IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

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](Senior member, IEEE)

[1]Computer Science Department, University of Milan, Italy
[2]Department of Engineering and Architecture, University of Trieste, Italy
[3]Department of Electrical Engineering and Computer Science, Khalifa University, UAE
[4]Chair of Process and Data Science, RWTH Aachen University, Germany

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

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, the learning procedure they follow ignores the constraints *concurrency* imposes on process data. 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 (Imran et al., 2023). 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 [4, 5, 6]. 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 [7]. Figure 1 point (1) highlights *data transformation*, *noise reduction*, and *feature engineering*. Figure 1 point (2) mentions *prediction*, *simulation*, and *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 [8, 9] that compute virtual activity logs, a problem which is also known as *log lifting* [10]. 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 [11, 12, 13, 14], to simplify the discovered models [15, 16, 17], or to support real-time analysis on

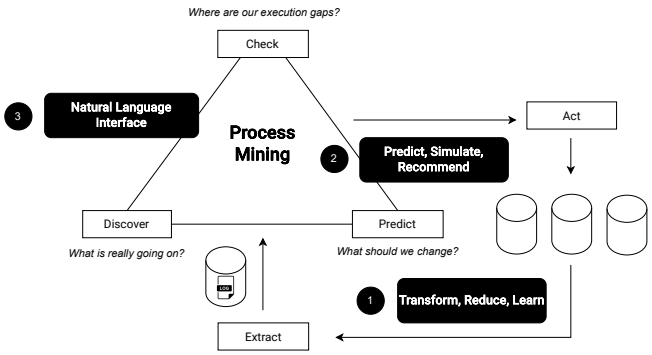


FIGURE 1. The PM tasks and their relation to ML

event streams [18, 19, 20]. 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 [21, 22, 23]. Root cause analysis [24] and data explainability [25] 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 [26] to *optimize* business processes' resource consumption and to provide insights to process *re-design* [27]. Computational support for PM appears to align with that of ML models from a technological standpoint [28, 29, 30]. This convergence might suggest a straightforward integration of these fields.

However, in practice, this integration is far from straightforward. When mapping PM tasks to ML tasks, it becomes imperative to construct training functions and select hyper-parameters 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 [31]. 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. 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. 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 the relevance of features for analysis purposes. This includes tasks like detecting *concept drift* and supporting *zero-shot learning* [32].

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 the event logs used to feed them [33]. Arbitrarily selecting algorithms leads to unfair evaluation and sub-optimal solutions. For example, a given model cannot be compared with another if their implementations consider different feature spaces [34]. It is also important to make sure that ML models are exposed to process-specific information, such as the processes' control-flow constraints. 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 an effort on these aspects must be jointly made by the PM and ML research communities. This call to 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. 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 VII 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 model 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 [35, 36], noise filtering [37, 38], and missing entry recovery [39], as these can be addressed by current statistical techniques. Rather, we will focus on issues specific to process data, including its statistical distribution and event concurrency. Indeed, careless assumptions about the encoding of input data can lead to biased models with reduced generalizability.

A. DATA DISTRIBUTION

When choosing an ML model for a PM task, it is tempting to assume that the process data fed to the model will follow a normal distribution. Indeed, data normality is beneficial for many types of ML models. Models like Gaussian, naive Bayes, logistic and linear regression explicitly rely on the assumption that the data distribution is bi-variate 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 always 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* trend that complies with the *Pareto principle* [31]. A normal distribution cannot always be assured also for the pairwise dependency relationship between activities, a key statistical information exploited by process discovery algorithms [40]. 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" [41] and "Receipt phase of an environmental permit application process" [42]). However, the normality of dependencies in less regular or not observed in "spaghetti" like processes, as in the "BPI Challenge 2015 Municipality 1"[43]. 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. Imbalanced data sets or non-stationary environments may also cause serious difficulties. For example, if the training data is skewed towards a particular class or outcome, the model may be more likely to predict that class or outcome even when it is not the most likely one. Independent component analysis [44] provides ways to reveal Gaussianity and non-Gaussianity. Of course, non-normal distributions can be transformed to normal ones using Box-Cox transformations [45], and unbalanced data sets can be balanced [46, 47] but, as we shall see, such data transformations should be applied with caution, as they have consequences on the performance of the models.

Of course, an explicit estimate of the data distribution may not even be necessary. Some ML models work well also in the case of non-normally distributed data. Simple yet effective ML models like decision trees and random forests do not assume any normality and work reasonably well on raw event data. Also, linear regression is statistically effective if the model errors are Gaussian, an assumption less stringent for process data than the normality of the entire data set. Kernel methods, e.g., Gaussian processes and support vector machines, provide flexible models that are practical to work with but require proper hyperparameter variables to fit the data.

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 [48], 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 [49]. To address this issue, it is important to provide ML models with control flow information about the iterative or concurrent execution of tasks as additional context to the event logs. One approach that has been explored is the use of Bi-directional Long-Short Term Memory (BiLSTM) architectures. Thapa et al. [50] used BiLSTM to detect concurrent human activities in a smart home environment. In addition, Thapa et al. [51] adapted the LSTM algorithm into a synchronous algorithm called sync-LSTM, enabling the model to handle multiple parallel input sequences and generate multiple synchronized output sequences. The field of predicting the behavior of highly concurrent systems using ML is rapidly evolving, as evidenced by the recent review by Neu et al. [52]. Researchers are actively exploring new techniques and methodologies to improve the understanding and prediction of concurrency in various domains.

C. NON-STATIONARY BEHAVIOUR

Even when the process data distributions can be fitted precisely, running processes, especially the ones involving resources that learn and age like people and equipment, change over time. This gives rise to *non-stationary* behavior. This problem is a critical one since ML models' learning capacity decreases under non-stationary conditions [20].Concept drift detection techniques are therefore needed. 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 [53]. 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 interdependence between the activities and the available resources or cycle time. Concept drift can compromise the accuracy of PPM models by causing degradation in performance due to evolving patterns and dynamics within the process data. Each aspect should be appropriately encoded and monitored using statistical analysis [54].

D. LLM FOR ZERO-SHOT LEARNING

Zero-shot learning involves identifying solutions that were not encountered during training [55]. 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 [56]. 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

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

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. LLMs can analyze logs, couple them with other data, and suggest operational actions to achieve goals. The challenge lies in learning the mapping between log prefixes and desired outcomes in the latent space constructed by GPT algorithms. One promising approach is to integrate LLMs with diffusion models, which are capable of learning conditional probability distributions in a latent space [57]. While caution is required due to training time constraints with current diffusion models designed for 3D representations, they can help probabilistically map latent space data points corresponding to known process activities to points encoding the process sequence in BPM contexts [58].

E. DATA ENCODING

ML algorithms are trained on collections of examples, each of which is encoded as a vector in a multidimensional feature space. Appropriate encoding methods can reduce the complexity of the samples and the space or time complexity of the model [59]. In PM, capturing the relationships between different process dimensions is critical because event logs contain information from several complementary dimensions, such as event data, execution traces, resource usage, and cycle time.

Each event in PM can be described as a multidimensional object whose value to process execution lies in its interdependence with other events, resources, and time constraints. Capturing constraints due to alternative, optional, or mandatory dependencies between events is essential. Coding methods should also identify features subject to *concept drift*, and covariance control [60] is necessary to account for hidden relationships between different dimensions. Surprisingly, the PM community has devoted relatively little effort to investigating the impact of encoding methods on PM pipeline performance. Comparative studies are scarce, with only a few examples, such as [61, 62, 63], available for reference.

In practice, basic techniques such as one-hot coding [64], frequency-based coding [13], and general statistics for numerical attributes [22] are often used. Sequential order is captured using text mining techniques [65, 66], k-gram encoding [12], and array representations [67]. While these techniques capture some control flow information, they may not fully account for concurrency. To better capture dependencies between activities, PM has turned to techniques from other domains, such as text mining [65, 66] and graph embedding [68, 69]. Graph embedding methods, while outperforming other techniques, come with increased time complexity and a loss of transparency in the resulting latent space [70]. Recent trends focus on encoding control-flow information into feature spaces, representing parallelism or optionality of activities [71, 72]. Emerging approaches include multi-perspective views of traces that combine both data-flow and control-flow [73, 74]. Additionally, *inter-case* encoding, capturing relationships between different cases, has been explored [75]. However, the application of these advanced encoding techniques often remains domain-dependent.

Moreover, the encoding procedures used to map PM data into ML models are poorly documented in the PM literature. The chosen feature space is often implicitly defined, specific encoding steps are unclear, and the actual code used is not disclosed. Ablation studies, which examine the impact of removing parts of the data representation on performance, 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 $\#_{\text{a}}(e)$ 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 $\#_{\text{a}}(e) = \perp$. It follows that an event can also be viewed as a tuple of attribute-value pairs $e = (\mathcal{A}_1, ..., \mathcal{A}_m)$, 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: $\sigma(i \rightarrow n) \mapsto \langle e_i, ..., e_n \rangle$, 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 $\overline{L} = [\langle a, b, c \rangle^3, \langle a, b, a \rangle^{11}, \langle a, c, b, a \rangle^{20}]$. 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 FORMALISATION OF PM DATA ENCODING

Despite the variety of encoding methods discussed in Section II, we argue that available approaches fail to capture key process-level information such as the interplay between cases, or between activity execution and availability of resources. Most of the encoding methods in use today focus on the *control-flow*, according to an *inter-case* view. Methods focusing on the *intra-case* view have been proposed but are rarely applied [76]. Similarly, proposals for encoding the *data-flow* [77] are available in the literature. The research paper of [78] presents a comprehensive survey and benchmarking of 27 encoding methods, highlighting their expressivity, scalability, correlation, and domain agnosticism. Another recent trend is stressing the need of capturing constraints connected to concurrency [71, 72]. In this section, we discuss in detail how PM data is encoded to suit ML models' training procedures. 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 $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)\}$ where inputs and outputs are paired. We can then define this training set as an example matrix $\mathbf{X} := [\mathbf{x}_1, ..., \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ and a label matrix $\mathbf{Y} := [\mathbf{y}_1, ..., \mathbf{y}_n]^\top \in \mathbb{R}^{n \times d'}$, 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.

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 $\Gamma : \Sigma \rightarrow \mathbf{X}^{n \times d}$ 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, $\pi : \Sigma \rightarrow \Sigma_\alpha$ 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 $\rho : \Sigma_\alpha \rightarrow D$ 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 $\eta : \Sigma_\alpha \rightarrow \mathbf{X}_\alpha^{n \times d}$, 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 $\nu : \mathbf{X}_\alpha^{n \times d} \rightarrow \mathbf{X}^{n \times d}$ to clarify the components of the two matrices are valuated differently.

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 $\nu : \mathbf{X}_\alpha^{n \times d} \rightarrow \{0, 1\}^{n \times d}$, 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 [11] 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

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%

TABLE I. *Managing Road Traffic Fines* Event Log

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.

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ \dots \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \begin{bmatrix} a & b & c \\ a & b & c \\ a & b & c \\ [a,a] & b & \perp \\ [a,a] & b & \perp \\ \dots & & \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 0 \\ 2 & 1 & 0 \\ \dots & & \end{bmatrix} \quad (1)$$

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-fow* 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 refers to the real-live event log of road traffic fines [41]. The events captured in the event log include creating a fine notice, recording the penalty amount, verifying if the payment is received, registering an appeal to the prefecture, and others. The reader interested in more details is referred to [79]. As illustrated in Table I, 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 that have 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.

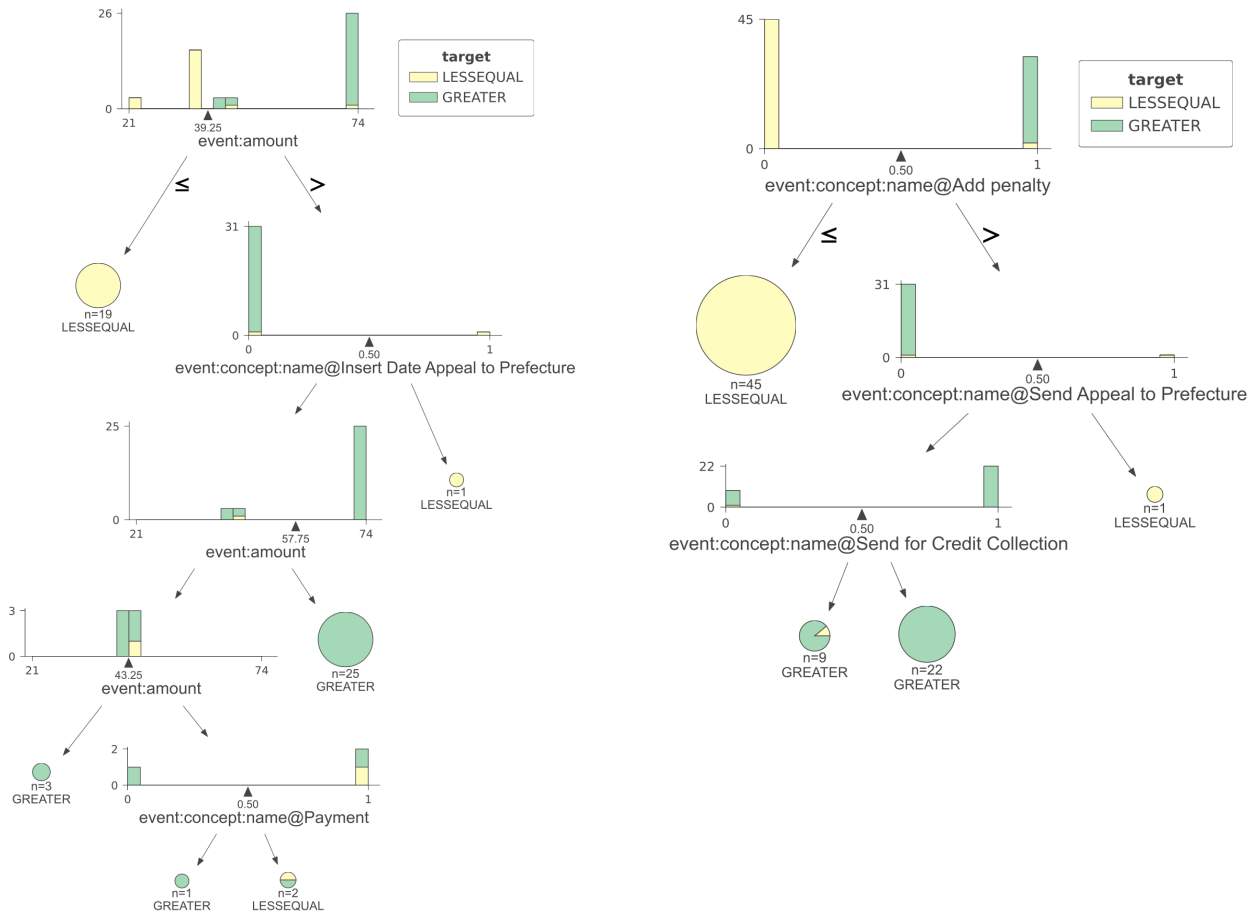
Let us now try to develop predictive analytics on this event log. For example, we could ask ourselves why certain cases exhibit a duration that is significantly longer than others. To study the problem, we are interested in searching for patterns correlated to long duration. Using encoding, we can represent the cases in the event log as vectors composed of categorical data, such as the executed activities, and of numerical data such as the number of penalties and the trace duration[2]. A decision tree can then be used to highlight the factors influencing case duration. We express it as a simple binary problem: being below or above a threshold of 200 days. Figure 2 illustrates the results we obtain. Figure 2a presents a decision tree conforming to the case distribution observed in the event log. The entire set of cases in L is encoded in $\mathcal{X}$. As a consequence, the most frequent variants take the lion's share of the examples used to train the decision tree. Figure 2b presents the decision tree obtained by balancing the case distribution among variants, oversampling those variants with low occurrence. This is, for example, achieved by creating $\mathcal{X}$ taking an equal number of occurrences to the traces in L.

Because the split points of the tree are chosen to best separate examples into two groups with minimum mixing, the cases with low occurrence tend to be ignored. Indeed, 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 categorical features only. This is due to the fact that the variants not associated with a penalty amount were quite rare, and by increasing their representation for balancing the data set we prevented the algorithm to use the penalty amount as a discrimination feature.

It is important to note that, in general, we cannot say if proactive balancing is better than using data as they are, and even which is the correct balancing factor to be applied. The strategy to be preferred strongly depends on our goal. If we want to analyse an event log in order to identify procedures that can be automated and learn the decision rule to be used, our interest is in the frequent behaviour. The real distribution of the event log, or even a distribution pruned from rare examples [31], must address the learning procedure we adopt. If our goal is anomaly detection [82] or root cause analysis [83] rare examples have to be represented.

Our next example is related to the need of capturing concurrency (Section II). While cases included in an event log are described as sequences of activities, the behaviour they describe should be interpreted differently based on the model that generated them. To capture control-flow behaviour, one needs to encode the dependency relationships in event logs. By executing the Heuristic Miner algorithm [80] on the "Road Traffic Fines" [41] event log, we observe 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" [81] will reveal the concurrent execution of the `Blood test`,

[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



(a) Unbalanced event log

(b) Balanced event log

FIGURE 2. Two *decision trees* generated from our sample 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 categorical features only. The decision tree is an example of an algorithm significantly affected by uncritical training using the case distribution of event logs.

`X-ray scan`, and `Physical test` activities. All these activities are required to complete the diagnostic stage, except for `X-ray scan`, which may be skipped, but the order of execution is not relevant. Thanks to process models, PM techniques do consider concurrency. Two sequences $\langle a, b, c\rangle$ and $\langle a, c, b\rangle$ can have the same conformance to the model if the model describes b and c as concurrent activities, while the conformance value will be different if b and c are in sequence or relate to alternative paths. Unfortunately, most ML models view event logs merely as sequential data. When cases get encoded into a vector space, the inference the ML model can produce is based on the distance in this space. The distance between $\langle a, b, c\rangle$, and $\langle a, c, b\rangle$ is accounted in the same way in the vector space, and we cannot differentiate between the sequences based on the reference process model. This limitation impairs capturing concurrent behaviour that is not detected by simply matching the two sequences. In terms of our example, an ML procedure could effectively predict the lead time of a case knowing that the `Payment` activity was executed. Training an ML algorithm to predict the conformance to the diagnostic protocol of a delivered treatment is more complex, and will require a higher amount of training data, as the ML model needs to incorporate examples on the equivalence of the different orders of execution of the `Blood test`, `X-ray scan`, and `Physical test` activities. Encoding this equivalence in vector space spaces, for example, defining suitable pictograms to feed a CNN, is still an open challenge.

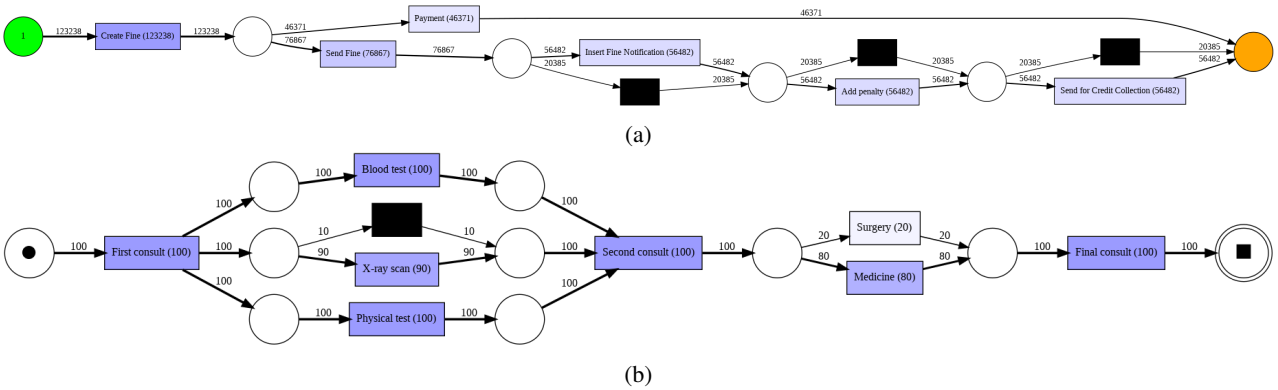


FIGURE 3. (a) The Heuristic Miner Algorithm [80] was used to discover a model from the "Road Traffic Fines" [41] event log. The discovered model specifies alternative routes that can be followed to complete the process. In particular, executing `Payment` or `Send Fine` implies the following alternative paths. (b) The Heuristic Miner Algorithm [80] is used to discover a model from the "Artificial Patient Treatment" [81] 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.

VI. TOWARD AN INTEGRATED METHODOLOGY

Guided by the above considerations about encoding, we will now outline the strategy to be used to properly integrate PM and ML. In the previous sections, we argued that when PM tasks are mapped to ML tasks, PM-specific assumptions should drive the construction of training functions and hyper-parameters selection. Simple ML classification and regression algorithms model the data by a single Gaussian grounded on mean and co-variance. On the other hand, kernel methods like Gaussian Processes and Support Vector Machines, have opened the possibility of flexible models that are practical to work with, but require non-trivial hyper-parameter tuning to fit behavioural data[84].

Figure 4 provides a synoptic view of mapping PM tasks to ML ones. As an example of non-trivial mapping, let us consider the non-linear relationship between data samples and the expected outcomes addressed by robust ML algorithms with adjusted hyper-parameters. At this point, linear projections as PCA are not effective as t-SNE visualisation [85] to obtain insights from the data. Other challenges with moderate difficulty are related to label availability and imbalanced scenarios [82]. In this case, semi-supervised ML techniques and generative models can tackle the label issue, as well as sampling or synthesising methods are the second ones. Problems related to data quality, in which the difficulty is to build an approximation to have a proper data distribution accentuated, can be solved by enlarging the training data and by a proper tuning of the ML algorithm. Alternatively, the training process can be enriched using generative models [86]. To handle the difficulties outlined in Section I, when using non-pictorial traces representation "process-friendly" GANs can be considered, like Sequence GANs (SGANs), in which the adversarial samples are designed from discrete sequences, like events. The application of GANs is not limited to data augmentation, as it can be used also for improving data quality for process model generalisation [86]. Preliminary results are available on using GAN-generated data to improve predictive tasks (e.g., lead time of incomplete cases) under an adversarial framework [87].

Coming from non-stationary process behaviour, sampling methods are a promising way to reduce the impact of non-stationary distributions of event log data [88]. After bringing the data to at least a near-stationary behaviour, the business process can naturally change its pattern over time, leading to a burdensome problem called concept drift [20, 54, 89, 90]. In dealing with this problem, a significant part of the PM community has focused on detecting and managing its onset. Regardless of the success of these attempts, we still consider this problem an open issue, since the event data stream is modelled as a complete trace stream, known from the start to the end activity. In reality, the drift onset occurs at an arbitrary position of the event stream, well before the endpoint is reached and the rest of the trace is known. Some researchers are addressing this information deficiency by using statistical adaptations based on the Hoeffding Bounds [91]. In principle, it is possible to rely on statistical assumptions about the confidence interval of the data to make a decision on the drift onset. In other words, it is possible to create ML models and perform predictions and monitoring supported by an approximated conjecture about the future, obtained from the available event log data. The use of "stateful" ML models with memory, in particular, Deep Learners based on the LSTM architecture, could enable handling drifts. However, this kind of challenge demands experienced ML practitioners and a robust computational structure.

A. HYPER-PARAMETER TUNING

Once a class of ML models has been chosen, hyper-parameter tuning must be performed to instantiate the ML model that delivers the desired accuracy (and possibly some required non-functional properties, like explainability). Searching the

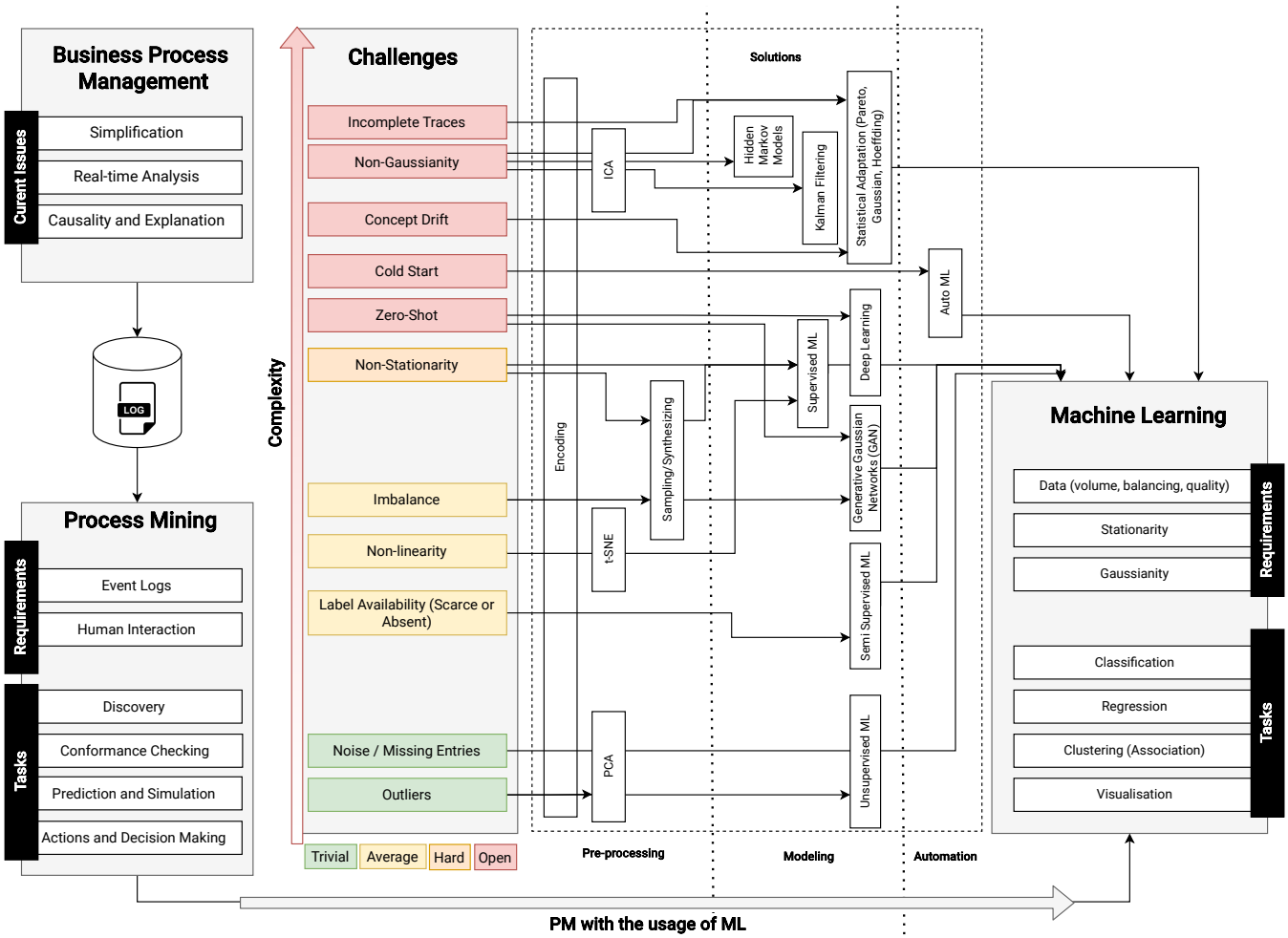


FIGURE 4. From task to task, an overview of PM and ML relationship

model space by trial and error can be burdensome. Automated Machine Learning (AutoML) is a reasonable alternative to tackle these problems grounded on sharing previous knowledge for similar tasks. AutoML can help to handle the classification problems called Zero-shot [92, 93] or Cold-start [94], for which little context information (and even the complete list of classes) may not be available at the start of the training, by taking advantage of meta-features and information on similar models, akin to how human experts start an old-fashioned search for desirable models driven by their experience on related tasks [95]. Some PM research works based on AutoML discusses how to find a suitable PM pipeline by recommending steps [14, 96, 97]. For example, [96] proposed a solution to suggest the encoding method, since the higher number of methods might lead to a tricky selection. Furthermore, there are encoding methods able to fit particular data. It is remarkable that traditional process mining tasks can be leveraged when matched with intelligent decision support approaches.

B. FINAL RECOMMENDATIONS

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

1) RECOMMENDATION 1: Choose data representation carefully

When working with PM data structures, it is crucial to carefully translate them into a metric feature space that can be manipulated by ML algorithms. Additionally, it is important to preserve context information, such as control-flow, *concurrency*, or *inter-case* constraints, which are essential for process analysis. The choice of encoding techniques should align with problem-specific goals and constraints.

2) RECOMMENDATION 2: Fit the data distributions

PM often deals with non-Gaussian, non-stationary distributions. To achieve optimal performance in production, it is advisable to estimate the data distribution instead of relying on the best Gaussian mix approximation. Building training sets interactively poses a significant challenge in PM. Leveraging ML approaches such as AutoML and Active Learning can help reduce the manual burden and improve the process.

3) RECOMMENDATION 3: Consider zero-shot learning

When training machine learning (ML) models, the full set of possible outcomes (the co-domain of f) may be only partially known. In domains such as process optimization, the costs of certain sequences may not be available during the training of a regression model. Assessing the completeness of the available information is essential when formulating the problem to ensure the quality of the model inference. Zero-shot learning, e.g. using LLMs, can support this goal. However, the correct adaptation of their response to PM tasks is still an open field.

4) RECOMMENDATION 4: Ensure minimum ML quality at an early stage via constraints

As the estimation of data distribution converges over time, an extended convergence period is unacceptable as it results in a high model error during training. It is possible to impose control flow constraints on ML models when they are known in advance based on domain requirements and regulations.

5) RECOMMENDATION 5: Incorporate domain knowledge

Domain knowledge plays a critical role in effective PM. Integrating domain-specific information and constraints into ML models can significantly enhance their performance and interpretability. It is important to actively involve domain experts in the feature engineering and model validation processes.

6) RECOMMENDATION 6: Evaluate model interpretability

PM tasks often require interpretable models to gain insight into process behavior and make informed decisions. It is essential to evaluate the interpretability of ML models and select algorithms that provide transparent explanations of their predictions. This is especially important when dealing with critical processes or compliance and regulatory requirements.

7) RECOMMENDATION 7: Continuously monitor and update ML models

Process environments are dynamic, and changes over time can affect the performance of ML models. Establishing a monitoring and evaluation framework enables the assessment of model performance and facilitates timely updates when needed. Continuous learning and retraining of models ensures their accuracy and relevance in evolving process scenarios.

8) RECOMMENDATION 8: Share knowledge and best practices

Promote knowledge sharing and collaboration within the PM community. Encourage the dissemination of successful case studies, research findings, and best practices to foster learning and advancement in the field. Engage in conferences, workshops, and online forums to connect with with other practitioners and researchers and stay abreast of the latest developments in PM.

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

VII. CONCLUSIONS

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. Table **??** presents a summary of the proposed recommendations, which we believe will serve as a valuable starting point for further advances in the field. 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. As such, 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, "Global process mining survey," Deloitte, Tech. Rep., 2021. [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, J. Buijs et al., "Process mining manifesto," in International conference on business process management. 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, 2023.

[4] 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," Business & Information Systems Engineering, vol. 65, no. 2, pp. 95–101, 2023.

[5] D. Chapela-Campa and M. Dumas, "From process mining to augmented process execution," Software and Systems Modeling, pp. 1–10, 2023.

[6] A. Berti, D. Schuster, and W. M. van der Aalst, "Abstractions, scenarios, and prompt definitions for process mining with llms: A case study," arXiv preprint arXiv:2307.02194, 2023.

[7] A. Berti, S. van Zelst, and D. Schuster, "Pm4py: a process mining library for python," Software Impacts, vol. 17, p. 100556, 2023.

[8] N. Tax, N. Sidorova, R. Haakma, and W. M. van der Aalst, "Event abstraction for process mining using supervised learning techniques," in Proceedings of SAI Intelligent Systems Conference. Springer, 2016, pp. 251–269.

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

[10] G. Tello, G. Gianini, R. Mizouni, and E. Damiani, "Machine learning-based framework for log-lifting in business process mining applications," in International Conference on Business Process Management. Springer, 2019, pp. 232–249.

[11] M. Song, C. W. Günther, and W. M. Van der Aalst, "Trace clustering in process mining," in International conference on business process management. Springer, 2008, pp. 109–120.

[12] R. J. C. Bose and W. M. Van der Aalst, "Context aware trace clustering: Towards improving process mining results," in proceedings of the 2009 SIAM International Conference on Data Mining. SIAM, 2009, pp. 401–412.

[13] A. Appice and D. Malerba, "A co-training strategy for multiple view clustering in process mining," IEEE transactions on services computing, vol. 9, no. 6, pp. 832–845, 2015.

[14] G. M. Tavares, S. Barbon Junior, E. Damiani, and P. Ceravolo, "Selecting optimal trace clustering pipelines with meta-learning," in Brazilian Conference on Intelligent Systems. Springer, 2022, pp. 150–164.

[15] 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 International Conference on Business Process Management. Springer, 2020, pp. 129–146.

[16] A. Senderovich, A. Shleyfman, M. Weidlich, A. Gal, and A. Mandelbaum, "To aggregate or to eliminate? optimal model simplification for improved process performance prediction," Information Systems, vol. 78, pp. 96–111, 2018.

[17] D. Chapela-Campa, M. Mucientes, and M. Lama, "Simplification of complex process models by abstracting infrequent behaviour," in International Conference on Service-Oriented Computing. Springer, 2019, pp. 415–430.

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

[19] G. M. Tavares, P. Ceravolo, V. G. T. Da Costa, E. Damiani, and S. B. Junior, "Overlapping analytic stages in online process mining," in 2019 IEEE International Conference on Services Computing (SCC). IEEE, 2019, pp. 167–175.

[20] P. Ceravolo, G. M. Tavares, S. B. Junior, and E. Damiani, "Evaluation goals for online process mining: a concept drift perspective," IEEE Transactions on Services Computing, 2020.

[21] 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: Springer International Publishing, 2019, pp. 348–361.

[22] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "Predictive process mining meets computer vision," in International Conference on Business Process Management. Springer, 2020, pp. 176–192.

[23] A. E. Márquez-Chamorro, M. Resinas, and A. Ruiz-Cortés, "Predictive monitoring of business processes: a survey," IEEE Transactions on Services Computing, vol. 11, no. 6, pp. 962–977, 2017.

[24] 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 2020 2nd International Conference on Process Mining (ICPM). IEEE, 2020, pp. 129–136.

[25] 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. 172 923–172 938, 2020.

[26] T. Wiel, "Process mining using integer linear programming," 2010.

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

[28] W. Van der Aalst and E. Damiani, "Processes meet big data: Connecting data science with process science," IEEE Transactions on Services Computing, vol. 8, no. 6, pp. 810–819, 2015.

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

[30] 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 BPM (Demos), 2017.

[31] W. M. van der Aalst, "On the pareto principle in process mining, task mining, and robotic process automation." in DATA, 2020, pp. 5–12.

[32] B. Hilprecht and C. Binnig, "One model to rule them all: towards zero-shot learning for databases," arXiv

preprint arXiv:2105.00642, 2021.

[33] G. MARQUES TAVARES et al., "Meta learning in process mining: Toward a systematic approach to design data analytics pipelines with event logs," 2023.

[34] E. Rama-Maneiro, J. Vidal, and M. Lama, "Deep learning for predictive business process monitoring: Review and benchmark," IEEE Transactions on Services Computing, pp. 1–1, 2021.

[35] M. F. Sani, S. J. van Zelst, and W. M. van der Aalst, "Applying sequence mining for outlier detection in process mining," in OTM Confederated International Conferences" On the Move to Meaningful Internet Systems". Springer, 2018, pp. 98–116.

[36] M. F. Sani, S. van Zelst, and W. M. van der Aalst, "Repairing outlier behaviour in event logs using contextual behaviour," Enterprise Modelling and Information Systems Architectures (EMISAJ), vol. 14, pp. 5–1, 2019.

[37] 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. 48 756–48 764, 2018.

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

[39] F. Fox, V. R. Aggarwal, H. Whelton, and O. Johnson, "A data quality framework for process mining of electronic health record data," in 2018 IEEE International Conference on Healthcare Informatics (ICHI). IEEE, 2018, pp. 12–21.

[40] A. Berti, "Statistical sampling in process mining discovery," in The 9th international conference on information, process, and knowledge management, 2017, pp. 41–43.

[41] M. De Leoni and F. Mannhardt, "Road traffic fine management process," 2015. [Online]. Available: http://dx.doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5

[42] J. Buijs, "Receipt phase of an environmental permit application process," 2014. [Online]. Available: http://dx.doi.org/10.4121/uuid:a07386a5-7be3-4367-9535-70bc9e77dbe6

[43] B. van Dongen, "Bpi challenge 2015 municipality 1," 2015. [Online]. Available: http://dx.doi.org/10.4121/uuid:a0addfda-2044-4541-a450-fdcc9fe16d17

[44] T.-W. Lee, "Independent component analysis," in Independent component analysis. Springer, 1998, pp. 27–66.

[45] R. M. Sakia, "The box-cox transformation technique: a review," Journal of the Royal Statistical Society: Series D (The Statistician), vol. 41, no. 2, pp. 169–178, 1992.

[46] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient online evaluation of big data stream classifiers," in Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, 2015, pp. 59–68.

[47] M. Roccetti, G. Delnevo, L. Casini, and S. Mirri, "An alternative approach to dimension reduction for pareto distributed data: a case study," Journal of big Data, vol. 8, no. 1, pp. 1–23, 2021.

[48] 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. Springer, 2021, pp. 3–17.

[49] C. Di 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 Business Process Management: 15th International Conference, BPM 2017, Barcelona, Spain, September 10–15, 2017, Proceedings 15. Springer, 2017, pp. 252–268.

[50] K. Thapa, Z. M. Abdullah Al, 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, 2020.

[51] K. Thapa, Z. AI, Y. Sung-Hyun et al., "Adapted long short-term memory (lstm) for concurrent human activity recognition." Computers, Materials & Continua, vol. 69, no. 2, 2021.

[52] D. A. Neu, J. Lahann, and P. Fettke, "A systematic literature review on state-of-the-art deep learning methods for process prediction," Artificial Intelligence Review, pp. 1–27, 2022.

[53] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," Information Fusion, vol. 37, pp. 132–156, 2017.

[54] L. Baier, J. Reimold, and N. Kühl, "Handling concept drift for predictions in business process mining," in 2020 IEEE 22nd Conference on Business Informatics (CBI), vol. 1. IEEE, 2020, pp. 76–83.

[55] M. Käppel, S. Schönig, and S. Jablonski, "Leveraging small sample learning for business process management," Information and Software Technology, vol. 132, p. 106472, 2021.

[56] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," Advances in neural information processing systems, vol. 35, pp. 22 199–22 213, 2022.

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

[58] C. Sun, J. Han, W. Deng, X. Wang, Z. Qin, and S. Gould, "3d-gpt: Procedural 3d modeling with large language models," arXiv preprint arXiv:2310.12945, 2023.

[59] S. Barbon Junior, P. Ceravolo, E. Damiani, and G. Marques Tavares, "Evaluating trace encoding methods in process mining," in International Symposium: From Data to Models and Back. Springer, 2020, pp. 174–

189.

[60] J. Wang, R. K. Wong, and X. Zhang, "Low-rank covariance function estimation for multidimensional functional data," Journal of the American Statistical Association, vol. 117, no. 538, pp. 809–822, 2022.

[61] S. Barbon Jr, P. Ceravolo, R. S. Oyamada, and G. M. Tavares, "Trace encoding in process mining: a survey and benchmarking," arXiv preprint arXiv:2301.02167, 2023.

[62] I. Teinemaa, M. Dumas, M. L. Rosa, and F. M. Maggi, "Outcome-oriented predictive process monitoring: review and benchmark," ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 13, no. 2, pp. 1–57, 2019.

[63] P. D. Koninck, S. vanden Broucke, and J. D. Weerdt, "act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes," in Business Process Management (BPM), ser. Lecture Notes in Computer Science, M. Weske, M. Montali, I. Weber, and J. vom Brocke, Eds., vol. 11080. Springer, 2018, pp. 305–321.

[64] N. Tax, I. Verenich, M. L. Rosa, and M. Dumas, "Predictive business process monitoring with LSTM neural networks," in Conference on Advanced Information Systems Engineering (CAiSE), ser. Lecture Notes in Computer Science, E. Dubois and K. Pohl, Eds., vol. 10253. Springer, 2017, pp. 477–492.

[65] S. M. Weiss, N. Indurkhya, and T. Zhang, Fundamentals of Predictive Text Mining, Second Edition, ser. Texts in Computer Science. Springer, 2015.

[66] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, ser. ICML'14. JMLR.org, 2014, p. II–1188–II–1196.

[67] P. Ceravolo, E. Damiani, M. Torabi, and S. Barbon, "Toward a new generation of log pre-processing methods for process mining," in International Conference on Business Process Management. Springer, 2017, pp. 55–70.

[68] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in International Conference on Knowledge Discovery and Data Mining (SIGKDD), ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 855–864.

[69] B. Perozzi, V. Kulkarni, H. Chen, and S. Skiena, "Don't walk, skip! online learning of multi-scale network embeddings," in International Conference on Advances in Social Networks Analysis and Mining (ASONAM), ser. ASONAM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 258–265.

[70] 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. Springer, 2020, pp. 296–308.

[71] 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), S. Chiusano, T. Cerquitelli, R. Wrembel, K. Nørvåg, B. Catania, G. Vargas-Solar, and E. Zumpano, Eds., vol. 1652. Springer, 2022, pp. 119–129.

[72] M. Vazifehdoostirani, L. Genga, and R. Dijkman, "Encoding high-level control-flow construct information for process outcome prediction," in 2022 4th International Conference on Process Mining (ICPM). IEEE, 2022, pp. 48–55.

[73] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "A multi-view deep learning approach for predictive business process monitoring," IEEE Transactions on Services Computing, 2021.

[74] A. Guzzo, M. Joaristi, A. Rullo, and E. Serra, "A multi-perspective approach for the analysis of complex business processes behavior," Expert Systems with Applications, vol. 177, p. 114934, 2021.

[75] J. Kim, M. Comuzzi, M. Dumas, F. M. Maggi, and I. Teinemaa, "Encoding resource experience for predictive process monitoring," Decision Support Systems, vol. 153, p. 113669, 2022.

[76] 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 (BPM), ser. Lecture Notes in Computer Science, J. Carmona, G. Engels, and A. Kumar, Eds., vol. 10445. Springer, 2017, pp. 306–323.

[77] M. de Leoni and W. M. P. van der Aalst, "Data-aware process mining: discovering decisions in processes using alignments," in Symposium on Applied Computing (SAC), S. Y. Shin and J. C. Maldonado, Eds. ACM, 2013, pp. 1454–1461.

[78] G. M. Tavares, R. S. Oyamada, S. B. Junior, and P. Ceravolo, "Trace encoding in process mining: A survey and benchmarking," Engineering Applications of Artificial Intelligence, vol. 126, p. 107028, 2023.

[79] 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: Springer International Publishing, 2016, pp. 377–392.

[80] A. Weijters, W. M. van Der Aalst, and A. A. De Medeiros, "Process mining with the heuristics miner-algorithm," Technische Universiteit Eindhoven, Tech. Rep. WP, vol. 166, no. July 2017, pp. 1–34, 2006.

[81] "Process mining in healthcare tutorial," 2020. [Online]. Available: https://gitlab.com/healthcare2/process-mining-tutorial

[82] 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 2020 2nd International

Conference on Process Mining (ICPM). IEEE, 2020, pp. 161–168.

[83] M. S. Qafari and W. van der Aalst, "Root cause analysis in process mining using structural equation models," in International Conference on Business Process Management. Springer, 2020, pp. 155–167.

[84] A. Melkumyan and F. Ramos, "Multi-kernel gaussian processes," in Twenty-second international joint conference on artificial intelligence, 2011.

[85] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." Journal of machine learning research, vol. 9, no. 11, 2008.

[86] J. Theis and H. Darabi, "Adversarial system variant approximation to quantify process model generalization," IEEE Access, vol. 8, pp. 194 410–194 427, 2020.

[87] 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 International Conference on Business Process Management. Springer, 2020, pp. 237–256.

[88] W. C. Cheung, D. Simchi-Levi, and R. Zhu, "Learning to optimize under non-stationarity," in The 22nd International Conference on Artificial Intelligence and Statistics. PMLR, 2019, pp. 1079–1087.

[89] R. J. C. Bose, W. M. van der Aalst, I. Žliobaitė, and M. Pechenizkiy, "Handling concept drift in process mining," in International Conference on Advanced Information Systems Engineering. Springer, 2011, pp. 391–405.

[90] J. Carmona and R. Gavalda, "Online techniques for dealing with concept drift in process mining," in International Symposium on Intelligent Data Analysis. Springer, 2012, pp. 90–102.

[91] P. Domingos and G. Hulten, "Mining high-speed data streams," in Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, 2000, pp. 71–80.

[92] W. Wang, V. W. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 10, no. 2, pp. 1–37, 2019.

[93] Z. Ji, G. Dai, and Y. Yu, "Multi-modality adversarial auto-encoder for zero-shot learning," IEEE Access, vol. 8, pp. 9287–9295, 2019.

[94] H. Chemingui, I. Gam, R. Mazo, C. Salinesi, and H. B. Ghezala, "Product line configuration meets process mining," Procedia Computer Science, vol. 164, pp. 199–210, 2019.

[95] R. L. Hu, C. Xiong, and R. Socher, "Correction networks: Meta-learning for zero-shot learning," 2019. [Online]. Available: https://openreview.net/forum?id=r1xurn0cKQ

[96] G. M. Tavares and S. B. Junior, "Process mining encoding via meta-learning for an enhanced anomaly detection," in European Conference on Advances in Databases and Information Systems. Springer, 2021, pp. 157–168.

[97] G. Marques Tavares and S. Barbon Junior, "Matching business process behavior with encoding techniques via meta-learning: An anomaly detection study." Computer Science & Information Systems, vol. 20, no. 3, 2023.

PAOLO CERAVOLO is currently an associate professor with the Dipartimento di Informatica, Università degli Studi di Milano, Italy. His research interests include data representation and integration, business process monitoring, empirical software engineering. On these topics, he has published several scientific papers. 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 in the Department of Engineering and Architecture at the 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 in the Computer Science Department at the State University of Londrina (UEL), Brazil. He earned his BSc degree in Computer Science in 2005, followed by an MSc degree in Computational Physics from the University of São Paulo in 2007. In 2008, he obtained a degree in Computational Engineering, and in 2011, he completed his Ph.D. degree at IFSC/USP, also in Computational Physics. His research interests encompass Computer Vision, Pattern Recognition, and Machine Learning, with a current emphasis on Meta-Learning, Stream Mining, and Process Mining.

ERNESTO DAMIANI is a full professor at the Università degli Studi di Milano, 20133, Milan, Italy, and 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. Damiani received his honorary doctorate degree for "his contribution to Big Data teaching and research" from Institute National des Sciences Appliquées de Lyon, France. Contact him at ernesto.damiani@ku.ac.ae.

ERNESTO DAMIANI is currently a Full Professor with RWTH Aachen University, leading the Process and Data Science (PADS) Group. He is also the Chief Scientist at Celonis, part-time affiliated with the Fraunhofer FIT, and a member of the Board of Governors of Tilburg University. He holds unpaid professorship positions with the Queensland University of Technology, since 2003, and Technische Universiteit Eindhoven (TU/e). He is a Distinguished Fellow of Fondazione Bruno Kessler (FBK), Trento, the Deputy CEO of the Internet of Production (IoP) Cluster of Excellence, and the Co-Director of the RWTH Center for Artificial Intelligence.