Unsupervised Feature Construction for Time Series Anomaly Detection - An Evaluation

Marine Hamon[1,2], Vincent Lemaire[1], Nour Eddine Yassine Nair-Benrekia[1], Samuel Berlemont[1], and Julien Cumin[1]

[1] Orange Research Lannion, Châtillon et Meylan, France
{marine.hamon, vincent.lemaire, noureddineyassine.nairbenrekia, samuel.berlemont, julien1.cumin}@orange.com
[2] Université de Rennes, France

Abstract. To detect anomalies with precision and without prior knowledge in time series, is it better to build a detector from the initial temporal representation or to compute a new, tabular representation using an existing automatic variable construction library? This article addresses this question by conducting an in-depth experimental study for two popular detectors: Isolation Forest and Local Outlier Factor. The results, obtained from experiments on five different datasets, show that the new representation, computed using the *tsfresh* library, allows Isolation Forest to improve its performance significantly.

Keywords: anomaly detection · time series · feature construction.

1 Introduction

The literature of time series deals with various learning tasks such as forecasting, clustering and classification. In this article, we address the problem of time series anomaly detection (TSAD) [3, 5, 6, 20, 33], specifically for univariate time series. We denote $\tau_i = \langle (t_1, x_1), \dots, (t_m, x_m) \rangle$ a univariate time series, where x_k is the value of the series at time t_k. In this article, the objective is to predict whether the pair (t_i, x_i) corresponds to an anomaly (point-wise detection).

Over the past few years, a consensus has emerged within the community that transforming time series from the temporal domain to an alternative representation space is one of the most effective ways of improving model accuracy. This has been observed in classification [1, 28, 32], early classification [4] and, to a lesser extent, in anomaly detection in a few very specific publications focused on particular use cases [36, 40].

This recent research work has motivated the study we present in this paper. We build on the idea of changing the representation space and then applying "usual" anomaly detectors designed for tabular data [32]. The question we address here is: can we achieve better anomaly detection performance in the computed feature space, or is it better to stay in the original temporal representation?

The rest of this article is organized as follows: Section 2 presents the background and key concepts used in this article, so that it can be properly positioned in the very large literature of anomaly detection for time series. Section 3 presents the proposed processing pipeline. Section 4 presents the experimental protocol. Detailed results are then presented in section 5 before concluding in the final section.

2 Context and Concepts Used

To allow the reader to correctly position the work carried out in this article, we outline below the main themes present in the literature of anomaly detection for time series [5, 6] and explain our positioning. As the topic of anomaly detection in time series is very widely covered in the literature, we do not claim to be exhaustive but rather aim to be factual for the purposes of this study.

2.1 Types of Anomalies

The literature distinguishes at least three main types of anomalies [10]:

- (i) **point anomalies** : for example, an unusually high financial transaction in relation to a customer's transaction history.
- (ii) **collective anomalies** (in the sense of a succession of correlated punctual anomalies): for example, a sudden drop in traffic on a website, due for example to a server failure or a denial-of-service attack.
- (iii) **contextual anomalies**: for example, an abnormal increase in electricity consumption in a given region, due for example to a snowstorm or an exceptional heat wave.

Given the objective of our study, we are placing ourselves in the most general case possible. We make the effort of being as "agnostic" as possible through the following assumptions. (i) We have no feedback loop from an expert user to contextualize the observed data (ii), to adjust the parameters of the methods (iii), or to adjust the sliding window size (see below). (iv) We also do not try to distinguish punctual anomalies from collective ones.

2.2 Online or Offline Analysis

For anomaly detection in time series, it is important to distinguish between online detection and offline analysis. Online detection refers to the real-time detection of anomalies as they occur. It is often used in real-time monitoring systems where quick response is essential to prevent undesirable events. Offline analysis, on the other hand, focuses on the retrospective examination of data to identify anomalies after the fact. It is generally used for post-hoc analysis, understanding the root causes of anomalies and improving online detection systems. In this article, we address offline analysis, considering that we are in an exploratory analysis scenario (the detection models are not deployed subsequently). Therefore, in the experimental section, all data will be used for training, and the results will also be presented in a training context.

2.3 Cross-knowledge

When it comes to analyzing multiple time series, pooling knowledge allows the extraction of richer information compared to treating each time series individually. By combining different series, it is possible to detect trends, correlations, or patterns that would not be visible when analyzing each time series separately. This approach provides a deeper understanding of the interactions and dynamics between different time series. By exploiting this cross-referenced information, it becomes possible to improve forecast accuracy and identify anomalies or unusual events. However, as mentioned in [39]: “each time series must be considered as totally independent of the others, unless we know the interactions or clearly the application domain in which the time series are generated”. In this article, however, we are as agnostic as possible. As a result, we will be studying time series one by one, without combining information. Each time series thus becomes a “dataset” in its own right, on which a detection method can be applied and performance results collected.

2.4 Typology of Approaches

The literature considers three main families of anomaly detection approaches [6]:

- (i) **supervised**, where each time series, or a portion of it, is labeled as either normal or anomalous.
- (ii) **semi-supervised**, where it is assumed that the beginning of the time series contains no anomalies; in this case, the model is trained on the normal data only and then deployed on the rest of the time series for anomaly detection.
- (iii) **unsupervised**, where no assumption is made: anomalies can occur at any point in the time series and and no labels (normal/anomalous) are available during training.

In this article, consistent with the idea of being agnostic and performing cold analysis, we place ourselves in the third case.

2.5 Detection Methods: Time-Series vs. Tabular

There are numerous anomaly detection methods for univariate time series [7,18] and just as many for tabular data [11]. It is interesting to note that these two fields share certain methods, ranging from the simplest (statistical methods, for example) to the most elaborate. Indeed, methods designed for tabular data often perform very well on temporal data. Examples of these include Isolation Forest (IF) [26] and Local Outlier Factor (LOF) [9]. In the oral presentation made by Boniol in [6], the interested reader will find a comparison of a fairly large number of methods (in particular slides 125 to 130), showing the very good positioning of IF and LOF on temporal data. Given the objective of the study presented in this article, and the fact that both methods work well in both the temporal and tabular domains, both will be the methods we use in the remainder of our comparative study.

2.6 Time Series Windowing

The final concept we need to introduce is “sliding windows” (and the associated parameters), which allows methods initially dedicated to tabular data to be used on temporal data. For a given time series (in which we wish to detect anomalies), the method consists in “slicing” it into a succession of F windows (of size W). These windows are then organized into a table of W columns and F rows, thus turning the time series into a table. The objective is then to identify which rows of this table contain anomalies.

The window passed over the time series can be either “jumping” (also known as non-overlapping), or “sliding” (overlapping). In the sliding case, an additional parameter is introduced: the shift step α which determines the degree of overlap between consecutive windows (jumping windows can be thought of as sliding windows with $\alpha = W$, i.e. no overlap). A fairly careful review of the literature [8,16,17,23,25,34,35] shows that most methods, unless they incorporate feedback from an expert, use a sliding window and $\alpha = 1$. This will also be our case in the remainder of this article.

The remaining challenge is to determine the window size (W). This point is quite crucial, yet surprisingly under-explored in the literature. Most studies set the value of W through preliminary cross-validation experiments. To our knowledge, only one paper has reasonably addressed the question of automatically determining W in the case of “periodic” time series [13], in which the authors test several methods for detecting “seasonality/periodicity”, within a semi-supervised framework. In our case, we will not make the assumption that the time series studied are periodic. The experimental section of this article will therefore test several values of W with increasing orders of magnitude.

2.7 Feature Extraction from Time Series

The approach involves transforming data from the temporal world to the tabular world using libraries that automatically extract features from time series. The features are diverse, allowing to capture different properties of the time series, such as seasonality, trends or auto-correlation, and can therefore be adapted to different application domains. This transformation thus captures the essence of time series data, while making it compatible with more traditional tabular data analysis techniques.

Stimulated by the development of the library HCTSA [15], several unsupervised feature engineering tools have been developed independently in different programming languages: for example, CATCH22 [27], FEATURETOOLS [22], TSFRESH [12], TSFEL [2], TSFEATURES [21], or FEASTS [29]. An extensive study presented in [32] (for the classification task) but also those presented in [36, 40] (for the anomaly detection task) show that TSFRESH [12] is among the best performers. It is therefore our selected feature extraction tool. The proposed approach could nonetheless use other tools, or even combine them (for example using a stacking method).

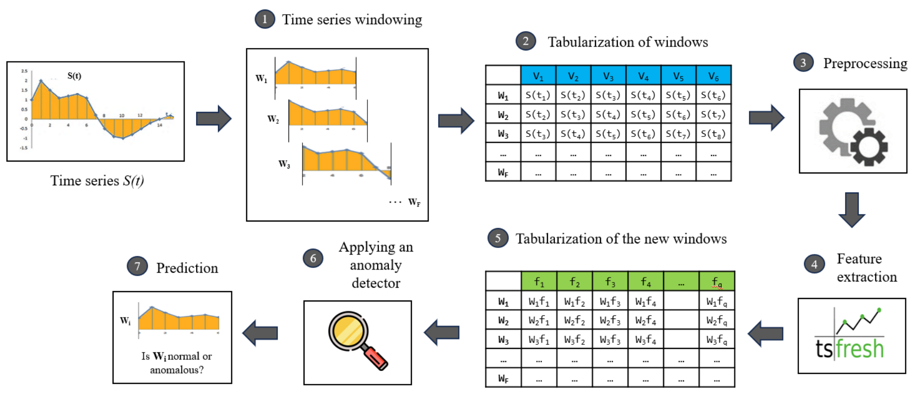
1 Time series windowing
2 Tabularization of windows
3 Preprocessing
4 Feature extraction
5 Tabularization of the new windows
6 Applying an anomaly detector
7 Prediction
Time series S(t)
Is Wi normal or anomalous?
tsfresh

Fig. 1. Proposed processing pipeline.

TSFRESH [3] (Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests) is a Python library that calculates up to 800 features (basic statistics, autocorrelations, entropy, Fourier coefficients, etc.) from a time series by combining 63 time series characterization methods [12]. It offers three pre-defined feature dictionaries ranging in size from 10 to almost 800 features. The library also enables feature selection based on statistical tests. This process can be parallelized, to reduce computation times. In this paper, we used the maximum number of features, as feature selection is only possible in a supervised framework.

3 Proposed Processing Pipeline

Based on all the concepts presented previously in this paper, the processing pipeline proposed for our comparative study is presented in figure 1. It consists of seven steps and is applied to each time series:

- Step 1: divide the series into windows ($W = 6$ for illustration purposes in figure 1);
- Step 2: put the obtained F windows, of size W, (W_1 to W_F) in a tabular data base;
- Step 3: preprocess data;
- Step 4: feature extraction using the TSFRESH library;
- Step 5: put the F obtained windows (W_1 to W_F), described by the q features calculated using TSFRESH, in a tabular data base;
- Step 6: apply an anomaly detector (IF or LOF);
- Step 7: obtain the anomaly prediction for each window.

The experimental study described in the following section will compare the results obtained with and without steps 4 and 5, to answer the research question raised in the introduction of this paper.

[3] `https://tsfresh.readthedocs.io/en/latest/index.html`

Table 1. Summary of information on the datasets used.

Name	#series	Length		Freq (Hz)	% anomalies		Domains
		Min	Max		Min	Max	
SVDB	76	230400	230400	0.008	0.34	45.54	ECG
NAB	46	1127	22695	variable	8.30	10.29	servers, tweets, traffic, advertising
AIOPS	13	16441	295414	[0.003-1.7]	0.06	7.50	performance indicators
NormA	5	2000	35040	unknown	3.08	9.13	aerospace, health, body language, electricity
UCR	247	6674	300262	diverse	0.0005	4.9	medical, meteorology, biology, industry

4 Experimental Protocol

This section describes the choices made during the experiments: user-defined parameters, sliding window size (W), etc. All experiments carried out can be reproduced using the code in [19]. Some preliminary tests, which will be briefly discussed below, are not fully presented in this article but interested readers can nevertheless find them in the "Additional material" file available in the GitHub mentioned above [19].

4.1 Datasets

Anomaly detection is currently the subject of a great deal of research, and numerous benchmark datasets have been established to enable comparing the performance of new algorithms to the state of the art. Among these, we have selected a number of datasets described below. In some cases, we did not retain all time series from these datasets. The reasons for excluding certain time series are given, and the exact list of time series (identifiers) used for each dataset is available in [19].

SVDB (MIT-BIH Supraventricular Arrhythmia Database) (available at [37]): this dataset initially contains 78 time series. However, two series have been discarded due to an extremely high contamination rate (CT) (>50%) which unrealistic.

NAB (Numenta Anomaly Benchmark) (available at [37]): created by Numenta in 2015, it is currently the second most widely used benchmark in the literature [39]. It consists of 58 time series divided into seven groups, two of which contain synthetic data only. We chose not to use these synthetic data, which correspond to 11 time series. The remaining five groups cover a variety of topics [4]). From these five groups, we removed one time series that contains no anomalies. In total, we used 46 time series from this dataset.

AIOPS 2018[5]: this dataset was created in 2018 for the AIOps challenge. 29 time series were collected from various companies such as Sogou, Tencent or eBay. They correspond to performance indicators that reflect the scale, quality of web services and health of a machine as explained in [31]. The data we used comes from the benchmark dataset [30]. Additionally, it is worth noting that

[4] more information at: `https://github.com/numenta/NAB/tree/master/data`
[5] Available at: `https://github.com/TheDatumOrg/TSB-UAD`

we have combined the training and testing parts for each time series. After analyzing the correlations among the 29 time series, we noticed that some of them are correlated, which could potentially introduce bias in the statistical tests conducted during the analysis of the results. Therefore, we only keep the uncorrelated series ($r < 0.3$), ultimately leading us to select 13 series for this dataset.

NormA (available at [37]): this dataset consists of 21 time series, of which 14 are synthetic and have been discarded. The seven real series can be grouped into four categories. Among the real-world time series, the first three are highly correlated, and we will thus only keep one of the three. In total, we thus keep five time series only for this dataset.

UCR Anomaly Archive[6] : The UCR Anomaly Archive dataset was released in 2020 as an alternative to several benchmarks deemed to be deficient by [39]. For this dataset, we chose to only exclude three series that are significantly longer than the others (due to computational time constraints). We thus used a total of 247 time series from this dataset.

To sum up, our five datasets[7] together represent a wide diversity of problems, anomaly rates and issues (see Table 1). When presenting the results, the AIOPS and NormA datasets, which contain a small number of time series, will be merged in order to perform statistical tests, such as the Wilcoxon test [38], which requires a minimum number of values to be used.

4.2 Size of Sliding Windows

As mentioned above, this point is quite crucial but surprisingly little discussed in the literature. To the best of our knowledge, only one paper has reasonably addressed the question of automatically adjusting the value of W in the case of "periodic" time series [13]. In our case, where we aim to remain agnostic and without feedback loops, we decided to simply test four window sizes corresponding to values observed in the literature. We started with a small size $W = 32$ and then doubled its size, hoping to increase the amount of information captured, several times. In the end, the tested values for W are 32, 64, 128, and 256. The aim of the experiments will not be to determine the optimal window size but to confirm that the conclusions do not depend on a very specific window size.

4.3 Hyperparameter Tuning

The TSFRESH library [12] (acronym for *Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests*) can be used to create varying numbers of

[6] Available at: `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`

[7] Norma and NAB have sometimes described having mislabeled ground truth, trivial or overestimated success by having repeated anomalies. But in our case, as mentioned above, by removing synthetic data, correlated time series, and manual inspection we have done our best to avoid these aspects for these two datasets (see [19] to have the exact identifiers of the retained time series).

features (the value of q mentioned in section 3). All the tests we conducted showed better results by pushing the feature extraction process to its limits, thus using a large number of features calculated by the library (i.e. its 'Efficient' version and $q = 777$ features). We point out that sometimes, for example with $W = 32$, it is not possible for TSFRESH to calculate all the requested features. In such cases, we removed the impacted columns (with missing values) from the table produced in step 5 of the pipeline described in section 3.

Two anomaly detectors for tabular data that have shown their worth on time series are used: Isolation Forest (IF) and Local Outlier Factor (LOF). In all the experiments carried out below, we used the implementation provided in the PyOD[8] library (version 1.1.3) with their default settings.

4.4 Results Evaluation Criterion

The literature offers numerous evaluation metrics, such as accuracy, F1 score, the "PA%K" criterion [24], the AUC (*Area Under the receiver operating characteristics Curve*) [14]. In our case, we decided not to use a criterion that requires a threshold value, as we place ourselves in an agnostic framework where there is no feedback loop with a domain expert. Indeed, this threshold can be a difficult parameter to select and/or may require starting from a heuristic on the distribution of anomaly scores. For this reason, we selected the AUC as our evaluation criterion.

The reader will note that in this case, but also for other potential criteria, ground truth labels are necessary to allow the evaluation of the predictions obtained in step 7 of our processing pipeline. For the predicted class, this information is given by the detection method used, here IF or LOF. For the class to be predicted (ground truth), we operate as follows (which corresponds to the observed usage): during the segmentation of the time series into windows (step 1), if an anomaly exists in the window, then the entire window is labeled as anomalous, and vice versa in the opposite case. Thus, it is possible to compare for each window the class predicted by a confidence score delivered by IF or LOF with the ground truth. The set of comparisons allows us to calculate the AUC for each time series (individually) from the datasets used. The collection of these AUCs will also enable us to conduct statistical tests and present results in terms of ranking or critical diagrams.

4.5 Preprocessing

Preliminary tests were carried out in order to analyze the benefits from "horizontally normalizing" the values contained in the sliding windows before applying the two approaches: 1) anomaly detection using the initial representation (called "TS" in Table 2) and 2) anomaly detection using the new feature representation calculated by TSFRESH (called "FE" in Table 2). To this end, a comparative study was carried out between "doing nothing" (No Normalization) and 3 usual

[8] `https://pyod.readthedocs.io/en/latest/`

Table 2. Comparison of the mean ranks of normalization approaches by window size (combining IF and LOF results).

			Without Normalization	MinMax	Median-IQR	MeanStd
SVDB	32	TS	2.250	2.036	3.063	2.651
		FE	1.474	2.250	3.135	3.141
	64	TS	2.211	2.227	2.908	2.655
		FE	1.750	2.260	2.934	3.056
	128	TS	2.174	2.243	2.609	2.974
		FE	2.049	2.418	2.474	3.059
	256	TS	2.079	2.638	2.424	2.859
		FE	2.260	2.563	2.230	2.947
AIOPS + NormA	32	TS	1.806	2.583	2.472	3.139
		FE	2.236	2.708	2.750	2.306
	64	TS	1.903	2.431	2.722	2.944
		FE	2.458	2.722	2.625	2.194
	128	TS	1.722	2.583	2.667	3.028
		FE	2.500	2.722	2.417	2.361
	256	TS	1.736	2.944	2.319	3.000
		FE	2.250	2.944	2.250	2.556

normalization methods, namely (i) Min-Max, (ii) Median-IQR and (iii) Mean-Standard Deviation.

This study is conducted based on (i) three of the datasets described above: SVDB, NormA and AIOPS among the five selected and (ii) the TSFRESH library parameter set to 'minimal' ($q = 10$). Since the NormA dataset only consists of five time series, its results are combined with those of AIOPS, as mentioned earlier.

The results presented in Table 2 show the mean rank of these four normalization techniques (1 being the best, 4 the worst), aggregated regardless of the detection method (IF or LOF) and versus the window size. We see that normalization approaches do not improve performance. The absence of normalization exhibits the best mean rank in most experiments, and thus yields the best results. Consequently, in the next section, only the results obtained with no normalization are presented. A more in-depth analysis of this study found in the supplementary material [19], according to the "detection method" axis (IF or LOF), further confirms this result.

5 Results

5.1 Performance Gains from the Extracted Features for Each Detector

Table 3 reports the obtained results: with (FE) or without (TS) the use of TSFRESH in the proposed processing pipeline; with the Isolation Forest (IF) or

Table 3. Comparison of detectors, by dataset, according to the window size used.

		Isolation Forest			Local Outlier Factor		
		TS	FE	p-value	TS	FE	p-value
SVDB	32	1.882	1.118	$\mathbf{1.522\times10^{-12}}$	1.329	1.671	$\mathbf{2.066\times10^{-3}}$
	64	1.855	1.145	$\mathbf{5.375\times10^{-9}}$	1.480	1.520	4.454×10^{-1}
	128	1.842	1.158	$\mathbf{5.553\times10^{-10}}$	1.750	1.250	$\mathbf{2.490\times10^{-6}}$
	256	1.592	1.408	1.161×10^{-1}	1.684	1.316	$\mathbf{5.541\times10^{-6}}$
AIOPS + NormA	32	1.889	1.111	$\mathbf{1.930\times10^{-3}}$	1.556	1.444	8.317×10^{-1}
	64	1.778	1.222	$\mathbf{1.930\times10^{-3}}$	1.556	1.444	7.660×10^{-1}
	128	1.778	1.222	$\mathbf{1.930\times10^{-3}}$	1.417	1.583	4.925×10^{-1}
	256	1.833	1.167	$\mathbf{4.745\times10^{-3}}$	1.389	1.611	1.674×10^{-1}
NAB	32	1.652	1.348	5.059×10^{-2}	1.261	1.739	$\mathbf{8.504\times10^{-4}}$
	64	1.630	1.370	$\mathbf{1.727\times10^{-2}}$	1.261	1.739	$\mathbf{7.430\times10^{-5}}$
	128	1.674	1.326	$\mathbf{5.525\times10^{-3}}$	1.217	1.783	$\mathbf{1.261\times10^{-4}}$
	256	1.641	1.359	$\mathbf{1.723\times10^{-2}}$	1.283	1.717	$\mathbf{3.163\times10^{-3}}$
UCR	32	1.826	1.174	$\mathbf{2.581\times10^{-29}}$	1.156	1.844	$\mathbf{5.159\times10^{-22}}$
	64	1.844	1.156	$\mathbf{9.264\times10^{-32}}$	1.170	1.830	$\mathbf{7.227\times10^{-26}}$
	128	1.848	1.152	$\mathbf{1.792\times10^{-32}}$	1.223	1.777	$\mathbf{2.247\times10^{-20}}$
	256	1.796	1.204	$\mathbf{8.983\times10^{-27}}$	1.243	1.757	$\mathbf{1.055\times10^{-19}}$

Local Outlier Factor (LOF) detectors. These results are detailed by dataset and window size. The presented values correspond to the mean ranks obtained over all time series of each dataset. The p-values are derived from the Wilcoxon test [38]: a bold value indicates a statistically significant difference in performance between the pair of detectors.

The results are clear: (i) for Isolation Forest, across 16 results (4 datasets x 4 window sizes), the proposed FE method (using TSFRESH) always obtains the best mean rank with 14/16 cases where the difference is statistically significant (ii) for Local Outlier Detection, the proposed FE method obtains the best mean rank 4 times, of which only 2 are significantly better. The proposed processing chain benefits IF, which is tree-based (thus relying on value ranks), but does not benefits LOF, which is density-based (thus relying on distance calculations).

It is likely that for LOF, the size of the vector produced by TSFRESH leads to a dimensional problem in the distance calculations used by this method, whereas IF does not suffer from this problem. The performance of LOF nevertheless raises questions. This is a model that requires more attention than Isolation Forest (based on rank statistics), which needed neither normalization nor optimization of its hyperparameters. It is also conceivable that adjusting the value of k in LOF could provide an improvement, but in the case without a feedback loop with a domain expert, we have not explored this possibility. These points will be explored in a future work. A normalization of the extracted features (vertically) was also evaluated (as an additional step between the step 5 and 6 in Figure 1) the results indicate that: 1) for UCR and (AIOPS + NormA) datasets, LOF-FE nearly always improved its performance but it was not enough to surpass LOF-

TS, 2) for the NAB dataset, LOF-FE consistently improved its performance and surpassed LOF-TS and 3) for SVDB, a decline in the performance of LOF-FE was observed. Our conclusion remains therefore unchanged.

5.2 Comparison of detectors

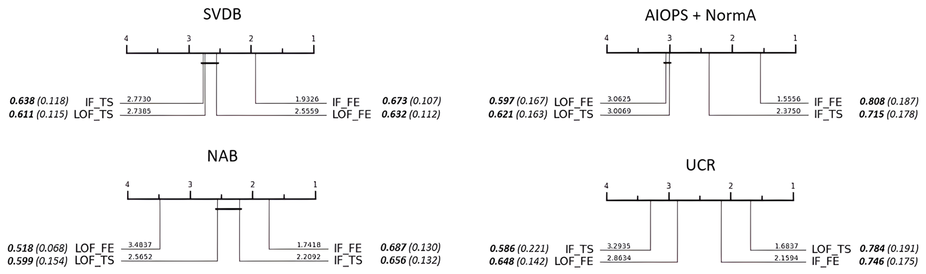

Fig. 2. Critical diagrams for each dataset, averaged over the four window sizes.

Figure 2 shows the critical diagram (CD) of methods (i) per dataset, then (ii) with or without the use of calculated features and (iii) averaged over all window sizes. In each sub-figure and for each bar of the critical diagram, we find in this order: the mean rank of the method, the method name (IF or LOF) prefixed with TS or FE (respectively for initial Time Series or FE for Feature Engineering via TSFRESH), then the corresponding mean AUC value with finally the standard deviation of the AUC in brackets. Interested readers will find further results with other axes of analysis in the supplementary material [19].

Once again, we note that, in general, Isolation Forest benefits most from the features calculated using TSFRESH, ranking first on 3 of the 4 datasets. For LOF, the results are much more contrasted: the creation of features brings little or no improvement compared to the initial temporal representation. These results confirm those presented in Table 3. Finally, LOF applied to temporal data achieves the best results on the UCR dataset. This dataset has the particularity of containing very few anomalies (see Section 4.1), with anomalies appearing only in the second part of each time series. This has a remarkable impact on IF which dramatically improves its performance using features of TSFRESH: AUC moves from 0.586 to 0.746, i.e. +36%.

6 Conclusion

In the context of anomaly detection within a time series, this paper proposed a processing pipeline which first transforms time series from the temporal domain to an alternative tabular representation space, and then apply anomaly detectors dedicated to tabular data. Our motivation is to consider the feature extraction process as a source of knowledge and information useful for detection. The central question was whether better results could be obtained in the computed feature space compared to the initial temporal representation? Through extensive experiments across five datasets and with two detectors (IF and LOF), we observed that the results significantly improved for IF, but not for LOF. Future work will focus on (i) extending the number of detectors in the comparison, (ii) testing other feature extraction libraries (or combining them), and (iii) considering tuning the window size and/or considering not a sliding window but a "jumping" window (with no overlapping between consecutive windows), even though this is not inherent to the proposed processing chain.

Acknowledgments. The authors would like to thank the reviewers for their insightful feedback and suggestions.

Disclosure of Interests. The authors declare no competing interests relevant to the content of this article.

References

1. Badi, O., Devanne, M., Ismail-Fawaz, A., Abdullayev, J., Lemaire, V., Berretti, S., Weber, J., Forestier, G.: Cocalite: A hybrid model combining catch22 and lite for time series classification. In: IEEE International Conference on Big Data (2024)
2. Barandas, M., Folgado, D., Fernandes, L., Santos, S., Abreu, M., Bota, P., Liu, H., Schultz, T., Gamboa, H.: Tsfel: Time series feature extraction library. SoftwareX **11**, 100456 (2020)
3. Blázquez-García, A., Conde, A., Mori, U., Lozano, J.A.: A review on outlier/anomaly detection in time series data. ACM computing surveys (CSUR) **54**(3), 1–33 (2021)
4. Bondu, A., Achenchabe, Y., Bifet, A., Clerot, F., Cornuejols, A., Gama, J., Hebrail, G., Lemaire, V., Marteau, P.F.: Open challenges for machine learning based early decision-making research. SIGKDD Explor. Newsl. **24**(2), 12–31 (dec 2022)
5. Boniol, P., Liu, Q., Huang, M., Palpana, T., Paparrizos, J.: Dive into time-series anomaly detection: A decade review. ArXiv **abs/2412.20512** (December 2024), `https://arxiv.org/pdf/2412.20512`
6. Boniol, P., Paparrizos, J., Palpanas, T.: Tutorial: New trends in time series anomaly detection. In: International Conference on Extending Database Technology (2023), [Slides ...] [Video ...]
7. Braei, M., Wagner, S.: Anomaly detection in univariate time-series: A survey on the state-of-the-art. ArXiv **abs/2004.00433** (2020), `https://api.semanticscholar.org/CorpusID:214743362`
8. Braei, M., Wagner, S.: Anomaly detection in univariate time-series: A survey on the state-of-the-art. arXiv preprint arXiv:2004.00433 (2020)
9. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. SIGMOD Rec. **29**(2), 93–104 (2000)
10. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM computing surveys (CSUR) **41**(3), 1–58 (2009)
11. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. **41**(3) (jul 2009)

12. Christ, M., Braun, N., Neuffer, J., Kempa-Liehr, A.W.: Time series feature extraction on basis of scalable hypothesis tests (tsfresh - A python package). Neurocomputing **307**, 72–77 (2018)
13. Ermshaus, A., Schäfer, P., Leser, U.: Window size selection in unsupervised time series analytics: A review and benchmark. In: ECML PKDD Workshop, AALTD. p. 83–101. Springer-Verlag (2023). `https://doi.org/10.1007/978-3-031-24378-3_6`
14. Fawcett, T.: An introduction to roc analysis. Pattern Recognition Letters **27**(8) (2006)
15. Fulcher, B.D., Jones, N.S.: hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. Cell systems **5**(5), 527–531 (2017)
16. Geiger, A., Liu, D., Alnegheimish, S., Cuesta-Infante, A., Veeramachaneni, K.: Tadgan: Time series anomaly detection using generative adversarial networks. In: 2020 ieee international conference on big data (big data). pp. 33–43. IEEE (2020)
17. Golmohammadi, K., Zaiane, O.R.: Time series contextual anomaly detection for detecting market manipulation in stock market. In: 2015 IEEE international conference on data science and advanced analytics (DSAA). pp. 1–10. IEEE (2015)
18. Gupta, M., Gao, J., Aggarwal, C.C., Han, J.: Outlier detection for temporal data: A survey. IEEE Transactions on Knowledge and Data Engineering **26**(9), 2250–2267 (2014)
19. Hamon, M.: Supplementary material of the paper: "Unsupervised feature construction for anomaly detection anomalies in time series - an evaluation" (2024), [Github ...] [Supplementary Material (pdf)...]
20. Han, S., Hu, X., Huang, H., Jiang, M., Zhao, Y.: Adbench: Anomaly detection benchmark. Advances in Neural Information Processing Systems **35**, 32142–32159 (2022)
21. Hyndman, R., Kang, Y., Montero-Manso, P., O'Hara-Wild, M., Talagala, T., Wang, E., Yang, Y.: tsfeatures: Time Series Feature Extraction (2023), https://pkg.robjhyndman.com/tsfeatures/, https://github.com/robjhyndman/tsfeatures
22. Kanter, J.M., Veeramachaneni, K.: Deep feature synthesis: Towards automating data science endeavors. In: 2015 IEEE int. conf. on Data Science and Advanced Analytics (DSAA). pp. 1–10. IEEE (2015)
23. Kapourchali, M.H., Banerjee, B.: Unsupervised feature learning from time-series data using linear models. IEEE Internet of Things Journal **5**(5), 3918–3926 (2018)
24. Kim, S., Choi, K., Choi, H.S., Lee, B., Yoon, S.: Towards a rigorous evaluation of time-series anomaly detection. In: AAAI Conference on Artificial Intelligence (2021), `https://api.semanticscholar.org/CorpusID:237490301`
25. Kim, S., Choi, K., Choi, H.S., Lee, B., Yoon, S.: Towards a rigorous evaluation of time-series anomaly detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 7194–7201 (2022)
26. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 IEEE International Conference on Data Mining (ICDM). pp. 413–422 (2008)
27. Lubba, C.H., Sethi, S.S., Knaute, P., Schultz, S.R., Fulcher, B.D., Jones, N.S.: catch22: Canonical time-series characteristics. Data Mining and Knowledge Discovery **33**(6), 1821–1852 (2019)
28. Middlehurst, M., Bagnall, A.: The freshprince: A simple transformation based pipeline time series classifier. In: El Yacoubi, M., Granger, E., Yuen, P.C., Pal, U., Vincent, N. (eds.) Pattern Recognition and Artificial Intelligence. pp. 150–161. Springer International Publishing (2022)

29. O'Hara-Wild, M., Hyndman, R., Wang, E., Cook, D., Talagala, T., Chhay, L., O'Hara-Wild, M.M.: Package 'feasts'. CRAN, Junio (2020)
30. Paparrizos, J., Kang, Y., Boniol, P., Tsay, R.S., Palpanas, T., Franklin, M.J.: Tsb-uad: an end-to-end benchmark suite for univariate time-series anomaly detection. Proceedings of the VLDB Endowment **15**(8), 1697–1711 (2022)
31. Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J., Zhang, Q.: Time-series anomaly detection service at microsoft. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 3009–3017 (2019)
32. Renault, A., Bondu, A., Lemaire, V., Gay, D.: Automatic feature engineering for time series classification: Evaluation and discussion. In: 2023 International Joint Conference on Neural Networks (IJCNN). pp. 1–10. IEEE (2023)
33. Samariya, D., Thakkar, A.: A comprehensive survey of anomaly detection algorithms. Annals of Data Science **10**(3), 829–850 (2023)
34. Schmidl, S., Wenig, P., Papenbrock, T.: Anomaly detection in time series: A comprehensive evaluation. Proceedings of the VLDB Endowment **15**(9), 1779–1797 (2022)
35. Shawe-Taylor, J., Žličar, B.: Novelty detection with one-class support vector machines. In: Advances in Statistical Models for Data Analysis. pp. 231–257. Springer (2015)
36. Teh, H.Y., Kevin, I., Wang, K., Kempa-Liehr, A.W.: Expect the unexpected: unsupervised feature selection for automated sensor anomaly detection. IEEE Sensors Journal **21**(16), 18033–18046 (2021)
37. Wenig, P., Schmidl, S., Papenbrock, T.: TimeEval datasets github page (2024), `https://timeeval.github.io/evaluation-paper/notebooks/Datasets.html`
38. Wilcoxon, F.: Individual Comparisons by Ranking Methods, pp. 196–202 (1992)
39. Wu, R., Keogh, E.J.: Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. IEEE transactions on knowledge and data engineering **35**(3), 2421–2429 (2021)
40. Zhang, W., Dong, X., Li, H., Xu, J., Wang, D.: Unsupervised detection of abnormal electricity consumption behavior based on feature engineering. IEEE Access **8**, 55483–55500 (2020)