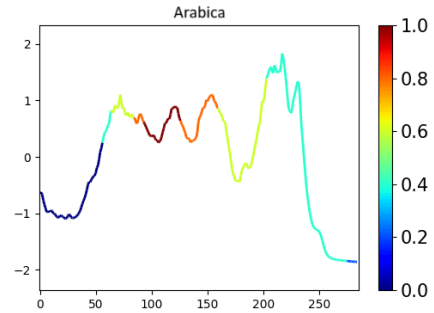


(a) Scale ($ks = 1.3$) with 1-NN-DTW model, $R_{scale} = 0.32$.

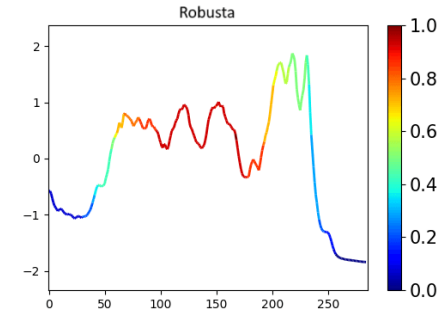


(b) Scale ($ks = 1.3$) with 1-NN-DTW model, $R_{scale} = 0.45$.

Figure 12. Experimental results for data samples labeled Gun (12a) and Point (12b) in the GunPoint dataset with the Scale transformation.

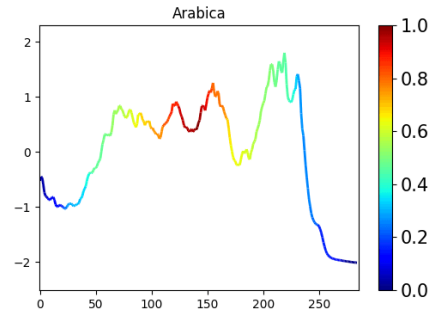


(a) Noise ($k_n = 7$) with BOSS model, $R_{scale} = 0.89$.

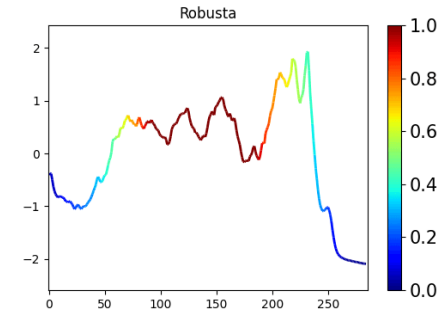


(b) Noise ($k_n = 7$) with BOSS model, $R_{scale} = 0.73$.

Figure 13. Experimental results for data samples labeled Arabica (13a) and Robusta (13b) in the Coffee dataset with Noise transformation.



(a) Slice with ST model, $R_{slice} = 0.11$.



(b) Slice with ST model, $R_{slice} = 0.49$.

Figure 14. Experimental results for data samples labeled Arabica (14a) and Robusta (14b) in the Coffee dataset with Slice transformation.

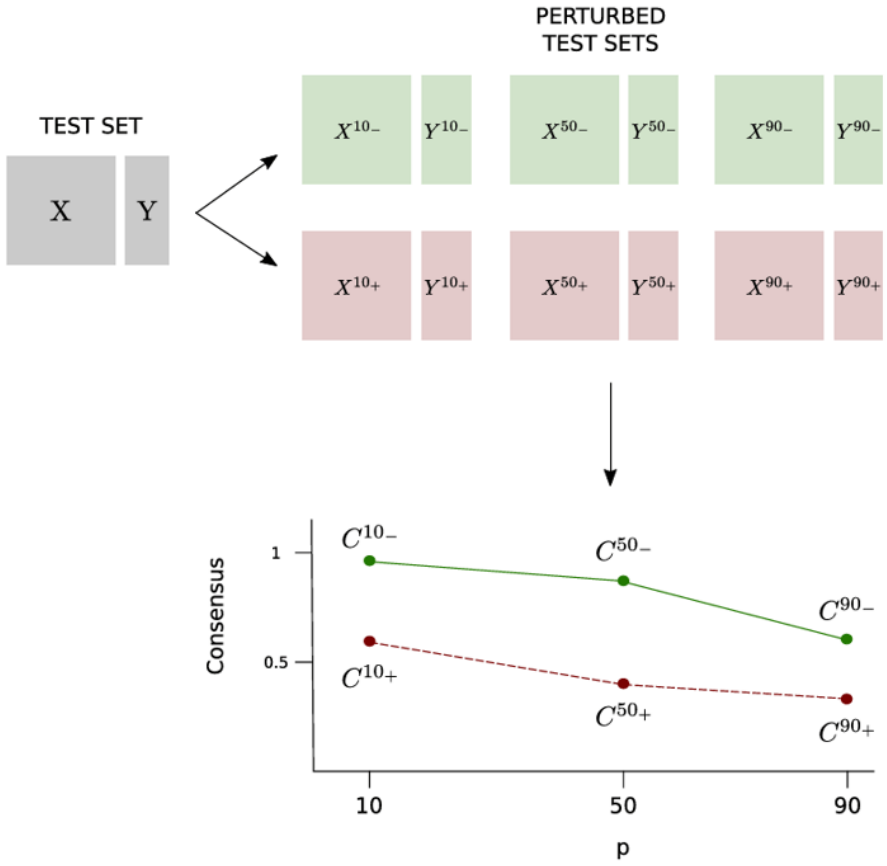


Figure 15. Diagram showing quantitative assessment method [23].

index is transformed. Then, we label the strings in the set X^{p-} (this label set is called Y^{p-}) and X^{p+} (called Y^{p+}). For 3 values of p: 10, 50 and 90, we calculate the probability that the label of the time series in the set X^{p-} (or X^{p+}) is different from the label of the test set using the formula:

$$C^{p+} = \frac{1}{m}\sum_{s=0}^{m} \mathbb{1}_{y_s = y_{s^{p+}}} \tag{10}$$

with:

$$\mathbb{1}_{y_s = y_{s^{p+}}} = \begin{cases} 1, & \text{if } y_s = y_{s^{p+}}; \\ 0, & \text{if } y_s \neq y_{s^{p+}}. \end{cases} \tag{11}$$

The probability value varies from 0 to 1, where 0 means that all labels in Y^{p+} (or Y^{p-}) are different from the labels in Y, while 1 means that all time series in the perturbed set are classified in the same class as the corresponding time series in the original test set. The computation is performed sequentially for different values of $p = (10, 50, 90)$, so that C^{p-} and C^{p+} form two curves (as shown in Figure 15).

The area under each line C^{p-} and C^{p+}, $eLoss1$ and $eLoss2$, are calculated using the trapezoidal rule. The area between the lines C^{p-} and C^{p+} is called $\Delta eLoss$ and is defined by:

$$\Delta eLoss = eLoss1 - eLoss2 \tag{12}$$

If $\Delta eLoss$ is positive, the explanation is

Table 2. Table of average $\Delta eLoss$ and Robustness values (Robustness values are shown in parentheses) over the entire dataset when applying transformations with different TSC models. For each transformation and TSC model on a dataset, the best $\Delta eLoss$ value is shown in bold

	Warp			**Scale**		
	1-NN-DTW	BOSS	ST	1-NN-DTW	BOSS	ST
GunPoint	**0.10 (0.86)**	0.01 (0.54)	0.09 (0.76)	**0.26 (0.42)**	0.15 (0.85)	0.00 (0.94)
Coffee	– (1.00)	**0.36 (0.82)**	0.17 (0.71)	**0.23 (0.45)**	0.16 (0.66)	0.11 (0.82)
	Noise			**Slice**		
	1-NN-DTW	BOSS	ST	1-NN-DTW	BOSS	ST
GunPoint	**0.29 (0.56)**	0.29 (0.40)	0.23 (0.51)	-0.06 (0.10)	0.02 (0.80)	**0.04 (0.40)**
Coffee	0.10 (0.62)	0.08 (0.86)	0.09 (0.58)	**0.15 (0.00)**	0.05 (0.86)	0.14 (0.74)

considered credible and if $\Delta eLoss$ is negative, the explanation is considered unreliable. We have evaluated all 4 transformations, 3 TSC models for 2 datasets GunPoint and Coffee, obtaining the results presented in Table 2.

Table 2 shows that the average $\Delta eLoss$ value is positive in most cases, with only one negative $\Delta eLoss$ case in italics in the table, which means that the explanation given is informative. The results also confirm that perturbing the important parts will have a higher chance of changing the classifier's prediction than perturbing the unimportant parts.

If we analyze the results by each transformation, we can see that the negative $\Delta eLoss$ value, i.e. the explanations are not informative, is in the Slice transformation. This may indicate that the Slice transformation is not suitable for our experimental datasets. If we analyze the results based on each TSC model, in most cases the best-explained model (highlighted) is 1-NN-DTW, which obtains the highest $\Delta eLoss$ value in 6 out of 8 cases.

4.3.3. Comment

This method provides an intuitive and semantically understandable explanation for the classification decision of the TSC model for time series. Specifically, the method shows that the speed of each action recorded in the GunPoint dataset is discriminative, i.e. if an action occurs faster or slower in a specific time interval, it can be classified into a different class than it would be if the change were not applied. In addition, a number is also provided to represent the model's reliability for a transformation, helping users to evaluate the suitability between time transformations and TSC models, thereby choosing the appropriate model for each specific problem.

However, besides the obvious advantages mentioned, this method still has limitations that need to be overcome: *Firstly*, determining the correct neighborhood is a big challenge and there is no general rule to prescribe. For example, if the time interval chosen to perform the transformations is too short compared to the length of the reference sequence, the generated neighbor sequences may all be labeled in the same class as the original sequence, leading to the inability to provide low-level explanations. *Secondly*, illogical cases may occur when applying the transformations. For example, with the GunPoint dataset, the values of the features (logically) only lie within the range $[-2, 2]$, but if we apply the Scale transformation with an unreasonable scale, the values of some features in

```
[-0.00656379  0.00588742 -0.00643476  0.00800835 -0.00776153 -0.02265182
  0.01358833  0.0076313   0.02060812  0.04597982  0.01359996 -0.00688044
  0.00408194  0.00196779 -0.00016988  0.00018479  0.00893332 -0.0073931
  0.03271896  0.03167794 -0.06060252  0.04543312  0.01595328  0.01894213
  0.02139157 -0.03427938  0.01895683 -0.01025941 -0.01049109  0.00230709]
```

Figure 16. Table of SHAP Values.

the sequence may lie outside this range, leading to the risk of errors when performing classification using TSC models. *Finally*, the consistency of the method may not be guaranteed, because each time we perform the initialization of neighbor sequences, the generated random intervals will often be completely different.

4.4. Explanation Method for TSC Using SHAP

4.4.1. Point Sample - GunPoint Dataset

The parameter values for this experiment are as follows: the label is set to 2 for the Point class, with 30 slices and a slice length of 5. The classification model used is ST.

After applying KernelSHAP, we obtain the SHAP values corresponding to each segment, as shown in Figure 16.

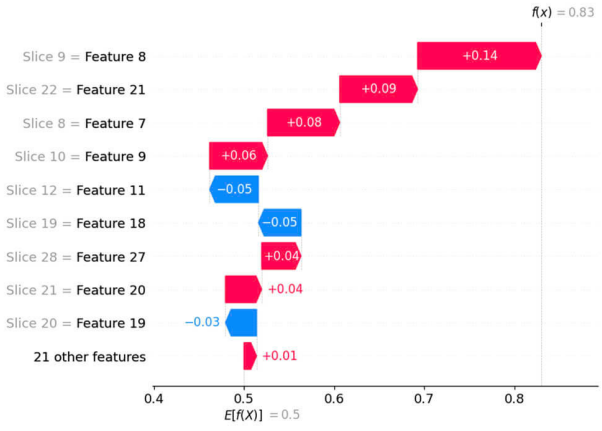


Figure 17. Waterfall Plot - SHAP for Point Sample.

To easily visualize the impact of each feature on the model's decision, we can use a Waterfall plot (Figure 17) or a Force-plot (Figure 18).

Features that have a positive impact on the decision are shown in red, and features with a negative impact are shown in blue. The magnitude of the impact for each feature is also clearly displayed.

Based on the Waterfall plot (Figure 17) and the Force-plot (Figure 18), we can easily understand and identify the model's decision.

$E[f(X)] = 0.5$ indicates that the expected value of the model is 0.5. This means that the probability of the label 2 in the training set is 0.5. Using the SHAP values calculated, we can also represent the final value of $f(x)$. In this case, $f(x) = 0.83$ indicates that the probability of the model assigning label 2 to the data point being considered is 0.83.

Based on the plots, we can also easily identify the impact of each feature on the value of $f(x)$. Besides, we can easily answer the following questions: Which features have the most positive and negative impact on the model's decision? What is the extent of their impact? The answer is that Slice 9 has a positive impact of +0.14 on the model, while Slice 12 has a negative impact of -0.05 on the model (the higher the value of $f(x)$, the higher the accuracy of the model).

From the calculated SHAP values, we can also represent it in a Colormap as shown in Figure 19. With this plot, we can simultaneously visualize the time series and the impact of each feature on the model's decision. Features with a larger impact will have a darker color, and features with a smaller impact will have a lighter color. This is displayed on the color scale on the right side of the plot.

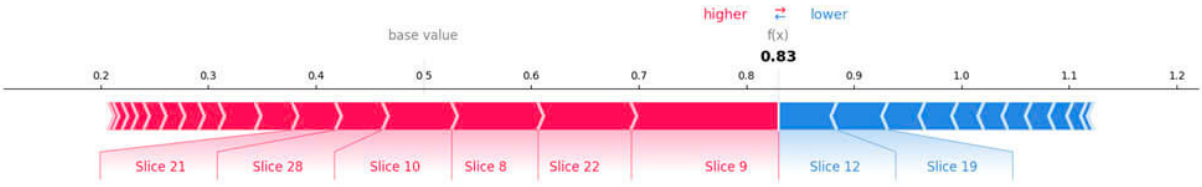


Figure 18. Force-Plot - SHAP for Point Sample.

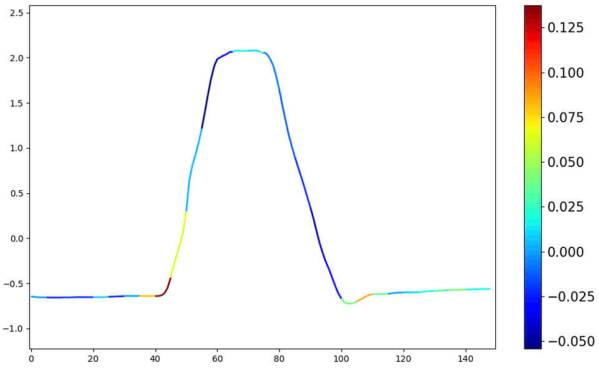


Figure 19. Colormap - SHAP for Point Sample.

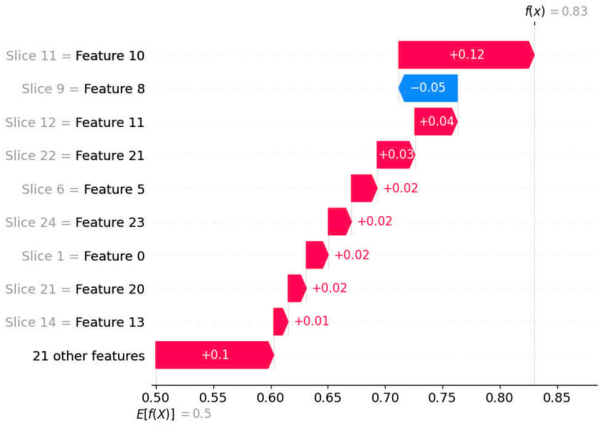


Figure 20. Waterfall Plot - SHAP for Gun Sample.

4.4.2. Experiments with Other Samples

Similar to the Point Sample - GunPoint Dataset, we conducted experiments on three samples: Gun Sample from the GunPoint Dataset, Robusta and Arabica Samples from the Coffee Dataset. For each sample, we present the Waterfall plot and Colormap plot to show the impact of each feature on the model's decision for each sample.

a) Gun Sample - GunPoint Dataset

The parameter values for this experiment are similar to those of the Point sample, with the only difference being the label, which is set to 2 for the Gun class.

The impact of each feature on the model's decision for the Gun Sample - GunPoint Dataset is shown in Figure 20 and Figure 21.

From the plots, we can see that Slice 11 has the highest positive impact on the model's decision for the Gun label with a value of +0.12, while Slice 9 has the highest negative impact on the model's decision for the Gun label with a value of −0.05.

b) Robusta Sample - Coffee Dataset

The parameter values for this experiment are as follows: the label is set to 1 for the Robusta class, with 57 slices and a slice length of 5. The classification model used is ST.

The impact of each feature on the model's decision for the Robusta Sample - Coffee Dataset is shown in Figure 22 and Figure 23.

From the plots, we can see that Slice 47 has the highest positive impact on the model's decision for the Robusta label with a value of +0.18 . In contrast, Slices 5 and 32 have the highest negative impact on the model's decision for the Robusta label, each with a value of −0.04.

c) Arabica Sample - Coffee Dataset

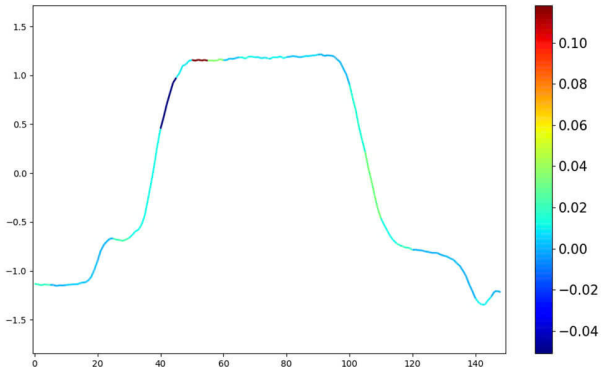


Figure 21. Colormap - SHAP for Gun Sample.

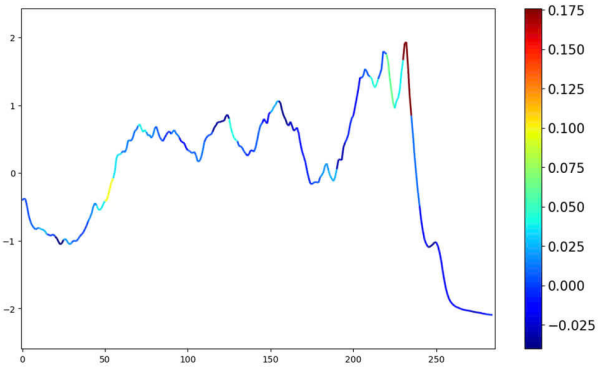


Figure 23. Colormap - SHAP for Robusta Sample.

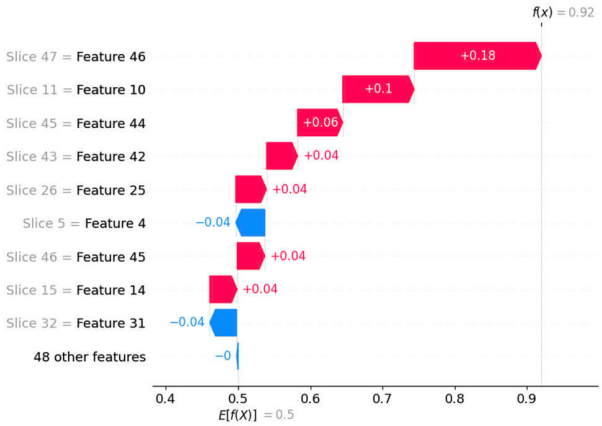


Figure 22. Waterfall Plot - SHAP for Robusta Sample.

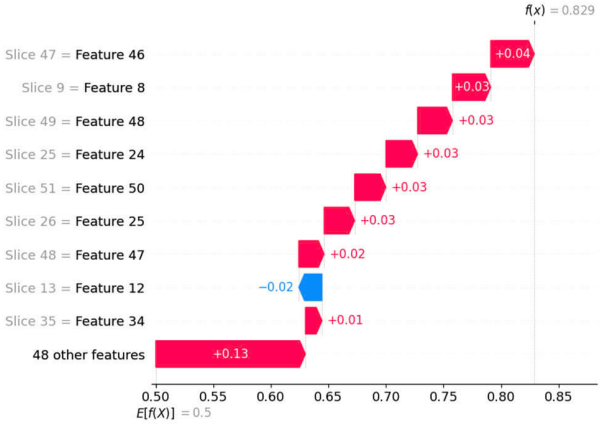


Figure 24. Waterfall Plot - SHAP for Arabica Sample.

The parameter values for this experiment are similar to those of the Robusta sample, with the only difference being the label, which is set to 0 for the Arabica class.

The impact of each feature on the model's decision for the Arabica Sample - Coffee Dataset is shown in Figure 24 and Figure 25.

From the plots, we can see that Slice 47 has the highest positive impact on the model's decision for the Arabica label with a value of +0.04, while Slice 13 has the highest negative impact on the model's decision for the Arabica label with a value of −0.02.

4.4.3. Compare Experiment Result with LIME

With the same data sample, Robusta from the Coffee dataset, and using the same classification model, ST, we obtained the results shown in Figure 14*b* for LIME and in Figure 23 for SHAP.

We can easily see that the fundamental difference between SHAP and LIME lies in the feature impact values: SHAP includes negative values, while LIME does not. This difference suggests that SHAP provides a more nuanced understanding of feature contributions, capturing both positive and negative impacts on the model's prediction, whereas LIME may offer a simpler but less detailed perspective.

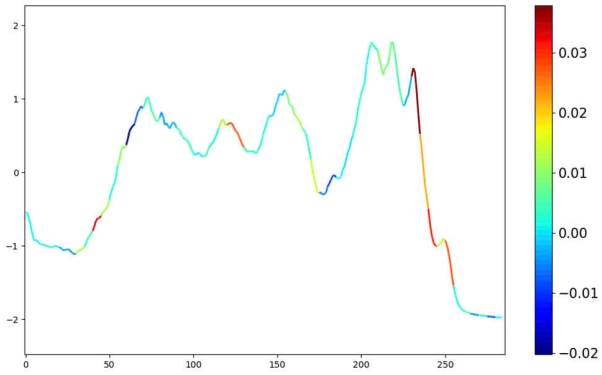


Figure 25. Colormap - SHAP for Arabica Sample.

5. Conclusion and Future Work

The local TSC explanation method based on LIME and SHAP that has been proposed can become a powerful tool for explaining TSC models. The key strength of this method lies in its ability to provide a precise explanation of the contribution value for each feature in the time series. Consequently, we can identify the degree of influence of each factor on the model's prediction results, including both positive and negative impacts. Additionally, the application of time series data transformation methods discussed in this work can provide additional useful information about the properties of time series, supporting the provision of effective semantic explanations. The detailed explanations for TSC models are visually represented through saliency maps, making it easy for users to comprehend the information provided by the explanations.

For the LIME-based explanation method, exploring the application of additional time series transformations (such as bootstrapping for time series [21]) or combining multiple transformations to generate neighboring sequences is a promising avenue to address the limitations discussed in Section 4.3.3.

Similarly to the problem of initializing neighbors in the LIME-based explanation method, determining the features to perform segmentation is also an important task to make SHAP work more smoothly and efficiently. Let's consider the following example:

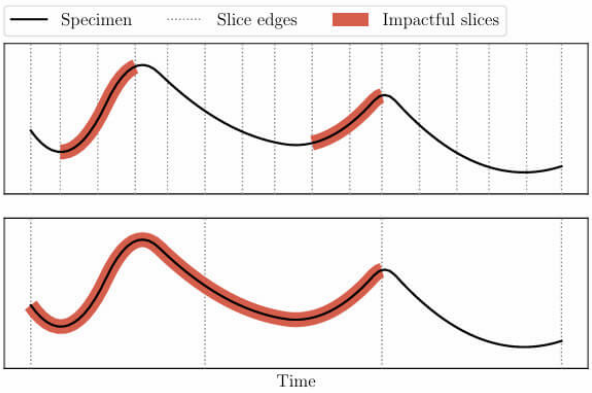


Figure 26. The difference in importance when slice size changes.

When the slice size is different, the impact of features on the model's decision also changes and may lead to inaccuracy. Therefore, in the future, this study can be further researched and developed to select the optimal slice size for each model and each specific data point to accurately calculate the impact.

Furthermore, beyond the challenge of determining an optimal number of slices for segmentation, our SHAP-based explanation method currently employs slices of uniform size. An alternative approach involves segmenting time series into slices of varying sizes, which could provide a more nuanced understanding of the features that genuinely influence the model's decision.

Acknowledgment

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2023.4.

References

[1] 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. 100700-100724, 2022. doi: 10.1109/ACCESS.2022.3207765.

[2] T. T. Nguyen, T. N. Le, and G. Ifrim, "A Model-agnostic Approach to Quantifying the Informativeness of Explanation Methods for Time Series Classification," in *ECML PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, pp. 77-94, 2020.

[3] F. Mujkanovic, V. Doskoc, M. Schirneck, P. Schafer, and T. Friedrich, "timeXplain - A Framework for Explaining the Predictions of Time Series Classifiers," *arXiv preprint*, arXiv:2009.13211, 2020.

[4] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," *arXiv preprint*, arXiv:1602.04938, 2016.

[5] M. Guillemé, V. Masson, L. Rozé, and A. Termier, "Agnostic Local Explanation for Time Series Classification," *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 432-439, 2019. doi: 10.1109/ICTAI.2019.00067.

[6] H. A. Dau, A. Bagnall, K. Kamgar, C. C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The UCR Time Series Archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293-1305, 2019. doi: 10.1109/JAS.2019.1911747.

[7] P. Esling and C. Agon, "Time-series Data Mining," *ACM Computing Surveys*, vol. 45, no. 1, pp. 1-34, 2012. doi: 10.1145/2379776.2379788.

[8] C. W. Tan, F. Petitjean, E. Keogh, and G. I. Webb, "Time Series Classification for Varying Length Series," *arXiv preprint*, arXiv:1910.04341, 2019.

[9] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data Augmentation for Time Series Classification Using Convolutional Neural Networks," in *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, 2016.

[10] R. Assaf, I. Giurgiu, F. Bagehorn, and A. Schumann, "MTEX-CNN: Multivariate Time Series EXplanations for Predictions with Convolutional Neural Networks," *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 952-957, 2019. doi: 10.1109/ICDM.2019.00106.

[11] Z. Wang, W. Yan, and T. Oates, "Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline," *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1578-1585, 2017. doi: 10.1109/IJCNN.2017.7966039.

[12] J. Crabbé and M. van der Schaar, "Explaining Time Series Predictions with Dynamic Masks," *arXiv preprint*, arXiv:2106.05303, 2021.

[13] P. Senin and S. Malinchik, "SAX-VSM: Interpretable Time Series Classification Using SAX and Vector Space Model," *2013 IEEE International Conference on Data Mining*, pp. 1175-1180, 2013. doi: 10.1109/ICDM.2013.52.

[14] Y. Wang, R. Emonet, E. Fromont, S. Malinowski, E. Menager, L. Mosser, and R. Tavenard, "Learning Interpretable Shapelets for Time Series Classification through Adversarial Regularization," *arXiv preprint*, arXiv:1906.00917, 2019.

[15] R. Guidotti, A. Monreale, F. Spinnato, D. Pedreschi, and F. Giannotti, "Explaining Any Time Series Classifier," *2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI)*, pp. 167-176, 2020. doi: 10.1109/CogMI50398.2020.00029.

[16] J. Labaien, E. Zugasti, and X. De Carlos, "Contrastive Explanations for a Deep Learning Model on Time-series Data," in *International Conference on Big Data Analytics and Knowledge Discovery*, pp. 235-244, 2020.

[17] I. Karlsson, J. Rebane, P. Papapetrou, and A. Gionis, "Explainable Time Series Tweaking via Irreversible and Reversible Temporal Transformations," *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 207-216, 2018. doi: 10.1109/ICDM.2018.00036.

[18] E. Delaney, D. Greene, and M. T. Keane, "Instance-based Counterfactual Explanations for Time Series Classification," *arXiv preprint*, arXiv:2009.13211, 2021.

[19] J.-P. Kreiss and S. N. Lahiri, "Bootstrap Methods for Time Series," in *Handbook of Statistics*, vol. 30, Elsevier, pp. 3-26, 2012. doi: 10.1016/B978-0-444-53858-1.00001-6.

[20] A. D. Hutson, "Resampling Methods for Dependent Data," *Technometrics*, vol. 46, no. 2, pp. 252-252, 2004. doi: 10.1198/tech.2004.s795.

[21] D. Rajapaksha and C. Bergmeir, "Limref: Local Interpretable Model Agnostic Rule-based Explanations for Forecasting, with an Application to Electricity Smart Meter Data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 11, pp. 12098-12107, 2022.

[22] J. Yoon, D. Jarrett, and M. Schaar, "Time-series Generative Adversarial Networks," 2019.

[23] A. Abanda, U. Mori, and J. A. Lozano, "Ad-hoc Explanation for Time Series Classification," *Knowledge-Based Systems*, vol. 252, p. 109366, 2022. doi: 10.1016/j.knosys.2022.109366.

[24] L. Devroye, P. Epstein, and J.-R. Sack, "On Generating Random Intervals and Hyperrectangles," *Journal of*

Computational and Graphical Statistics, vol. 2, no. 3, pp. 291-307, 1993.

[25] F. Hausdorff, "Set Theory," Chelsea Publ. Co., New York, 1962.

[26] R. Briandet, E. K. Kemsley, and R. H. Wilson, "Discrimination of Arabica and Robusta in Instant Coffee by Fourier Transform Infrared Spectroscopy and Chemometrics," *Journal of Agricultural and Food Chemistry*, vol. 44, no. 1, pp. 170-174, 1996. doi: 10.1021/jf950305a.

[27] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *arXiv preprint*, arXiv:1201.0490, 2018.

[28] M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines, and F. J. Király, "Sktime: A Unified Interface for Machine Learning with Time Series," *arXiv preprint*, arXiv:1909.07872, 2019.

[29] P. Virtanen et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261-272, 2020. doi: 10.1038/s41592-019-0686-2.

Appendix .1. ***Hausdorff distance between two time intervals***

Named after Felix Hausdorff (1868-1942), the Hausdorff distance [25] is stated as "the maximum distance of one set to the nearest point in the other set".

More specifically, the Hausdorff distance from set A to set B is a maximin function, defined by the formula:

$$h(A, B) = max_{a \in A}[min_{b \in B} d(a, b)]$$

where a and b are points in sets A and B, respectively, and $d(a, b)$ is an arbitrary distance between these points; for simplicity, we will take $d(a, b)$ to be the Euclidian distance between a and b.

If A and B are two sets of points, the Hausdorff distance between them will be:

$$HD(A, B) = max\{h(A, B), h(B, A)\}$$

In the context of our work, we have two intervals $A = [a_1, a_2]$ and $B = [b_1, b_2]$ with $a, b \in \mathbb{R}$, the Hausdorff distance from A to B is determined by the following formula:

$$d_H(A, B) = max\{|a_1 - b_1|, |a_2 - b_2|\}$$

Note that the Slice transform uses the union of intervals, so the extension of the Hausdorff distance calculation with the union [**?**] is used for this transform.