Computer Science and Information Systems 20(3):1207–1233 https://doi.org/10.2298/CSIS220110005T

Matching Business Process Behavior with Encoding Techniques via Meta-Learning: An anomaly detection study*

Gabriel Marques Tavares[1] and Sylvio Barbon Junior[2]

[1] Università degli Studi di Milano (UNIMI), Milan, Italy
gabriel.tavares@unimi.it
[2] Università degli Studi di Trieste (UniTS), Trieste, Italy
sylvio.barbonjunior@dia.units.it

Abstract. Recording anomalous traces in business processes diminishes an event log's quality. The abnormalities may represent bad execution, security issues, or deviant behavior. Focusing on mitigating this phenomenon, organizations spend efforts to detect anomalous traces in their business processes to save resources and improve process execution. However, in many real-world environments, reference models are unavailable, requiring expert assistance and increasing costs. The considerable number of techniques and reduced availability of experts pose an additional challenge for particular scenarios. In this work, we combine the representational power of encoding with a Meta-learning strategy to enhance the detection of anomalous traces in event logs towards fitting the best discriminative capability between common and irregular traces. Our approach creates an event log profile and recommends the most suitable encoding technique to increase the anomaly detection performance. We used eight encoding techniques from different families, 80 log descriptors, 168 event logs, and six anomaly types for experiments. Results indicate that event log characteristics influence the representational capability of encodings. Moreover, we investigate the process behavior's influence for choosing the suitable encoding technique, demonstrating that traditional process mining analysis can be leveraged when matched with intelligent decision support approaches.

Keywords: Anomaly detection, Meta-learning, Encoding, Process mining, Recommendation.

1. Introduction

Organizations rely on the correct execution of business processes to achieve their goals. However, anomalous instances in event logs are harmful to process quality. This way, stakeholders are interested in detecting and mitigating anomalies so that business processes correspond to their expected behavior. Detecting anomalies is not only beneficial for resource-saving but also to avoid security issues [46]. Process Mining (PM) is devoted to extracting valuable information from organizational business processes. Within PM, conformance checking methods are dedicated to finding anomalies. Conformance

* This article is an extended version of the paper "Process Mining Encoding via Meta-Learning for an Enhanced Anomaly Detection" [41].

techniques compare process models and event logs, quantifying deviations and, consequently, identifying anomalies [39]. Traces not complying with the model are interpreted as anomalous, either from a control-flow or data-flow perspective.

Although conformance checking supports the recognition of anomalous traces, the methods are model-dependent, hindering their applicability since the model is not available in many scenarios. Moreover, resorting to expert knowledge is time-consuming and may often result in subjective and approximate decisions that do not allow the effective automation of PM-related tasks [30]. Various approaches have been proposed for detecting anomalies in business processes as a compliance verification task. For instance, Bohmer and Rinderle-Ma [10] use likelihood graphs to model process behavior and support anomaly detection. The method is applicable to control- and data-flow attributes, although the quality of discovered models limits its performance. Recently, many methods are relying on Machine Learning (ML). To overcome the mismatch at the representational level between PM and traditional data mining [16], a shared characteristic of ML-based approaches is to transform the process or trace representations. The feature vector obtained is capable of describing instances and connections for further ML modelling.

As pointed by Barbon et al. [6], a suitable encoding technique is crucial for the quality of posterior methods applied to the event log. This way, by finding the appropriate encoding, one can improve the identification of anomalous instances. In other words, a suitable encoding technique allows the best representation of typical behavior and adjusted discriminant capacity of anomalous traces. However, considering the multitude of available encoding methods, selecting the correct algorithm is challenging [24,38]. To grasp the algorithm selection problem in the context of PM, we propose an approach based on a Meta-learning (MtL) strategy to recommend the best encoding method for a given event log. The approach aims at maximizing the number of identified anomalies by profiling event log behavior based on its features. MtL has been applied as a recommender system, succeeding in emulating expert decisions for a wide range of applications [23,47]. Taking advantage of structural and statistical lightweight meta-features extracted from event logs, our MtL approach suggests the most suitable encoding technique. For that, it was built using 80 meta-features, trained over 168 event logs (meta-instances) for guiding the best one of eight promising encoding methods (meta-target). Moreover, the proposed approach is easily scalable, allowing the inclusion of additional encoding techniques and log descriptors. Results show that the MtL approach outperforms current baselines and can improve the detection of anomalous traces independently of the applied learning algorithm.

This work is an extension of [41]. The reference work investigated the MtL approach and compared its performance with baselines. In this extension, we propose a more in-depth analysis of the results and their insights, also including more encoding techniques. First, by adopting explainability techniques, we strengthen the understanding of process behavior's impact on the encoding quality. For that, we present an analysis connecting event log features to encoding methods, hence, providing a better problem understanding. Furthermore, we developed experiments to test the internal validity of our research design. The contributions of the presented approach are manifold. The main advantage is providing guided recommendations of the suitable encoding technique for a given event log, which is beneficial for both experts and non-experienced users. For end-users, the recommendation saves experimentation time and speeds up the pipeline design. For ex-

perts, the created meta-database serves as an analytical tool to correlate business process behavior with optimal techniques, giving further insight into the problem.

The remainder of this paper is organized as follows. Section 2 presents an extensive discussion about anomaly detection and the encoding of business processes. Section 3 lays down fundamental aspects regarding the application and where this problem sits in relation to the literature. Section 4 presents the MtL-based methodology proposed in this paper. Moreover, the section explores the extracted meta-features, and the encoding techniques used along with the MtL approach. Section 5 details the experimental setup (including event logs) and compares our approaches's performance with two baselines. Furthermore, a comprehensive anomaly analysis is shown along with a discussion about implications of the different anomalies. Lastly, Section 6 concludes the paper and highlights our contributions.

2. Related Work

This section explores the discussion in literature focusing on anomaly detection and encoding techniques. Since there are no works identifying the best encoding techniques for anomaly detection, the goal is to present how traditional anomaly detection is performed within PM and compare it with ML-based anomaly detection.

2.1. Anomaly Detection

Early on, Bezerra et al. [9] defined anomaly in business processes based on a set of assumptions: (i) the set of traces can be divided in normal and anomalous subsets, (ii) an anomalous execution is infrequent in comparison to normal executions, and (iii) normal behavior generates more comprehensive process models than models generated from anomalous traces. Within business process literature, traditional anomaly detection is performed by conformance checking techniques [9,8]. For that, the conformance approach is based on the comparison between process model and event log [1]. Therefore, non-compliant business instances are interpreted as anomalous while conforming ones are regarded as normal behavior. It is important to note that there are several different conformance checking methods and no consensus of which one is the best [26].

Most traditional techniques rely on token-based replay [9,39,3]. These approaches replay trace sequences into the model by consuming executed activities according to model constraints. Trace fitness is measured by counting missing and remaining tokens. For instance, Bezerra et al. [9] proposed a method that uses domain knowledge to filter the event log and then applied a process discovery technique to generate a model from the filtered log. Then, traces are classified based on model fitting (i.e., non-fitting traces are considered anomalous). However, more recent techniques rely on alignments due to being more robust, especially for logs with noise [26]. These techniques also propose a model-log comparison but directly relate a trace to valid execution sequences allowed by the model. Therefore, alignment methods apply a synchronous approach where normal behavior is defined by the accordance of moves between trace and model. Finally, multiple alignments can be produced, and the goal is to find the optimal one, which can be costly. Although used in industry, conformance checking techniques depend on a high-quality

process model, which is a constraint in many real scenarios. This happens because often models are unavailable, inadequate, or incorrect [5]. Moreover, creating or inferring them from data is complex as the discovery process has to balance between precision and generality [40].

To overcome the issue of identifying anomalous instances in scenarios without an available process model, ML-based techniques have been gaining traction in the literature. Nolle et al. [33] use an autoencoder to model process behavior and detect irregular cases. The same authors in another research use a deep neural network trained to predict the next event [34]. An activity with a low probability score (extracted from the network) is recognized as an anomaly. The paper of Tavares and Barbon ([40]) use language-inspired trace representations to model process behavior. Cases isolated in the feature space are identified as abnormal. Techniques based in deep learning such as [33,34] may be computationally expensive, while traditional ML techniques inject bias when representing traces in high-dimensional spaces.

2.2. Encoding Business Processes

A common characteristic of ML-based approaches is the need to transform the event log representation into formats expected by traditional ML techniques. In other words, one may need to apply a function that projects each trace to a n-dimensional feature space where the anomaly detection can take place. This transformation step is often referred to as encoding.

Trace encoding is often applied [11,17,18,27,44] in PM but has been shallowly discussed in the literature [6]. Due to a mismatch at the representational level, it is mandatory to apply trace encoding when applying data mining techniques to event logs. In the context of predictive models, Leontjeva et al. [27] proposed a sequence encoder based on Hidden Markov Models, whereas Polato et al. [37] used a last state encoding method. Word embedding techniques have also been applied [17]. Koninck et al. [17,40,22] experimented with word2vec and doc2vec to encode traces as words and paragraphs, respectively. Furthermore, the authors also extrapolated these encodings to other representational levels, such as logs and models. Hake et al. [22] used the same word2vec and combined it with recurrent neural networks to label nodes in business models.

Barbon et al. [6] identified three major encoding families: PM-based, word embeddings, and graph embeddings. PM-inspired encoding assumes that measures extracted from conformance checking techniques can be used as features and, hence, as an encoding technique. Word embeddings are traditionally associated with natural language processing and information retrieval. When applied to business processes, these techniques assume that activity names are words and sequence of activities are sentences. This interpretation allows the application of both shallow techniques such as one-hot encoding and modern approaches such as word2vec. It follows that activities and their direct-follows relationships can also be interpreted as nodes and edges in a graph. Naturally, business process models are also represented as graphs, which matches graph-based techniques. Therefore, graph embeddings, which follow up on word embeddings, are also applicable for trace encoding. In this scope, the authors assessed complementary perspectives in trace encoding techniques considering the complexity, time consumption and injected bias, among others. Overall, the authors found that there is no best encoding technique for every scenario (i.e., different event logs may be better encoded by different techniques).

2.3. Meta-learning in Process Mining

Recently, automation approaches based on MtL have been explored in PM [43,42]. The main hypothesis behind these works assumes a relationship between process behavior and optimal pipelines. Barbon et al. [43] use MtL to identify the discovery technique that maximizes performance in multiple criteria. Thus, given a new event log, its behavior is retrieved through descriptive features, which are then associated with the best discovery technique. A more complex issue is investigated in [42]. In this work, the authors propose a technique to recommend the complete pipeline for trace clustering. The pipeline includes the encoding technique, clustering algorithm, and its hyperparameters, and the step's intercorrelation makes the problem more challenging. In both works, experiments have surpassed baseline performances, indicating that there is indeed a relationship between process behavior and PM pipelines. In the same fashion, we extrapolate this assumption for encoding techniques. The aim is to improve ML-based anomaly detection by identifying the suitable encoding technique.

3. Problem Statement

Considering the multiple algorithms that encode event logs, choosing the best method is a challenging task even for experts. Furthermore, in many scenarios, not enough attention is given to this step in the PM pipeline. In this section, we present basic notions and state where the problem sits in the literature.

Definition 1 (Event, Attribute, Case, Event log). *Let Σ be the event universe, i.e. the set of all possible event identifiers. Σ^* denotes the set of all sequences over Σ. Let $\mathcal{AN}$ be the set of attribute names. For any event $e \in \Sigma$ and an attribute $a \in \mathcal{AN}$, then $\#_a(e)$ is the value of attribute n for event e. Let C be the case universe, that is, the set of all possible identifiers of a business case execution. C is the domain of an attribute $case \in \mathcal{AN}$. An event log L can be viewed as a set of cases $L \subseteq \Sigma^*$ where each event appears only once in the log.*

Definition 2 (Encoding). *Let an event log L, encoding is a function f_e that maps L to a feature space, i.e., $f_e : L \rightarrow \mathcal{R}^n$ where $\mathcal{R}^n$ is a n-dimensional real vector space.*

Encoding event logs bridges the gap between PM and data mining. That is, once the different log granularity levels are condensed into an n-dimensional numerical feature space, the combination with traditional data mining techniques is natural.

Definition 3 (Algorithm Selection Problem). *Let $x \in \mathcal{P}$ be a problem in a problem space, let $f(x) \in \mathcal{F}$ be a function that extracts features from the problem x, let $S(f(x))$ be a function that selects the mapping between the problem space to the algorithm space $A \in \mathcal{A}$, and let $p(A, x)$ be a function that maps the performance of an algorithm to the performance measure space. The goal is to determine $S(f(x))$ (the mapping of problems to algorithms) that maximizes the performance of the algorithm.*

In this context, MtL is a strategy to solve the algorithm selection problem. For that, meta-learning aims at mapping the relationship between the problem space and the algorithm performance space. Automatically selecting an encoding technique is, then, beneficial for the end-user. Moreover, it could also provide insights into event log behavior and optimal encoding methods.

4. Methodology

This section presents the proposed methodology to enhance anomaly detection in event logs. The method is based on the combination of encoding representational power with MtL as the learning paradigm. We also present the design details, including all steps of the MtL pipeline.

4.1. Meta-Learning for Anomaly Detection in Process Mining

In this work, we investigate encoding methods that boost the performance of traditional ML algorithms for the anomaly detection task in business processes. For that, our proposed approach relies on MtL. The primary assumption is that the event log characteristics (i.e., descriptors) can support the choice of the best encoding method. The best encoding is the technique that produces the highest anomaly detection rates (i.e., accuracy) when combined with a given ML-induced model. The boosted capability provided by a particular encoding method relies on the correctly and effectively discriminative capacity to represent traces [6]. However, identifying the best encoding is very tricky depending on the expert's experience in the particular domain. Here, we follow the assumption that this experience could be emulated using MtL.

Figure 1 presents an overview of our approach. Starting from a collection of event logs, the first step is the *Meta-feature extraction*, which mines descriptors that characterize the event logs. The extracted meta-features are capable of capturing the business process behavior from complementary perspectives. The idea is that the combination of multiple descriptors creates a representation of log behavior in a vector space. Next, we submit the event logs to encoding techniques. The encoding methods work at the trace level, and the encoded traces serve as input for an ML algorithm aiming to detect anomalies. Hence, we assess a performance metric (namely, F-score) that ranks the encoding algorithms for each event log. This step is called *Meta-target definition*, where each event log is submitted to all encoding methods, and the best encoding (meta-target) is identified by ranking performances. Then, the *Meta-database creation* joins the two previous steps. A database is created using meta-features (descriptors) and the meta-targets (best encoding for a given process) extracted from the logs. Consequently, each meta-instance is a set of log descriptors associated with an encoding technique that leverages anomaly detection performance for that event log. Once the meta-database is created, we induce a *Meta-model* in the *Meta-learner* step. The meta-model learns the distribution of the process data, hence, it associates process behavior to encoding techniques. This way, the meta-model is the final product of our workflow. Given a new event log, its meta-features are extracted and fed to the meta-model, which provides a data-driven recommendation of the best encoding technique for that event log.

4.2. Log descriptors – Meta-Feature Extraction

Extracting high-quality descriptors is fundamental for the performance of the meta-model. Moreover, meta-feature extraction should have a low computational cost, otherwise, the MtL pipeline is unjustified. This way, we selected a group of lightweight features that contains reliable representational capacities. To retrieve a multi-perspective view of event logs, we extract features from several process layers: activities, traces, and logs. These

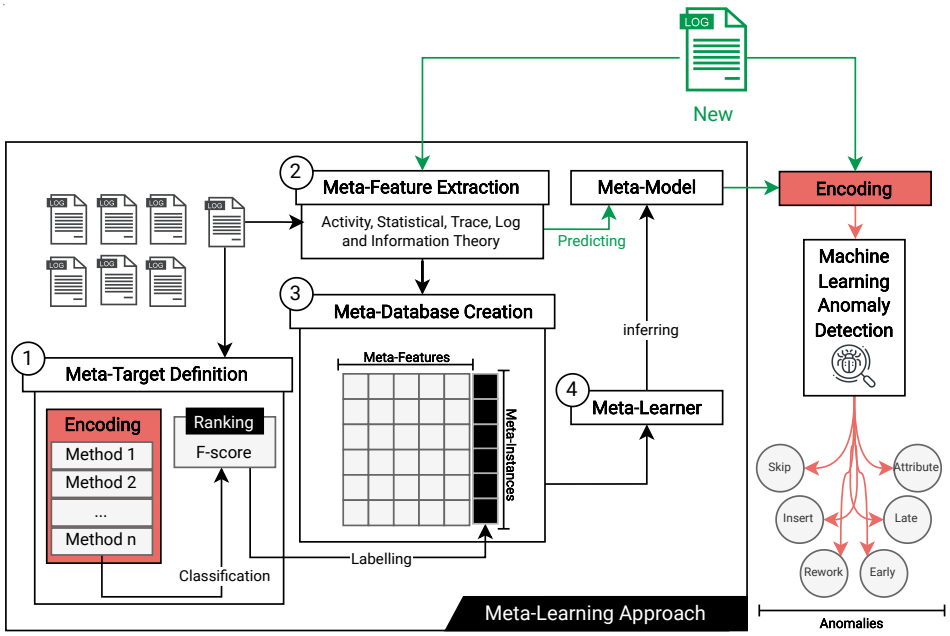


Fig. 1. Overview of the proposed approach for recommending encoding techniques based on process behavior. First, meta-features are extracted from a collection of event logs. Then, by applying each encoding technique in conjunction with a ML classification model, it is possible to rank their performance and find the best encoding method. By joining the two previous steps, we construct a meta-database, which in turn serves as the basis to infer a meta-model aiming at recommending an encoding technique for a new (unseen) event log

features were first proposed in [43], which combines business process features from different sources. Table 1 presents the features and groups they belong.

Overall, 80 features were extracted from the event logs. They capture complementary elements of business processes, containing information such as statistical dispersion, probability distribution shape and tendency, and log complexity.

4.3. Encodings – Meta-Target Definition

In this work, the application of encoding for anomaly detection in PM is a fundamental step toward building the meta-database. Ultimately, the encodings are the meta-targets associated with log features. Given a log and its meta-features, we associate it with an encoding that maximizes the anomaly detection performance. Therefore, encoding techniques play a major role as they can excel in detecting anomalous instances for certain types of log behaviors. The application of encoding in PM has already been explored by several researches [6,17,27,37]. Barbon et al. [6] extensively evaluated trace encoding methods using complexity metrics to assess encoding capacity. Moreover, the authors submit the encoding methods to a classification task for anomaly detection. The

Table 1. Meta-features extracted from event logs. This set of meta-features aim at capturing several complementary levels of business process' behaviors

Family	# Number	Name
all activities	12	number of activities, minimum, maximum, mean, median, standard deviation, variance, the 25th and 75th percentile of data, interquartile range, skewness, and kurtosis coefficients
start activities	12	number of activities, minimum, maximum, mean, median, standard deviation, variance, the 25th and 75th percentile of data, interquartile range, skewness, and kurtosis coefficients
end activities	12	number of activities, minimum, maximum, mean, median, standard deviation, variance, the 25th and 75th percentile of data, interquartile range, skewness, and kurtosis coefficients
trace lenght	29	minimum, maximum, mean, median, mode, standard deviation, variance, the 25th and 75th percentile of data, interquartile range, geometric mean and standard variation, harmonic mean, coefficient of variation, entropy, and a histogram of 10 bins along with its skewness and kurtosis coefficients
trace variants	11	mean, standard variation, skewness coefficient, kurtosis coefficient, the ratio of the most common variant to the number of traces, and ratios of the top 1%, 5%, 10%, 20%, 50% and 75% to the total number of traces
log level	4	number of traces, unique traces, traces ratio and the number of events

work proposes the application of three encoding families to event logs: PM-based encoding, word embedding, and graph embedding. The PM-based encodings are conformance checking techniques that compare an event log to a process model, measuring deviance and producing a fitness value along with token counting results [39]. Word embeddings can naturally be applied in event logs when considering activities and traces as words and sentences [5,17]. These techniques rely on context information captured by neural networks' weights trained for context prediction. Lastly, graph embeddings are techniques that encode graph information, such as nodes, vertices, and their attributes. Graph embeddings are particularly interesting in the PM domain as they can represent process models (with limitations such as not capturing concurrency) and traces, modeling entity links, and long-term relations.

Considering the three encoding families presented in [6], we selected representatives of each encoding family. This way, we aim to reduce the representative bias and evaluate if there is a relation between encoding techniques and log behavior. For PM-based encodings, we used alignments and token-based replay as they have been considered the state-of-the-art conformance checking method [15]. Alignments compare the event log and process model and measure the deviations between the two. For that, it relates traces to valid execution sequences allowed by the model. This evaluation unfolds into three types. Synchronous moves are observed when both the trace and model can originate a move. Model-dependent moves originate only from the model, and log-dependent moves are derived from traces but are not allowed by the model. Synchronous moves represent the expected behavior when model and log executions agree. The alignment technique

searches for an optimal alignment (i.e., when the fewest number of the model- and log-moves are necessary). This process, measured by a cost function, produces a fitness value and other statistics regarding the states consumed by the model. Token-replay follows a similar approach by comparing log and model. For that, it replays traces in the model by consuming executed activities according to model constraints. A conformance measure is produced based on missing and remaining activities.

Word embedding techniques in PM have mostly relied upon *word2vec* and *doc2vec* to encode traces. In Barbon et al. [5], the authors propose the word2vec encoding in conjunction with one-class classification to detect anomalies in business processes. Koninck et al. [17] use both word2vec and doc2vec to encode activity and trace information, respectively. Subsequent work also builds trace representations based on doc2vec [28]. In this work, we adopt several text-based encodings: count2vec, doc2vec, hash2vec, one-hot and word2vec. The count vectorizer (count2vec) encodes words by accounting their frequencies in a text document, producing a matrix of word counts. The one-hot encoding produces a similar matrix but binarizing the values, i.e., accounting for the appearance or not of a word in a document. For both techniques, feature length is determined by the number of unique words in the vocabulary, which tends to produce sparse vectors. The hash vectorizer (hash2vec) maps a feature with a word using a hashing function and computes frequencies based on previously mapped indices. Although hash2vec allows for predetermined vector sizes, hash collisions may happen. Doc2vec is an extension of word2vec adapted to documents, independently of their length. Word2vec creates numerical representations for words and, for that, a neural network is trained to reconstruct the linguistic context of words in a corpus [32]. The word embeddings come from the weights of the induced neural network. The main advantage is that words appearing in similar contexts produce similar encoding vectors. However, this method is limited to unique word representations. Doc2vec extends word2vec by adding a paragraph vector in the encoding process [25]. This way, the document context is captured by the encoding.

For the graph embedding family, we employed *node2vec*, another encoding technique built on top of word2vec. Node2vec's primary goal is to encode graph data while maintaining graph structure. Given a graph, node2vec performs random walks starting from different nodes [21]. This process creates a corpus, which is used as input for word2vec. The second-order random walks balance a trade-off between breadth and width, capturing neighbor and neighborhood information. Hence, the method can represent complex neighborhoods given its node exploration approach.

Using the eight defined encodings (alignment, count2vec, doc2vec, hash2vec, node2vec, one-hot, token-replay and word2vec), we performed the *Meta-target definition* step shown in Figure 1. The goal is to identify which encoding enhances the detection of anomalous traces in event logs. For that, we applied a traditional ML pipeline where we combine an encoding technique with a classification algorithm for detecting anomalous traces. This way, each log is first encoded, then, its traces are divided into an 80%/20% holdout strategy where 80% of traces are used for model inferring while 20% of traces are used for testing (i.e., evaluating if the trained model can correctly label traces). This process is repeated 30 times with different splits to avoid outlier performances and biased splits. We employed the Random Forest (RF) algorithm [12] in the meta-target definition step due to its robustness and easiness of interpretability. After 30 iterations, the average F-score for each technique is obtained, allowing us to rank the encoding techniques based

on their average performance. Thus, the best encoding technique for a given log is the one that produces the highest average F-score when combined with an ML technique for the anomaly detection task. We chose F-score as the ranking metric as it successfully balances different performance perspectives.

As an example of the ranking step, consider event log L, encoding techniques E_1, E_2 and E_3, and a classification model M. We submit L to E_1 and combine it with M to detect anomalous traces. Following the described steps, this process is repeated multiple times and the average F-score of M is retrieved. The same is done for E_2 and E_3. Since the model is the same, the only variable influencing the final performances is the encoding method. Let 0.75, 0.6 and 0.85 be the average F-score for E_1, E_2 and E_3, respectively. Therefore, the encoding technique that enhances anomaly detection in this scenario is E_3 because it produces the highest average F-score when combined with M. This way, the meta-target definition associates the meta-target (E_3) with its meta-instance (L), as shown in Figure 1.

4.4. Meta-database and Meta-model

We create the meta-database by combining the meta-features extracted from the logs with the defined meta-targets. Once the meta-database is built, the meta-learner step takes place. The meta-learner embeds a traditional ML pipeline, that is, the meta-database is submitted to an ML algorithm that infers a meta-model. The meta-model is the final object produced by the MtL approach. This way, given a new event log (previously unseen), its meta-features are extracted and, based on them, the meta-model is able to recommend a suitable encoding technique. Furthermore, the meta-database is also an interesting byproduct because it provides a mapping between log behavior (meta-features) and optimal encoding techniques (meta-targets). Thus, it can be used to analyze the relationship between the problem space and the performance space.

5. Evaluation

In this section, we present the performance of our approach and compare it with two baseline methods. Moreover, we develop a discussion regarding the impact of the anomalies in the encoding.

5.1. Experimental Setup

This section describes the main aspects of the experimental configuration, such as the used event logs and the algorithm applied in the meta-learner step.

Event logs – Meta-instances The more instances available, the more representative is the meta-database as it contains more examples of business process behaviors. Experiments on anomaly detection benefit from labeled datasets since one can compute traditional performance metrics to evaluate if anomalous cases are indeed captured. Considering these constraints, we built our meta-database from two groups of synthetic event logs composed of a wide range of behaviors originating from 12 different process models and

disturbed by six types of anomalies. We highlight that the use of real event logs is not possible in this scenario as the performance assessment relies on labeled event logs at the trace level. That is, information regarding normality or abnormality of traces is needed. Therefore, since we suffer from the availability of benchmark data and labeled event logs, we chose to proceed with synthetic event logs where we can assert which traces are normal or anomalous. Thus, avoiding bias in the interpretation of results.

The first group of event logs was initially presented by Nolle et al. [34] and replicated in [5] and [40]. Six models were generated using the PLG2 tool [14]. PLG2 randomly generates process models representing several business patterns such as sequential, parallel, and iterative control-flows. Moreover, PLG2 allows the configuration of the number of activities, breadth and width, hence, providing a complex set of models that capture diverse behavior. One additional process model, P2P, extracted from [35] was added to the pool. Then, the authors adopted the concept of likelihood graphs [10] to introduce long-term control-flow dependencies. The likelihood graphs can mimic complex relations between event to event transitions and attributes attached to these events. This way, the control-flow perspective is constrained by probability distributions that coordinate the model simulation. For instance, an activity may follow another given a probability. Combining stochastic distributions with a set of process models leverages the similarity between produced event logs and real-world logs. Four event logs were simulated in each process model, generating a total of 28 logs. The final step added anomalies to the traces within the synthetic event logs, which is a traditional practice in related work [9,10]. We applied six anomaly types for all event logs with a 30% incidence: i) *skip*: a sequence of 3 or less necessary events is skipped; ii) *insert*: 3 or less random activities are inserted in the case; iii) *rework*: a sequence of 3 or less necessary events is executed twice; iv) *early*: a sequence of 2 or fewer events executed too early, which is then skipped later in the case; v) *late*: a sequence of 2 or fewer events executed too late; vi) *attribute*: an incorrect attribute value is set in 3 or fewer events.

The second group of synthetic event logs was proposed by Barbon et al. [6]. The authors also used the PLG2 tool to create five process models representing scenarios of increasing complexity (i.e., a higher number of activities and gateways). Then, the process models were simulated using the *Perform a simple simulation of a (stochastic) Petri net* ProM plug-in[3], producing 1000 cases for each log. As a post-processing step, the same anomalies used for the previous set of logs were applied in this set but with different configurations. The authors implemented four anomaly incidences (5%, 10%, 15% and 20%). Moreover, the dataset contains binary and multi-class event logs, meaning that some logs incorporate normal behavior and only one anomaly type (binary), and some logs contain both normal behavior and all anomalies at the same time (multi-class). The latter configuration is especially challenging given the higher complexity as more behaviors are present in the same log. In total, this set contains 140 event logs.

For all event logs, anomalies sit on the event level, but they can be easily converted to the case level. That is, cases containing events affected by any anomaly are considered anomalous cases. Table 2 shows the event log statistics for all event logs used in this work. As demonstrated, the set of logs presents a significant behavioral variation because they contain several different anomalies (combined in binary and multi-class scenarios),

[3] http://www.promtools.org/doku.php

injection rates, and process characteristics. These characteristics support the creation of a heterogeneous meta-database, increasing business process representability.

Table 2. Event log statistics: each log contains different levels of complexity

Name	#Logs	#Cases	#Events	#Activities	Trace length	#Variants
P2P	4	5k	38k-43k	25	5-14	513-655
Small	4	5k	43k-46k	39	5-13	532-702
Medium	4	5k	28k-31k	63	1-11	617-726
Large	4	5k	51k-57k	83	8-15	863-1143
Huge	4	5k	36k-43k	107	3-14	754-894
Gigantic	4	5k	28k-32k	150-155	1-14	693-908
Wide	4	5k	29k-31k	56-67	3-10	538-674
Scenario1	28	1k	10k-11k	22-380	6-16	426-596
Scenario2	28	1k	26k	41-333	23-30	1k
Scenario3	28	1k	43k-44k	64-348	39-50	1k
Scenario4	28	1k	11k-13k	83-377	1-30	383-536
Scenario5	28	1k	18k-19k	103-406	1-37	637-737

To avoid concerns regarding synthetic data representativeness, we further assess the distribution of log behavior compared to real event logs. For that, we extracted all meta-features (listed in Table 1) of the synthetic event logs plus three real business processes (Business Process Intelligence Challenges (BPIC) 2012[4], BPIC 2013 and helpdesk[5]). Then, we applied a dimensionality reduction technique, namely, the Principal Component Analysis (PCA) algorithm [45], to visualize the event logs in the feature space. Figure 2 presents the results of the feature space reduced to two dimensions and populated by the event logs. The union of the two Principal Components (PC) explains 92.30% of the data variance. Therefore, most distances in the higher dimension space are respected after the dimensionality reduction. Figure 2 shows that the real event logs sit close to the synthetic event logs in the feature space. Therefore, although no real event logs are employed in the experiments (due to the lack of labels), this analysis indicates that the synthetic event logs are representative of real scenarios.

A complementary mean of assessing data behavior is by extracting the log complexity. For that, we retrieved the complexities of all synthetic event logs and the same three real processes. The measure chosen was the normalized variant entropy, recently proposed by Augusto et al. [4], and is based on graph entropy. According to the authors, graph-based entropy is particularly suited for event logs as it captures size, variation and distance in an integral way. Figure 3 presents the normalized variant entropy distribution in the form of a boxplot. The three red dots refer to the real event logs. As we can see, groups of logs spread accross the x-axis. The complexity score for real event logs is similar to the synthetic logs, reaffirming the suitability of the synthetic business processes.

[4] https://www.tf-pm.org/resources/logs
[5] https://doi.org/10.17632/39bp3vv62t.1

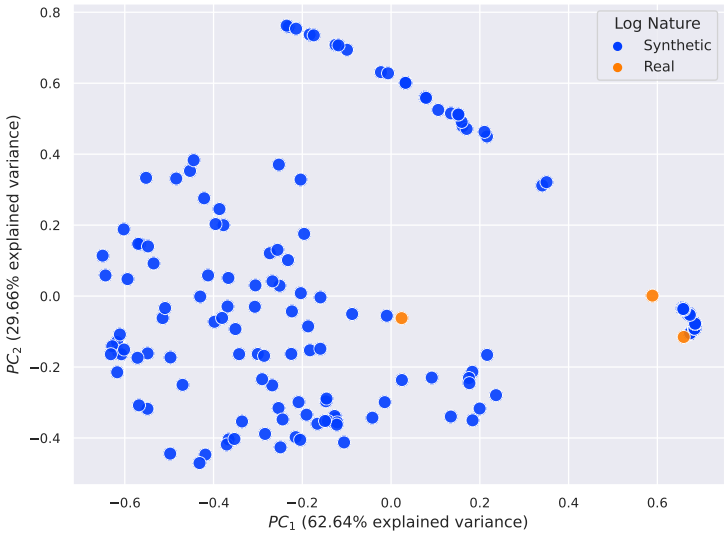


Fig. 2. Reduced feature space after applying PCA. Meta-features capturing log behavior are extracted from both real and synthetic event data. The real event logs populate the same region in the feature space as the synthetic ones

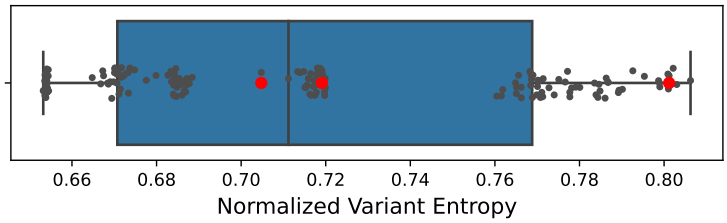


Fig. 3. Boxplot representing the normalized variant entropy extracted from all event logs. Red dots are the three real event logs

Meta-model The last step of the methodology is inferring the meta-model from the meta-data (as seen in Figure 1). In this scenario, we transform the algorithm selection problem into a classification problem. This way, we enable the use of traditional ML classifiers for this task. The meta-model creation is a step related to the usage of a supervised ML algorithm to map meta-features as an inference model. Our proposal was fashioned as a framework, allowing expansions and recombination of algorithms following the constraints established. Considering consistency and the same arguments presented in Section 4.3, we also employ the RF in this step. In particular, a RF model can provide insights into the quality of the features and their contribution to the prediction procedure. A RF model is composed of decision trees as base learners, which are built using the most informative features, reducing the number of features such as a feature selection step. Another impor-

tant aspect is the high predictive power, a simple tuning procedure, and also the reduced possibility of generating an overfitted model. Furthermore, we clarify that deep learning methods are not suitable for tabular data and, hence, cannot be used in this context. To measure the performance of our approach (explored in Section 5.2), we use a traditional configuration of ML pipelines. For that, we divide the meta-instances into two sets: 80% are used for model training while the remaining 20% for performance measurement. This holdout strategy is repeated 30 times (with different data splits) to avoid outlier performances. We assess both F-score and accuracy and report the results in comparison with two baselines.

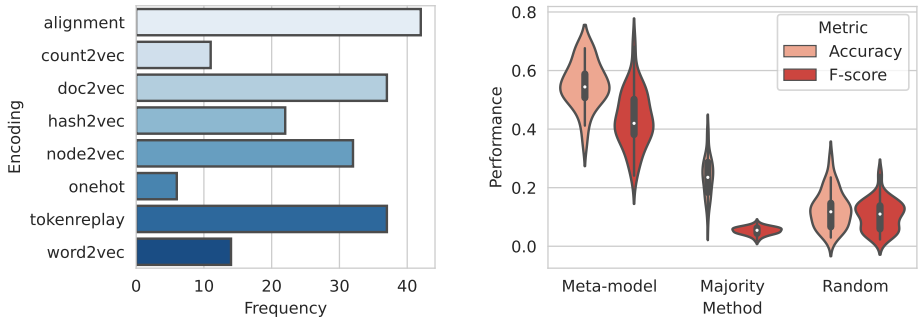
5.2. Results and Discussion

This section reports the MtL recommendation results. Moreover, it provides an in-depth exploration of anomalous scenarios and their relationship with encoding techniques. Finally, we assess the relevant features used by the meta-model and test the validity of the experimental design.

Meta-learning Performance First, we report the results of the meta-target definition step (i.e., ranking the encodings for each meta-instance). Since we are following a data-driven strategy, it is worth observing the balance regarding meta-targets. For that, Figure 4a shows the frequency of the encoding techniques in the first position in the ranking step (see Section 4.3). The ranking is built using the average F-scores obtained by each encoding algorithm. This analysis brings insights about the balanced scenario when selecting an encoding technique, illustrating the "no free lunch theorem" [2]. Four encoding techniques appeared most frequently in the first position. The alignment method was the best encoding for 42 event logs, while doc2vec was optimal for 35 logs, token-replay for 30 logs and node2vec for 27 logs. In other words, Figure 4a shows that alignment was the meta-target for 42 meta-instances, doc2vec was the meta-target for 35 meta-instances, and token-replay and node2vec were the meta-target for 30 and 27 meta-instances, respectively. These results highlight a balanced distribution between the best-ranked encodings. Regarding the other encodings, one-hot and count2vec were the least frequent best encodings. This outcome is expected as these encodings are very shallow, produce sparse vectors, and do not capture process constraints, such as loops. Word2vec also performed poorly mostly because of the small vocabulary of event logs. This technique might require more examples to produce its best outputs. Finally, hash2vec did not perform so well due to collisions in the mapping space, which decrease the encoding quality and, hence, harm the anomaly detection performance.

Figure 4b reports the performance of our approach for the task of recommending the encoding technique that leverages anomaly detection in event logs. Considering the lack of literature in the area, we compare the MtL performance with two baselines: majority and random selection. Majority regards the encoding method with the highest frequency in the meta-database, hence always recommending the alignment encoding. Although a simple baseline, majority voting is a suitable comparison in ML applications, clearly specifying the minimum performance threshold. Random selection works by arbitrarily selecting one of the possible meta-targets for each event log. This approach simulates a PM practitioner in a scenario without the availability of experts, a common situation in

real environments. From both accuracy and F-score perspectives, our approach outperforms the others with a large advantage. The meta-model obtains an average accuracy of 55% and an average F-score of 0.43. The violin visualization also demonstrates the MtL robustness since the density curve is compressed (i.e., most recordings are near the average mark). The majority approach produced accuracy and F-score averages of 24% and 0.05, respectively. The random method achieves 13% accuracy and 0.11 F-score. It is worth mentioning that these results report the performance for selecting the best encoding method.



(a) Encodings ranking frequency **(b)** Recommendation performance

Fig. 4. Encoding ranking extracted from the Meta-target definition step. The ranking is ruled by the F-score obtained by recommending a suitable encoding technique. Given a new event log, the meta-model recommends the best encoding considering the meta-features derived from the log. Fig. 4b demonstrates the meta-model performance in comparison with baseline approaches

Anomaly analysis As introduced in Section 5.1, the event logs contain six different anomaly types. Moreover, a subset of the logs is struck by all six anomalies at the same time. Considering the anomaly perspective, Table 3 reports the F-score performance of all encodings and compares them with the MtL approach. Naturally, detecting anomalous instances in logs affected by *all* anomalies is the most difficult task. Hence, the F-score values are the worst in this scenario. At the same time, this scenario is the most common in real environments, where event logs contain traces with multiple deviation types. Nonetheless, we observe that MtL reports the highest mean F-score, reaching 0.49. It is followed by alignments (0.47), doc2vec (0.43), hash2vec (0.43) and node2vec (0.43). The performance rises considerably in the other anomaly types as the problem is binary in these cases.

Insert, *rework* and *skip* are the most detectable anomalies because they deeply affect the control-flow perspective of traces, therefore, this behavior change is easily captured by the encodings. For these anomalies, MtL reaches the highest performance values, producing F-score values close to 1. For *rework* and *skip* anomalies, hash2vec and node2vec

Table 3. Comparison of anomaly detection performance using fixed encoding methods and MtL recommendation. Mean and standard deviation (in parenthesis) F-score values are reported for each anomaly type. Bold values indicate the best method for each anomaly

Encoding	*all*	*attribute*	*early*	*insert*	*late*	*rework*	*skip*
alignment	0.47 (0.15)	0.92 (0.04)	0.93 (0.04)	0.98 (0.02)	0.93 (0.03)	0.97 (0.02)	0.98 (0.02)
count2vec	0.42 (0.15)	**0.93** (0.04)	0.92 (0.04)	0.93 (0.03)	0.93 (0.04)	0.98 (0.01)	0.98 (0.01)
doc2vec	0.43 (0.21)	**0.93** (0.03)	0.94 (0.03)	0.93 (0.03)	**0.94** (0.03)	0.94 (0.03)	0.95 (0.03)
hash2vec	0.43 (0.13)	**0.93** (0.04)	0.93 (0.04)	0.98 (0.01)	0.93 (0.04)	**0.99** (0.01)	**0.99** (0.01)
node2vec	0.43 (0.12)	**0.93** (0.04)	0.92 (0.04)	0.98 (0.01)	0.93 (0.04)	**0.99** (0.01)	**0.99** (0.01)
onehot	0.31 (0.1)	**0.93** (0.04)	0.92 (0.04)	0.93 (0.03)	0.93 (0.04)	0.93 (0.04)	0.98 (0.01)
token-replay	0.36 (0.08)	**0.93** (0.03)	0.94 (0.03)	0.96 (0.02)	**0.94** (0.03)	0.97 (0.02)	0.96 (0.02)
word2vec	0.4 (0.14)	**0.93** (0.04)	0.92 (0.04)	0.98 (0.02)	0.93 (0.04)	0.98 (0.02)	0.98 (0.02)
MtL	**0.49** (0.15)	**0.93** (0.03)	**0.95** (0.03)	**0.99** (0.01)	**0.94** (0.03)	**0.99** (0.01)	**0.99** (0.01)

tie with the MtL approach. Several other encodings follow closely, such as alignment, count2vec and word2vec. The encoding order changes when observing *early* and *late* anomalies. In these scenarios, MtL remains the best technique, reaching 0.95 F-score for *early* and 0.94 F-score for *late*, but now is followed more closely by doc2vec and token-replay, both reaching 0.94 in the two anomalies. Finally, all techniques (except alignments) tie for the *attribute* anomaly. Interpreting performance from the anomaly perspective reinforces the hypothesis that encodings perform differently in different scenarios, that is, log behavior is determinant when choosing the appropriate encoding. Hence, the MtL efficiency in this experiment exposes the influence of event log behavior on the encoding representational power. The results indicate that anomaly detection is enhanced when the relationship between event log descriptors and encodings is appropriately mapped. This mapping is mastered by our proposed MtL method, which outperforms the use of fixed encodings for all event logs.

We compared the F-score obtained by classifying all event logs using statistical analysis grounded on the non-parametric Friedman test to determine any significant differences between the usage of a unique encoding technique and meta-recommended ones. We used the post-hoc Nemenyi test to infer which differences are statistically significant [19]. As Figure 5 shows, differences between populations are significant, i.e., the MtL framework is statistically superior to other methods. Furthermore, other groups with no significant difference can be identified, e.g., aligment, node2vec, hash2vec and token-replay. One-hot encoding was statistically the worst performing encoding technique, separated from other groups and algorithms. Thus, MtL for recommending individual encoding methods to maximize the predictive performance achieved superior results statistically different from the usage of only one encoding. In other words, the performance obtained using MtL was statistically superior to a single encoding technique.

Meta-model Comparison To better assess meta-model performance, we evaluated the same recommendation problem using several traditional classifiers. The choice of classifiers used is based on the relevance in the ML literature and their different nature, therefore, possibly capturing if some heuristics influence in the recommendation quality. Along with RF, the classifiers are: Decision Tree (DT) [13], Logistic Regression (LR), Support

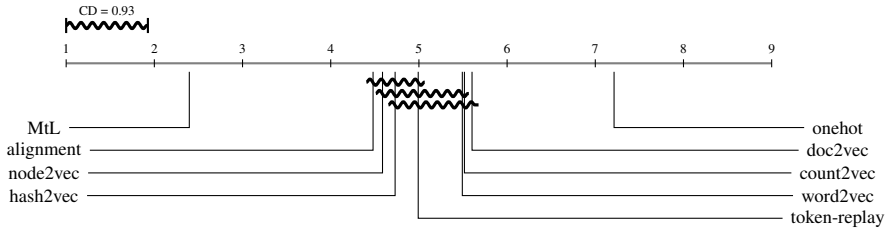


Fig. 5. Nemenyi post-hoc test (significance of $\alpha = 0.05$ and critical distance of 0.93) considering the F-score obtained from all event log classifications

Vector Machine (SVM) [36], k-Nearest Neighbor, and Gradient Boosting (GB) [20]. Figure 6 exposes the performances (accuracy and F-score) of all classification algorithms (in this case, meta-models). RF and GB appear as the best meta-models for both metrics, producing the highest average performances. In terms of F-score, SVM and kNN perform poorly with a significant distance to other methods. DT and LR remain in the middle group, not achieving the best performances, but producing a more stable recommendation performance than the worst algorithms.

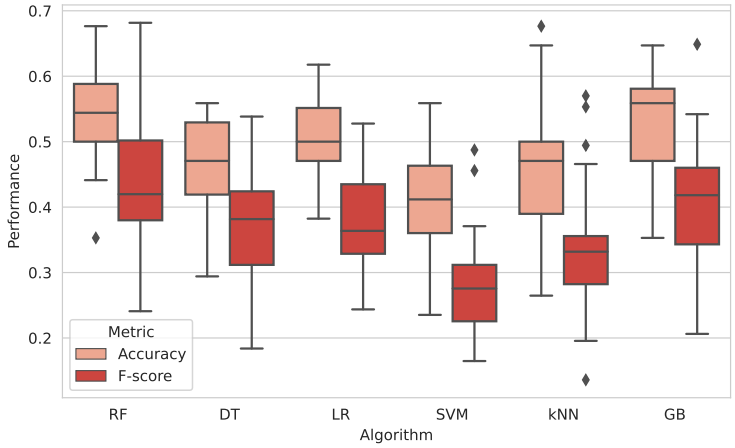


Fig. 6. Performance report for several classifiers in the recommendation task. RF and GB stand out as the most appropriate meta-models

To complement the previous analysis, we performed a statistical analysis again grounded on the non-parametric Friedman test. Figure 7 presents the results of the statistical test. As identified before, RF and GB are not statistically different within themselves. However, this group is superior to other algorithms as their distance is higher than the critical distance. The second group (GB, DT, LR and kNN) separates itself from SVM, which performs quite poorly in general. This analysis confirms the suitability of the RF as

a stable algorithm that performs well without the need of tuning. Automatic ML methods could be used to improve even further the configuration of the meta-model, although this is out of scope of the current research.

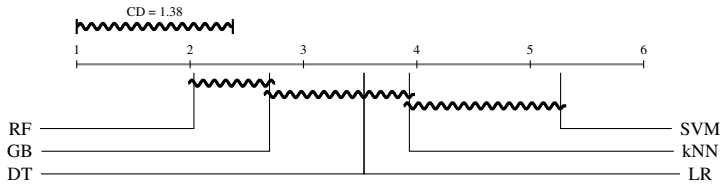


Fig. 7. Nemenyi post-hoc test (significance of $\alpha = 0.05$ and critical distance of 1.38) of several classifiers considering the F-score obtained for recommending the optimal encoding technique

Matching Process Behavior and Encoding Techniques With the goal of providing a more refined analysis, in the next sections we reduce the set of encoding techniques to the best representative of each family: alignment, doc2vec and node2vec. Moreover, considering the results from the previous test, we use the RF algorithm as the meta-model. The recommendations provided via MtL are based on data-driven assumptions constructed using meta-features from historical data. In this section, we present the meta-feature importance for predicting and explain the impact of the meta-features for recommending the encoding method (meta-target) in each anomaly context. The RF importance of the obtained model was used to discuss the relevance of features. To enlighten the interpretation of models, we exploit the Shapley Additive Explanations (SHAP), proposed in [29], explaining the obtained predictions in the different anomaly environments. We chose to apply two interpretation methods (RF importance and SHAP) because they are complementary and capture different aspects of feature relevance. By using RF importance, we can assess the most impactful meta-features for predicting any meta-target. As expected though, these importances are influenced by the bias of the ML model. Complimentarily, SHAP provides a more in-depth analysis, both at the class level (meta-targets) and for subsets of meta-instances (e.g., groups of specific anomalies). This way, we aim to provide a comprehensive study of the relationship between process behavior and optimal techniques.

Figure 8 shows all 80 features sorted by importance (i.e., ranked using RF importance) and colored using different colors for each feature family (*activity*, *trace*, and *log*). The top-ranked features were from trace and activity families with 65% and 35%, respectively, in the top 20 most important features. In general, trace-related features were the most successful in capturing the process behavior, which indicates that information sitting on the case-level may be more important than event-level information when describing normal and anomalous executions. A suitable explanation for such a pattern is that trace features have access to the complete case context. Thus, the difference between cases is more easily detected by this feature family. Particularly, $trace\_len\_entropy$, $trace\_len\_hist6$ and $trace\_len\_skewness\_hist$ were the best from this group. $Trace\_len\_entropy$ captures

the entropy of trace lengths of a process, meaning that anomalies affect the distribution of trace sizes, which turns to be an efficient indicator of process behavior. $Trace\_len\_hist6$ and $trace\_len\_skewness\_hist$ refer to a particular histogram bin (the sixth) and the skewness of this histogram, respectively. Histograms are fair data descriptors and have already been used in the PM domain [7]. These results indicate that processes have different distributions, and their asymmetry is an important indicator of process behavior. Since anomalies directly affect the executed activities, activity-based features also produce meaningful content. The standard deviation of the number of unique ending activities ($end\_activities\_std$) was the most influential feature from this family. Indeed, anomalies affecting the trace tail may be more detectable since the number of possible end activities is more limited. Many of the most important activity-level descriptors are related to start and end activities variance, implying that anomalies substantially affect activity distribution. Log-based features were considered the less important by RF, and none was featured in the top-20. The best-ranked feature from this family is the number of events in the process. On the other hand, the number of traces was useless for this context. The number of variants and their ratio were also less relevant for choosing the appropriate encoding. Usually, trace variants are an indicative of log complexity, and for traditional PM tasks such as process discovery and conformance checking, the number of variants is very influential on the results. However, in this scenario where we aim at finding the best encoding method, variants are less influential because word embedding and graph embedding techniques are used to deal with more extensive corpora. This way, the application of these methods cannot be easily identifiable by the number of variants or their ratio. We also observe that features based on simpler statistical tools such as average and median were predominantly less important than features relying on more powerful statistical concepts.

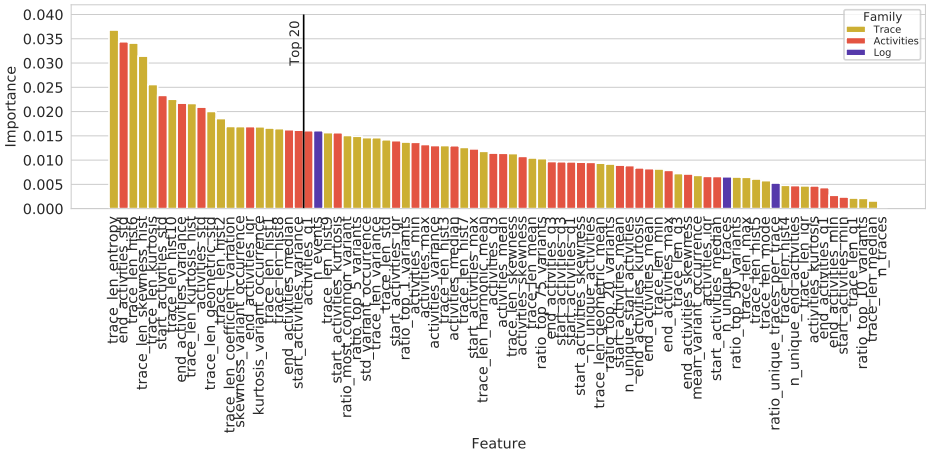


Fig. 8. RF feature importance sorted by their importance and colored by feature family

It is expected that different anomaly scenarios demonstrate particularities when having their pattern modeled, which demands information provided by specific groups of features. We used the SHAP method to comprehend these particularities and pave the way for an extended discussion on the features' importance and their contribution to the description of particular scenarios.

The top 20 most impacting features addressing the recommendation of encoding methods under all anomalies under study are shown in Figure 9. In a scenario comprised of several anomalies, the SHAP method indicated the impact of $trace\_len\_hist6$ and $end\_activities\_std$ as the most influential features. The former mostly supports the decision between doc2vec and node2vec, with a smaller impact when choosing alignment as the suitable encoding method. Particularly, $trace\_len\_hist6$ was remarkably important for doc2vec, meaning that particular distributions are better encoded by this method. Furthermore, $end\_activities\_std$ is more suitable to indicate a decision between alignment and node2vec, being the most important meta-feature in recognizing the suitability of node2vec for an event log. Moreover, $end\_activities\_std$ and $start\_activities\_std$ (both from the activity family) were the most influential features when associating a process with alignment. Indeed, the configuration of starting and ending activities highly affects the alignment's performance, hence, it was clear to the meta-model how these features affected the decision for alignment. On the other hand, descriptors such as the entropy of trace lengths, number of events, and most frequent activity had a minor impact.

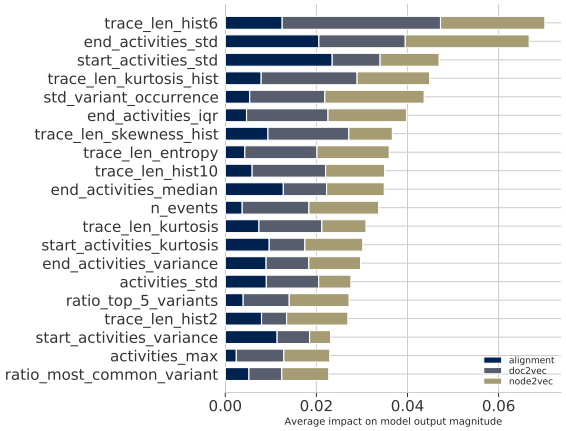


Fig. 9. Top 20 meta-features ranked using SHapley Additive exPlanations (SHAP) for all anomalies

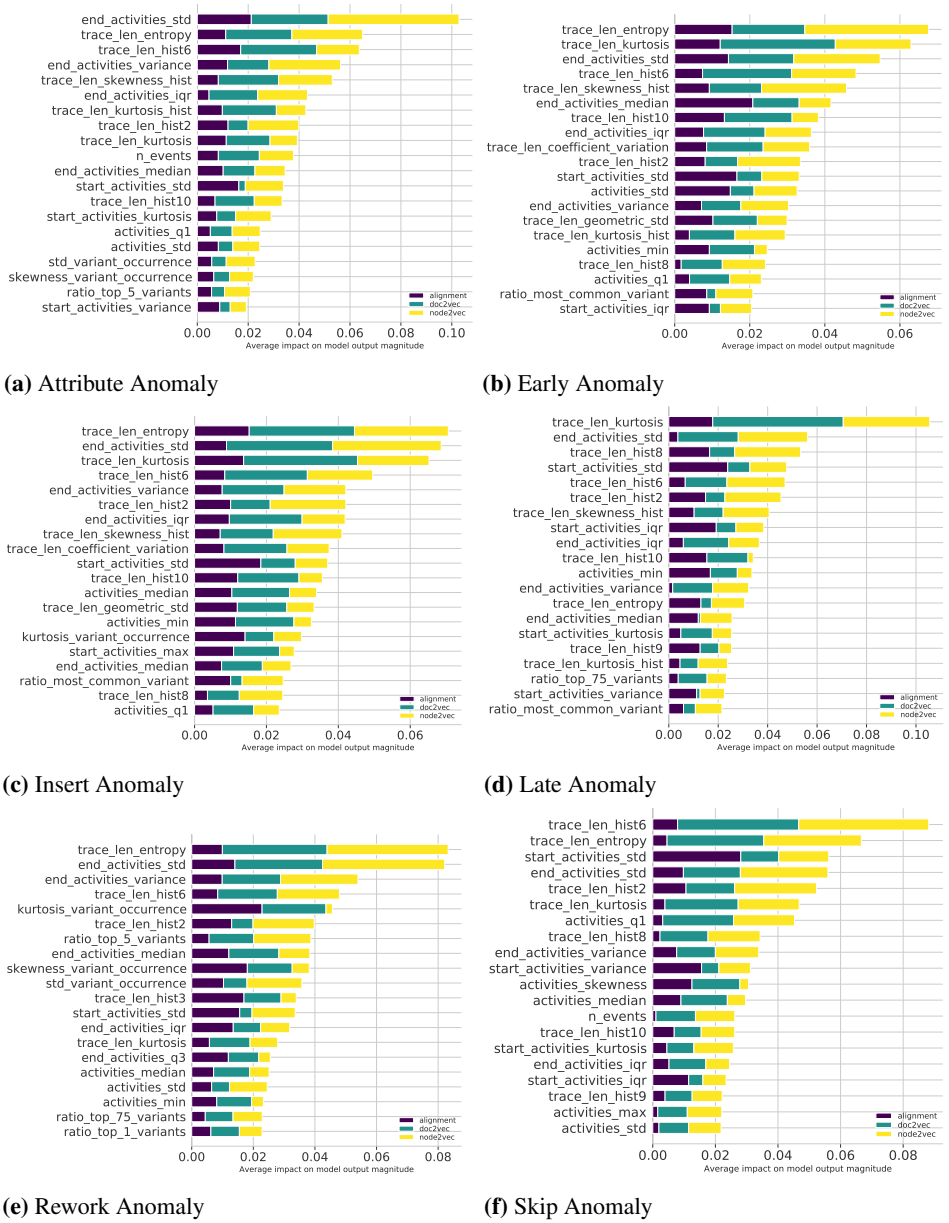
Aiming at a better comprehension of the relationship between anomaly patterns and process behavior, Figure 10 depicts the most influential features for each anomaly type (*attribute*, *early*, *insert*, *late*, *rework*, and *skip*). As in the previous analysis, $end\_activities\_std$ and $trace\_len\_entropy$ had a high impact for most anomalies, confirming again their competence of correctly capturing the process behavior for the studied problem. The exception was the *late* anomaly, in which $trace\_len\_entropy$ occupied only the 13^{th} posi-

tion in the rank. In particular, in a scenario compromised by *late* anomaly (Figure 10d), the $trace\_len\_kurtosis$ performed an important descriptive role, reaching a value that is double of the value of the 2^{nd} most impacting feature. This scenario exposes the importance of observing when a sequence of events is executed too prematurely, creating a deviation in the normal distribution, which is then captured by the $trace\_len\_kurtosis$ meta-feature.

When analyzing the relationship between features importance and specific anomalies, we observe that in *late* and *skip* anomalies, the meta-features had a lower impact in deciding for the alignment method. On the other hand, in *insert* and *rework* the features had a greater impact on the same technique. Indeed, the latter anomalies may be more easily detected by the alignments. The *late* anomaly is also challenging for the other encodings (i.e., the importance for node2vec and doc2vec was lower with this deviation). The highest average impact for doc2vec and node2vec happened for *insert* and *early* anomalies, respectively.

The single feature from log family present among the top 20 was n_events. This feature contributed to identifying *attribute* and *skip* anomalies, consequently also appearing when recommending an encoding method in a scenario with all anomalies (Figure 9). Again, the trace and activity families were predominant in the top 20 for all anomalies. Although different features are selected and sorted among the top 20, the contributions are similar to those presented using RF importance. In other words, all the presented meta-features contributed to provide the recommendation by the proposed MtL approach, except for n_traces. Prematurely ignoring some characteristics interferes in the recommendation of particular scenarios, indicating that the high descriptive power is reached by the combination of multiple descriptors. Such outcome indicates that including additional features might lead to better representations and, hence, an improved recommendation performance. Furthermore, by using the RF algorithm when modeling the recommendation system, we obtained a model composed only of features able to contribute to the final prediction since the RF properties guarantee a model that is built upon features capable of supporting the most accurate results. Overall, considering the results of both analysis, we can conclude that process behavior (captured by meta-features) does influence the quality of the encoded event log. Therefore, characteristics of the underlying business process should be considered when choosing an appropriate encoding technique, a problem that is tackled by our proposal.

Experimental validity Mendling et al. [31] raised a discussion about algorithm engineering and its impact on accuracy results, questioning if improved performances are a merit of algorithm design or a result of biases in both data and technique. We adopted their proposed framework to assess the internal validity of our research design and experiments. As stated by the authors, when the manipulation of research artifacts is causally responsible for an effect, its research design is internally valid. Therefore, experiments built on randomization constrain the effect of confounding factors. Since our dataset might contain biases due to an imbalance in the representational level of meta-features, we apply a resampling strategy based on randomization to assess our research design quality. For that, we compare the resampling results with the previously reported in Figure 4b. Essentially, we submitted the meta-database to a binning procedure using the $trace\_len\_entropy$ meta-feature as it was the best ranked according to the RF importance (Figure 8). Fol-



(a) Attribute Anomaly

(b) Early Anomaly

(c) Insert Anomaly

(d) Late Anomaly

(e) Rework Anomaly

(f) Skip Anomaly

Fig. 10. Top 20 features ranked using SHapley Additive exPlanations (SHAP) for each anomaly type

lowing, we selected a percentage of samples per bin, creating a resampled meta-database, which was later fed to the same research pipeline as described in Section 4. Table 4 reports the results of the resampling experiment with a gradual increase of sampling size

percentages. As expected, when smaller percentages of meta-instances are selected for each bin, we observe a reasonable decrease in both performance metrics. Such behavior is expected simply because the learning model has fewer examples to generalize from. With the increasing percentage of meta-instances, both accuracy and F-score also rise in performance, indicating convergence to the result observed in Figure 4b. This behavior confirms the internal validity of our research design, which, given a meta-database of increasing representational quality, improves the recommendation performance.

Resampling size	10%	20%	30%	40%	50%
Accuracy	0.52	0.56	0.58	0.64	0.68
F-score	0.43	0.50	0.53	0.62	0.65

Table 4. Resampling experiment prediction performance

Complimentarily, we report the confusion matrices for each resampling percentage in Table 5. In general, the performance for each class increases along with the resampling size. However, we note that the alignment class is particularly challenging to identify. For instance, there is a high number of false positives associated with this class. Although they decrease as the meta-database grows, there is still a significant amount of misclassifications with this particular target. This pattern confirms the analysis of feature importance presented in Figures 9 and 10. Considering that it is the most difficult class to decide from (the meta-features have a lower average impact), the misclassifications are a consequence. Doc2vec and node2vec are more easily identified by the classifier, which corroborates to a better performance for these classes.

resampling / encoding	10%			20%			30%			40%			50%		
alignment (*A*)	0.28	0.46	0.26	0.42	0.33	0.26	0.38	0.41	0.21	0.47	0.34	0.19	0.49	0.33	0.18
doc2vec (*D*)	0.18	0.57	0.25	0.23	0.58	0.19	0.23	0.63	0.13	0.22	0.68	0.1	0.19	0.74	0.07
node2vec (*N*)	0.18	0.21	0.61	0.22	0.19	0.59	0.18	0.14	0.68	0.19	0.09	0.73	0.17	0.08	0.75
	A	*D*	*N*	*A*	*D*	*N*	*A*	*D*	*N*	*A*	*D*	*N*	*A*	*D*	*N*

Table 5. Resampling experiment confusion matrix. This analysis extends Table 4 by providing a more in-depth notion of which encoding techniques are more difficult to be identified and how the performances change for an increasing size of the meta-database

6. Conclusion

Organizations are interested in detecting anomalous instances in their business processes as a method to leverage process quality, avoid resource waste, and mitigate security issues. In this work, we proposed to combine encoding techniques with MtL to enhance the detection of anomalous traces in event logs. Our strategy relies on a powerful set of meta-features extracted from the event logs. We showed its viability by recommending the best

encoding technique with an F-score of 0.43, statistically overcoming the baselines. MtL boosted the anomaly detection by fitting the optimal encoding technique for each event log, statistically outperforming the usage of a single encoding technique.

Furthermore, in this extension of the original research [41], we dove into the influence of meta-features on the recommended encoding technique. For that, we extracted descriptors' importance from the ML model and also exploited the SHAP method to explain the predictions. This analysis leveraged the understanding of which features (and their families) better capture process behavior in the context of anomaly detection. Considering concerns about algorithm engineering, we assessed the internal validity of the research design by applying a resampling strategy. The results indicated the validity of the experiments in this environment. For future works, we plan to include more encoding techniques and propose additional features for the meta-feature extraction step.

References

1. van der Aalst, W.: Process Mining: Data Science in Action. Springer Berlin Heidelberg (2016), https://doi.org/10.1007/978-3-662-49851-4
2. Adam, S.P., Alexandropoulos, S.A.N., Pardalos, P.M., Vrahatis, M.N.: No Free Lunch Theorem: A Review, pp. 57–82. Springer International Publishing, Cham (2019), https://doi.org/10.1007/978-3-030-12767-1_5
3. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Alignment based precision checking. In: La Rosa, M., Soffer, P. (eds.) Business Process Management Workshops. pp. 137–149. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
4. Augusto, A., Mendling, J., Vidgof, M., Wurm, B.: The connection between process complexity of event sequences and models discovered by process mining. Information Sciences 598, 196–215 (2022), https://www.sciencedirect.com/science/article/pii/S0020025522002997
5. Barbon Jr., S., Ceravolo, P., Damiani, E., Omori, N.J., Tavares, G.M.: Anomaly detection on event logs with a scarcity of labels. In: 2020 2nd International Conference on Process Mining (ICPM). pp. 161–168 (2020)
6. Barbon Jr., S., Ceravolo, P., Damiani, E., Tavares, G.M.: Evaluating trace encoding methods in process mining. In: Bowles, J., Broccia, G., Nanni, M. (eds.) From Data to Models and Back. pp. 174–189. Springer International Publishing, Cham (2021)
7. Barbon Jr., S., Tavares, G.M., da Costa, V.G.T., Ceravolo, P., Damiani, E.: A framework for human-in-the-loop monitoring of concept-drift detection in event log stream. In: Companion Proceedings of the The Web Conference 2018. p. 319–326. WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2018), https://doi.org/10.1145/3184558.3186343
8. Bezerra, F., Wainer, J., van der Aalst, W.M.P.: Anomaly detection using process mining. In: Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R. (eds.) Enterprise, Business-Process and Information Systems Modeling. pp. 149–161. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
9. Bezerra, F., Wainer, J.: Algorithms for anomaly detection of traces in logs of process aware information systems. Information Systems 38(1), 33–44 (2013), https://www.sciencedirect.com/science/article/pii/S0306437912000567
10. Böhmer, K., Rinderle-Ma, S.: Multi-perspective anomaly detection in business process execution events. In: On the Move to Meaningful Internet Systems: OTM 2016 Conferences. pp. 80–98. Springer International Publishing, Cham (2016)

11. Bose, R.P.J.C., van der Aalst, W.M.: Context Aware Trace Clustering: Towards Improving Process Mining Results, pp. 401–412 (2019), https://epubs.siam.org/doi/abs/10.1137/1.9781611972795.35
12. Breiman, L.: Random forests. Machine learning 45(1), 5–32 (2001)
13. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees. Routledge (1984)
14. Burattin, A.: Plg2: Multiperspective processes randomization and simulation for online and offline settings (2015)
15. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018)
16. Ceravolo, P., Tavares, G.M., Barbon Jr., S., Damiani, E.: Evaluation goals for online process mining: a concept drift perspective. IEEE Transactions on Services Computing pp. 1–1 (2020)
17. De Koninck, P., vanden Broucke, S., De Weerdt, J.: act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) Business Process Management. pp. 305–321. Springer International Publishing, Cham (2018)
18. Delias, P., Doumpos, M., Grigoroudis, E., Matsatsinis, N.: A non-compensatory approach for trace clustering. International Transactions in Operational Research 26(5), 1828–1846 (2019), https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12395
19. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (Dec 2006), http://dl.acm.org/citation.cfm?id=1248547.1248548
20. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. The Annals of Statistics 29(5), 1189–1232 (2001)
21. Grover, A., Leskovec, J.: Node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 855–864. KDD '16, ACM, New York, USA (2016)
22. Hake, P., Zapp, M., Fettke, P., Loos, P.: Supporting business process modeling using rnns for label classification. In: Frasincar, F., Ittoo, A., Nguyen, L.M., Métais, E. (eds.) Natural Language Processing and Information Systems. pp. 283–286. Springer International Publishing, Cham (2017)
23. He, X., Zhao, K., Chu, X.: Automl: A survey of the state-of-the-art. Knowledge-Based Systems 212, 106622 (2021), https://www.sciencedirect.com/science/article/pii/S0950705120307516
24. Kotthoff, L.: Algorithm selection for combinatorial search problems: A survey. In: Bessiere, C., De Raedt, L., Kotthoff, L., Nijssen, S., O'Sullivan, B., Pedreschi, D. (eds.) Data Mining and Constraint Programming: Foundations of a Cross-Disciplinary Approach. pp. 149–190. Springer International Publishing, Cham (2016)
25. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Xing, E.P., Jebara, T. (eds.) Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 32, pp. 1188–1196. PMLR, Bejing, China (22–24 Jun 2014)
26. Lee, W.L.J., Verbeek, H., Munoz-Gama, J., van der Aalst, W.M., Sepúlveda, M.: Recomposing conformance: Closing the circle on decomposed alignment-based conformance checking in process mining. Information Sciences 466, 55–91 (2018), https://www.sciencedirect.com/science/article/pii/S0020025518305413
27. Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) Business Process Management. pp. 297–313. Springer International Publishing, Cham (2015)
28. Luettgen, S., Seeliger, A., Nolle, T., Mühlhäuser, M.: Case2vec: Advances in representation learning for business processes. In: Leemans, S., Leopold, H. (eds.) Process Mining Workshops. pp. 162–174. Springer International Publishing, Cham (2021)

29. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf
30. Măruşter, L., Weijters, A.T., Van Der Aalst, W.M., Van Den Bosch, A.: A rule-based approach for process discovery: Dealing with noise and imbalance in process logs. Data mining and knowledge discovery 13(1), 67–87 (2006), https://doi.org/10.1007/s10618-005-0029-z
31. Mendling, J., Depaire, B., Leopold, H.: Theory and practice of algorithm engineering (2021), https://arxiv.org/abs/2107.10675
32. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013), https://arxiv.org/abs/1301.3781
33. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: Analyzing business process anomalies using autoencoders. Machine Learning 107(11), 1875–1893 (2018), https://doi.org/10.1007/s10994-018-5702-8
34. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: Binet: Multi-perspective business process anomaly classification. Information Systems 103, 101458 (2022), https://www.sciencedirect.com/science/article/pii/S0306437919305101
35. Nolle, T., Seeliger, A., Mühlhäuser, M.: Binet: Multivariate business process anomaly detection using deep learning. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) Business Process Management. pp. 271–287. Springer International Publishing, Cham (2018)
36. Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: ADVANCES IN LARGE MARGIN CLASSIFIERS. pp. 61–74. MIT Press (1999)
37. Polato, M., Sperduti, A., Burattin, A., Leoni, M.d.: Time and activity sequence prediction of business process instances. Computing 100(9), 1005–1031 (Sep 2018), https://doi.org/10.1007/s00607-018-0593-x
38. Rice, J.R.: The algorithm selection problem. Advances in Computers, vol. 15, pp. 65–118. Elsevier (1976), https://www.sciencedirect.com/science/article/pii/S0065245808605203
39. Rozinat, A., van der Aalst, W.: Conformance checking of processes based on monitoring real behavior. Information Systems 33(1), 64–95 (2008), https://www.sciencedirect.com/science/article/pii/S030643790700049X
40. Tavares, G.M., Barbon, S.: Analysis of language inspired trace representation for anomaly detection. In: Bellatreche, L., Bieliková, M., Boussaïd, O., Catania, B., Darmont, J., Demidova, E., Duchateau, F., Hall, M., Merčun, T., Novikov, B., Papatheodorou, C., Risse, T., Romero, O., Sautot, L., Talens, G., Wrembel, R., Žumer, M. (eds.) ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium. pp. 296–308. Springer International Publishing, Cham (2020)
41. Tavares, G.M., Barbon Jr., S.: Process mining encoding via meta-learning for an enhanced anomaly detection. In: Bellatreche, L., Dumas, M., Karras, P., Matulevičius, R., Awad, A., Weidlich, M., Ivanović, M., Hartig, O. (eds.) New Trends in Database and Information Systems. pp. 157–168. Springer International Publishing, Cham (2021)
42. Tavares, G.M., Barbon Junior, S., Damiani, E., Ceravolo, P.: Selecting optimal trace clustering pipelines with meta-learning. In: Xavier-Junior, J.C., Rios, R.A. (eds.) Intelligent Systems. pp. 150–164. Springer International Publishing, Cham (2022)
43. Tavares, G.M., Junior, S.B., Damiani, E.: Automating process discovery through meta-learning. In: Sellami, M., Ceravolo, P., Reijers, H.A., Gaaloul, W., Panetto, H. (eds.) Cooperative Information Systems. pp. 205–222. Springer International Publishing, Cham (2022)

44. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. ACM Trans. Knowl. Discov. Data 13(2) (mar 2019), https://doi.org/10.1145/3301300
45. Tipping, M.E., Bishop, C.M.: Mixtures of Probabilistic Principal Component Analyzers. Neural Computation 11(2), 443–482 (02 1999)
46. van der Aalst, W., de Medeiros, A.: Process mining and security: Detecting anomalous process executions and checking process conformance. Electronic Notes in Theoretical Computer Science 121, 3–21 (2005), proceedings of the 2nd International Workshop on Security Issues with Petri Nets and other Computational Models (WISP 2004)
47. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Artificial Intelligence Review 18(2), 77–95 (Jun 2002), https://doi.org/10.1023/A:1019956318069

Gabriel Marques Tavares is a Ph.D. candidate in Computer Science at the University of Milan (UNIMI), Italy. He received his BSc and MSc at the Londrina State University (UEL) in 2019, while in 2014 he participated in an exchange program at the University of Michigan. Currently, his research activities focus on leveraging Process Mining analysis using Machine Learning methods. More specifically applying Meta-learning and Automated Machine Learning to unveil relationships between data characteristics and optimal pipelines. He has investigated themes such as automated Process Discovery, Trace Clustering, Event log encoding, Data Representation, Process Explainability, and Online Process Mining with particular attention to Concept Drift Detection.

Sylvio Barbon Junior is an Associate Professor at the Department of Engineering and Architecture at the University of Trieste (UNITS), Italy. Currently, he is part of the Machine Learning Lab. Previously, he was the leader of the research group that studies machine learning in the Computer Science Department at the State University of Londrina (UEL), Brazil. He received his BSc degree in Computer Science in 2005, MSc degree in Computational Physics from the University of São Paulo (2007), a degree in Computational Engineering in 2008, and a Ph.D. degree (2011) from IFSC/USP such as the MSc degree. In 2017, he was visiting researcher at the University of Modena and Reggio Emilia (Italy) working on multispectral analysis and machine learning. Still, in 2017, he was visiting researcher at Università Degli Studi Di Milano (Italy) focused on Data Stream and Process Mining. He is currently a professor in postgraduate and graduate programs. His research interests include Computer Vision, Pattern Recognition, and Machine Learning. Currently, focusing on Meta-Learning, Stream Mining, and Process Mining.

Received: January 10, 2022; Accepted: November 10, 2022.