IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 July 2022, accepted 25 August 2022, date of publication 1 September 2022, date of current version 16 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3203523

APPLIED RESEARCH

Variable-Length Multivariate Time Series Classification Using ROCKET: A Case Study of Incident Detection

AGNIESZKA BIER[1,2], AGNIESZKA JASTRZĘBSKA[1,3], AND PAWEŁ OLSZEWSKI[1]
[1]DTiQ Poland Sp. z o. o, 44-100 Gliwice, Poland
[2]Faculty of Applied Mathematics, Silesian University of Technology, 44-100 Gliwice, Poland
[3]Faculty of Mathematics and Information Science, Warsaw University of Technology, 00-662 Warsaw, Poland

Corresponding authors: Agnieszka Jastrzębska (agnieszka.jastrzebska.mini@pw.edu.pl) and Agnieszka Bier (agnieszka.bier@polsl.pl)

This work was supported by the European Regional Development Fund, Intelligent Development Operational Program for 2014-2020, sub-measure 1.1.1 ''Industrial research and development work implemented by enterprises'' for the implementation of the project POIR.01.01.01-00-1141/19 ''Development of intelligent loss prevention system and monitoring the quality of the quick service restaurants chain.''

ABSTRACT Multivariate time series classification is a machine learning problem that can be applied to automate a wide range of real-world data analysis tasks. RandOm Convolutional KErnel Transform (ROCKET) proved to be an outstanding algorithm capable to classify time series accurately and quickly. The textbook variant of the multivariate time series classification problem assumes that time series to be classified are all of the same length, while in real-world applications this assumption not necessarily holds. The literature of this domain does not pay enough attention to data processing pipelines for variable-length time series. Thus, in this paper, we present a thorough analysis of three preprocessing pipelines that handle variable-length time series that need to be classified with a method that requires the data to be of equal length. These three methods are truncation, padding, and forecasting of missing value. Experiments conducted on benchmark datasets, showed that the recommended procedure involves padding. Forecasting ensures similar classification accuracy, but comes at a much higher computational cost. Truncation is not a viable option. Furthermore, in the paper, we present a novel domain of application of multivariate time series classification algorithms, that is incident detection in cash transactions. This area poses substantive challenges for automated model training procedures since the data is not only variable-length, but also heavily imbalanced. In the study, we list various incident types and present trained classifiers capable to aid human auditors in their daily work.

INDEX TERMS Classification, incident detection, multivariate time series, ROCKET, varying-length time series.

I. INTRODUCTION

Time series classification has become a vital domain of machine learning. The multitude of exciting real-life applications drives the development of the field and inspires fruitful research that aim at delivering new approaches, improving the existing ones, and adapting them to new types of data. This paper focuses on the task of classifying multivariate time series of unequal lengths. That is, each multivariate series can have a different length.

The study presented in this paper is related to our project that deals with *incident detection in cash transaction data*. The business context of the project is *loss prevention in quick-service restaurant* operations. In our context, an incident is

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

when a cashier (server, employee) is under-ringing items, voiding items, or performing other intentional operations that cause money losses for the restaurant.

Thanks to many years of successful cooperation with our clients in the quick-service restaurant industry, we have accumulated an adequate database of transactions, and we rely on human experts to conduct incident audits. The human-auditor-based loss prevention scheme employed at the moment is the cornerstone of the daily operations of our company. However, there is a pressing need to automate audit tasks.

We want to build an automated classifier that detects operations in which a cashier intentionally mishandles the transaction process. Our goal is to deliver a custom classification model that will assign one of two possible labels: fraudulent operation or legitimate operation to each transaction committed in the system. We envision the incident recognition module as an offline audit tool that will analyze data on past restaurant operations. It becomes apparent that the problem at hand is a binary classification task, which may appear trivial to the reader at first glance. However, the nuances of the domain make this task far from easy.

In this paper, we present fundamental challenges and solutions that we have encountered and worked on while constructing one particular variant of our incident detector. We must emphasize that this study is unique and novel. As we will detail in the literature review on fraud detection in cash transactions, practitioners are well aware of the magnitude of the problems. One can find studies that list a wide range of frauds that can take place. Unfortunately, the literature does not offer a satisfactory range of solutions to automate the detection of fraud in cash transactions.

The construction of an automated machine learning model for cash transactions fraud detection requires a noticeable technological advancement in the computer system that handles daily operations on the client side. Without surprise, there are not that many solutions of this kind available on the market. Such tools are typically built as an additional feature of a broader system that handles daily operations. Nonetheless, some commercial applications offer such functionalities. Naturally, our company, DTIQ, offers such a service. Next, we may mention NCR Solutions,[1] whose store management application includes alerts about detected suspicious transactions (for example, suspicious refunds).

In order to build such a system, one must have access to data concerning payments. On that note, based on our experience, the following data-related challenges take place:

- The first essential obstacle is a heavy imbalance. In a representative chunk of transactions data only around 5–10% are deemed to be fraudulent.
- There are many "creative" ways of how to manipulate a cash-based transaction in a restaurant, or generally speaking, in retail. There is no one repetitive scenario taken by a dishonest cashier, tactics change in time.

[1]https://www.ncr-hospitality.com/en/solutions/theft-detection-20/

- Data is available in a non-typical structure and it is very varied. We envision that for a given cashier, for a given date we have a shift transactions history and (any) one or more transactions this time span may be fraudulent. The length of the history of transactions may be substantially different, ranging from less than 10 up to even 500 transactions per shift. Transactions history can concern operations of various types such as cash payments, card payments, discounts, technical procedures, etc.
- We envision that the data can be represented as multivariate time series made of a sequence of transactions where the variables describe "parameters" of transactions such as purchased items quantity and the variety of products (foods, drinks, condiments, etc.), gross value, discount, cash paid and change required. There are many variables, and time series may be very short. Length can be shorter than the number of variables.

Summarizing the concerns raised above, let us state that we deal with multivariate time series classification task but with highly challenging data. These data characteristics are highly dissimilar to the characteristics of time series from benchmark dataset repositories. In the case of time series classification task, the very often used resource is https://timeseriesclassification.com/. However, on this page, there are no time series datasets with properties similar to ours.

Let us emphasize that the output of the multivariate time series classification area is rich with compelling ideas and algorithms. We can briefly distinguish the following groups of methods:

- classifying ensembles, where weak learners are built for univariate series and multiple variables are handled at a sampling stage when weak learners are built;
- classification pipelines composed of two steps: feature extraction and standard classification, where feature extraction is performed for all variables in time series to handle the issue of multiple variables;
- neural networks, which integrate feature extraction and classification step in a single algorithm and are suitable for processing multivariate sets.

Recently, Ruiz *et al.* [32] published a comprehensive review of multivariate time series classification methods. Ruiz *et al.* present a systematic and very thorough comparison of the existing methods. The study covered experiments with aforementioned benchmark datasets on the https://timeseriesclassification.com/ website. Their results clearly show that the algorithm called RandOm Convolutional KErnel Transform, ROCKET, authored by Dempster *et al.* [9], is the best-performing one for multivariate time series classification. ROCKET uses random convolutional kernels to extract features and then logistic regression to execute the classification step. The authors conclude their summary saying that *"ROCKET is the best ranked and by far the fastest classifier and would be our recommendation as the default choice for MTSC problems."* [32].

Importantly, Ruiz *et al.* compared ROCKET with very competitive algorithms, including several neural network-based approaches (such as ResNet and InceptionTime). We must mention that the superiority of ROCKET was also reported in independent works for other (not necessarily multivariate) time series classification tasks, including the studies delivered by Dempster *et al.* [9], [10], [11], Salehinejad *et al.* [33], Pantiskas *et al.* [30], and more. Among papers emphasizing the superiority of ROCKET is the work of Dhariyal *et al.* [8] who focused on the comparisons of ROCKET with neural approaches. The authors conclude their findings saying that *"recent deep learning MTSC methods do not perform as well as expected."*.

The mentioned-above findings motivated us to investigate the feasibility of applying the ROCKET algorithm for incident detection. In this paper, we report on the outcomes of these efforts. It must be stressed, that the work we performed aimed at providing a novel adaptation of the ROCKET procedure. ROCKET is an algorithm suitable for same-length time series. The default strategy of how to use it with time series of varying lengths is to do padding. In our experiments, we have tested several strategies that allow to apply the feature extraction step the same as in ROCKET but for multivariate time series with varying lengths. In particular, we have tested the following techniques:

- padding – the baseline strategy which relies on placing a constant value to make time series lengths even,
- trimming – cutting time series,
- forecasting future values using an ARIMA model.

A. THE NOVELTY OF THE CONTRIBUTION

Let us briefly summarize the novel contributions addressed in this paper:

- We present a comparative study on how various approaches to varying-length time series length equalization contribute to the classification accuracy of ROCKET.
- We introduce and analyze the feasibility of ARIMA-based forecasting used for time series length manipulation in varying-length time series forecasting problem.
- We present the application of multivariate time series classification to a new domain, cash transaction fraud detection.

B. THE PRACTICAL SIGNIFICANCE OF THE UNDERTAKEN TOPIC

Fraud detection in retail and services can be positioned under the umbrella of loss prevention mechanisms crucial for the daily operations of businesses. There are manifold aspects of this problem spanning from physical monitoring of locations, video analytics, to transaction data monitoring. In this paper, we narrow down the attention to the analysis of information stored in a conventional relational database concerning transactions. *We demonstrate how one can leverage this resource for fraud detection.*

The remainder of this manuscript is structured as follows. Section II presents a literature review on the topic of multivariate time series classification. Section III mentions existing studies on a wide context of the incident detection task. Section IV introduces the issue of variable-length time series. Section V presents empirical experiments we have performed to evaluate time series preprocessing pipelines that deal with variable-length data. Section VI addresses a case study, in which we apply a selected processing pipeline (padding) to the incident detection task. Section VII concludes the paper.

II. LITERATURE REVIEW ON MULTIVARIATE TIME SERIES CLASSIFICATION

Classification, in general, is the process of predicting a class label of an object. We describe the object using measurable attributes, also called features. We extract the same set of attributes for each object, and a decision algorithm, called a classifier, is used to distinguish between objects from different classes. In the case of time series classification, attributes are consecutive observations ordered in time. In the case of multivariate time series, we have more than one sequence making a time series. Multivariate time series contain simultaneously collected signals concerning one entity.

The baseline approach to univariate time series classification would treat each time series data point as a single attribute. We may apply any standard classifier, for example, random forest, to a data frame in which one row corresponds to one time series and one column corresponds to one moment in time. We would have to ensure that each time series is of the same length and starts at a comparable moment in time. Surveys show that this baseline approach achieves surprisingly satisfying results for univariate time series datasets [5]. However, when we deal with multivariate time series, the problem's dimensionality grows quickly and the plain-classifier-based approach may become infeasible.

There are three types of approaches suitable for multivariate time series. We address them separately in the following subsections.

A. MULTIPLE VARIABLES HANDLING WITH THE USE OF WEAK LEARNERS IN ENSEMBLE CLASSIFIERS

A straightforward solution to the problem of multivariate time series classification is to make an ensemble in which weak learners are trained on different variables in the data. The algorithms do not take the benefit of possible relations between the variables present in the data. Regardless, such a technique is relatively easy to implement, and one can use a wide choice of classifiers to instantiate a weak learner.

A prominent algorithm that allows for such a processing scheme is the Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) [4], [28]. It is an ensemble of a wide variety of base univariate time series classifiers. Its latest version uses as weak learners classifiers such as Shapelet Transform Classifier (STC) [3], Time Series Forest (TSF) [1], Contractable Bag of Symbolic-Fourier Approximation Symbols (CBOSS) [29], and Random Interval Spectral Ensemble (RISE) [25]. The notorious flaw of HIVE-COTE is its extremely high time complexity. As the authors of this algorithm earnestly report [28] (let us give just one example), for a dataset named HandOutlines, HIVE-COTE takes 18.53 hours to train. In contrast, an Inception neural network takes 7.11 hours, and ROCKET takes 0.23 hours on a high-end computer.

In the group of ensemble algorithms suitable to deal with multivariate data, we can also find the generalized Random Shapelet Forest (gRSF) [20]. It generates a set of shapelet-based decision trees. A shapelet is deemed to be a distinctive part of a time series. Karlsson *et al.* [20] generate shapelets randomly. Nonetheless, there are algorithms dedicated just to this task. Several such methods were delivered by Ji *et al.* [19]. The gRSF algorithm randomizes variables for which the shapelets are extracted to handle multivariate time series. As a result, a single tree (weak learner) in the gRSF algorithm is built for one of the variables available in the data. Several approaches utilizing this scheme benefit from a preprocessing step based on Dynamic Time Warping (DTW) that flexibly aligns time series.

B. MULTIPLE VARIABLES HANDLING USING FEATURE EXTRACTION PROCEDURE OPERATING ON ALL VARIABLES

The second approach to the multivariate time series classification task is independently extracting features from all variables. Then, one can use these features to build a single classifier. There is a wide variety in how different algorithms extract the features and build these classifiers. Admittedly, one can still use an ensemble classifier in the end, but handling multiple variables happens at the feature extraction level.

We find the Canonical Interval Forest (CIF) [27] classifier in this family of methods. It combines the TSF classifier [1] with an approach for feature extraction named Catch22 [26]. The feature extraction step ensures handling multiple variables when Catch22 samples intervals from different variables that are a base for features computation. Similarly, Baldan and Benitez's [6] Complexity Measures and Features for Multivariate Time Series (CMFMTS) extracts features for all variables, which are then processed using a traditional classifier.

Subsequently, let us mention the so-called dictionary-based methods adapted for multivariate data. In this family, we find the Word Extraction for Time Series Classification (WEASEL) [34] technique that uses a feature extraction method named Multivariate Unsupervised Symbols and Derivatives (MUSE). The entire processing pipeline is termed WEASEL + MUSE [35]. It builds a feature vector using a sliding window strategy applied to each time series variable. Then, it extracts discrete features per variable and window. Irrelevant features are removed using a technique based on a Chi-squared test, and then, finally, a logistic regression classifier is trained to perform class label assignment. A similar idea is present in an algorithm named Multiple Representation Sequence Learner (MrSEQL) [23].

Finally, let us mention an algorithm termed RandOm Convolutional KErnel Transform (ROCKET) [9]. It is a fast and accurate time series classification method for dealing with multivariate datasets. Its superiority is attributed to a unique manner of time series feature extraction. It uses a concept analogous to the one present in the feature extraction step in Convolutional Neural Networks (CNNs), namely convolutional kernels.

The CNN model is frequently used in image classification. CNNs fuse in a single processing stream feature extraction and classification steps. The input image is represented as a numeric matrix. Two types of layers are included in the feature extraction part of a CNN: convolutional and pooling [31].

A convolutional layer extracts features from the input image. It is performed using kernels that can be interpreted as filters. A kernel is represented with a numeric matrix of a specific size. Typically, this size is relatively small. A kernel moves on the image matrix with a certain stride. In each position, it performs a multiplication of the values in the kernel by the values in the underlying part of the image. The results of these multiplications are then added, and they become an element in the convolution layer output matrix [13]. Convolutional kernels, when applied to images, capture high-level features such as edges and texture. By analogy, convolutional kernels can capture patterns in time series.

Apart from size, weights (kernel matrix elements), padding (specifying behavior on edges), and stride, we shall mention two more kernel parameters: bias and dilation. Dilation is used to spread the kernel over the input data, so with dilation equal to d, every d-th element of the input data "covered" by the kernel will be convolved with the kernel weights. Dilation allows kernels to capture the same pattern at different scales. Bias term is added to the convolution result between the input numbers and kernel weights [24].

ROCKET extracts features from time series data using convolutional kernels just like a CNN. The main idea of applying convolutional kernels for time series is that we can successfully equate a time series with an image. In the case of univariate time series, we can interpret a time series as an image made of $1 \times time\_series\_length$ pixels. In the case of multivariate time series, we can interpret it as an image made of $number\_of\_variables \times time\_series\_length$ pixels. In both scenarios, we can apply kernels to extract features. Several traits distinguish ROCKET from convolutional layers used in typical CNNs:

- ROCKET uses only a single convolutional layer and a massive number of kernels. On top of that, kernel weights are not learned but randomly selected.
- ROCKET uses dilation randomly sampled for each kernel instead of increasing it exponentially with depth. In consequence, it allows for capturing patterns at different frequencies and scales.

- Apart from random weights and dilation, ROCKET uses kernels with random length, padding, and bias. This differs from classical CNNs, where it is common for groups of kernels to share the same size, dilation, and padding.
- Besides the global max pooling, ROCKET incorporates a novel pooling feature – the proportion of positive values (PPV). It enables ROCKET to control the prevalence of a specific pattern in a time series.

ROCKET is relatively robust to different choices for many parameters [9].

C. MULTIPLE VARIABLES HANDLING USING NEURAL NETWORKS

Neural networks appear as a natural choice when it comes to multivariate time series classification. There is an obvious correspondence between the multiple time series variables and multiple neurons that can be used to instantiate an input layer of neural architecture. Thus, a substantial research effort was devoted to developing various neural models for this domain.

AlexNet neural network architecture was among the first CNN-based architectures tested for time series data [14]. However, the literature studied it only in the context of univariate time series, and more advanced models quickly dethroned it.

Significant progress was achieved when a CNN architecture called ResNet utilizing the so-called residual connections (also known as skip connections) was applied to time series data by Wang *et al.* [39]. One of the dilemmas of training neural networks is that we usually want deeper neural networks for better accuracy and performance. However, the deeper the network, the harder it is for training to converge. ResNet allows deep neural architecture training by allowing shortcut connections in each residual block. In practice, it means that there is an alternative path for data to reach the latter parts of the neural network that skips some layers. ResNet turned out to be a huge success when applied to multivariate time series data. It is also deemed the best for univariate time series [32].

InceptionTime is a neural network built specifically for multivariate time series classification. It achieves high accuracy by combining the ResNet architecture with the so-called inception modules. An inception module is a specifically constructed layer that takes on the input multivariate time series and uses multidimensional kernels to perform the convolution step. Max pooling is applied to reduce the problem dimensionality. Mentioned steps (convolution and pooling) are preceded and followed by a bottleneck layer that thins the data even more [14].

Finally, let us mention the Time Series Attentional Prototype Network (TapNet) [44]. The two key operations of this network are named Random Dimension Permutation and Multivariate Time Series Encoding. Random Dimension Permutation is used to produce groups of randomly selected time series variables. Multivariate Time Series Encoding is performed for each group based on one-dimensional convolutional layers followed by batch normalization, Leaky Rectified Linear Units, and a global pooling layer. In addition to the listed encoding process, the raw data is passed through a parallel path through an LSTM and a global pooling layer. This makes the TapNet a fusion of a convolutional and a recurrent network. Then, signals from the CNN and LSTM are merged into a fully connected layer, followed by yet another fully connected layer.

III. LEVERAGING TRANSACTIONS DATA FOR INCIDENT DETECTION – PROBLEM DESCRIPTION AND LITERATURE REVIEW

Let us reiterate that the problem at hand is transaction incident detection in retail and services such as restaurants. The incidents we are interested in occur around employee-client interaction when an employee handles order payments, returns, etc. As Hines and Youssef underline [17], occupational fraud costs the average organization 5% of yearly revenues. Restaurants are especially susceptible to these negative phenomena. The authors give an estimate of 3–6% range to evaluate the restaurant loss scale.

Even though occupational frauds occur typically at points of sale [17], very few companies report using dedicated fraud detection systems. To illustrate the overall complexity of the issue, let us distinguish the main categories of internal incidents, which in our practice (this paper describes an ongoing applied project) take place:

- under-ringing product/service value;
- bogus refunds;
- intentional deletion/voiding of items from an order and pocketing the money for deleted items;
- registering different items than ordered by a customer;
- sweethearting (handing additional products to a customer);
- violations of coupon policies that result in either pocketing the discount value or the customer paying less than he is supposed to.

Of course, the list mentioned above is not exhaustive; it covers critical incidents that take place around points of sales in different businesses. What is more, some frauds occur as a single instance event, and some are performed via a chain of operations keyed into the system as a sequence of operations. The details depend on the specifics of the services or products a given business offers.

This paper addresses a method that operates on data concerning point of sale transactions logged to a relational database. Such a scenario has the advantage of universality. A relational database with numerical information about transactions is a common resource of various businesses. Alternative data sources are video monitoring which is not always present at each point of sale and is always custom-tailored for a particular company.

The downside of using numerical transaction data is that the data is "blind". By this, we mean that in a commercial setting an audit is typically conducted by a human auditor who is simultaneously looking at the transaction data and the video monitoring and seeks for discrepancies between what took place and what was logged to the system.

The underlying challenge required before considering the construction of an incident detection system is collecting an appropriate dataset [2]. These would differ depending on the company and its operational procedures. There are no open datasets of this kind. The collection of such a dataset is done with the help of human loss prevention specialists who manually annotate data. Then, as the literature suggests, experts define numerical features using transaction data which are the base for classifier construction [43], [45]. Some attempts utilize only transaction frequency data [22]. We shall mention that the literature offers some insights into fraud detection in credit card transactions or online payments. A recent survey by Lim *et al.* [37] points out that a variety of machine learning algorithms, including outlier detection techniques, neural networks, rule-based systems, classifiers such as SVM, and others have been applied to solve this task [36]. Some papers take an even broader perspective and discuss financial fraud in general [40]. Notably, the problem tackled in this paper is more challenging than credit card fraud analysis. In retail and services, it is much easier for a dishonest employee to manipulate a cash transaction than a credit card transaction. Money paid in via a credit/debit card is transferred directly to the service provider (business), and a typical employee is not able to install a system circumventing the legitimate payment process. There is also one psychological effect of ''encouraging'' frauds for cash transactions: the only trace a client has is a receipt that a client often does not read or even take from a counter.

To the best of our knowledge, only a few research papers address fraud detection in the context of loss prevention in restaurants, which is our application domain. There are two papers by Hines and Youssef. One article is focused on describing what they call a ''rotating check'' fraud [16]. It is a type of fraud similar to deleting items from an open receipt after a customer paid and pocketing the deleted value. The second paper is focused on discussing outlier detection techniques and how they can be applied to identify frauds [17]. In the latter case, the authors address two types of fraud. These are, again, rotating check fraud and another type they call ''bartender no sale'', which means that a person is registering a transaction as a no-sale transaction and pocketing the money.

An engaging survey was presented by Collins [7]. His paper, first and foremost, lists a range of fraudulent behaviors that can take place in a restaurant. Beyond the list presented at the beginning of this section with the essential fraud types, Collins mentions [7]:

- Servers reprinting the same receipt throughout a shift and handing it repeatedly to different customers. An employee can obfuscate and hide better if he convinces the persons to order similar food items. This can also be done in collaboration with other servers (one reprints receipts for, say, coke and pie, another, for fish and chips).
- Violations regarding transferring a list of transactions between different servers, in which one server gains another server's tip without the latter's consent.

In our operational model (in many quick-service restaurants), both situations do not happen because our point-of-sale devices do not allow for reprinting of a receipt, and tips are not registered in the system and are always handled instantly. Unfortunately, Collin's paper does not propose how to automate the detection of the described frauds. Described countermeasures require substantial human effort (manual verification).

Kelly [21] more recently mentioned the issues of voiding items and discount policy abuses, but the cited paper neither described new fraud methods nor addressed some concrete solution to the problem.

To sum up, while the literature acknowledges the issue of fraud detection in cash transactions, the particular data-driven solutions that aim to detect these issues are almost absent in the literature. The ones that were described are limited to one specific fraud technique.

IV. THE PROBLEM OF VARIABLE-LENGTH TIME SERIES

There are a few fundamental reasons why time series generated by variants of a single prototypical process may end up having different lengths. As Tan *et al.* [38] underline, one of these mechanisms corresponds to variation in the relative frequency at which the process is observed. For instance, the generating processes might unfold at differing speeds, or the sensors might operate at different frequencies. This case is especially valid for remote sensing and human activity recognition applications. In the former case, weather conditions may influence how samples are generated. In the latter case, sensors are typically programmed in an energy-saving manner. When the same type of activity is recorded over some time (for instance, when a subject is sitting), the sampling frequency drops to preserve battery life.

Another reason for unequal-length time series generation is a variation concerning the points during the process at which the recorded observations begin and end. This issue affects especially datasets that originate from audio signals. As Tan *et al.* [38] underline, just a handful of strategies have been devised to address the classification of time series with differing lengths. The existing studies propose the following solutions for unequal-length data:

- uniform scaling, which stretches shorter time series to the length of the longest time series in a uniform manner;
- padding, which adds either a fixed value or a low amplitude noise to the suffix or the prefix of shorter time series to obtain sequences as long as the longest one;
- direct processing of varying-length time series.

The first two approaches lack in some aspects. Uniform scaling is applicable to series that differ in length due to varying frequencies but not to those that differ due to variations in the start and the end point. This solution cannot be used if we wish to obtain a data-agnostic processing stream. Tan *et al.* report that padding is unsuccessful when the length of time series differs by a lot [38].

Direct processing of varying-length time series means that an algorithm has an intrinsic capability to produce features from such data. Unfortunately, not many algorithms can do so. What is more, the algorithms capable of processing varying-length time series directly are not in the group of the top-performing ones. ROCKET, which is deemed the optimal choice due to its speed and accuracy, cannot handle unequal-length data and employs padding by default. Examples of algorithms that handle unequal-length time series are some DTW-based methods, BOSS, and Proximity Forest. However, these algorithms achieve inferior performance compared to the top performers: ROCKET, HIVE-COTE, CIF [32].

To the best of our knowledge, the topic of unequal-length time series classification is directly studied in only one paper available in the literature, and it is in the context of univariate data. Some papers indirectly mention this issue. However, the problem at hand is too significant to be handled in such a shallow manner. This area of research needs more contributions since modern data analysis schemes require handling of such datasets.

In this paper, we present an empirical evaluation of the efficiency of several strategies for dealing with variable-length time series data:

- truncation,
- padding (with a constant value),
- forecasting values with an ARIMA model.

The first two strategies are straightforward; therefore, we do not address them in greater detail.

AutoRegressive Integrated Moving Average (ARIMA) model consists of three elements:

- autoregressive model (AR),
- moving average model (MA),
- model integration (I).

The first part, autoregressive model, is a process in which each value is assumed to be a linear combination of previous values. In other words, it uses memory to describe the current value. Autoregressive model order determines how many previous values are taken to compute the current value. Let us denote an autoregressive model of order p as AR(p). Its general form is given as

$$z'_t = c + b_1 z_{t-1} + b_2 z_{t-2} + \ldots + b_p z_{t-p}, \tag{1}$$

where z'_t is a prediction of the current value z_t, c is an intercept, describing a drift. We can easily notice that the autoregressive model is analogous to the multiple regression model. Parameters $b_1, b_2, \ldots, b_p$ describe how strong is a relationship between history and a current value. The description of the current value is an approximation of the real value z_t with the error ε_t

$$z_t = c + b_1 z_{t-1} + b_2 z_{t-2} + \ldots + b_p z_{t-p} + \varepsilon_t, \tag{2}$$

where ε_t is white noise.

A moving average model, another component of ARIMA, uses past forecast errors in a regression-style model:

$$a_{t-1}\varepsilon_{t-1} + a_{t-2}\varepsilon_{t-2} + \ldots + a_{t-q}\varepsilon_{t-q} \tag{3}$$

q denotes the order of the moving average model; we denote it as MA(q). $a_1, a_2, \ldots, a_q$ are discovered coefficients. While the autoregressive model uses historical values, the moving average uses historical distortions to model a time series.

The third component of the ARIMA model is integration (I). Integration, in this context, is the action opposite to differentiating. If we join the three components together, we obtain ARIMA(p, d, q) model, where d is the degree of first differentiating applied. The model can be written as:

$$\begin{aligned} \text{ARIMA}(p, d, q) = {} & c + b_1 z'_{t-1} + b_2 z'_{t-2} + \ldots + b_p z'_{t-p} \\ & + a_{t-1}\varepsilon_{t-1} + a_{t-2}\varepsilon_{t-2} + \ldots \\ & + a_{t-q}\varepsilon_{t-q} + \varepsilon_t \end{aligned} \tag{4}$$

To automatically detect the structure of the model, that is p, d and q, one may use the Hyndman-Khandakar algorithm [18]. It combines unit root tests, minimization of the Akaike information criterion (AIC), and Maximum Likelihood Estimation to obtain an ARIMA model. In our case, a modified version of the AIC was applied the details of which are given in Section V.

In this study, we have narrowed the scope of interest to forecasting with ARIMA. We wished to validate whether the forecasting path is worthy of following in this case. ARIMA is a well-known, rather classic method. Naturally, there are other time series precision methods available to use instead of ARIMA.

V. EMPIRICAL EVALUATION USING BENCHMARK DATA

To provide as high replicability of the conducted study as possible, we completed the empirical evaluation of the proposed approach at first for benchmark datasets that are publicly available. Secondly, we applied ROCKET to process our data, which is a private asset of our company and, thus, we cannot share it. In this section, we present the application to publicly available benchmark datasets.

A. BENCHMARK DATASETS AND DATA STAGING

The first part of the experiment concerned publicly available datasets. Their summary is given in Table 1.

Selected datasets are of substantially distinct properties. We have datasets with many classes, such as Phoneme with 39 classes, but also three sets with binary labels. The starting time series length is also very different. Take EigenWorms set, in which time series are made of 17,984 observations. In contrast, ERing dataset is made of series 64 elements long. We also made sure that the cardinality of samples differs. In this regard, let us point to the set named StandWalkJump

TABLE 1. Summary of benchmark dataset properties. dims. – the number of variables, cl.num. – the number of classes, tr.s. – train size, te.s. – test size.

dataset name	dims.	cl.num.	length	tr.s.	te.s.
ArticularyWordRecognition	9	25	144	275	300
AtrialFibrillation	2	3	640	15	15
BasicMotions	6	4	100	40	40
Cricket	6	12	1197	108	72
EigenWorms	6	5	17984	128	131
Epilepsy	3	4	206	137	138
ERing	4	6	65	30	270
EthanolConcentration	3	4	1751	261	263
FingerMovements	28	2	50	316	100
Handwriting	3	26	152	150	850
Heartbeat	61	2	405	204	205
Libras	2	15	45	180	180
MotorImagery	64	2	3000	278	100
NATOPS	24	6	51	180	180
Phoneme	11	39	217	3315	3353
RacketSports	6	4	30	151	152
SelfRegulationSCP1	6	2	896	268	293
SelfRegulationSCP2	7	2	1152	200	180
StandWalkJump	4	3	2500	12	15
UWaveGestureLibrary	3	8	315	120	320

with just 12 samples in the train set (it is a 3-class balanced set, so we have only four samples in a class). In contrast, we have MotorImagery set with 139 samples in each class. The selected 20 datasets intentionally have distinct properties.

The variables present in the tested datasets are also of various properties. For example, in the RacketSports dataset, observations were collected at a rate of 10 Hz. The sampling frequency of the Epilepsy dataset was 16 Hz. In contrast, in the FingerMovements dataset, we have information concerning 28 EEG channels which translates to 28 variables, each sampled with the 100 Hz frequency. Finally, the MotorImagery dataset contains recordings performed with a sampling rate of 1000 Hz. The source of observations also differs. We have data collected using smart watches (RacketSports), spectrometers (EthanolConcentration, Heartbeat), ECG (AtrialFibrillation), accelerometers and gyroscopes (BasicMotions), and more. There is also a dataset with recordings converted from curves obtained from videos (Libras). We are not placing a detailed description of all variables because there are 258 variables (the sum of the values presented in the second column in Table 1).

The source website already provides the data split into train and test sets. We did not interfere with this split.

The data from https://timeseriesclassification.com is of equal length. Thus, to perform wider tests, we had to trim it. This was performed according to the following rules:

1) The trimming was executed in the same way separately for each class and for the train and test set. As a result, the trimming was stratified; we did not want to treat classes differently and insert by accident some class label-related bias.
2) We assume that for each class 1/3 of the instances will have a length in the range [10%,40%] of the original length, 1/3 of the instances will have a length in the range (40%,70%], and the remaining 1/3 (70%,100%).
3) Assuming that for a given instance, we ought to do the trimming to the [10%,40%] range, we randomly draw a value from this range multiply it by time series length and round to an integer value.

Because of the nondeterminism related to the above-presented data staging procedure, we repeated all experiments ten times and averaged the results.

B. PROPOSED ADAPTATIONS IN PRACTICE

The course of experiments concerned testing the identified methods for dataset length equalization for selected 20 datasets. The quality of a method is measured with classification accuracy that we compute for test sets (train sets are not used for quality evaluation, only for model building). Each experiment was repeated 10 times, in each run, we have been effectively working with a different set because the trimming procedure is random. Therefore, we give average accuracy, maximum accuracy, and standard deviation. Table 2 contains the obtained results. Benchmark datasets are balanced. Thus, the accuracy measure (given in Eq. (8)) was enough to evaluate quality at this stage.

We have tuned the values of two relevant parameters for each dataset: the number of kernels and regularization parameter λ for the Ridge regression classifier. The tuning procedure relied on 10 repetitions of the classification procedure on full-length data for selected combinations of parameters. We checked for 50, 100, 500, 1000, 5000, and 10000 kernels and ten values of λ generated using an exponential sequence ranging from 10^{-3} to 10^{3}. The number of kernels translates to the number of features that are then used in Ridge regression. Therefore, it should be tuned individually for each dataset due to the varying properties that may call for more or less features. In contrast, λ determines the shrinkage penalty. $\lambda = 0$ implies that the penalty term has no effect. In consequence, the regularization effect will not be visible. Increasing the λ increases the impact of the shrinkage penalty. In consequence, regression coefficients will get closer to zero. After establishing suitable parameters for full-length data, we used the same settings for all other experiments.

The ARIMA model fitting stage was executed with the use of the *forecast* R package. For each time series, we have fitted an optimal ARIMA model according to the Akaike information criterion (AIC) measure adjusted for a small sample size (AICc):

$$\mathrm{AICc} = \mathrm{AIC} + \frac{2k^2 + 2k}{n - k - 1}, \tag{5}$$

where n is the sample size and k is the number of parameters. One can observe that, AICc adds to the bare AIC an additional penalty term for the number of parameters. The AIC is given

TABLE 2. Results concerning selected 20 datasets. Average (avg), maximum (max), and standard deviation (sd) of classification on test sets for selected datasets. The values are based on ten repetitions of the experiment. Different columns mark different time series equalization methods. 100 kernels were used. Values are given in %. Bold font is used to mark the best outcome for reduced-length sets.

dataset name	whole set			padding			truncation			ARIMA		
	avg	max	sd	avg	max	sd	avg	max	sd	avg	max	sd
ArticularyWordRecognition	98.83	100	0.63	80.13	84.67	2.60	30.20	34.67	2.40	82.03	**85.00**	2.71
AtrialFibrillation	16.00	20.00	3.44	15.33	26.67	7.06	14.00	20.00	2.11	20.00	**33.33**	9.43
BasicMotions	100	100	0	89.25	**97.50**	6.02	76.75	85.00	7.55	90.25	95.00	4.16
Cricket	74.44	77.78	2.79	88.89	94.44	3.07	83.47	84.72	0.44	90.00	**95.83**	2.91
EigenWorms	56.26	58.78	2.25	72.37	**78.63**	4.56	69.92	**78.63**	4.20	72.44	76.34	2.70
Epilepsy	97.61	98.55	0.69	75.65	81.16	5.03	59.71	63.77	2.35	76.23	**85.51**	4.54
ERing	77.63	78.89	0.86	94.78	**96.67**	1.34	78.89	83.70	5.63	23.93	75.93	18.27
EthanolConcentration	45.55	49.43	2.16	25.93	27.38	0.82	34.91	**38.40**	1.70	25.70	28.52	1.50
FingerMovements	46.30	48.00	1.25	50.40	**61.00**	4.65	48.90	50.00	0.74	51.10	58.00	4.46
Handwriting	23.96	24.59	0.99	14.87	**21.88**	3.89	6.29	7.06	0.53	15.98	19.77	3.14
Heartbeat	72.20	72.20	0	71.56	**72.68**	1.28	72.20	72.20	0	71.12	**72.68**	1.45
Libras	49.00	54.44	2.31	39.78	44.44	3.84	23.33	26.11	1.80	40.72	**46.67**	5.21
MotorImagery	53.20	59.00	3.80	51.10	54.00	2.73	50.90	**59.00**	4.43	52.70	56.00	3.06
NATOPS	65.72	67.78	1.44	51.00	**56.67**	3.44	25.28	29.44	2.69	49.44	56.11	3.69
PhonemeSpectra	17.79	19.57	0.90	12.46	**13.75**	0.65	7.49	8.05	0.34	13.02	**13.75**	0.44
RacketSports	70.59	75.66	2.74	69.34	**74.34**	3.66	67.96	73.03	2.83	66.91	**74.34**	4.15
SelfRegulationSCP1	69.90	76.11	3.05	83.04	**87.03**	2.11	76.79	79.18	1.76	81.37	86.01	2.66
SelfRegulationSCP2	52.72	55.56	2.17	51.56	**58.89**	3.78	48.67	50.56	1.21	50.61	56.11	3.22
StandWalkJump	28.00	33.33	4.22	24.67	**46.67**	8.92	36.67	**46.67**	28.67	28.67	40.00	6.33
UWaveGestureLibrary	66.25	67.81	1.36	62.93	66.25	2.66	23.38	25.94	1.61	64.72	**69.06**	2.40

as

$$\mathrm{AIC} = 2k - 2\ln(\hat{L}), \tag{6}$$

where k is the number of parameters in the model and $\hat{L}$ is the maximum value of the likelihood function. We are computing the AICc for a set of models, and the one deemed the best has the lowest AIC value. Since the step of fitting the optimal ARIMA model is performed multiple times (the number of all time series times in each dataset multiplied by the number of experiment repetitions, which is 10), we are not giving the final estimates in the paper.

Table 2 shows that truncation is the worst possible choice regarding data preprocessing. In two cases, truncation was better than padding and ARIMA-based time series extension. We have decided to include results on full-length time series even though the target processing stream was working on variable-length data. On the one hand, comparing the accuracy of trimmed and full-length data allows inferring the possible redundancy in full-length time series.

In some domains, it is not an obvious choice how to cut time series for classification if they are being extracted from a data stream. This applies to sensor data. One can easily produce a dataset of equal length time series in these domains. However, not all data that needs to be subjected to classification comes from sensors. In multiple other domains, time series length is not collected as a stream.

In this context, we would like to point out that for a dataset called EigenWorms, the accuracy obtained on trimmed time series is 20% higher than when the data is considered in the original length. When we look at the dataset repository, we can easily explain this, as this dataset contains sensors recording the motion of worms [41]. The same goes for the AtrialFibrillation dataset, which is cut from ECG recordings, StandWalkJump, which is collected with sensors recording human activities, and so on. There are a few more cases when working with shorter data resulted in slightly higher accuracy. In each case, we can interpret this phenomenon in the same manner – the data was recorded by sensors, then cut from a wider stream, and we envision that a distinct pattern is not present in the entire time series length that was cut.

Our focus is on three techniques for variable-length data preprocessing: padding, truncation, and forecasting further values using an ARIMA model. The first conclusion from the values in Table 2 is that truncation is the worst possible strategy. It results in models that classify data with a substantially worse accuracy than the other two methods. The strategy that turned out to be most favorable is padding. We are most likely to achieve a top-performing classifier with this preprocessing strategy. The average accuracy of the best classifier for the considered 20 datasets for truncation is 47.26%, while it is 62.49% for padding. In the middle, we have ARIMA-forecasts-based preprocessing, which produced models with 61.20% accuracy on average. Thus, ARIMA is slightly worse than padding but much better than truncation. If we inspect standard deviations, the average standard deviation in 10 repetitions of the experiment was the highest for truncation and equal to 6.03, which means that truncation not only provided the worst accuracy but was also the least stable. The slightest deviations, on average, are attributed to padding (2.03). ARIMA-based preprocessing gave models that varied on average by 3.71% for the 20 considered datasets.

Achieved results clearly show that padding is the most reliable preprocessing tactic when dealing with variable-length time series data that need to be made equal-length for further classification.

To illustrate this discussion, we present Fig. 1, which shows the distribution of best accuracies achieved for the 20 considered datasets for the four cases: full-length datasets and preprocessed using the three studied methods.

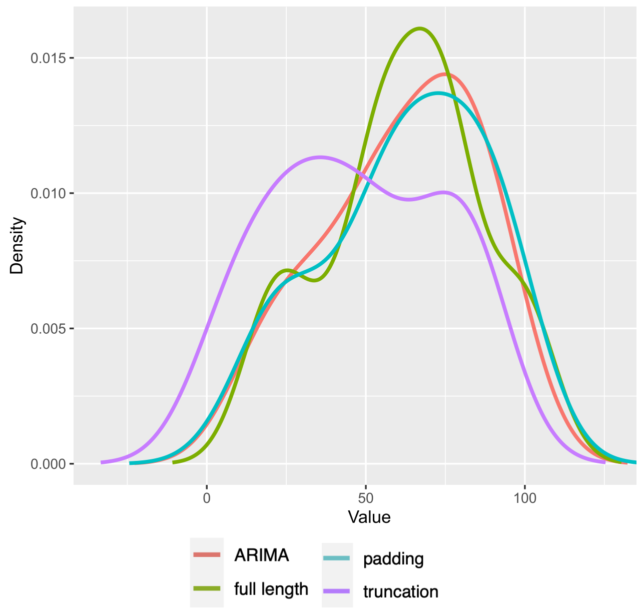

FIGURE 1. Accuracy distribution for the experiments covering 20 datasets using four variant of the preprocessing procedure (raw data and three methods for dealing with variable-length data).

Fig. 1 confirms that truncation should not be considered a viable preprocessing choice for the described data classification stream. ARIMA-forecasts-based and padding-based preprocessing variants ensure satisfying accuracy. However, the use of ARIMA entails a substantial increase in the computational complexity of the model. Thus, padding is the recommended preprocessing strategy.

Analogous observations can be made when we plot a critical difference (CD) diagram with Wilcoxon-Holm post-hoc analysis, which illustrates the significance of the differences between various approaches. The CD diagram is visible in Fig. 2. It was computed for four data processing streams: full-length time series and shortened using (i) truncation, (ii) padding, (iii) ARIMA forecasts applied to 20 datasets. The tests were run for $\alpha = 0.05$.

Note that in the CD plot, we can see the ranking of methods using the horizontal scale. It goes from two to four, and

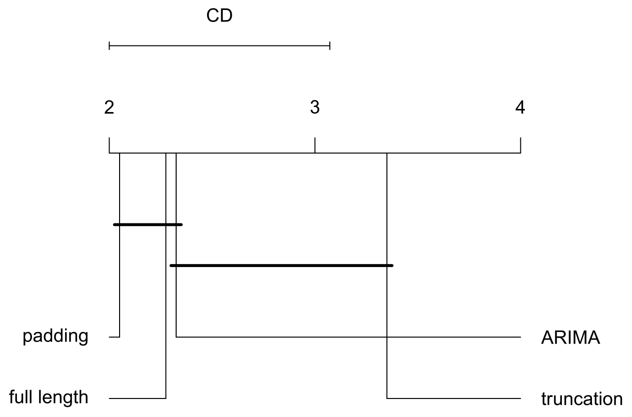

FIGURE 2. Critical difference diagram with Wilcoxon-Holm post-hoc analysis for comparing different strategies to variable-length time series classification using ROCKET. $\alpha = 0.05$.

smaller values are better. In other words, classification accuracy for variable-length time series extended using padding turned out to be the best. Surprisingly, these classification rates are even higher than for full-length time series. The conducted statistical tests indicate that this difference is statistically insignificant. A horizontal bar links the results obtained with padding and full-length time series.

The processing stream that utilizes ARIMA forecasts turned out to be very close in ranking with the full-length series. Interestingly, the differences between ARIMA-based method and truncation are statistically insignificant. However, there is a clear difference between the overall efficiency of these two methods visible in the values in the ranking.

VI. CASE STUDY OF INCIDENT DETECTION

After we had established that padding is the recommended preprocessing strategy when dealing with variable-length time series, we proceeded with experiments using our data concerning cash transaction incidents using padding.

A. DATASET DESCRIPTION

Our project aims to develop an incident detection mechanism that will recognize fraudulent cash transactions. The challenge is that we want to use data logged to a relational database, the backbone of a system that handles payments in a restaurant. This implies that we do not have many available descriptors concerning a transaction.

Transactions audit in our company is primarily a domain of human experts who prepared a dataset with labeled samples for us. The problem of incident detection is imbalanced. Fraudulent transactions make up about 5% of the overall transaction volume. Table 3 presents the number of items in two considered classes: legitimate and fraudulent transactions and the number of different fraud types in the dataset.

Let us note that the values given in Table 3 concern cases manually labeled by our human annotators. It is only a tiny fraction of our database's total quantity of transactions. However, for the sake of model creation, we had to establish

TABLE 3. The number of fraudulent and legitimate instances in our dataset.

class	cardinality
legitimate transactions	24149
fraudulent transactions:	
– Items different than registered	97
– Sweethearting	131
– Items deletes fraud	145
– Refund/Void fraud	288
– Unauthorized coupon discount	381
– Unauthorized employee discount	238
– Other discount violations	191
total # of fraudulent transactions	1471

a learning set of instances that we will use to construct a model and evaluate its quality.

Table 3 presents different fraud types covered by our auditors. These are

- Items different than registered – server (employee) hands items different than registered. The situation typically concerns a case when registered items are of a higher value than handed items, and the money difference ends up with a server.
- Sweethearting – server hands additional items to a customer.
- Items deletes fraud – server incorrectly deletes items from a receipt. Typically customer pays for the deleted items beforehand and the money for deleted items is taken by the employee.
- Refund/Void fraud – server calls a refund or voids a whole or a part of a committed order when they should not (a customer did not ask for refund/void). Money resulting from this operation is pocketed by the employee.
- Unauthorized coupon discount – employee applies a coupon discount without grounds. There are two possible scenarios. One, where a customer pays less when they should pay more. Two, where a customer pays the right price, and the employee takes the difference between the right and discounted order value.
- Unauthorized employee discount – employee applies an employee discount for a meal that is not handed out to an employee.
- Other discount violations – various discount policy violations that differ for particular clients, for instance, issuing a discount higher than the maximal allowed discount or applying two coupons when only one can be applied.

We want to emphasize that in this study, we present the results for automated detection of the above-listed frauds. To the best of our knowledge, this is the first type of a study that does not focus on an automated detection of a single fraud type.

Detection of the listed fraud types was performed based on variables extracted from a relational database that handles the daily operations of our various business partners. Table 4 presents considered features. Extending this list is our future work direction.

We are working with multivariate time series. Thus, we can represent a single instance with Eq. (7).

$$\begin{bmatrix} x_1^1 & x_1^2 & x_1^3 & \dots & x_1^{L1} \\ x_2^1 & x_2^2 & x_2^3 & \dots & x_2^{L1} \\ x_3^1 & x_3^2 & x_3^3 & \dots & x_3^{L1} \end{bmatrix} \tag{7}$$

In Eq. (7), $x_i^j \in \mathbb{R}$, $i = 1, \dots, M$ denotes variable index, M is the total number of variables, $j = 1, \dots, L$ denotes the index of the element in the time series. In particular, x_i^j describes a transaction immediately before x_i^{j+1}. L is the length of the longest time series we have in a dataset.

Since the data is of variable length, several first $x$es may be unavailable (we denote them technically as *NA*). Values denoted with NA are replaced with actual values at the stage of padding.

Let us recall that a single instance describes the history of transactions handled by a given employee. We created the learning dataset to take all data annotated manually by our human experts. We had to develop a data model under real-world constraints. The essential restriction is that human experts attach labels to a single transaction for a given audit. That is, we have a history of transactions for a given employee for a particular day, and a human auditor attaches a label (one of the labels present in Table 3) to one transaction the auditor looked at. The remaining transactions in the day are not checked. Auditor work is manual expert labor. It is not feasible to ask an auditor to label the entire history of transactions for a more significant number of days because a single manual transaction audit takes about 40 minutes and an employee handles hundreds of transactions during one shift.

Thus, the data we have ends with a transaction that an auditor manually verified. In our dataset, each instance, as formalized in Eq. (7) is paired with a label indicating fraud type or a lack thereof. Subsequently, we split the data into train and test parts. The train set is used for model construction, and the test set is used for model evaluation. These two sets are disjoint, which means quality evaluation concerns samples previously unseen by the classifier.

We envision the incident detection model as a tool that shall help a human auditor. An incident detector suggests checking several transactions, which a human auditor manually verifies. Let us assume that a full manual audit of a single transaction takes 40 minutes. Let us assume that 40 minutes is split into 30 minutes of searching for the possible incident and 10 minutes of report writing and other technical details.

B. EMPIRICAL EVALUATION

The initial experiment relied on building a binary classifier that recognized between fraudulent and legitimate instances.

TABLE 4. Used feature names, descriptions, and types.

Feature name	Description	Type
Gross	Sales gross value	currency
Net	Sales net value	currency
Tax	Tax	currency
TypeID	transaction types such as 'Sale', 'Cancel', 'Cash drop', 'Paid Out', 'No Sale' 'Usage Adjustment', 'Refund', 'Void', 'Voided Refund', 'Suspend', and so on	categorical
Discount	Discount value	currency
Tip	Tip amount	currency
Change	Change amount	currency
TimeLapse	Time that has passed since the last transaction	time (min.)
CorrectedProductsValue	Sum of products with flag IsCorrected set to True in transaction	currency
EmployeeMealCount	Count of employee discounts applied in the transaction	int
EmployeeMealValue	Value of a meal purchased using employee discount	currency
CanceledItemsQuantity	Quantity of products that within a given transaction were canceled	int
CanceledItemsAmount	Money value of products that within a given transaction were canceled	currency
VoidedItemsQuantity	Quantity of products that within a given transaction were voided	int
VoidedItemsAmount	Money value of products that within a given transaction were voided	currency
CorrectedDiscountsQuantity	Quantity of discounts that within a given transaction were corrected	int
CorrectedDiscountsAmount	Money value of discounts that within a given transaction were corrected	currency
DiffProductCount	Number of different products in transaction	int
DiscountQuantity	Number of discounts assigned to transaction	int
DrinkCount	Number of products categorized as drinks assigned to the given transaction	int
FoodCount	Number of products categorized as food assigned to a given transaction	int
MealSetCount	Number of products categorized as meal sets assigned to a given transaction	int
ModifierCount	Number of items categorized as meal modifiers in a given transaction	int
ProductCount	Number of products in transaction	int
UnitCount	Number of food units in a transaction; unit is a main, basic product of a given brand	int

Because the dataset is not balanced, we tested the following train/test set sampling procedure:

- Randomly select 50% of fraudulent transactions.
- The number of legitimate transactions $= K\cdot$the number of fraudulent transactions, K is a positive integer, and sampling is random.
- All remaining samples are placed in the test set.

K is a parameter of our procedure.

Furthermore, since the dataset is not balanced, in addition to accuracy (given in Eq. (8), we had to use precision and recall to evaluate classifier performance. Precision and recall are defined in Eq. (9) and Eq. (10), respectively.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

where:

- true positive (TP) – the number of fraudulent transactions correctly identified as frauds
- true negative (TN) – the number of legitimate transactions correctly identified as legitimate
- false positive (FP) – the number of legitimate transactions incorrectly identified as frauds
- false negative (FN) – the number of fraudulent transactions incorrectly identified as legitimate

We envision that the proposed incident detector will work in a way that it will suggest a human auditor which transactions to look at first. There is a higher cost associated with a false positive than a false negative. In other words, a false positive means an auditor spent (precious) time verifying a legitimate transaction. False negative is a case when we miss a fraud. Thus, precision is of primary importance to us.

In the experiments, we have tested models with 1000 and 10000 kernels and parameter $K = 1, 2, 3, 5, 10, 12, 20$. The results achieved for various parameter combinations are given in Table 5.

ROCKET has proved to be a good algorithm for the multivariate time series classification task. Including mode samples from the legitimate transactions class, which increased the imbalancedness of a train set, turned out to be the essential step in determining the outcome of incident detection. In Table 5, we see that the precision grows with the increase of parameter K determining the imbalancedness degree of a train set. Simultaneously, recall drops. These two rates are accompanied by very high accuracy, which is not an

TABLE 5. Quality of incident detection for various parameter combinations. *K* is the imbalancedness degree of a train set. Precision, recall, and accuracy are given in %.

Kernels	K	Precision	Recall	Accuracy	TP	FP	TN	FN
1000	1	8.58	57.75	79.95	425	4531	18883	311
	2	15.74	41.30	91.20	304	1628	21051	432
	3	20.43	31.25	93.82	230	896	21048	506
	5	21.19	19.29	94.71	142	528	19946	594
	10	35.22	7.61	95.54	56	103	16696	680
	12	38.60	5.98	95.26	44	70	15259	692
	20	60.00	0.82	92.79	6	4	9445	730
10000	1	21.73	28.40	94.36	209	753	21191	527
	2	15.70	35.73	91.95	263	1412	21267	473
	3	18.80	31.79	93.33	234	1011	20933	502
	5	25.06	15.35	95.47	113	338	20136	623
	10	37.41	7.47	95.59	55	92	16707	681
	20	59.57	3.80	94.75	28	19	13105	708

TABLE 6. Selected models that achieved the highest and the second-highest precision when trained in a manner one class versus all. We give precision (in %) and the number of true positives associated with the best model.

	highest		second-highest	
class	**prec.**	**TP**	**prec.**	**TP**
Items different than registered	50.00	1	31.13	33
Sweethearting	50.00	1	27.68	31
Items deletes fraud	71.43	5	31.94	23
Refund/Void fraud	100	2	42.86	3
Unauthorized coupon discount	50.00	3	50.00	3
Unauthorized employee discount	41.38	12	40.74	11
Other discount violations	66.67	2	36.91	31
Sum	–		–	

informative measure in this case because of the vast imbalancedness of the test set.

If we look at the *TP* value in Table 5, we see that when precision increases, *TP* drops. It translates to the situation when our model is more valuable to human auditors since it suggests more precisely fraudulent transactions. The downside is that the number of *TP*s drops. In other words, the model indicates more precisely which transactions are fraudulent, but the overall number of suggestions drops. For the model with the precision of 59.57%, we get 28 transactions correctly identified as fraudulent, and only 19 legitimate transactions were incorrectly identified as fraudulent. This is a good outcome; such a tool can be useful for a human auditor. It must be noted that this model correctly tagged 13105 transactions as *TN*, which means that a lot of data was filtered out.

Let us recall that we assume that a single audit takes 40 minutes, out of which 30 minutes is spent on searching. Assume the model correctly identified 28 transactions as fraudulent. This results in 28*30 minutes of human time saved. The required precision level should be evaluated with respect to the time gained when using the model.

Subsequently, we proceed with experiments to detect particular fraud classes mentioned in our study. We have constructed six binary classifiers. We have relabeled the data so that class "positive" was made of samples from a single incident class. All remaining samples were made as "negative".

Table 6 presents models that achieved the highest and the second-highest precision in the experiments for analogous combinations of parameters as presented in Table 5.

In this case, the number of available "positive" instances drops significantly. Without surprise, ROCKET struggles to produce better models. Though we can achieve a precision of about 60%, recall drops. Please note that the number of *TP* associated with the highest precision is typically tiny. We envision that these seven models will be used jointly. Thus, in the last row in Table 6, we present the sum of *TP*s. This sum, equal to 26, is comparable to 28 obtained when applying a single model. Models that produced the second-highest precision returned together 135 True Positives, which is substantially more than 26. The average precision of these models was 37.32%, which is higher than the precision of a model presented in Table 5 that produced a similar number of *TP*s.

All in all, we believe that the proposed strategy for incident detection is worthy of further investigation. The delivered tools can automate this task and reduce manual labor.

VII. CONCLUSION

In the paper, we have applied benchmark datasets to test three preprocessing strategies for dealing with variable-length time series that we wish to classify with an algorithm that requires equal-length data: padding, truncation, and forecasting further time series values. The experiments have shown that padding is the recommended preprocessing strategy, achieving high classification accuracy. Truncation is not recommended. Classification accuracy substantially dropped when it was employed. Forecasting further time series values leads to a satisfying accuracy as well. However, the computational cost of this extra step is too high to consider it worthy of attention.

At this stage of our studies, we were working with benchmark datasets that we cut to make them variable-length. Two arguments motivated the choice of such a procedure. One that we wanted to demonstrate our methods using publicly available data, commonly adopted by researchers in this domain. Two that we assumed that if we classify this full-length data using ROCKET, we will get an "ideal" classification quality that we can refer to later. The experiments have shown that our second assumption was missed. We achieved a higher classification accuracy in a few cases using the shortened dataset. It was not because of some random factor since we reached these results with ten repetitions of the whole procedure involving random time series cuts. A closer inspection of these datasets showed that they typically originated from sensors, and the time series to be classified were cut arbitrarily. This outcome postulates the need to develop new early time series classification methods, which do not attract enough attention now.

The above conclusion refers to the methodological layer of our study. From the perspective of the applied project that we work on, early time series classifiers do not seem indispensable, as the data we work on is not cut from a stream. Instead, the length is always determined by the number of transactions in a server's shift.

From the perspective of our applied task, our future work will concentrate on the technical aspect of data classification. We will keep improving the procedure by, for example, introducing a variable evaluation step, which will result in removing unpromising variables before ROCKET is run. This will decrease the run-time memory required by the program.

REFERENCES

[1] A. Abanda, U. Mori, and J. A. Lozano, "Time series classifier recommendation by a meta-learning approach," *Pattern Recognit.*, vol. 128, Aug. 2022, Art. no. 108671.

[2] A. Abdallah, M. A. Maarof, and A. Zainal, "Fraud detection system: A survey," *J. Netw. Comput. Appl.*, vol. 68, pp. 90–113, Jun. 2016.

[3] M. Arul and A. Kareem, "Applications of shapelet transform to time series classification of earthquake, wind and wave data," *Eng. Struct.*, vol. 228, Feb. 2021, Art. no. 111564.

[4] A. Bagnall, M. Flynn, J. Large, J. Lines, and M. Middlehurst, "On the usage and performance of the hierarchical vote collective of transformation-based ensembles version 1.0 (HIVE-COTE v1.0)," in *Proc. 5th ECML PKDD Workshop*, Ghent, Belgium, 2020, pp. 3–18.

[5] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Mining Knowl. Discovery*, vol. 31, no. 3, pp. 606–660, May 2017.

[6] F. J. Baldán and J. M. Benítez, "Multivariate times series classification through an interpretable representation," *Inf. Sci.*, vol. 569, pp. 596–614, Aug. 2021.

[7] G. Collins, "Safeguarding restaurants from point-of-sale fraud: An evaluation of a novel theft deterrent application using artificial intelligence," *J. Hotel Bus. Manag.*, vol. 2, no. 1, pp. 1–5, 2013.

[8] B. Dhariyal, T. L. Nguyen, S. Gsponer, and G. Ifrim, "An examination of the state-of-the-art for multivariate time series classification," in *Proc. Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2020, pp. 243–250.

[9] A. Dempster, F. Petitjean, and G. I. Webb, "ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining Knowl. Discovery*, vol. 34, no. 5, pp. 1454–1495, Sep. 2020.

[10] A. Dempster, D. F. Schmidt, and G. I. Webb, "MiniRocket: A very fast (almost) deterministic transform for time series classification," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*. New York, NY, USA: Association for Computing Machinery, Aug. 2021, pp. 248–257.

[11] A. Dempster, D. F. Schmidt, and G. I. Webb, "HYDRA: Competing convolutional kernels for fast and accurate time series classification," 2022, *arXiv:2203.13652*.

[12] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Inf. Sci.*, vol. 239, pp. 142–153, Aug. 2013.

[13] J. Djolonga, J. Yung, M. Tschannen, R. Romijnders, L. Beyer, A. Kolesnikov, J. Puigcerver, M. Minderer, A. D'Amour, D. Moldovan, S. Gelly, N. Houlsby, X. Zhai, and M. Lucic, "On robustness and transferability of convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 16458–16468.

[14] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Müller, "Deep learning for time series classification: A review," *Data Mining Knowl. Discovery*, vol. 33, no. 4, pp. 917–963, Jul. 2019.

[15] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Mining Knowl. Discovery*, vol. 28, no. 4, pp. 851–881, Jul. 2014.

[16] C. Hines and A. Youssef, "Machine learning applied to rotating check fraud detection," in *Proc. 1st Int. Conf. Data Intell. Secur. (ICDIS)*, Apr. 2018, pp. 32–35.

[17] C. Hines and A. Youssef, "Machine learning applied to point-of-sale fraud detection," in *Proc. Mach. Learn. Data Mining Pattern Recognit.*, in Lecture Notes in Computer Science, vol. 10934, 2018, pp. 283–295.

[18] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: The forecast package for R," *J. Stat. Softw.*, vol. 27, no. 1, pp. 1–22, 2008.

[19] C. Ji, S. Liu, C. Yang, L. Pan, L. Wu, and X. Meng, "A shapelet selection algorithm for time series classification: New directions," *Proc. Comput. Sci.*, vol. 129, pp. 461–467, 2018.

[20] I. Karlsson, P. Papapetrou, and H. Boström, "Generalized random shapelet forests," *Data Mining Knowl. Discovery*, vol. 30, no. 5, pp. 1053–1085, Sep. 2016.

[21] C. Kelly, "Experiential methods for identifying and reducing point of sale retail fraud," *EDP Audit, Control, Secur. Newslett.*, vol. 59, no. 5, pp. 13–20, 2019.

[22] R. Laimek, N. Kaothanthong, and T. Supnithi, "ATM fraud detection using outlier detection," in *Intelligent Data Engineering and Automated Learning* (Lecture Notes in Computer Science), vol. 11314. Cham, Switzerland: Springer, 2018.

[23] T. Le Nguyen and G. Ifrim, "A short tutorial for time series classification and explanation with MrSQM," *Softw. Impacts*, vol. 11, Feb. 2022, Art. no. 100197.

[24] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 10, 2021, doi: 10.1109/TNNLS.2021.3084827.

[25] J. Lines, S. Taylor, and A. Bagnall, "Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles," *ACM Trans. Knowl. Discovery Data*, vol. 12, no. 5, pp. 52:1–52:35, 2018.

[26] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones, "*catch22*: Canonical time-series Characteristics," *Data Mining Knowl. Discovery*, vol. 33, no. 6, pp. 1821–1852, Nov. 2019.

[27] M. Middlehurst, J. Large, and A. Bagnall, "The canonical interval forest (CIF) classifier for time series classification," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2020, pp. 188–195.

[28] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall, "HIVE-COTE 2.0: A new meta ensemble for time series classification," *Mach. Learn.*, vol. 110, nos. 11–12, pp. 3211–3243, Dec. 2021.

[29] M. Middlehurst, W. Vickers, and A. Bagnall, "Scalable dictionary classifiers for time series classification," in *Proc. Int. Conf. Intell. Data Eng. Automated Learn.*, in Lecture Notes in Computer Science, vol. 11871, 2019, pp. 11–19.

[30] L. Pantiskas, K. Verstoep, M. Hoogendoorn, and H. Bal, "Taking ROCKET on an efficiency mission: Multivariate time series classification with LightWaveS," 2022, *arXiv:2204.01379*.

[31] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, "Do vision transformers see like convolutional neural networks?" 2021, *arXiv:2108.08810*.

[32] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, "The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Min. Knowl. Discovery*, vol. 35, no. 2, pp. 401–449, Mar. 2021.

[33] H. Salehinejad, Y. Wang, Y. Yu, T. Jin, and S. Valaee, "S-rocket: Selective random convolution kernels for time series classification," 2022, *arXiv:2203.03445*.

[34] P. Schäfer and U. Leser, "Fast and accurate time series classification with WEASEL," in *Proc. ACM Conf. Inf. Knowl. Manag.*, Singapore, Nov. 2017, pp. 637–646.

[35] P. Schafer and U. Leser, "Multivariate time series classification with WEASEL+MUSE," in *Proc. ECML/PKDD Workshop AALTD*, 2017, pp. 637–646.

[36] D. Shaohui, G. Qiu, H. Mai, and H. Yu, "Customer transaction fraud detection using random forest," in *Proc. IEEE Int. Conf. Consum. Electron. Comput. Eng. (ICCECE)*, Jan. 2021, pp. 144–147.

[37] K. S. Lim, L. H. Lee, and Y.-W. Sim, "A review of machine learning algorithms for fraud detection in credit card transaction," *Int. J. Comput. Sci. Netw. Secur.*, vol. 21, no. 9, pp. 31–40, 2021.

[38] C. W. Tan, F. Petitjean, E. Keogh, and G. I. Webb, "Time series classification for varying length series," 2019, *arXiv:1910.04341*.

[39] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 1578–1585.

[40] J. West, M. Bhattacharya, and R. Islam, "Intelligent financial fraud detection practices: An investigation," in *Proc. Int. Conf. Secur. Privacy Commun. Netw.*, in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 153. Cham, Switzerland: Springer, 2015, pp. 186–203.
[41] E. Yemini, T. Jucikas, L. J. Grundy, A. E. X. Brown, and W. R. Schafer, "A database of caenorhabditis elegans behavioral phenotypes," *Nature Methods*, vol. 10, no. 9, pp. 877–879, Sep. 2013.
[42] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015, *arXiv:1511.07122*.
[43] Y.-L. Zhang, J. Zhou, W. Zheng, J. Feng, L. Li, Z. Liu, M. Li, Z. Zhang, C. Chen, X. Li, Y. Qi, and Z.-H. Zhou, "Distributed deep forest and its application to automatic detection of cash-out fraud," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 5, pp. 1–19, Sep. 2019.
[44] X. Zhang, Y. Gao, J. Lin, and C. Lu, "TapNet: Multivariate time series classification with attentional prototypical network," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 6845–6852.
[45] Z. Zhang, L. Chen, Q. Liu, and P. Wang, "A fraud detection method for low-frequency transaction," *IEEE Access*, vol. 8, pp. 25210–25220, 2020.

AGNIESZKA BIER received the M.Sc. (Eng.) degree in mathematics and the M.Sc. (Eng.) degree in computer science from the Silesian University of Technology, Gliwice, Poland, in 2005 and 2008, respectively, and the Ph.D. degree in mathematics from the University of Silesia, Poland, in 2010.

Since then, she has been working at the Faculty of Applied Mathematics, Silesian University of Technology, as an Assistant Professor. For over three years, she was the Leader of Data Science Team at DTiQ. Her research interests include certain topics in group theory and algorithmic problems on the border of mathematics and computer science.

AGNIESZKA JASTRZĘBSKA received the B.Sc. degree in information technology from the University of Derby, Derby, U.K., in 2009, the M.Sc. (Eng.) degree in computer engineering from the Rzeszow University of Technology, Rzeszow, Poland, in 2010, and the Ph.D. and D.Sc. degrees in computer science from the Warsaw University of Technology, Warsaw, Poland, in 2016 and 2021, respectively.

She is currently an Associate Professor at the Faculty of Mathematics and Information Science, Warsaw University of Technology, where she has been working, since the beginning of her research career. She joined DTiQ, in 2021, as a Machine Learning Specialist. Her research interests include machine learning, computational intelligence, and fuzzy modeling. She is an Associate Editor of the journal *Applied Soft Computing*.

PAWEŁ OLSZEWSKI received the B.Sc. degree in mathematics and the M.Sc. degree in computer science from the Białystok University of Technology, Białystok, Poland, in 2015 and 2017, respectively.

His research interest includes machine learning. Moreover, he has experience in applied research on abstract algebra. He has been a member of Data Science Team at DTiQ, since 2020.