Tuning Machine Learning to Address Process Mining Requirements

PAOLO CERAVOLO[1], (Member, IEEE), SYLVIO BARBON JUNIOR[2], ERNESTO DAMIANI[3], (Senior Member, IEEE), AND WIL VAN DER AALST[4], (Fellow, IEEE)

[1]Department of Computer Science, University of Milan, 20133 Milan, Italy
[2]Department of Engineering and Architecture, University of Trieste, 34127 Trieste, Italy
[3]Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi, United Arab Emirates
[4]Chair of Process and Data Science, RWTH Aachen University, 52056 Aachen, Germany

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

The work was partially supported by the Multilayered Urban Sustainability Action (MUSA) project, funded by the European Union–NextGenerationEU, under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strengthening of research structures and creation of R&D ''innovation ecosystems'', set up of ''territorial leaders in R&D'' (CUP G43C22001370007, Code ECS00000037). The work was also partially supported by the SERICS project (PE00000014) under the NRRP MUR program funded by the EU–NextGenerationEU.

ABSTRACT Machine learning models are routinely integrated into *process mining* pipelines to carry out tasks like data transformation, noise reduction, anomaly detection, classification, and prediction. Often, the design of such models is based on some *ad-hoc* assumptions about the corresponding data distributions, which are not necessarily in accordance with the *non-parametric* distributions typically observed with process data. Moreover, mainstream machine-learning approaches tend to ignore the challenges posed by *concurrency* in operational processes. Data *encoding* is a key element to smooth the mismatch between these assumptions but its potential is poorly exploited. In this paper, we argue that a deeper understanding of the challenges associated with training machine learning models on process data is essential for establishing a robust integration of process mining and machine learning. Our analysis aims to lay the groundwork for a methodology that aligns machine learning with process mining requirements. We encourage further research in this direction to advance the field and effectively address these critical issues.

INDEX TERMS Process mining, machine learning, non-parametric distribution, concurrency, non-stationary, zero-shot learning, encoding, training.

I. INTRODUCTION

Process Mining (PM) is an established discipline rooted in *data mining* and *business process management*. The use of traditional PM tasks such as *process discovery* and *conformance checking* is now commonplace in many organizations [1], [2]. The benefits of integrating PM with traditional process monitoring, bringing automation, transparency and efficiency to the forefront, are now widely recognized [3]. However, the last decade has witnessed a surge of new insights from the field of artificial intelligence that has captured the attention of the PM research community [4]. Figure 1 illustrates the key steps in applying data science to PM. Data from the event logs of information systems is *extracted and prepared* for *process discovery* and *conformance checking*, the combined output of these tasks is used for *predictive monitoring* and *action-oriented decision making*. Artificial intelligence enhances these processes by facilitating various downstream operations. Currently, there is a notable focus on leveraging Large Language Models (LLM) to seamlessly interface PM algorithms with natural language [5], [6], [7], as illustrated in point (3) of Figure 1. But in today's practice, is ad-hoc Machine Learning (ML) models that are routinely integrated into PM pipelines, performing various tasks that are today part of PM libraries [8]. Figure 1 point (1) highlights *data transformation*, *noise reduction*, and *feature engineering*. Figure 1 point (2) mentions *prediction*, *simulation*, and

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

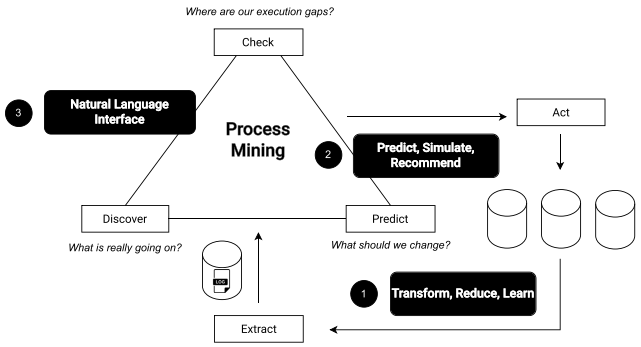


FIGURE 1. The PM tasks and their relation to ML.

recommendation. Our focus in this paper is on consistent procedures to train ML models for their ad-hoc integration into PM pipelines.

For example, ML is playing a key role in the interface between PM and sensor platforms. Advances in sensing technologies have made it possible to deploy distributed monitoring platforms capable of detecting fine-grained events. The granularity gap between these events and the activities considered by classic PM analysis has often been bridged using ML models [9], [10] that compute virtual activity logs, a problem that is also known as *log lifting* [11]. ML has been proposed as a key technology to *strengthen* existing techniques, for example, using trace clustering to reduce the diversity that a process discovery algorithm must handle in analyzing an event log [12], [13], [14], [15], to simplify the discovered models [16], [17], [18], or to support real-time analysis on event streams [19], [20], [21]. ML is adopted to apply predictive models to the executing cases of a process. This research area, known as predictive process monitoring, exploits event log data to foresee future events, remaining time, or the outcome of cases, in support of decision making [22], [23], [24]. Root cause analysis [25], [26] and explainability [27], [28] are other tasks that can be applied to event log data using ML techniques, in order to improve our understanding of a business process. ML models have also been used in addition to (or in lieu of) classic linear programming [29] to *optimize* business processes' resource consumption and to provide insights to process *redesign* [30]. Computational support for PM appears to align with that of ML models from a technological standpoint [31], [32], [33]. This convergence might suggest a straightforward integration of these fields.

Nevertheless, in practice, this integration is far from straightforward. When mapping PM tasks to ML tasks, it becomes imperative to construct training functions and select hyperparameters guided by business process-specific assumptions. Many of these assumptions stem from the inherent characteristics of human social systems. For instance, it is widely recognized that process variants follow *non-parametric* distributions [34]. In contrast, ML models often benefit from data normality, and skewed data distributions can introduce bias into their predictions. Furthermore, the conventional ML perspective on event log data often oversimplifies the reality. Properly encoding the procedural nature of event log traces poses significant challenges [35]. Frequently, the sequence of executed events is represented solely by a fixed-length prefix. Even more intricate is the encoding of *concurrency* and the *interactions* that govern events within a business process [36]. The process of encoding event log data into a feature space compatible with ML algorithms is a pivotal design choice. It has profound implications for *sample complexity*, *data distribution*, and *feature relevance* for analysis purposes. This includes tasks such as detecting *concept drift* [37] and enabling *zero-shot learning* principles on PM tasks [38].

Today, much of the research on integrating ML with PM focuses on developing ML models to attain high performance in specific business process management scenarios. Less attention has been paid to designing a general methodology to select and adapt ML models based on the nature of the PM problem, taking into account the specific properties of the process data. We argue that, when using ML models in PM pipelines, it is important to prevent any *mismatch* between the assumptions on input data underlying the ML models and the information captured by the event logs used to feed them. Indeed, careless assumptions about the encoding of input data can lead to biased models with reduced generalizability. Arbitrarily selecting ML algorithms leads to unfair evaluation and sub-optimal solutions. For example, a given ML model cannot be compared with another if their implementations consider different feature spaces [39]. It is also important to make sure that ML models are exposed to process-specific information, such as the processes' control-flow constraints [36]. In this paper, we attempt to identify some of the causes of this mismatch and suggest how to remove them, with the aim of fostering research on a sound methodology to address the integration between PM and ML.

We believe that the PM and ML research communities must work together to address these issues. This call for collaboration is valid in general, but particularly in business process management, where data analysis has to leave the safe harbor of experimental science to sail into the open sea of decision science, where reproducibility is often a challenge due to the unique contextual factors influencing each business process. We believe that the community should invest in defining *common practices* and *benchmarks* to promote fair comparison, by specifying common settings, and tracking progress in the field over time [40]. In this paper we discuss the challenges in a specific direction, i.e. from PM to ML. More specifically, in Section II, we discuss the issues leading to the PM-to-ML mismatch. In Section III, we introduce some basic PM notions. In Section IV, we link them to ML principles. Section V clarifies the discussion by presenting a couple of samples. Section VI proposes research lines for advancing in the direction of a general methodology that integrates ML models into PM pipelines. Section VIII closes the paper.

II. THE ISSUES LANDSCAPE

An important issue underlying our discussion is how to account for the specificity of process data in ML algorithm selection and hyper-parameter tuning. Of course, processing event logs presents all the usual challenges of data preprocessing and preparation. We will not discuss standard data preprocessing techniques such as outlier removal [41], [42], noise filtering [43], [44], and missing entry recovery [45], as these can be addressed by current statistical techniques. Rather, we will focus on issues specific to process data.

A. DATA DISTRIBUTION

When selecting an ML model for a PM task, there may be a temptation to assume that the processed data fed to the model will conform to a normal distribution. Indeed, the assumption of data normality can be advantageous for various types of ML models. Specifically, models such as Gaussian, naive Bayes, logistic regression, and linear regression explicitly rely on the assumption that the data distribution is bivariate or multivariate normal. Many phenomena of interest for business process analysis, such as the duration of some activities, are known to follow normal or log-normal distributions.[1] For other PM data, however, assuming normality is not advisable. For example, process variants are specific activity sequences that occur through a process from start to end. Variants' occurrence in an activity log is typically following a *non-parametric* power-law trend that complies with the *Pareto principle* [34]. A normal distribution cannot always be assured also for the pairwise dependency relationship between activities, a key statistical information exploited by process discovery algorithms [46]. Indeed, in this case, the normality assumption has been verified for some event logs, including some popular benchmarks we will discuss in Section V (the "Road traffic fines" [47] and "Receipt phase of an environmental permit application process" [48]). However, the normality of dependencies in less regular or not observed in "spaghetti" like processes, as in the "BPI Challenge 2015 Municipality 1" [49]. There are reasons to believe that dependencies in loosely specified logs may follow some power-law trend as well, and require careful parameter fitting in statistical analysis.

At first glance, this may not seem particularly problematic, since most ML algorithms, such as decision trees, support vector machines, or neural networks, are generally considered to be robust to non-normal distributions and can handle a variety of data types and distributions. However, it is important to note that certain characteristics of a data set, such as the presence of outliers, skewed distributions, or unbalanced classes, can still cause serious difficulties for ML algorithms [50]. If the training data is skewed toward a particular class or outcome, as in power-law or log-normal distributions, the model may be more likely to predict that class or outcome, even if it is not the most likely. Independent Component Analysis provides ways to detect Gaussianity and non-Gaussianity [51]. Of course, non-normal distributions can be transformed into normal ones using Box-Cox transformations [52], and unbalanced data sets can be balanced [53], [54], but as we will see in Section V, such data transformations should be applied with caution, as they have consequences for the performance of the models.

[1] See, for instance, the "lunch break" duration distributions at https://www.statista.com/statistics/995991/distribution-of-lunch-breaks-by-length-in-europe/

B. CONCURRENCY

Another area of focus is concurrency. How to use ML to predict the behavior of highly concurrent systems and processes is still an open problem, and research in the AI community has only scratched the surface. Most ML approaches view event logs as merely sequential data [55], rather than sequential manifestations of a concurrent system. This can lead to under-sampling of the log space and insufficient training to handle seemingly out-of-order event sequences [56]. To effectively address this challenge, ML models must be provided with additional context about the concurrent execution of tasks or the underlying control flow in the system. One avenue that has been explored to integrate this information is the use of bidirectional long-short term memory (BiLSTM) architectures. For example, Thapa et al. [57] used BiLSTM to identify concurrent human activities in a smart home environment. In addition, their subsequent work introduced a synchronous algorithm called sync-LSTM [58], which adapts the Long Short-Term Memory (LSTM) algorithm to handle multiple parallel input sequences and generate synchronized output sequences. Predicting the behavior of highly concurrent systems using ML remains an actively evolving field, as evidenced by the recent comprehensive review by Neu et al. [59]. Researchers are exploring novel techniques and methodologies to improve our understanding and predictive capabilities with respect to concurrency in various domains. Despite the progress that has been made, concurrency poses an ongoing challenge, stimulating continued exploration and innovation in the field.

C. NON-STATIONARY BEHAVIOR

Even when process data distributions can be fitted accurately, ongoing processes, especially those involving resources that learn and age, such as people and equipment, change over time. This gives rise to *non-stationary* behavior. Concept drift detection techniques are therefore needed [21]. In traditional data mining applications, *concept drift* is identified when a concept, i.e. the relationship between a data instance and its associated class, changes at two different points in time [60]. Concept drift can compromise the accuracy of predictive process monitoring techniques by causing performance degradation due to evolving patterns and dynamics within the process data. In PM, many aspects of drift should be carefully monitored, including the appropriateness of the event trace with respect to the model, the dependency

relationship between activities, and the interdependency between activities and available resources or cycle time. Each aspect should be appropriately coded and monitored using statistical analysis [37]. While there has been progress in research, the practical application of concept drift detection in real process mining scenarios is still an evolving area [61], [62], [63], [64]. The adoption of these techniques in industrial settings depends on the development of robust and scalable methods that can handle the complexity of dynamic processes.

D. LLM FOR ZERO-SHOT LEARNING

Zero-shot learning involves identifying solutions that were not encountered during training [65]. This approach uses unstructured auxiliary information encoded during training instead of explicit labels. The system learns to associate new input elements with encodings that have the highest similarity in terms of auxiliary information, allowing it to propose results not seen during training. In PM, where labeled process data may be limited or inaccessible, zero-shot learning becomes critical. LLMs, exemplified by Generative Pretrained Transformers (GPT), have emerged as fundamental models for zero-shot learning applications [66]. Pre-trained on rich linguistic data, LLMs capture complex patterns, context, and semantics in natural language. Organizations can use LLMs for various machine learning tasks without the need for extensive task-specific training datasets, streamlining development and increasing efficiency. LLMs' ability to understand prompts and generate human-like text or multimedia data can also greatly simplify user interaction with PM algorithms.

Integrating LLMs into PM shifts the focus from analysis to synthesis of activities to achieve desired goals [67]. LLMs can analyze logs, couple them with other data, and suggest operational actions to achieve goals [7]. The challenge lies in learning the mapping between log prefixes and desired outcomes in the latent space constructed by GPT algorithms. One promising strategy is to combine LLMs with diffusion models, which have demonstrated the ability to learn conditional probability distributions within a latent space [68]. While it's important to exercise caution, especially given the training time constraints associated with current diffusion models, which are primarily designed for 3D representations, these models provide valuable assistance in probabilistically mapping latent space data points. For example, mapping known prerequisites for performing process activities to a decision path within a particular business process [69].

E. DATA ENCODING

ML algorithms are trained using datasets, where each data point is a vector in a multi-dimensional feature space. PM, on the other hand, deals with event logs, which capture business processes in the form of sequential records of events over time. Each event is defined by dimensions such as execution time, resource usage, and data exchanged. However, the primary focus is on viewing each event as a step in a sequence of activities that together form a specific business process case. This difference in the data structures of ML and PM is a challenge. In PM, the order of activities is critical and is the most important information to convey. On the other hand, ML training sets typically don't consider the order of examples. Although one could encode the sequence of events in a case as a single data point, this compromises the representation of events as multidimensional objects. Reconciling the focus on temporal order in PM with the feature-driven representation of ML algorithms is not trivial. To complicate matters, events in a business process are not only influenced by intra-case dependencies. They are also susceptible to inter-case dependencies, such as sharing the same set of resources with events in other cases. Effective event log encoding should ideally capture all of these aspects.

Surprisingly, the PM community has devoted relatively little effort to investigating the impact of encoding methods on PM pipeline performance. In practices, basic techniques such as one-hot encoding [70], frequency-based encoding [14], and general statistics for numerical attributes [23] are often used. These methods excel at distinguishing cases based on their constituent activities, frequency, or other characteristics, but fail in capturing the sequential order of activity execution. To improve the representation of dependencies between activities, PM has integrated techniques from several domains, including text mining [71], [72] and graph embedding [73], [74]. While graph embedding methods often outperform other techniques, they introduce increased time complexity and a decrease in transparency in the resulting latent space [75]. Recent advances emphasize encoding control flow information in feature spaces to represent the parallelism or optionality of activities [36], [76]. Innovative approaches include multi-perspective views of traces that integrate both data-flow and control-flow information [77], [78]. In addition, the exploration of *inter-case* encoding, which captures relationships between different cases, has attracted attention [79]. However, the application of these advanced encoding techniques often remains domain-specific and cannot be widely adopted by the community.

Furthermore, the PM literature lacks comprehensive documentation of the encoding techniques used to map PM data to ML algorithms. Comparative studies are rare and only a limited number of examples, such as [35], [80], and [81], are available for reference. The chosen feature space is often implicitly defined, specific encoding steps remain unclear, and the actual code used is not disclosed. Ablation studies, which evaluate the performance impact of removing parts of the data representation, are still the exception rather than the rule. We argue that formalizing the encoding procedure can provide a rationale for this crucial design choice, aligning it with specific analytical goals and assumptions relevant to the algorithms under consideration. In Section IV, we present a proposal for such a formalization.

III. BASIC NOTIONS IN PM

To make this paper self-contained, in this section we recall some of the basic concepts of PM. An *event log* is a collection of *events* generated in a temporal sequence and stored as *tuples*, i.e., recorded values from a set of *attributes*. Events are aggregated by *case*, i.e., the end-to-end execution of a business process. For the sake of classification, all cases following the same *trace*, i.e., performing the same sequence of business process activities, can be considered equal as they belong to the same process *variant*.

Definition 1 (Event, Attribute): Let Σ be the *event universe*, i.e., the set of all possible event identifiers; Σ^* denotes the set of all finite sequences over Σ. Events have various *attributes*, such as timestamp, activity, resource, associated cost, and others. Let $\mathcal{AN}$ be the set of attribute names. For any event $e \in \Sigma$ and attribute $a \in \mathcal{AN}$, the function $\#_a(e)$ returns the value of the attribute a for event e.

The set of possible values of each attribute is restricted to a domain. For example, $\#_{\text{activity}} : \Sigma \rightarrow \mathcal{A}$, where $\mathcal{A}$ is the set of the legal activities of a business process, e.g. $\mathcal{A} = \{a, b, c, d, e\}$. If e does not contain the attribute value for some $a \in \mathcal{AN}$, then $\#_a(e) = \perp$. It follows that an event can also be viewed as a tuple of attribute-value pairs $e = (\mathcal{A}_1, \ldots, \mathcal{A}_m)$, where m is the cardinality of $\mathcal{AN}$.

Definition 2 (Sequence, Sub-sequence): In a sequence of events $\sigma \in \Sigma^*$, each event appears only once and time is non-decreasing, i.e., for $1 \leq i \leq j \leq |\sigma| : \#_{\text{timestamp}}(e_i) \leq \#_{\text{timestamp}}(e_j)$. Thus $\langle e_1, e_2, e_3 \rangle$ denotes three subsequent events. A sequence can also be denoted as a function generating the corresponding event for each position in the sequence: $\sigma(i \rightarrow n) \mapsto \langle e_i, \ldots, e_n \rangle$, with e_n the last event of a sequence. In this way, we can define a sub-sequence as a sequence $\sigma(i \rightarrow j)$ where $0 \leq i < j < n$.

Definition 3 (Case, Event Log): Let $\mathcal{C}$ be the *case universe*, that is, the set of all possible identifiers of a business case execution. $\mathcal{C}$ is the domain of an attribute $\#_{\text{case}} \in \mathcal{AN}$. We denote a case $c \in \mathcal{C}$ as $\langle e_1, e_2, e_3 \rangle_c$, meaning that all events are in a sequence and share the same case. For a case $\langle e_1, e_2, e_3 \rangle_c$ we have $\#_{\text{case}}(e_1) = \#_{\text{case}}(e_2) = \#_{\text{case}}(e_3) = c$. An *event log* L is a set of cases $L \subseteq \Sigma^*$ where each event appears only once in the log, i.e., for any two different cases, the intersection of their events is empty. When the case identifier is not used as a grouping attribute, an *event log* $\hat{L}$ can be simply viewed as a set of events, thus $\hat{L} \subseteq \Sigma$.

Definition 4 (Variant, Event Log): The cases c_1 and c_2 follow the same variant if $\langle e_1, e_2, e_3 \rangle_{c_1}$ and $\langle e_4, e_5, e_6 \rangle_{c_2}$ have the same sequence of activities, e.g. $\#_{\text{activity}}(e_1) = \#_{\text{activity}}(e_4) = a$, $\#_{\text{activity}}(e_2) = \#_{\text{activity}}(e_5) = b$, $\#_{\text{activity}}(e_3) = \#_{\text{activity}}(e_6) = a$. We call this sequence a trace. This implies an event log can also be viewed as a multi-set of traces. We denote an event log as a multi-set by writing $L = [\langle a, b, c \rangle^3, \langle a, b, a \rangle^{11}, \langle a, c, b, a \rangle^{20}]$. The superscript number of a trace details the number of cases following this variant. For example, $\langle a, b, a \rangle^{11}$ means we have a variant with 11 cases following the trace $\langle a, b, a \rangle$.

IV. A FORMALIZATION OF PM DATA ENCODING

In Section II, we discussed the challenges associated with fully encoding event logs into a format suitable for downstream ML algorithms. One of the aspects we need to capture is the *control-flow*, which represents the sequence of activities and the path the process follows. Understanding the control-flow is critical to identifying patterns, bottlenecks, deviations, and opportunities for improvement within a business process. In addition, many analytics can benefit from capturing the *data-flow*, which focuses on how data is generated, processed, and transmitted within the various activities of a process. Another important distinction lies in *inter-case* and *intra-case* dependencies within a business process. Intra-case examines interactions and dependencies within a single case, while inter-case examines those between different cases. Furthermore, events in an event log are not just characterized by the activities or tasks performed. They encompass several dimensions, including the resources consumed, the agent performing the activities, and the outcome produced. Capturing this *multidimensional* perspective may be essential for developing effective analytics. Ideally, an encoding process should capture all of these aspects. However, the techniques currently used, especially in real-world applications, fall short of this ideal. Most encoding methods today focus primarily on the *control-flow*, predominantly from a *inter-case* perspective [35]. Events are often represented simply by the activity performed, and cases are represented as sequences of events, often with a fixed-length prefix. This oversimplification does not fully capture the wealth of information contained in event logs. While there are proposed methods that emphasize the *intra-case* view, they are rarely implemented in practice [82]. Although the literature provides encoding proposals for *data-flow* [83], these are typically tailored to specific domains. A recent trend underscores the importance of capturing concurrency constraints [36], [76]. In this section, we aim to shed light on how PM data is encoded to align with ML algorithms, and why this process remains a significant research challenge.

For the sake of space, we limit our discussion to supervised learning, probably the most widely applied ML approach. Generally speaking, supervised techniques train models to compute functions $f : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ where the input is a d-dimensional vector $\mathbf{x}$ and the output is a d'-dimensional vector $\mathbf{y}$. Each dimension is a measurable piece of data, a.k.a feature or attribute. For popular ML tasks, the output is mono-dimensional. In regression, the output is a real-valued scalar value, while in classification, the output is a natural number indexing a "class". However, nothing prevents having multidimensional vectors in output. In structured learning, input and output may be a structure like a block matrix, divided into sub-matrices to represent algebraic entities such as graphs, tensors, etc. The training process to approximate f requires a set of examples $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ where inputs and outputs are paired. We can then define this training set as an example matrix $\mathbf{X} := [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ and

a label matrix $\mathbf{Y} := [\mathbf{y}_1, \ldots, \mathbf{y}_n]^\top \in \mathbb{R}^{n \times d'}$, given by the number n of vectors and the number d of dimensions in the vector space.

In their original format, PM log entries do not belong to a vector space. This is because the events in an event log are grouped by case and this grouping is essential to keep a connection with business process execution. Because the number of events per case is variable, data transformations are needed to create vectors of a predefined length.

Our goal here is to formalize the procedure to encode the cases into vectors in a way that can be used as a template to describe the specific encoding chosen for a PM application. Our starting point is $\hat{L} \subseteq \Sigma$, a log view as a set of event identifiers. This representation can be mapped into a vector space $\mathbf{X}$ by applying a suitable *transformation function* grouping event by case and returning vectors of size equal to or less than the event size.

Definition 5 (Encoding function): Given an event log $\hat{L} \subseteq \Sigma$, an *encoding function* $\Gamma : \Sigma \to \mathbf{X}^{n \times d}$ represents $\hat{L}$ in the vector space $\mathbf{X}$. The encoding function Γ is valid if it defines a transformation where two elements of Σ, e_i and e_j are aggregated on the same element $\mathbf{x} \in \mathbb{R}^d$ if $\#_{\text{case}}(e_i) = \#_{\text{case}}(e_j)$, with $n \leq |\mathcal{C}|$, i.e. the vectors in $\mathbf{X}$ are a subset of the cases in $\mathcal{C}$.

We propose a canonical representation of Γ as a composition of a *filtering function* π, a *dimensioning function* ρ, a *grouping function* η, and a *valuation function* ν, i.e., $\Gamma = \nu \circ \eta \circ \rho \circ \pi$. One or more of these components can implement the identity function with null effects.

In particular, $\pi : \Sigma \to \Sigma_\alpha$ imposes a condition on the events' attributes or the attributes' values, $\forall e \in \hat{L} \wedge \mathrm{a} \in \mathcal{AN} : P(\#_{\mathrm{a}}(e))$, where P is a predicate, thus $|\Sigma_\alpha| \leq |\Sigma|$. For example, filtering the events by their timestamp $\forall e \in \hat{L} : \texttt{YYYY-MM-DD} \geq \#_{\text{timestamp}}(e) \leq \texttt{YYYY-MM-DD}$. The function $\rho : \Sigma_\alpha \to D$ defines the dimensions of the vector space, creating new dimensions based on a range of values in the original dimensions or, less commonly, grouping multiple dimensions into a single one. Often, the set D is the union of multiple attribute domains, i.e. $D = \mathcal{A}_{k=1} \cup \mathcal{A}_{k=2} \cup \cdots \mathcal{A}_{k=l}$. The function $\eta : \Sigma_\alpha \to \mathbf{X}_\alpha^{n \times d}$, with $d = |D|$, assigns to $\mathbf{X}_\alpha$ the values of the attributes in e and groups events by case so that $\forall \mathbf{x} \forall \mathrm{a}_k : \mathbf{x}_{i,j} = \#_{\mathrm{a}_k}(e) \iff \#_{\mathrm{a}_k}(e) = D_j \wedge \#_{\text{case}}(e) = c_i$. The number of elements in the vector space equals the number of cases to include in the example matrix, thus $n \leq |\mathcal{C}|$. Because the sets Σ_α and D can be view as columnar matrices $M_{\Sigma_\alpha}^{n \times 1}$ and $M_D^{d \times 1}$, the size of $\mathbf{X}_\alpha$ is equal to $M_{\Sigma_\alpha} \times M_D^\top$, i.e. the set of events we selected with π is multiplied by the dimensions we identified with ρ. It is worth mentioning that, when grouping is applied, each vector component becomes an array of attribute values rather than a single value. The function ν aims at transforming these arrays of attribute values into real-valued scalar values. We define $\nu : \mathbf{X}_\alpha^{n \times d} \to \mathbf{X}^{n \times d}$ to clarify the components of the two matrices are valuated differently.

TABLE 1. ***The road traffic fines*** **event log.**

Cases	Number of Variants	Coverage of Cases
56482	1	37,6%
102853	2	68,4%
132758	4	88,3%
142926	7	95,0%
148887	17	99,0%
150270	131	99,9%
150370	231	100,0%

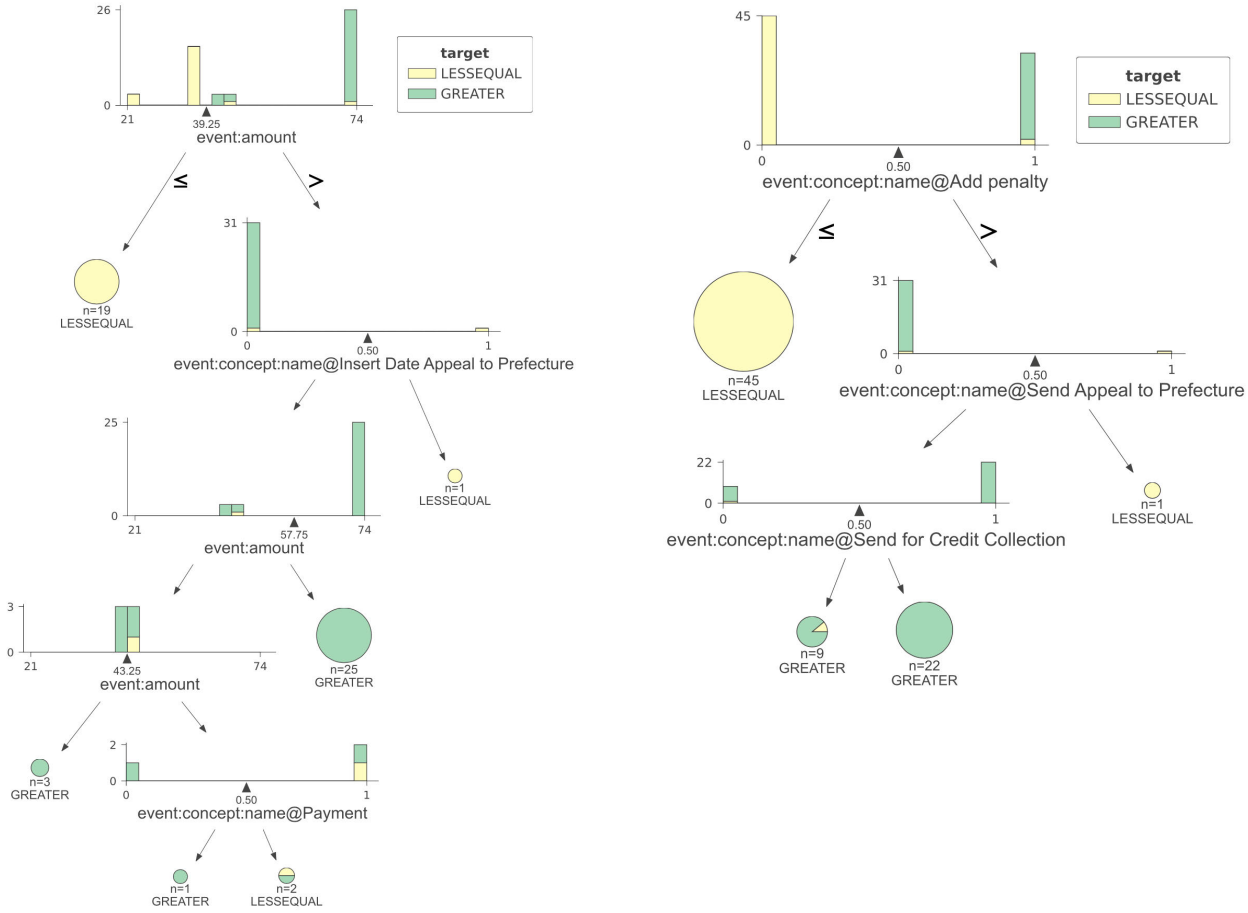
For example, the basic *one-hot* encoding schema corresponds to a null π, a ρ with $D = \bigcup_{k=1}^{l} \mathcal{A}_k$, an η for grouping the events of a same case, and a $\nu : \mathbf{X}_\alpha^{n \times d} \to \{0, 1\}^{n \times d}$, returning $\mathbf{x}_{i,j} = 1$ if at least a value $\#_{\mathrm{a}_k}(e) = D_j$ is observed for the case $\#_{\text{case}}(e) = c_i$, and 0 if not. The popular *activity profile* schema [12] encodes an event log into a vector of activity values by simply counting all events of a case that include that activity. The encoding function maps the events in $\hat{L}$ into $\mathbf{X}$ by executing the four canonical transformations as follows. First, it verifies to consider only events associated with activity values $\forall e \in \Sigma : \#_{\text{activity}}(e) \neq \perp$. Then it defines the dimensions of $\mathbf{X}$ with ρ so that $D = \mathcal{A}$, where $\mathcal{A}$ is the set of legal business process activities. Third, it aggregates the data by case with η. Finally, it performs the evaluation with ν, assigning the count of the components in $\mathbf{x}_{i,j}$ for each case c_i. For instance, the log $\overline{L} = [\langle a, b, c\rangle^3, \langle a, b, a\rangle^{11}, \langle a, c, b, a\rangle^{20}]$, is transformed in the first matrix in 1 with π, in the second matrix with ρ, in the third matrix in with η, to finally get the fourth matrix in 1 with ν.

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ \cdots \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \begin{bmatrix} a & b & c \\ a & b & c \\ a & b & c \\ [a,a] & b & \perp \\ [a,a] & b & \perp \\ \cdots & & \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 0 \\ 2 & 1 & 0 \\ \cdots & & \end{bmatrix} \tag{1}$$

We believe that if the PM community would get used to clarifying the definition of the following functions when defining an encoding procedure, the literature will benefit in terms of the comparability of the results. For example, a *data-flow* approach will require clarifying the contribution of the different dimensions in encoding cases. An *intra-case* approach will require modifying the η function to encode multiple cases into a single vector.

V. ILLUSTRATIVE EXAMPLES

We will now use two examples to illustrate the concepts introduced above. The first example relates to the impact of data distribution on PM analytics, as discussed in Section II-A. We refer to the real-life event log of "Road Traffic Fines" [47]. The events recorded in the event log include creating a fine notice, recording the fine amount, verifying that payment has been received, registering



(a) Unbalanced event log

(b) Balanced event log

FIGURE 2. Two *decision trees* generated from the "Road Traffic Fine" event log. In 2a the data in input conforms to the case distribution observed in the event log. As a consequence, the most frequent variants take the lion's share and the numeric feature *amount* decides multiple split points. In 2b data is balanced oversampling those variants with low occurrence. The split points in the tree use now categorical features. The decision tree is an example of an algorithm significantly affected by uncritical training using the case distribution of event logs.

an appeal with the prefecture, and others. The reader interested in more details is referred to [84]. As shown in Table 1, the occurrence of trace variants follows a Pareto distribution, with only 4 variants covering more than 88% of the recorded cases, and with 100 variants having a single occurrence. The most occurring variant is $\langle$*Create Fine*, *Send Fine*, *Insert Fine Notification*, *Add Penalty*, *Send for Credit Collection*$\rangle^{56482}$, the second is $\langle$*Create Fine*, *Payment*$\rangle^{46371}$, the third is $\langle$*Create Fine*, *Send Fine*$\rangle^{20385}$, and so on.

Now let us try to develop predictive analytics on this event log. For example, we might ask why certain cases have a significantly l onger d uration t han o thers. T o investigate the problem, we are interested in looking for patterns that correlate with long duration. The encoding method we used represent the cases in the event log as vectors composed of categorical data, such as the activities performed, and numerical data, such as the number of penalties and the trace duration.[2] A decision tree can then be used to highlight the factors that influence case duration. We express it as a simple binary problem: to be below or above a threshold of 200 days. Figure 2 illustrates the results we obtain. Figure 2a shows a decision tree corresponding to the case distribution observed in the event log. The whole set of cases in L is encoded in $\mathcal{X}$. As a result, the most frequent variants make up the lion's share of the examples used to train the decision tree. Figure 2b shows the decision tree obtained by balancing the case distribution among variants by oversampling the variants with low frequency. This is achieved, for example, by creating $\mathcal{X}$ which takes an equal number of occurrences for the variants in L. Since the splitting points of the tree are

[2]The methods used for encoding the event log in a vector space are available in the PM4PY library https://pm4py.fit.fraunhofer.de/documentation#decision-trees

chosen to best separate the examples into two groups with minimal mixing, the low-occurrence cases tend to be ignored. In fact, the tree in figure 2a relies on the numeric feature *amount* to decide on multiple split points. On the contrary, the tree in figure 2b defines the split points using only categorical features. This is due to the fact that the variants not associated with a penalty amount were quite rare, and by increasing their representation to balance the data set, we prevented the algorithm from using the penalty amount as a discriminative feature. It is important to note that, in general, we cannot say whether proactive balancing is better than using the data as it is, or even which balancing factor to apply. However, it is clear that the distribution of the data did affect our results. The choice of strategy is highly dependent on our specific objective. If the goal is to analyze an event log to identify automatable procedures and extract decision rules, the emphasis should be on frequent behavior [34]. On the other hand, if the goal is anomaly detection [87] or root cause analysis [26], the representation of rare examples becomes imperative.

Our next example relates to the need to capture concurrency, as discussed in Section II-A. While the events in an event log are described as sequences of activities, the behavior they describe should be interpreted differently based on the model that generated them. By running the Heuristic Miner algorithm [85] on the "Road Traffic Fines" [47] event log, we observe that alternative paths can be followed to complete the process. If a case includes the execution of the `Payment` activity, it will not include `Send Fine` and the following activities. The same algorithm applied to the "Artificial Patient Treatment" [86] will reveal the concurrent execution of the `Blood Test`, `X-ray Scan`, and `Physical Exam` activities. All of these activities are required to complete the diagnostic phase, except for the `X-ray scan` which can be skipped, but the order of execution is not relevant. Thanks to process models, PM techniques take concurrency into account. Two sequences $\langle a, b, c\rangle$ and $\langle a, c, b\rangle$ can have the same model compliance if the model describes b and c as concurrent activities, while the compliance value will be different if b and c are sequential or refer to alternative paths. Unfortunately, most ML models consider event logs as sequential data only. When cases are encoded in a vector space, the inference the ML model can make is based on the distance in that space. The distance between a sequence $\langle a, b, c\rangle$ and $\langle a, c, b\rangle$ is treated the same in the vector space, and we cannot distinguish between the sequences based on the reference process model. In terms of our example, an ML procedure could effectively predict the lead time of a case, knowing that the `Payment` activity was performed. While, training an ML algorithm to predict the compliance of a treatment with the diagnostic protocol is more complex and will require a larger amount of training data, since the ML model must incorporate examples of the equivalence of the different execution orders of the `Blood test`, `X-ray scan`, and `Physical test` activities. Encoding this equivalence in vector space spaces is still an open challenge.

VI. TOWARD AN INTEGRATED METHODOLOGY

Guided by the above considerations, we will now outline the strategy to be used to properly integrate PM and ML. Figure 4 provides a synoptic view of the mapping of PM tasks to ML tasks. The challenges discussed are organized vertically based on their complexity, starting from methods that have already entered practice in PM to open challenges that have not yet been solved by the research community.

For example, preprocessing techniques such as outlier removal [41], [42], noise filtering [43], [44], and missing entry recovery [45], are today addressed using unsupervised ML such as PCA and clustering.

Other challenges with moderate difficulty are related to label availability and unbalanced scenarios [87]. In this case, semi-supervised ML techniques, such as Active Learning, and generative models, such as Generative Adversarial Network (GAN) can be valuable solutions. Active learning [88] is valuable in PM because of its reduces the manual effort involved in labeling by intelligently selecting instances for annotation and optimizing the labeling process. This adaptability is critical for capturing evolving patterns in dynamic processes, allowing the model to iteratively refine its understanding of complex behaviors over time. Active learning also addresses challenges such as unbalanced data sets by strategically focusing on underrepresented classes, and helps reduce annotation bias by prioritizing uncertain or difficult instances. This iterative approach not only improves model performance, but also strategically leverages human expertise for more impactful contributions. Alternatively, the training process can be enriched using generative models [89]. To deal with the sequential nature of event logs Sequence GANs can be considered, in which the adversarial samples are designed from discrete sequences, such as events. The application of GANs is not limited to data augmentation, as they can also be used to improve data quality for process model generalization [89]. Preliminary results are available on using GAN-generated data to improve predictive tasks (e.g., lead time of incomplete cases) under an adversarial framework [90], [91].

When dealing with non-stationary process behavior, the use of sampling techniques proves to be a promising strategy for mitigating the effects of non-stationary distributions in event log data [92]. Once the data approaches a state of near-stationary behavior, it is noteworthy that the underlying business process may naturally change its pattern over time, leading to the phenomenon known as concept drift [21], [37], [93]. Despite efforts to address this challenge, we still consider it an open problem due to the fact that in the literature the event stream is typically modeled as a complete trace stream, assuming knowledge from the initiation to the completion of the activity. In reality, drift can begin at any point within the event stream, long before the endpoint is reached and the rest

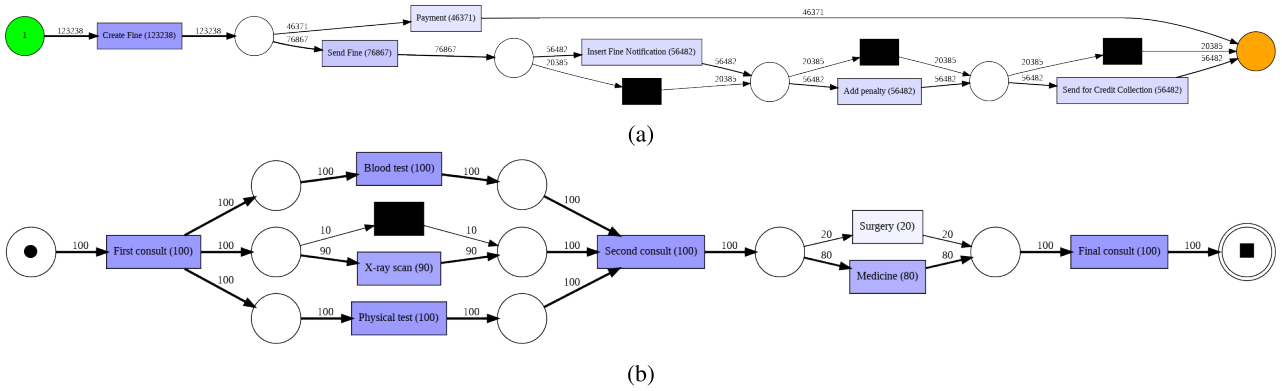


FIGURE 3. **(a) The Heuristic Miner Algorithm [85] was used to discover a model from the "Road Traffic Fines" [47] event log. The discovered model specifies alternative routes that can be followed to complete the process. In particular, executing `Payment` or `Send Fine` implies different subsequent paths. (b) The Heuristic Miner Algorithm [85] is used to discover a model from the "Artificial Patient Treatment" [86] event log. The discovered model specifies that `Blood test`, `X-ray scan`, and `Physical test` are executed in parallel. Any order can be followed in executing these activities.**

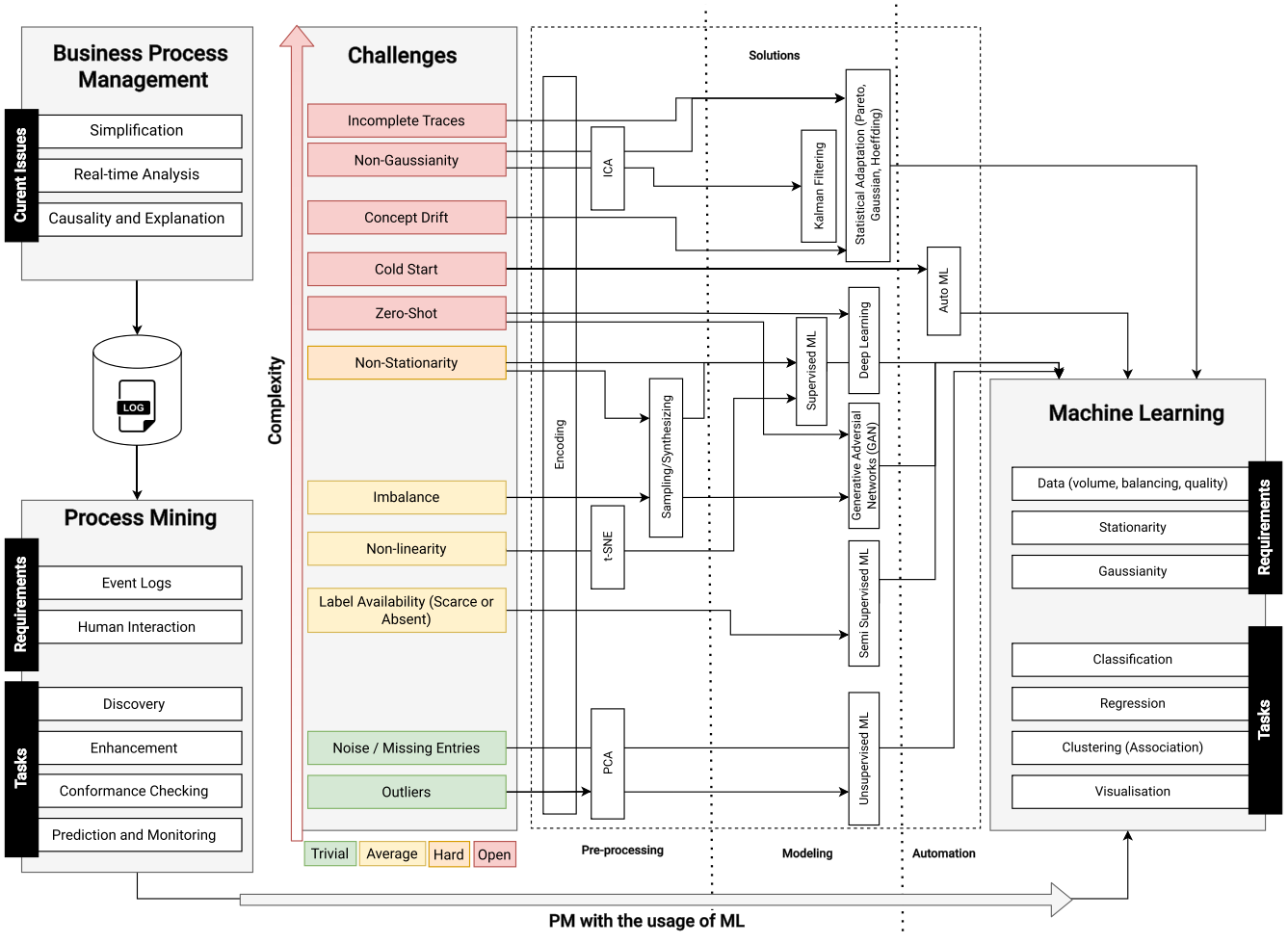


FIGURE 4. **From task to task, an overview of PM and ML relationship.**

of the trace is revealed. To address this information gap, some researchers make statistical adjustments based on Hoeffding bounds [94]. Essentially, using statistical assumptions about the confidence interval of the data becomes a means of making decisions about the onset of drift. The use of "stateful" ML models, especially deep learners based on

the LSTM architecture, holds the potential to effectively handle drifts. However, addressing such challenges requires the expertise of experienced machine learning practitioners and the establishment of a robust computing infrastructure.

A. HYPER-PARAMETER TUNING

Once a class of ML models is selected, hyperparameter tuning must be performed to instantiate the ML model that provides the desired accuracy (and possibly some required non-functional properties, such as explainability). Searching the model space by trial and error can be tedious. Automated Machine Learning (AutoML) is a reasonable alternative to address these problems, based on sharing prior knowledge for similar tasks. AutoML can help solve cold-start [95], for which little contextual information (and even the full list of classes) may not be available at the start of training, by exploiting meta-features and information about similar models, similar to how human experts start an old-fashioned search for desirable models driven by their experience on related tasks [96]. Some AutoML-based PM research discusses how to find an appropriate PM pipeline by recommending steps [15], [97]. For example, [97] suggests the encoding method, since the large number of available techniques can make the selection difficult. Furthermore, there are encoding methods that can be adapted to certain data.

VII. FINAL RECOMMENDATIONS

In this final section, we present a set of recommendations that aim to be valuable for both PM practitioners and researchers.

A. RECOMMENDATION 1: CHOOSE DATA REPRESENTATION CAREFULLY

When dealing with PM data structures, it is crucial to translate them into a metric feature space suitable for manipulation by ML algorithms. Preserve contextual information, such as control-flow, concurrency, and inter-case constraints, which are essential for thorough process analysis. The selection of encoding techniques should align with problem-specific goals and constraints.

B. RECOMMENDATION 2: FIT THE DATA DISTRIBUTIONS

PM often involves non-Gaussian, non-stationary distributions. At the same time, skewed distributions can significantly influence the learning process. Testing different class balance configurations may be required for reliable training. Use AutoML to reduce manual effort and improve ML models.

C. RECOMMENDATION 3: EMBRACE ZERO-SHOT LEARNING

Given the unpredictability of outcomes in PM tasks, especially in domains such as process optimization where the full set of possible outcomes may be partially known, incorporating zero-shot learning, such as using LLMs, can be beneficial. However, effectively adapting these models to PM tasks remains an open challenge.

D. RECOMMENDATION 4: ENSURE EARLY QUALITY WITH CONSTRAINTS

Given the gradual convergence of data distribution estimation, avoiding a long convergence period is essential to prevent a significant increase in model error during training. Impose control-flow or resource-usage constraints on ML models is still an open challenge.

E. RECOMMENDATION 5: LEVERAGE DOMAIN KNOWLEDGE

Domain knowledge plays a critical role in effective PM. Integrating domain-specific information and constraints into ML models significantly improves their performance and interpretability. Actively involving domain experts in feature engineering and model validation processes ensures that the developed models align with the intricacies of the business processes. Consider employing Active Learning as a valuable technique to iteratively engage domain experts and refine models based on their insights.

F. RECOMMENDATION 6: EVALUATE MODEL INTERPRETABILITY

PM tasks often require interpretable models to gain insight into process behavior and make informed decisions. Evaluate the interpretability of ML models and select algorithms that provide transparent explanations of predictions, especially in contexts involving critical processes or compliance and regulatory requirements.

G. RECOMMENDATION 7: CONTINUOUSLY MONITOR AND UPDATE ML MODELS

Process environments are dynamic, and changes over time can affect ML model performance. Establishing a monitoring and evaluation framework facilitates timely updates to ensure accuracy and relevance in evolving process scenarios. Concept-drift detection and continuous learning is essential.

H. RECOMMENDATION 8: ENCOURAGE KNOWLEDGE SHARING AND COLLABORATION

Encourage knowledge sharing and collaboration within the PM community. Encourage the dissemination of successful case studies, research, and best practices defining and using benchmarks. Participate in conferences, workshops, and online forums to connect with other practitioners and researchers and stay abreast of the latest ML/PM developments.

By following these recommendations, PM practitioners and researchers can improve the effectiveness and efficiency of process mining applications, enabling better process understanding, optimization, and decision making.

VIII. CONCLUSION

The growing use of ML methods in PM requires a robust and comprehensive methodology for integrating these algorithmic techniques. The purpose of this paper was to address

the challenges associated with ML/PM mapping and to identify the basic principles for establishing a methodological foundation in this area. Through the analysis conducted in this study, we have provided a set of recommendations that can guide practitioners and researchers in effectively applying ML to PM tasks. These recommendations cover various aspects of the PM process, from data representation to model evaluation and monitoring. By following these recommendations, PM practitioners and researchers can improve the effectiveness and efficiency of their ML-driven process mining applications. It is important to recognize that the field of ML in PM is constantly evolving, and new challenges and opportunities will continue to emerge. Therefore, ongoing research and collaboration between practitioners and researchers is essential to refine and extend the proposed recommendations. By adopting a methodological foundation that integrates ML techniques into PM, we can unlock the full potential of process mining and harness the power of data-driven insights to drive process understanding, optimization, and decision making across multiple domains and industries.

REFERENCES

[1] Deloitte. (2021). *Global Process Mining Survey*. [Online]. Available: https://mpm-processmining.com/en/global-process-mining-survey-2021/

[2] W. Van Der Aalst, A. Adriansyah, A. K. A. De Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. Van Den Brand, R. Brandtjen, and J. Buijs, "Process mining manifesto," in *Proc. Int. Conf. Bus. Process Manag.* Cham, Switzerland: Springer, 2011, pp. 169–194.

[3] M. Imran, S. Hamid, and M. A. Ismail, "Advancing process audits with process mining: A systematic review of trends, challenges, and opportunities," *IEEE Access*, vol. 11, pp. 68340–68357, 2023.

[4] O. Loyola-González, "Process mining: Software comparison, trends, and challenges," *Int. J. Data Sci. Analytics*, vol. 15, no. 4, pp. 407–420, May 2023.

[5] T. Teubner, C. M. Flath, C. Weinhardt, W. van der Aalst, and O. Hinz, "Welcome to the era of chatgpt et al. the prospects of large language models," *Bus. Inf. Syst. Eng.*, vol. 65, no. 2, pp. 95–101, Apr. 2023.

[6] D. Chapela-Campa and M. Dumas, "From process mining to augmented process execution," *Softw. Syst. Model.*, vol. 22, no. 6, pp. 1977–1986, Dec. 2023.

[7] A. Berti, D. Schuster, and W. M. van der Aalst, "Abstractions, scenarios, and prompt definitions for process mining with LLMs: A case study," in *Proc. Int. Conf. Bus. Process Manag.* Cham, Switzerland: Springer, 2023, pp. 427–439.

[8] A. Berti, S. van Zelst, and D. Schuster, "PM4Py: A process mining library for Python," *Softw. Impacts*, vol. 17, Sep. 2023, Art. no. 100556.

[9] N. Tax, N. Sidorova, R. Haakma, and W. M. van der Aalst, "Event abstraction for process mining using supervised learning techniques," in *Proc. SAI Intell. Syst. Conf.* Cham, Switzerland: Springer, 2016, pp. 251–269.

[10] S. J. van Zelst, F. Mannhardt, M. de Leoni, and A. Koschmider, "Event abstraction in process mining: Literature review and taxonomy," *Granular Comput.*, vol. 6, no. 3, pp. 719–736, Jul. 2021.

[11] G. Tello, G. Gianini, R. Mizouni, and E. Damiani, "Machine learning-based framework for log-lifting in business process mining applications," in *Proc. Int. Conf. Bus. Process Manage.* Cham, Switzerland: Springer, 2019, pp. 232–249.

[12] M. Song, C. W. Günther, and W. M. Van der Aalst, "Trace clustering in process mining," in *Proc. Int. Conf. Bus. Process Manage.*, Cham, Switzerland: Springer, 2008, pp. 109–120.

[13] R. J. C. Bose and W. M. Van der Aalst, "Context aware trace clustering: Towards improving process mining results," in *Proc. SIAM Int. Conf. Data Mining*, 2009, pp. 401–412.

[14] A. Appice and D. Malerba, "A co-training strategy for multiple view clustering in process mining," *IEEE Trans. Services Comput.*, vol. 9, no. 6, pp. 832–845, Nov. 2016.

[15] G. M. Tavares, S. Barbon Junior, E. Damiani, and P. Ceravolo, "Selecting optimal trace clustering pipelines with meta-learning," in *Proc. Brazilian Conf. Intell. Syst.* Cham, Switzerland: Springer, 2022, pp. 150–164.

[16] A. Kalenkova, A. Polyvyanyy, and M. La Rosa, "A framework for estimating simplicity of automatically discovered process models based on structural and behavioral characteristics," in *Proc. Int. Conf. Bus. Process Manag.* Cham, Switzerland: Springer, 2020, pp. 129–146.

[17] A. Senderovich, A. Shleyfman, M. Weidlich, A. Gal, and A. Mandelbaum, "To aggregate or to eliminate? Optimal model simplification for improved process performance prediction," *Inf. Syst.*, vol. 78, pp. 96–111, Nov. 2018.

[18] D. Chapela-Campa, M. Mucientes, and M. Lama, "Simplification of complex process models by abstracting infrequent behaviour," in *Proc. Int. Conf. Service-Oriented Comput.* Cham, Switzerland: Springer, 2019, pp. 415–430.

[19] V. P. Mishra, B. Shukla, and A. Bansal, "Analysis of alarms to prevent the organizations network in real-time using process mining approach," *Cluster Comput.*, vol. 22, no. 3, pp. 7023–7030, May 2019.

[20] G. M. Tavares, P. Ceravolo, V. G. T. Da Costa, E. Damiani, and S. B. Junior, "Overlapping analytic stages in online process mining," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Jul. 2019, pp. 167–175.

[21] P. Ceravolo, G. M. Tavares, S. B. Junior, and E. Damiani, "Evaluation goals for online process mining: A concept drift perspective," *IEEE Trans. Services Comput.*, vol. 15, no. 4, pp. 2473–2489, Jul. 2022.

[22] N. Di Mauro, A. Appice, and T. M. A. Basile, "Activity prediction of business process instances with inception CNN models," in *AI*IA 2019—Advances in Artificial Intelligence*. M. Alviano, G. Greco, and F. Scarcello, Eds. Cham, Switzerland: Springer, 2019, pp. 348–361.

[23] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "Predictive process mining meets computer vision," in *Proc. Int. Conf. Bus. Process Manage.* Cham, Switzerland: Springer, 2020, pp. 176–192.

[24] A. E. Márquez-Chamorro, M. Resinas, and A. Ruiz-Cortés, "Predictive monitoring of business Processes: A survey," *IEEE Trans. Services Comput.*, vol. 11, no. 6, pp. 962–977, Nov. 2018.

[25] Z. D. Bozorgi, I. Teinemaa, M. Dumas, M. La Rosa, and A. Polyvyanyy, "Process mining meets causal machine learning: Discovering causal rules from event logs," in *Proc. 2nd Int. Conf. Process Mining (ICPM)*, Oct. 2020, pp. 129–136.

[26] M. S. Qafari and W. van der Aalst, "Root cause analysis in process mining using structural equation models," in *Proc. Int. Conf. Bus. Process Manag.* Cham, Switzerland: Springer, 2020, pp. 155–167.

[27] K. M. Hanga, Y. Kovalchuk, and M. M. Gaber, "A graph-based approach to interpreting recurrent neural networks in process mining," *IEEE Access*, vol. 8, pp. 172923–172938, 2020.

[28] N. Guo, C. Liu, C. Li, Q. Zeng, C. Ouyang, Q. Liu, and X. Lu, "Explainable and effective process remaining time prediction using feature-informed cascade prediction model," *IEEE Trans. Services Comput.*, early access, Jan. 15, 2024, doi: 10.1109/TSC.2024.3353817.

[29] M. Prodel, V. Augusto, X. Xie, B. Jouaneton, and L. Lamarsalle, "Discovery of patient pathways from a national hospital database using process mining and integer linear programming," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2015, pp. 1409–1414.

[30] Y. Al-Anqoudi, A. Al-Hamdani, M. Al-Badawi, and R. Hedjam, "Using machine learning in business process re-engineering," *Big Data Cognit. Comput.*, vol. 5, no. 4, p. 61, Nov. 2021.

[31] W. V. D. Aalst and E. Damiani, "Processes meet big data: Connecting data science with process science," *IEEE Trans. Services Comput.*, vol. 8, no. 6, pp. 810–819, Nov. 2015.

[32] W. van der Aalst, "Academic view: Development of the process mining discipline," in *Process Mining in Action*. Berlin, Germany: Springer, 2020, pp. 181–196.

[33] F. Veit, J. Geyer-Klingeberg, J. Madrzak, M. Haug, and J. Thomson, "The proactive insights engine: Process mining meets machine learning and artificial intelligence," in *Proc. BPM (Demos)*, 2017, pp. 1–5.

[34] W. van der Aalst, "On the Pareto principle in process mining, task mining, and robotic process automation," in *Proc. 9th Int. Conf. Data Sci., Technol. Appl.*, 2020, pp. 5–12.

[35] G. M. Tavares, R. S. Oyamada, S. Barbon, and P. Ceravolo, "Trace encoding in process mining: A survey and benchmarking," *Eng. Appl. Artif. Intell.*, vol. 126, Nov. 2023, Art. no. 107028.

[36] M. Vazifehdoostirani, L. Genga, and R. Dijkman, "Encoding high-level control-flow construct information for process outcome prediction," in *Proc. 4th Int. Conf. Process Mining (ICPM)*, Oct. 2022, pp. 48–55.

[37] L. Baier, J. Reimold, and N. Kühl, "Handling concept drift for predictions in business process mining," in *Proc. IEEE 22nd Conf. Bus. Informat. (CBI)*, vol. 1, 2020, pp. 76–83.

[38] B. Hilprecht and C. Binnig, "One model to rule them all: Towards zero-shot learning for databases," 2021, *arXiv:2105.00642*.

[39] E. Rama-Maneiro, J. C. Vidal, and M. Lama, "Deep learning for predictive business process monitoring: Review and benchmark," *IEEE Trans. Services Comput.*, vol. 16, no. 1, pp. 739–756, Jan. 2023.

[40] J. Thiyagalingam, M. Shankar, G. Fox, and T. Hey, "Scientific machine learning benchmarks," *Nature Rev. Phys.*, vol. 4, no. 6, pp. 413–420, Apr. 2022.

[41] M. F. Sani, S. J. van Zelst, and W. M. van der Aalst, "Applying sequence mining for outlier detection in process mining," in *Proc. OTM Confederated Int. Conf. Move Meaningful Internet Syst.* Cham, Switzerland: Springer, 2018, pp. 98–116.

[42] M. F. Sani, S. van Zelst, and W. M. van der Aalst, "Repairing outlier behaviour in event logs using contextual behaviour," *Enterprise Model. Inf. Syst. Architectures (EMISAJ)*, vol. 14, pp. 1–5, Dec. 2019.

[43] W. Li, H. Zhu, W. Liu, D. Chen, J. Jiang, and Q. Jin, "An anti-noise process mining algorithm based on minimum spanning tree clustering," *IEEE Access*, vol. 6, pp. 48756–48764, 2018.

[44] X. Sun, W. Hou, D. Yu, J. Wang, and J. Pan, "Filtering out noise logs for process modelling based on event dependency," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jul. 2019, pp. 388–392.

[45] F. Fox, V. R. Aggarwal, H. Whelton, and O. Johnson, "A data quality framework for process mining of electronic health record data," in *Proc. IEEE Int. Conf. Healthcare Informat. (ICHI)*, Jun. 2018, pp. 12–21.

[46] A. Berti, "Statistical sampling in process mining discovery," in *Proc. 9th Int. Conf. Inf., Process, Knowl. Manag.*, Mar. 2017, pp. 41–43.

[47] M. De Leoni and F. Mannhardt, "Road traffic fine management process," Eindhoven Univ. Technol., The Netherlands, Tech. Rep., 2015, doi: 10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5.

[48] J. Buijs, "Receipt phase of an environmental permit application process," Eindhoven Univ. Technol., The Netherlands, Tech. Rep., 2014, doi: 10.4121/uuid:a07386a5-7be3-4367-9535-70bc9e77dbe6.

[49] B. van Dongen, "Bpi challenge 2015 municipality 1," Eindhoven Univ. Technol., The Netherlands, Tech. Rep., 2015, doi: 10.4121/uuid:a0addfda-2044-4541-a450-fdcc9fe16d17.

[50] S. Maghool, E. Casiraghi, and P. Ceravolo, "Enhancing fairness and accuracy in machine learning through similarity networks," in *Proc. Int. Conf. Cooperat. Inf. Syst.* Cham, Switzerland: Springer, 2023, pp. 3–20.

[51] T.-W. Lee, "Independent component analysis," in *Independent Component Analysis*. Berlin, Germany: Springer, 1998, pp. 27–66.

[52] R. M. Sakia, "The box-cox transformation technique: A review," *J. Roy. Stat. Soc., Ser. D*, vol. 41, no. 2, pp. 169–178, 1992.

[53] M. Roccetti, G. Delnevo, L. Casini, and S. Mirri, "An alternative approach to dimension reduction for Pareto distributed data: A case study," *J. Big Data*, vol. 8, no. 1, pp. 1–23, Dec. 2021.

[54] V. Bellandi, E. Damiani, V. Ghirimoldi, S. Maghool, and F. Negri, "Validating vector-label propagation for graph embedding," in *Cooperative Information Systems*, M. Sellami, P. Ceravolo, H. A. Reijers, W. Gaaloul, and H. Panetto, Eds. Cham, Switzerland: Springer, 2022, pp. 259–276.

[55] W. M. van der Aalst, "Concurrency and objects matter! Disentangling the fabric of real operational processes to create digital twins," in *International Colloquium on Theoretical Aspects of Computing*. Berlin, Germany: Springer, 2021, pp. 3–17.

[56] C. D. Francescomarino, C. Ghidini, F. M. Maggi, G. Petrucci, and A. Yeshchenko, "An eye into the future: Leveraging a-priori knowledge in predictive business process monitoring," in *Proc. Int. Conf. Bus. Process Manag.*, Barcelona, Spain. Cham, Switzerland: Springer, 2017, pp. 252–268.

[57] K. Thapa, Z. M. Abdullah, B. Lamichhane, and S.-H. Yang, "A deep machine learning method for concurrent and interleaved human activity recognition," *Sensors*, vol. 20, no. 20, p. 5770, Oct. 2020.

[58] K. Thapa, Z. M. Abdhulla AI, and Y. Sung-Hyun, "Adapted long short-term memory (LSTM) for concurrent human activity recognition," *Comput., Mater. Continua*, vol. 69, no. 2, pp. 1653–1670, 2021.

[59] D. A. Neu, J. Lahann, and P. Fettke, "A systematic literature review on state-of-the-art deep learning methods for process prediction," *Artif. Intell. Rev.*, vol. 55, no. 2, pp. 801–827, Feb. 2022.

[60] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Wozniak, "Ensemble learning for data stream analysis: A survey," *Inf. Fusion*, vol. 37, pp. 132–156, Sep. 2017.

[61] D. M. V. Sato, S. C. De Freitas, J. P. Barddal, and E. E. Scalabrin, "A survey on concept drift in process mining," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 1–38, Dec. 2022.

[62] J. N. Adams, S. J. van Zelst, L. Quack, K. Hausmann, W. M. van der Aalst, and T. Rose, "A framework for explainable concept drift detection in process mining," in *Business Process Management*. Rome, Italy: Springer, 2021, pp. 400–416.

[63] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "DARWIN: An online deep learning approach to handle concept drifts in predictive process monitoring," *Eng. Appl. Artif. Intell.*, vol. 123, Aug. 2023, Art. no. 106461.

[64] B. Scheibel and S. Rinderle-Ma, "An end-to-end approach for online decision mining and decision drift analysis in process-aware information systems," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.* Cham, Switzerland: Springer, 2023, pp. 17–25.

[65] M. Käppel, S. Schönig, and S. Jablonski, "Leveraging small sample learning for business process management," *Inf. Softw. Technol.*, vol. 132, Apr. 2021, Art. no. 106472.

[66] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 22199–22213.

[67] M. Vidgof, S. Bachhofner, and J. Mendling, "Large language models for business process management: Opportunities and challenges," 2023, *arXiv:2304.04309*.

[68] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," 2020, *arXiv:2011.13456*.

[69] C. Sun, J. Han, W. Deng, X. Wang, Z. Qin, and S. Gould, "3D-GPT: Procedural 3D modeling with large language models," 2023, *arXiv:2310.12945*.

[70] N. Tax, I. Verenich, M. L. Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.*, vol. 10253, E. Dubois and K. Pohl, Eds. Cham, Switzerland: Springer, 2017, pp. 477–492.

[71] S. M. Weiss, N. Indurkhya, and T. Zhang, *Fundamentals of Predictive Text Mining* (Texts in Computer Science), 2nd ed. Berlin, Germany: Springer, 2015.

[72] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. 31st Int. Conf. Int. Conf. Mach. Learn.*, vol. 32, 2014, pp. 1188–1196.

[73] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. KDD*. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 855–864.

[74] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, "Don't walk, skip: Online learning of multi-scale network embeddings," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*. New York, NY, USA: Association for Computing Machinery, Jul. 2017, pp. 258–265.

[75] G. M. Tavares and S. Barbon, "Analysis of language inspired trace representation for anomaly detection," in *ADBIS, TPDL and EDA 2020 Common Workshops and Doctoral Consortium*. Berlin, Germany: Springer, 2020, pp. 296–308.

[76] A. Chiorrini, C. Diamantini, L. Genga, M. Pioli, and D. Potena, "Embedding process structure in activities for process mapping and comparison," in *New Trends in Database and Information Systems (ADBIS)*, vol. 1652, S. Chiusano, T. Cerquitelli, R. Wrembel, K. Nørvåg, B. Catania, G. Vargas-Solar, and E. Zumpano, Eds. Berlin, Germany: Springer, 2022, pp. 119–129.

[77] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "A multi-view deep learning approach for predictive business process monitoring," *IEEE Trans. Services Comput.*, vol. 15, no. 4, pp. 2382–2395, Jul. 2022.

[78] A. Guzzo, M. Joaristi, A. Rullo, and E. Serra, "A multi-perspective approach for the analysis of complex business processes behavior," *Exp. Syst. Appl.*, vol. 177, Sep. 2021, Art. no. 114934.

[79] J. Kim, M. Comuzzi, M. Dumas, F. M. Maggi, and I. Teinemaa, "Encoding resource experience for predictive process monitoring," *Decis. Support Syst.*, vol. 153, Feb. 2022, Art. no. 113669.

[80] I. Teinemaa, M. Dumas, M. L. Rosa, and F. M. Maggi, "Outcome-oriented predictive process monitoring: Review and benchmark," *ACM Trans. Knowl. Discovery Data (TKDD)*, vol. 13, no. 2, pp. 1–57, 2019.

[81] P. D. Koninck, S. V. Broucke, and J. D. Weerdt, "act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes," in *Business Process Management* (Lecture Notes in Computer Science), vol. 11080, M. Weske, M. Montali, I. Weber, and J. vom Brocke, Eds. Berlin, Germany: Springer, 2018, pp. 305–321.
[82] A. Senderovich, C. D. Francescomarino, C. Ghidini, K. Jorbina, and F. M. Maggi, "Intra and inter-case features in predictive process monitoring: A tale of two dimensions," in *Business Process Management* (Lecture Notes in Computer Science), vol. 10445, J. Carmona, G. Engels, and A. Kumar, Eds. Berlin, Germany: Springer, 2017, pp. 306–323.
[83] M. de Leoni and W. M. P. van der Aalst, "Data-aware process mining: Discovering decisions in processes using alignments," in *Proc. Symp. Appl. Comput. (SAC)*, S. Y. Shin and J. C. Maldonado, Eds. 2013, pp. 1454–1461.
[84] F. Mannhardt, M. de Leoni, H. A. Reijers, and W. M. P. van der Aalst, "Decision mining revisited - discovering overlapping rules," in *Advanced Information Systems Engineering*, S. Nurcan, P. Soffer, M. Bajec, and J. Eder, Eds. Cham, Switzerland: Springer, 2016, pp. 377–392.
[85] A. J. M. M. Weijters, W. M. P. van Der Aalst, and A. K. A. De Medeiros, "Process mining with the heuristics miner-algorithm," Technische Univ. Eindhoven, The Netherlands, Working Papers, 2006, vol. 166. [Online]. Available: https://pure.tue.nl/ws/portalfiles/portal/2388011/
[86] (2020). *Process Mining in Healthcare Tutorial*. [Online]. Available: https://gitlab.com/healthcare2/process-mining-tutorial
[87] S. B. Junior, P. Ceravolo, E. Damiani, N. J. Omori, and G. M. Tavares, "Anomaly detection on event logs with a scarcity of labels," in *Proc. 2nd Int. Conf. Process Mining (ICPM)*, 2020, pp. 161–168.
[88] D. Schuster, S. J. van Zelst, and W. M. P. van der Aalst, "Cortado: A dedicated process mining tool for interactive process discovery," *SoftwareX*, vol. 22, May 2023, Art. no. 101373. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352711023000699
[89] J. Theis and H. Darabi, "Adversarial system variant approximation to quantify process model generalization," *IEEE Access*, vol. 8, pp. 194410–194427, 2020.
[90] F. Taymouri, M. L. Rosa, S. Erfani, Z. D. Bozorgi, and I. Verenich, "Predictive business process monitoring via generative adversarial nets: The case of next event prediction," in *Proc. Int. Conf. Bus. Process Manag.* Cham, Switzerland: Springer, 2020, pp. 237–256.
[91] C. van Dun, L. Moder, W. Kratsch, and M. Röglinger, "ProcessGAN: Supporting the creation of business process improvement ideas through generative machine learning," *Decis. Support Syst.*, vol. 165, Feb. 2023, Art. no. 113880.
[92] W. C. Cheung, D. Simchi-Levi, and R. Zhu, "Learning to optimize under non-stationarity," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 1079–1087.
[93] J. Carmona and R. Gavalda, "Online techniques for dealing with concept drift in process mining," in *Proc. Int. Symp. Intell. Data Anal.* Berlin, Germany: Springer, 2012, pp. 90–102.
[94] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2000, pp. 71–80.
[95] H. Chemingui, I. Gam, R. Mazo, C. Salinesi, and H. B. Ghezala, "Product line configuration meets process mining," *Proc. Comput. Sci.*, vol. 164, pp. 199–210, Jan. 2019.
[96] R. L. Hu, C. Xiong, and R. Socher. (2019). *Correction Networks: Meta-Learning for Zero-Shot Learning*. [Online]. Available: https://openreview.net/forum?id=r1xurn0cKQ
[97] G. M. Tavares and S. B. Junior, "Process mining encoding via meta-learning for an enhanced anomaly detection," in *Proc. Eur. Conf. Adv. Databases Inf. Syst.* Cham, Switzerland: Springer, 2021, pp. 157–168.

PAOLO CERAVOLO (Member, IEEE) is currently an Associate Professor with the Department of Computer Science, University of Milan, Italy. His research interests include data representation and integration, business process monitoring, and empirical software engineering. On these topics, he has published several scientific articles. As a data scientist, he was involved in several international research projects and innovative startups. For more information please visit: http://www.di.unimi.it/ceravolo.

SYLVIO BARBON JUNIOR is an Associate Professor with the Department of Engineering and Architecture, University of Trieste (UNITS), Italy. He is currently a Co-Director of the Machine Learning Lab. Prior to this, from 2012 to 2021, he led a research group dedicated to the study of machine learning with the Computer Science Department, State University of Londrina (UEL), Brazil. His research interests include encompass computer vision, pattern recognition, and machine learning, with a current emphasis on meta-learning, stream mining, and process mining.

ERNESTO DAMIANI (Senior Member, IEEE) received the Honorary Doctorate degree for "his contribution to big data teaching and research" from the Institute National des Sciences Appliques de Lyon, France. He is currently a Full Professor with the University of Milan, Milan, Italy, and the Founding Director of the Center for Cyber-Physical Systems, Khalifa University, United Arab Emirates. His research interests include cybersecurity, big data, and cloud/edge processing.

WIL VAN DER AALST (Fellow, IEEE) is currently an Alexander-von-Humboldt Professor with RWTH Aachen University, leading the Process and Data Science (PADS) Group. He is also the Chief Scientist at Celonis, and part-time affiliated with the Fraunhofer FIT. He is the Deputy CEO of the Internet of Production (IoP) Cluster of Excellence and the Co-Director of the RWTH Center for Artificial Intelligence. He is an IFIP Fellow, ACM Fellow, and an elected member of the Royal Netherlands Academy of Arts and Sciences, the Royal Holland Society of Sciences and Humanities, the Academy of Europe, the North Rhine-Westphalian Academy of Sciences, Humanities and the Arts, and the German Academy of Science and Engineering.