Discriminative Rule Learning for Outcome-Guided Process Model Discovery

Ali Norouzifar(✉) and Wil van der Aalst

RWTH University, Aachen, Germany
{ali.norouzifar,wvdaalst}@pads.rwth-aachen.de

Abstract. Event logs extracted from information systems offer a rich foundation for understanding and improving business processes. In many real-world applications, it is possible to distinguish between desirable and undesirable process executions, where desirable traces reflect efficient or compliant behavior, and undesirable ones may involve inefficiencies, rule violations, delays, or resource waste. This distinction presents an opportunity to guide process discovery in a more outcome-aware manner. Discovering a single process model without considering outcomes can yield representations poorly suited for conformance checking and performance analysis, as they fail to capture critical behavioral differences. Moreover, prioritizing one behavior over the other may obscure structural distinctions vital for understanding process outcomes. By learning interpretable discriminative rules over control-flow features, we group traces with similar desirability profiles and apply process discovery separately within each group. This results in focused and interpretable models that reveal the drivers of both desirable and undesirable executions. The approach is implemented as a publicly available tool and it is evaluated on multiple real-life event logs, demonstrating its effectiveness in isolating and visualizing critical process patterns.

Keywords: Process Mining · Process Discovery · Discriminative Process Modeling

1 Introduction

Event logs extracted from information systems provide a valuable foundation for analyzing and improving business processes. Process discovery aims to automatically derive process models from event logs, enabling further analysis such as conformance checking and performance optimization. In many real-world applications, process executions can be distinguished as desirable or undesirable. While desirable traces reflect efficient and compliant behavior, undesirable ones may involve inefficiencies, violations, delays, or resource waste. This distinction presents an opportunity: desirable traces illustrate desirable behavior, whereas undesirable ones reveal what should be avoided.

A common strategy in such contexts is to discover a process model that supports desirable traces while excluding undesirable ones [12]. However, a single global model often obscures the differences between the desirable and undesirable behavior of the

C. Cappiello et al. (Eds.): CoopIS 2025, LNCS 15535, pp. 15–32, 2026.
https://doi.org/10.1007/978-3-032-15538-2_2

processes. It lacks the capacity to reveal how different execution paths contribute to process outcomes or what specific control-flow characteristics distinguish desirable behavior from undesirable behavior.

Table 1. Trace variants and frequencies in L^+ and L^-.

Trace Variant	Freq in L^+	Freq in L^-
$\langle p, a, l \rangle$	100	0
$\langle p, l, a \rangle$	100	0
$\langle a, p, l \rangle$	50	50
$\langle a, l, p \rangle$	50	50
$\langle l, a, p \rangle$	0	100
$\langle l, p, a \rangle$	0	100

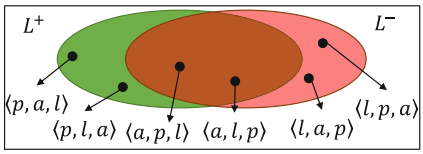


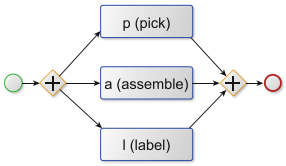
Fig. 1. Distribution of trace variants across desirable L^+ and undesirable L^- event logs.

In this paper, we propose a method to bridge this gap by identifying key control-flow patterns that effectively distinguish between desirable and undesirable traces. To this end, we introduce a methodology for selecting a subset of strong discriminative patterns, ensuring that only the most representative pattern is retained among those satisfied by similar trace subsets. Based on the selected patterns, we group traces and apply process discovery independently within each group. This leads to focused and interpretable process models that more accurately reflect the behavioral characteristics of distinct trace subsets. Our approach not only highlights the control-flow structures associated with trace desirability but also uncovers the underlying drivers of both desirable and undesirable process behaviors.

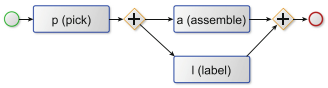
Consider an order-handling process that involves three distinct sequential activities: (p) **p**ick items from shelves. (a) **a**ssemble package by preparing for shipment. (l) Print shipping **l**abel. Ideally, orders follow a logical workflow such as $\langle p, a, l \rangle$ or $\langle p, l, a \rangle$. An analysis of historical data reveals two distinct sets of cases: a desirable event log L^+ representing workflows with an on-time delivery, and an undesirable event log L^- capturing inefficient or problematic workflows. The frequency of trace variants in each event log is shown in Table 1. Figure 1 positions the traces based on their belonging to desirable or undesirable behavior of the process.

Although L^+ and L^- share overlapping trace variants, their differing frequency distributions reflect distinct process behaviors. Discovering a process model that accounts for all observed traces without considering trace desirability can result in the model shown in Fig. 2a. Even if the model is discovered using only L^+, it may produce the same generalized process model, especially if sequences such as $\langle l, a, p \rangle$ and $\langle l, p, a \rangle$ are considered plausible but unobserved (i.e., allowed by generalization). As a result, the discovered model permits all possible orderings of the activities and achieves perfect fitness with respect to both the desirable and undesirable event logs, while failing to capture their behavioral differences.

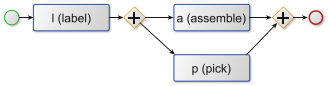
We leverage the predictive power of supervised learning models to identify discriminative control-flow characteristics that distinguish desirable from undesirable traces.



(a) Discovered process model for L^+ or $L^+ \cup L^-$.



(b) Discovered model for L^+.



(c) Discovered model for L^-.

Fig. 2. Process models for desirable and undesirable traces.

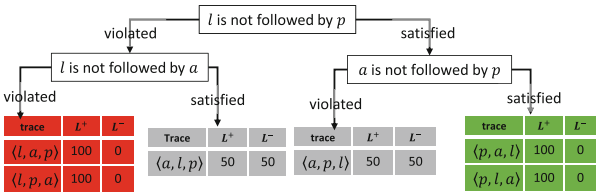


Fig. 3. Decision tree classifying traces into desirable or undesirable.

Traces are first encoded into a feature space constructed from declarative constraints, which effectively capture relationships between activities. The most informative features are then selected to explain behavioral differences, and subsequently used to extract and filter discriminative process variants that represent the core distinctions between the two classes.

A simple decision tree, shown in Fig. 3, suggests that traces are likely to be classified as desirable if l is not eventually followed by p, and a is also not eventually followed by p. Based on this logic, the process model depicted in Fig. 2b captures the control-flow of a subset of desirable traces, covering 66.7% of L^+ while excluding all traces from L^-. Unlike the generalized model in Fig. 2a, which permits all observed behavior including undesirable variants, this model emphasizes the representation of preferred behavior while explicitly avoiding undesirable patterns.

Conversely, Fig. 2c presents a process model focused on undesirable traces, corresponding to the red-labeled subgroup of traces in Fig. 3. This model covers 66.7% of L^- while excluding all traces from L^+, effectively capturing behavior specific to undesirable executions. This example demonstrates how discriminative control-flow features can guide the extraction of focused process variants that clearly separate desirable and undesirable behavior. While the other branches of the decision tree in Fig. 3 could also be used to derive additional process models, their associated subsets do not exhibit a dominant desirability class.

In real-life scenarios, effectively guiding process discovery requires a comprehensive approach that includes selecting a meaningful feature space to capture activity relationships, choosing a supervised learning model that balances predictive power with interpretability, and leveraging the model's explanations to derive event logs focused

on the distinguishing aspects. These targeted logs can then be used to discover process models that explain the underlying behavioral differences. This paper presents such an approach in detail.

2 Related Work

Declarative process discovery focuses on identifying a set of constraints that restrict the allowed behavior, ranging from no constraints permitting all behavior to highly restrictive combinations that allow no behavior at all. Some approaches, such as [5] and [4], begin with a model that permits only desirable traces and iteratively introduce declarative constraints to exclude undesirable behavior. Another example is the rejection miner proposed in [17], which selects patterns from a given input set that are satisfied by desirable traces and then minimizes this set to include patterns that reject undesirable traces. While such methods effectively yield declarative models that characterize forbidden behavior in the process, our approach differs in focus: we aim to identify discriminative patterns and subsequently discover imperative models that describe how process executions are actually carried out.

The discovery of imperative process models, such as BPMNs or Petri nets, represents another important line of research. For example, the approach introduced in [11] and [9] do not assume explicit access to undesirable traces. Instead, they generate artificial negative events to approximate undesired behavior and use these to evaluate generalization and guide the discovery process away from such behavior. A different line of work, such as [12], incorporates both desirable and undesirable event logs in a recursive process discovery framework, where each iteration seeks a process structure that effectively captures the desirable behavior while avoiding the undesirable one.

Clustering techniques without considering the desirability of cases can help to reduce the complexity of the control-flow by grouping similar behaviors. However, they do not guarantee that desirable and undesirable traces are assigned to separate clusters [19]. Other approaches, such as process variant identification [14] and concept drift detection [16], aim to identify control-flow variability across different dimensions, such as the time or performance dimensions.

Comparative process analysis offers a variety of techniques to analyze and contrast different subprocesses [2]. The field of predictive process monitoring investigates supervised learning approaches capable of effectively classifying different groups of cases [18]. While these techniques are highly effective for prediction tasks, they often lack interpretability and are not intended to serve as descriptive models of process behavior. The area of deviance mining explores the application of supervised learning methods to identify the key factors that distinguish normal from deviating cases [8]. These approaches aim to uncover the driving features behind behavioral deviations, providing valuable insights into the root causes of anomalies.

The deviance mining pipeline introduced in [8] constructs a feature space by combining declarative constraints, sequential features, and data attributes to represent traces. White-box supervised models, such as decision trees, are then trained to classify traces, and the paths from root to leaf nodes are reported as the most relevant features contributing to the classification. In contrast, our approach treats such rules as an intermediate

feature space that captures relationships among features. On top of this space, we train a sparse regression model capable of assigning importance scores to the rules, thereby identifying those that contribute most to the classification and can serve as explanations of the differences between variants. We further cluster the rules and leverage the most distinguishing ones to project event logs and discover process models that represent the variants satisfying the corresponding distinguishing rules.

In [15], a decision tree-based rule extraction approach is introduced to support study planning. However, the robustness and generalization ability of single decision trees for identifying discriminative rules can be limited. In this paper, we propose an enhanced rule extraction method based on ensemble tree models, combined with importance scoring derived from a supervised linear regression model. Our approach is inspired by the framework presented in [7], and leverages declarative feature extraction to capture control-flow characteristics. These insights are subsequently translated into interpretable process models using imperative process discovery techniques.

3 Preliminary

Given a set of elements A, the notation $\mathcal{P}(A)$ refers to the power set of A, A^* refers to all sequences we can generate over the elements of this set and A^n with $n \in \mathbb{N}$ is the universe of all vectors with n elements from set A. Considering a vector $b \in A^n$, $b[i]$ refers to the i-th element of the vector where $1 \leq i \leq n$. In the following, we introduce an event log formally.

Definition 1 (Event Log). *Let $\mathcal{C}$ denote the universe of case identifiers and $\mathcal{A}$ denote the universe of activities. A trace is a tuple $\sigma = (c, t)$, where $\pi_c(\sigma) = c \in \mathcal{C}$ is the case identifier and $\pi_t(\sigma) = t \in \mathcal{A}^*$ is the control flow of this trace. An event log $L \subseteq \mathcal{C} \times \mathcal{A}^*$ is a set of traces such that the case identifier of each trace is unique, i.e., considering any $\sigma, \sigma' \in L$, $\pi_c(\sigma) \neq \pi_c(\sigma')$. $\mathcal{L}$ is the universe of all event logs.*

For instance, $L = \{(1, \langle p, a, l\rangle), (2, \langle a, l, p\rangle), (3, \langle p, a, l\rangle), (4, \langle p, a, l\rangle), (5, \langle a, l, p\rangle), (6, \langle l, p, a\rangle)\}$ is an event log.

Definition 2 (Case Label). *Let $L \in \mathcal{L}$ be an event log. $\kappa_L : L \rightarrow \{0, 1\}$ is a trace labeling function that assigns 0 to a trace if it is undesirable and 1 if it is desirable. This function is assumed to be given as input. If the context is clear, we omit the subscript L and write $\kappa(\sigma)$ for $\sigma \in L$.*

For example, $\kappa = \{((1, \langle p, a, l\rangle), 1), ((2, \langle a, l, p\rangle), 0), ((3, \langle p, a, l\rangle), 1), ((4, \langle p, a, l\rangle), 1), ((5, \langle a, l, p\rangle), 1), ((6, \langle l, p, a\rangle), 0)\}$ is a labeling function with four desirable traces and two undesirable traces.

Declarative constraints are specific instances of predefined constraint templates that are applied to activities extracted from an event log. These constraints capture behavioral relationships and conditions in a flexible and rule-based manner. There exists a variety of well-established declarative constraint templates, including the following examples: $CoExistence(x_1, x_2)$: If activity x_1 occurs in a trace, then activity x_2 must also occur, and vice versa. $AtLeast1(x_1)$: Activity x_1 must occur at least once in the

trace. $ChainResponse(x_1, x_2)$: If activity x_1 occurs, activity x_2 must occur immediately afterward. For a detailed explanation of various declarative constraint templates, refer to [6].

Definition 3 (Declarative Constraints). *Let $\mathcal{T}$ be a non-empty set of templates, where each template is a relation $d(x_1, \ldots, x_m) \in \mathcal{T}$ defined over variables $x_1, \ldots, x_m$, with $m \in \mathbb{N}$ representing the arity of d. For a given set of activities $a_1, \ldots, a_m \in \mathcal{A}$, a declarative constraint $d(a_1, \ldots, a_m)$ is derived as a specific instantiation of the template d, binding its variables to the corresponding activities. $\mathcal{T}_A$ is the universe of declarative constraints that we can define over $A \subseteq \mathcal{A}$.*

Each trace can either satisfy a constraint, violate it, or lack sufficient information to evaluate it (vacuous satisfaction).

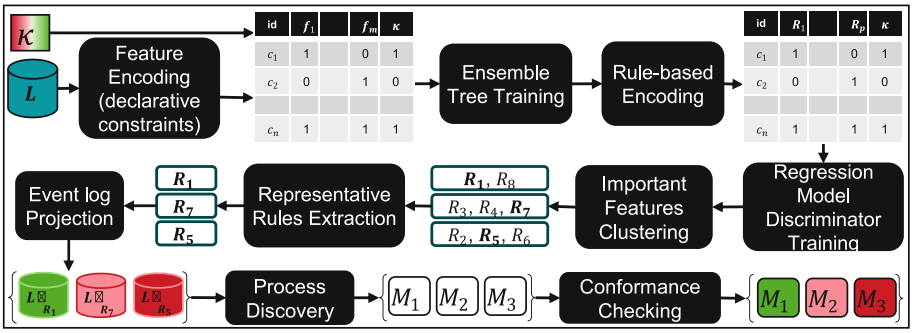


Fig. 4. An overview of our proposed framework.

Definition 4 (Declarative Constraints Evaluation). *Let $A \subseteq \mathcal{A}$ be a set of activities. Each constraint $d \in \mathcal{T}_A$ consists of an* activation condition $d_{\triangleright}$ *and a* target condition $d_{\square}$. *The activation condition determines when the constraint becomes relevant (i.e., triggered), and the target condition defines the behavior expected once the constraint is activated. Given a trace $\sigma \in \mathcal{A}^*$, the evaluation of constraint d over σ yields one of the following three outcomes:*

- *Violation: $\sigma \nvDash d$ if $d_{\triangleright}$ is satisfied in σ but $d_{\square}$ is not.*
- *Satisfaction: $\sigma \vDash d$ if both $d_{\triangleright}$ and $d_{\square}$ are satisfied in σ.*
- *Vacuous Satisfaction: $\sigma \| d$ if $d_{\triangleright}$ is not satisfied in σ.*

Considering $\mathcal{S} = \{satisfied, violated, vac\text{-}satisfied\}$ as the set of possible evaluation outcomes, the evaluation function $eval_L : L \times \mathcal{T}_{act(L)} \rightarrow \mathcal{S}$ is defined by

$$eval_L(\sigma, d) = \begin{cases} violated, & \text{if } \sigma \nvDash d, \\ satisfied, & \text{if } \sigma \vDash d, \\ vac\text{-}satisfied, & \text{if } \sigma \| d. \end{cases}$$

For example, considering the trace σ with $\pi_t(\sigma) = \langle p, l \rangle$, the constraint $CoExistence(a, p)$ is violated, since a does not occur, while p does. The constraint $ChainResonse(a, p)$ cannot be evaluated, as activity a does not appear in the trace (vacuous satisfaction). The constraint $AtLeast1(p)$ is satisfied.

4 Discriminative Process Variants Discovery

Figure 4 illustrates the proposed framework. The process starts by encoding traces with declarative constraints, which provide interpretable control-flow features. Ensemble tree-based models are then trained to capture complex interactions among these features. From the trained models, decision rules are extracted to construct an enriched feature space. A sparse logistic regression model is subsequently trained, and the most important features are clustered based on their similarity in representing traces within the feature space. For each cluster, the most representative rule is selected and used to filter traces, enabling the discovery of focused process variant models. Finally, these models are evaluated using dedicated metrics to ensure they effectively highlight the behavioral contrasts between L^+ and L^-.

4.1 Feature Encoding

The algorithm begins by discovering all declarative constraints that are satisfied by at least one trace in the log. These constraints are then pruned using the subsumption hierarchy: when multiple constraints have a subsumption relationship and yield identical evaluations across the event log, we retain only the most restrictive constraint [6]. Consider the function $declare : \mathcal{L} \rightarrow \mathcal{T}_{\mathcal{A}}$ as a function extracting such constraints. The resulting set of constraints is used to encode the event log into a structured feature space.

Definition 5 (Feature Space). *Let $L \in \mathcal{L}$ be an event log and $declare(L) \subseteq \mathcal{T}_{act(L)}$ be a set of declarative constraints extracted from it. We define the feature space as $\mathcal{F}_L = declare(L) \times \mathcal{S}$. An encoding function $encode_L : L \rightarrow \{0,1\}^{|\mathcal{F}_L|}$ maps each trace $\sigma \in L$ to a binary feature vector. For every feature $f_i = (d, s) \in \mathcal{F}_L$ where $1 \leq i \leq |\mathcal{F}_L|$, the corresponding encoding is defined as*

$$encode_L(\sigma)[i] = \begin{cases} 1, & \text{if } eval(\sigma, d) = s, \\ 0, & \text{otherwise.} \end{cases}$$

4.2 Ensemble Tree-Based Feature Extraction

To enhance the feature space, we further leverage tree ensemble models such as random forests and gradient boosting. In a *random forest*, each decision tree is trained on a bootstrap sample of L, and at each node, a random subset of features is considered for splitting. This introduces randomness that reduces variance and improves generalization. *Gradient boosting* constructs trees sequentially, where each tree is trained to correct the residual errors of the ensemble constructed so far, thereby reducing bias.

Definition 6 (Rule Extraction). *A rule $R \in \mathcal{P}(\mathcal{F}_L)$ is defined as a set of features. Let Ψ denote an ensemble of decision trees, which may be obtained from either a random forest or a gradient boosting model. For each decision tree $\psi \in \Psi$, $\mathcal{R}_\psi$ denotes the set of all rules extracted from ψ, where every path from the root to a leaf node in a decision tree corresponds to a rule. $\mathcal{R}_\Psi = \bigcup_{\psi \in \Psi} \mathcal{R}_\psi$ is the set of all rules extracted from an ensemble tree.*

Considering the decision tree represented in Fig. 3 as a tree from and ensemble of trees, $\{(NotSuccession(l,p), violated), (NotSuccession(l,a), violated)\}$ is one of the four rules we can extract from this decision tree.

Definition 7 (Rule-based Encoding). *Let Ψ be an ensemble tree and $\mathcal{R}_\Psi$ be the set of all rules extracted from it. The function $\phi_L{:}L \to \{0,1\}^{|\mathcal{R}_\Psi|}$ is a mapping from the traces to the rule-based encoding such that considering $R_i \in \mathcal{R}_\Psi$, $\phi_L(\sigma)[i] = \bigwedge_{f_j \in R_i} encode(\sigma)[j]$, where $1 \leq i \leq |\mathcal{R}_\Psi|$ and $1 \leq j \leq |\mathcal{F}_L|$.*

4.3 Regression Model

This new feature space encapsulates the ensemble-derived features and is used to augment our original feature set. Our methodology combines the strengths of declarative constraint encoding, ensemble tree-based feature extraction, and sparsity-inducing logistic regression to produce interpretable and efficient process models that distinguish between desirable and undesirable behaviors.

Definition 8 (Regression Classifier). *Let $L \in \mathcal{L}$ be an event log and $L' \subseteq L$ be a subset of traces selected for training of the regression model. Considering $\sigma \in L'$ as a trace from this event log, a logistic regression classifier with L1 regularization is defined as:*

$$cls(\sigma) = \begin{cases} 1, & \text{if } g\left(w^\top \phi(\sigma) + b\right) \geq 0.5, \\ 0, & \text{otherwise}, \end{cases}$$

where $w \in \mathbb{R}^{|\mathcal{R}_\Psi|}$ is the learned weight vector, and $b \in \mathbb{R}$ is the bias term and $g(z) = \frac{1}{1+\exp(-z)}$. w and b are obtained by minimizing the following regularized logistic loss:

$$\min_{w,b} \frac{1}{|L'|} \sum_{i=1}^{|L'|} \log\left(1 + \exp\left(-y_i\left(w^\top \phi(\sigma_i) + b\right)\right)\right) + \lambda \|w\|_1,$$

where $\|w\|_1 = \sum_{j=1}^{|\mathcal{R}_\Psi|} |w_j|$, $\sigma_i \in L'$ are training traces, $y_i = \kappa_L(\sigma_i)$ are the corresponding labels, and $\lambda > 0$ is the regularization strength controlling the sparsity of w.

We consider the weights learned by the logistic regression model as the importance of each rule $R_j \in \mathcal{R}_\Psi$, denoted as $importance(R_j) = w[j]$ where $1 \leq j \leq |\mathcal{R}_\Psi|$.

4.4 Hierarchical Clustering and Rule Selection

Although the important rules extracted from the regression model may overlap in the traces they cover, we aim to identify a representative and diverse subset of them. To this end, we apply hierarchical clustering to group rules that refer to similar sets of traces. As a dissimilarity measure, we use the *Jaccard distance* between binary feature vectors, allowing us to cluster and select distinctive rules more effectively.

Definition 9 (Jaccard Distance between Rules). *Let $L \in \mathcal{L}$ be an event log and $R_i, R_j \in \mathcal{R}_\Psi$ be two rules extracted from the ensemble of trees where $1 \leq i, j \leq |\mathcal{R}_\Psi|$. We define the Jaccard distance between rules R_i and R_j as:*

$$Jac(R_i, R_j) = 1 - \frac{\sum_{\sigma \in L} \big(\phi(\sigma)[i] \wedge \phi(\sigma)[j]\big)}{\sum_{\sigma \in L} \big(\phi(\sigma)[i] \vee \phi(\sigma)[j]\big)},$$

After computing the pairwise Jaccard distances between all rule pairs, we apply agglomerative hierarchical clustering, iteratively merging clusters of rules that exhibit the minimal average inter-cluster distance

$$dist(G, G') = \frac{1}{|G| \cdot |G'|} \sum_{R \in G} \sum_{R' \in G'} Jac(R, R'),$$

where G and G' are two clusters of rules.

Once clusters of similar rules are obtained, we proceed by selecting a representative rule for each cluster. Specifically, for each cluster $G \subseteq \mathcal{R}_\Psi$, we choose the rule with the highest importance as the representative rule:

$$R^*_G = \text{argmax}_{R_j \in G} |importance(R_j)|,$$

where $1 \leq j \leq |\mathcal{R}_\Psi|$. Subsequently, we filter the original set of traces to retain only those that satisfy the chosen representative rule R^*_G, yielding the subset of traces $L|_G = \{\sigma \in L \mid \phi(\sigma)[j] = 1 \text{ where } R_j = R^*_G\}$.

5 Evaluation Metrics

Given a set of K rule clusters, we focus on each cluster G_k where $1 \leq k \leq K$ and the corresponding filtered event log $L|_{G_k}$, which contains traces satisfying the representative discriminative rule of that cluster. For each $L|_{G_k}$, we independently apply process discovery to derive a process model that captures the behavior specific to the associated traces. This results in interpretable, cluster-specific models that not only describe the observed behavior but also facilitate distinguishing between desirable and undesirable cases.

To assess the representativeness of the models, we employ well-established conformance checking techniques [3] including *trace fitness* ($t\text{-}fit$), i.e., the percentage of traces that perfectly fit the model. *Alignment fitness* ($a\text{-}fit$), i.e., the weighted average of alignment-based fitness values across traces. *Precision* (prc), i.e., an escaping-edges-based measure that quantifies the amount of behavior allowed by the model but not observed in the event log.

Furthermore, we evaluate the discriminative power of the models using metrics introduced in [12], which are defined in terms of trace and alignment fitness.

Definition 10 (Discriminative Evaluation Metrics). *Let $L \in \mathcal{L}$ be an event log and κ_L a labeling function that partitions L into desirable and undesirable traces: $L^+ = \{\sigma \in L \mid \kappa_L(\sigma) = 1\}$ and $L^- = \{\sigma \in L \mid \kappa_L(\sigma) = 0\}$. Let $\mathcal{M}$ be the universe of Petri net*

models and $M \in \mathcal{M}$ denote a Petri net model. We define the complement of the fitness measures as $\overline{a\text{-}fit}(L, M) = 1 - a\text{-}fit(L, M)$ and $\overline{t\text{-}fit}(L, M) = 1 - t\text{-}fit(L, M)$.

Alignment-based accuracy (a-acc), trace-based accuracy (t-acc), alignment-based F1-score (a-F1), and trace-based F1-score (t-F1) are defined as follows:

- $a\text{-}acc(L^+, L^-, M) = a\text{-}fit(L^+, M) - a\text{-}fit(L^-, M)$
- $t\text{-}acc(L^+, L^-, M) = t\text{-}fit(L^+, M) - t\text{-}fit(L^-, M)$
- $a\text{-}F1(L^+, L^-, M) = \frac{2 \times a\text{-}fit(L^+, M) \times \overline{a\text{-}fit}(L^-, M)}{a\text{-}fit(L^+, M) + \overline{a\text{-}fit}(L^-, M)}$
- $t\text{-}F1(L^+, L^-, M) = \frac{2 \times t\text{-}fit(L^+, M) \times \overline{t\text{-}fit}(L^-, M)}{t\text{-}fit(L^+, M) + \overline{t\text{-}fit}(L^-, M)}$

The metrics $a\text{-}acc(L^+, L^-, M)$ and $t\text{-}acc(L^+, L^-, M)$ range from -1 to 1. A value closer to 1 indicates that the alignment or trace fitness is substantially higher with respect to the desirable event log. The metrics $a\text{-}F1(L^+, L^-, M)$ and $t\text{-}F1(L^+, L^-, M)$ range from 0 to 1, where a value of 1 denotes that the model balances well between representing desirable event log and avoiding the undesirable event log. The accuracy metrics primarily emphasize the difference in fitness values between the groups, whereas the F1 measures focus on the trade-off between increasing fitness for the desirable log and reducing it for the undesirable log. For example, if $a\text{-}fit(L^+, M) = 1$ and $a\text{-}fit(L^-, M) = 1$, then $a\text{-}acc(L^+, L^-, M) = 0$, indicating a mid-range value. However, $a\text{-}F1(L^+, L^-, M) = 0$, which is the minimum value, as the perfect fitness with respect to L^- means that one of the main objectives that is avoiding the undesirable behavior is sacrificed.

Table 2. The event logs used in the experiments.

Event Log	Desirable/Undesirable Event Logs	N. Traces
BPIC12	L^+: duration 4 to 28 days	3719
	L^-: duration more than 28 days	1433
BPIC17	L^+: duration more than 28 days	20282
	L^-: duration less than 28 days	11227
Hospital Billing	L^+: duration more than 1 day	74806
	L^-: duration less than 1 day	25194

6 Evaluation

The proposed framework has been fully implemented and is publicly available[1]. We evaluated the framework using several real-life event logs. This section presents an overview of the event logs used, the experimental setup, key findings, and a discussion of the framework's limitations.

[1] https://github.com/aliNorouzifar/discriminative_process_discovery.

6.1 Event Logs

Since the choice of L^+ and L^- depends on the specific application context, we adopted the method proposed in [14] to construct these labeled logs for process discovery. This method ranks traces based on a selected performance indicator and identifies significant control-flow changes using a two-sided sliding window in combination with the earth mover's distance. We selected case duration as the performance dimension and used the identified change points to define the desirable and undesirable event logs. The resulting trace groups are summarized in Table 2.

6.2 Experimental Setup

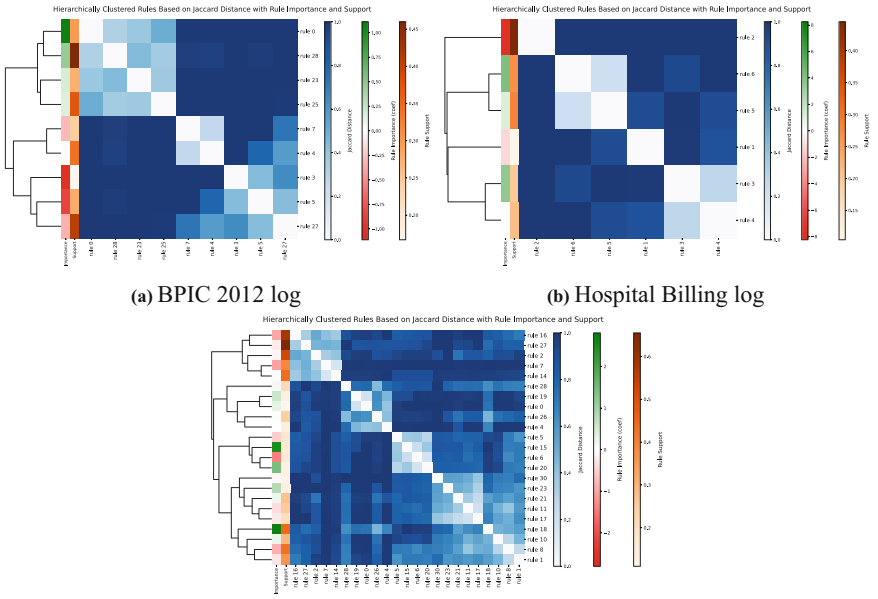
For each event log, we used 70% of the traces for feature extraction, training the ensemble models and logistic regression classifier, and reserved the remaining 30% for testing. Since class imbalance can negatively impact the learning process, we applied undersampling in training to ensure equal representation of desirable and undesirable cases. We evaluated different parameter settings for the supervised learning models using 5-fold cross-validation on the training data to determine the optimal configuration. It includes different random forest and gradient boosting settings for the rule generation and regression model parameters.

We applied agglomerative hierarchical clustering and determined the appropriate number of clusters by visually inspecting the resulting dendrograms. To avoid relying on the representational bias of a single discovery technique, we experimented with multiple process discovery algorithms:

- Rule-guided Inductive Miner (IMr) with a support threshold of $sup = 0.2$ [13]. The parameter settings for the discovery of declarative rules is set to $support = 0.005$ and $confidence = 0.95$.
- Inductive Miner with infrequency filtering (IMf) with a frequency threshold of $f = 0.2$ [10].
- Split Miner (SM) with parameters $\eta = 0.4$ and $\epsilon = 0.1$ [1].

6.3 Analysis of the Results

Figure 5 represents heatmaps of the Jaccard distances between rules extracted for each event log. The rows and columns correspond to rules obtained from regression models after initial feature generation using declarative constraints and ensemble tree-based rule extraction. The intensity of the blue color reflects the magnitude of the Jaccard distance, i.e., $Jac(R, R')$ where R is the rule corresponding to the row and R' is the rule corresponding to the column, so that lighter shades indicate that the corresponding rules are satisfied by similar groups of traces. In addition to the distance information, the heatmap includes a column on the left displaying the support for each rule in orange, and another column representing the importance magnitude using a color scale ranging from green to red with darker green denoting high positive coefficients and darker red indicating high negative coefficients. The dendrogram on the left, derived from hierarchical clustering based on the Jaccard distance, helps to determine an appropriate number of clusters for achieving well-separable groups of traces.



(a) BPIC 2012 log **(b)** Hospital Billing log

(c) BPIC 2017 log

Fig. 5. Jaccard distance heatmaps between extracted rules for each event log. Lighter blue indicates greater similarity. Rule support (orange) and importance (green-red) are shown on the left. Dendrograms guide the identification of trace clusters. (Color figure online)

Based on the dendrograms represented in Fig. 5, we choose the number of clusters 3 for the BPIC2012, 4 for the hospital billing, and 4 for the BPIC2017 log. In Table 3, we report the accuracy of the trained machine learning models for each event log ($ML\text{-}acc$), that is the ratio of the correctly classified traces (true positives and true negatives) to the number of traces. For every extracted cluster, the most important rule is shown in the table, along with its alignment accuracy, trace accuracy, importance magnitude, and support values in both the desirable and undesirable event logs. The reported alignment and trace accuracy values are calculated considering the models discovered using the IMr discovery technique.

Figure 6 presents a comparative overview of various evaluation metrics computed for different trace groups using models discovered by three process discovery techniques. Specifically, Figs. 6a, b, and c illustrate the results for models obtained using the IMr, IMf, and SM techniques, respectively. Each subfigure visualizes a distinct evaluation metric calculated across the three event logs used in the study. Trace groups, represented by different colors, include both the clusters identified in Fig. 5 and described in Table 3, as well as the baseline groups composed solely of desirable traces and undesirable traces.

BPIC 2012. As shown in Fig. 6, the models discovered considering the representative rule of cluster 1 achieve higher $a\text{-}acc$ and $a\text{-}F1$ scores compared to the baseline model derived from the desirable event log in both the IMr and IMf experiments. The $t\text{-}acc$

Table 3. Most important rule per discovered cluster with alignment accuracy, trace accuracy, coefficient, and support in L^+/L^-. Accuracy corresponds to the classifier trained for each event log; discovery and evaluation use the IMr technique.

Log	**ML-acc**	**Cluster**	**Rule**				
			a-acc	**t-acc**	**Coef**	**sup**(L^+)	**sup**(L^-)
BPIC12	0.74	1 (rule 0)	$(NotSuccession(\text{O\_Cre}, \text{O\_Sel}), satisfied)$=1∧ $(NotCoExistence(\text{A\_Act}, \text{A\_Dec}), satisfied)$=1				
			0.19	0.43	1.11	0.50	0.07
		2 (rule 7)	$(AlternateSuccession(\text{A\_Fin}, \text{O\_Can}), satisfied)$=1∧ $(NotCoExistence(\text{A\_Pre}, \text{A\_Can}), violated)$=1				
			0.00	-0.12	-0.31	0.19	0.32
		3 (rule 3)	$(AlternateSuccession(\text{A\_Sub}, \text{O\_SentB}), satisfied)$=0∧ $(AlternateSuccession(\text{A\_Acc}, \text{A\_Dec}), vac-satisfied)$=1				
			-0.14	-0.29	-1.07	0.02	0.31
HB	0.95	1 (rule 6)	$(NotCoExistence(\text{CHD}, \text{COO}), violated)$=1∧ $(AlternateSuccession(\text{FIN}, \text{JP}), violated)$=1				
			0.59	0.45	4.13	0.54	0.00
		2 (rule 3)	$(NotCoExistence(\text{FIN}, \text{JP}), vac-satisfied)$=0∧ $(AlternateSuccession(\text{CHD}, \text{BILLED}), violated)$=1				
			0.57	0.77	3.64	0.40	0.00
		3 (rule 1)	$(ChainSuccession(\text{RLS}, \text{STAT}), violated)$=0∧ $(End(\text{NEW}), violated)$=1				
			-0.45	-0.89	-1.26	0.11	0.11
		4 (rule 2)	$(End(\text{NEW}), satisfied)$=1				
			-0.62	-0.89	-8.24	0.00	0.89
BPIC17	0.83	1 (rule 7)	$(ChainSuccession(\text{A\_Val}, \text{A\_Den}), vac-satisfied)$=1				
			-0.17	-0.55	-1.05	0.07	0.71
		2 (rule 19)	$(CoExistence(\text{A\_Can}, \text{O\_Sent}), satisfied)$=0∧ $(ChainSuccession(\text{O\_Ret}, \text{O\_Acc}), violated)$=0				
			0.03	0.19	0.58	0.24	0.02
		3 (rule 15)	$(CoExistence(\text{A\_Val}, \text{A\_Comp}), violated)$=0∧ $(End(\text{O\_Can}), satisfied)$=1				
			-0.04	-0.52	2.51	0.19	0.14
		4 (rule 18)	$(End(\text{O\_Can}), violated)$=1				
			0.09	0.26	2.80	0.75	0.15

and t-$F1$ scores for cluster 1 may be slightly higher or lower, depending on the experiment. Considering Table 3, the most important rule in cluster 1 indicates that in the corresponding traces O_Created is not followed by O_Selected and A_Activated and A_Declined do not coexist. The illustrated model in Fig. 7c shows the discovered model

based on this event log that contains the control flow corresponding to the accepted and denied cases.

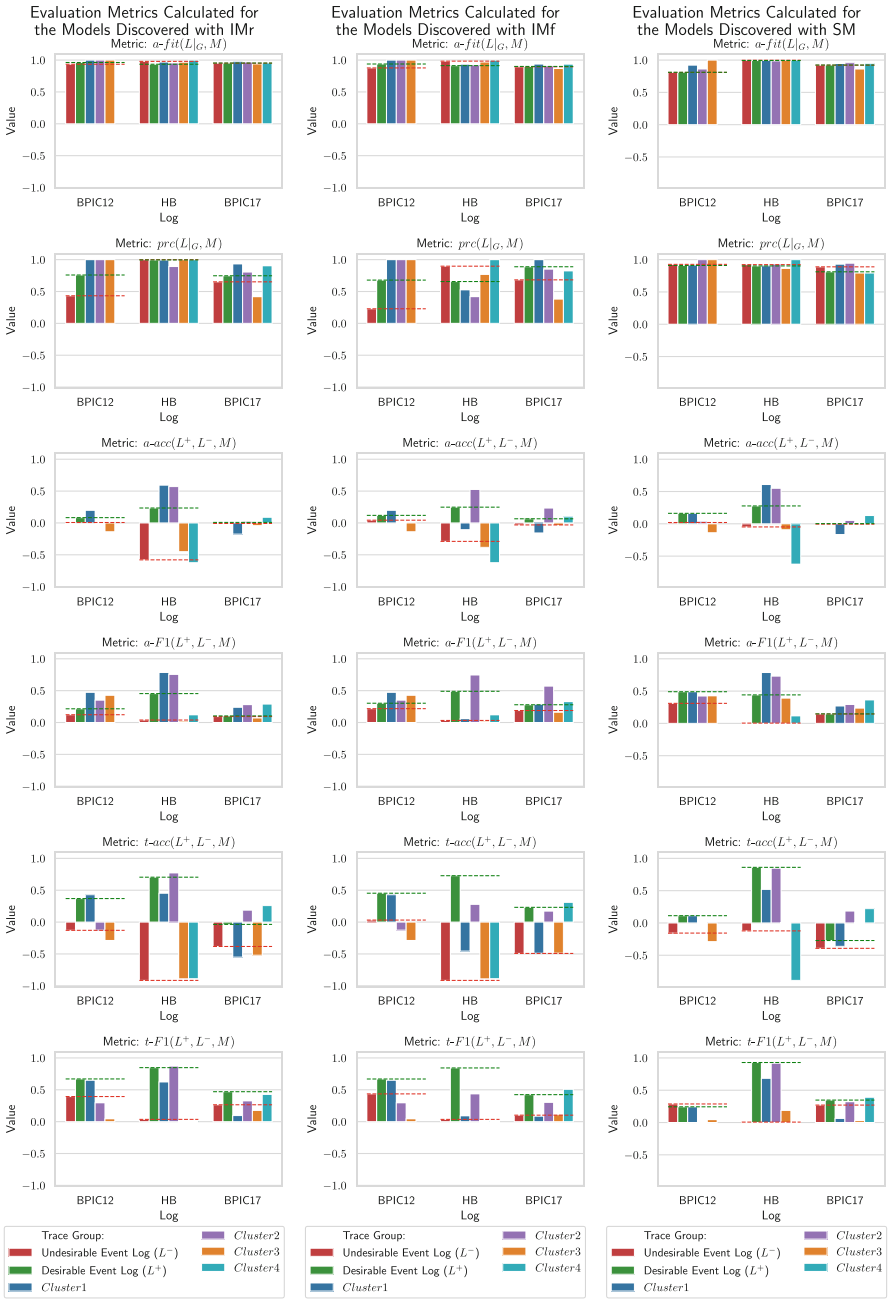
For the undesirable traces, models discovered from cluster 3 consistently outperform the baseline across all experiments by achieving lower $a\text{-}acc$, $t\text{-}acc$, and $t\text{-}F1$ values. The increase in the $a\text{-}F1$ score is due to a shift from a situation where $a\text{-}fit(L^-, M)$ is disproportionately high to a situation where a better balance between $a\text{-}fit(L^-, M)$ and $a\text{-}fit(L^+, M)$ is achieved. For example, in the IMr experiment, $a\text{-}fit(L^+, M)$ decreases from 0.94 to 0.39, and $a\text{-}fit(L^-, M)$ from 0.93 to 0.53. While $a\text{-}acc$ decreased, indicating greater distinguishability, the score $a\text{-}F1$ increases due to the more balanced representation of desirable and avoidance of undesirable traces.

Cluster 3 includes the traces in which the alternative succession of A_Accepted and A_Declined is vacuously satisfied. The model illustrated in Fig. 7e shows the discovered model for this event log that contains the control flow pertain to the canceled applications. Similarly, cluster 2 with the discovered model represented in Fig. 7d shows the control flow for a bigger group of canceled applications. Comparing these models to the discovered models based on the desirable event log, i.e., Fig. 7b and the undesirable event log, i.e., Fig. 7a show that the discovered models from the clusters more specifically represents the differences between the desirable and undesirable cases.

Hospital Billing. The evaluation metrics presented in Fig. 6 highlight that, the models discovered based on the representative rule of cluster 2 outperform those derived from the desirable event log in terms of all four metrics $a\text{-}acc$, $a\text{-}F1$, $t\text{-}acc$, and $t\text{-}F1$ across the IMr and SM experiments. Similarly, models discovered using the representative rule of cluster 1 yield higher $a\text{-}acc$ and $a\text{-}F1$ scores compared to the baseline. However, this improvement comes at the cost of slightly reduced $t\text{-}acc$ and $t\text{-}F1$ scores.

Although different process discovery techniques use the same group of traces to discover process models, the quality of the discovered models and the extent to which they generalize the observed behavior can vary significantly due to the representational bias of each discovery technique. In cluster 1, models discovered using IMr and SM demonstrate substantially higher alignment accuracy and F1-score compared to those discovered solely from the desirable event log. However, models generated using IMf tend to overgeneralize the behavior, allowing both desirable and undesirable traces to fit the model. This leads to lower scores in evaluation metrics that assess discriminative power. The low precision of IMf models further suggests that these models permit behaviors not present in the event logs. While this may indicate flexibility, it does not clarify whether the additional allowed behavior reflects acceptable generalization or undesirable drift.

Clusters 4 and 3 correspond to behaviors that are more representative of the undesirable event log. However, their performance does not consistently surpass that of the models discovered directly from the undesirable event log. The evaluation metrics across different experiments indicate that, in some cases, the models based on these clusters achieve better scores than the baseline, but this superiority is not observed uniformly across all settings.

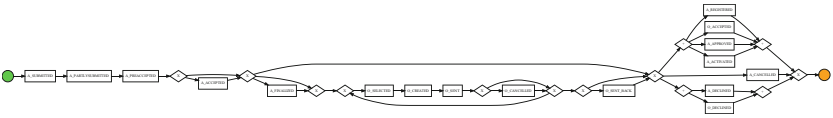


(a) Models discovered with the IMr process discovery technique.

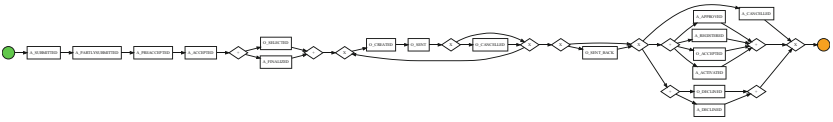
(b) Models discovered with the IMf process discovery technique.

(c) Models discovered with the SM process discovery technique.

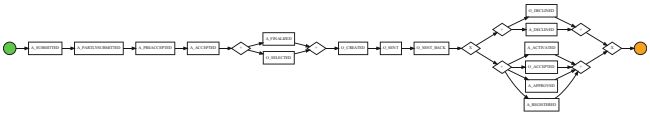
Fig. 6. Evaluation metrics for discovered models across trace groups. Bar heights show values based on IMr (left), IMf (middle), and Split Miner (right). Results are grouped by evaluation metric for each cluster and baseline (L^+ and L^-).



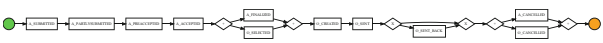
(a) Discovered model from the undesirable event log



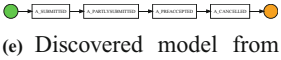
(b) Discovered model from the desirable event log



(c) Discovered model from cluster 1



(d) Discovered model from cluster 2



(e) Discovered model from cluster 3

Fig. 7. The discovered model from BPIC 12 event log using the IMr process discovery algorithm.

BPIC 2017. Based on the evaluation metrics shown in Fig. 6, the models discovered from different trace groups reveal interesting patterns. Clusters 1 and 3 are particularly effective in capturing undesirable behavior, while clusters 2 and 4 are better suited for representing desirable behavior. For example, in the IMr experiments, the model derived from cluster 1 exhibits significantly lower $a\text{-}acc$, $t\text{-}acc$, and $t\text{-}F1$ scores compared to the undesirable log, indicating a strong alignment with undesirable traces. Cluster 3 follows a similar pattern, though with less significant differences. On the other hand, the model from cluster 4 achieves significantly higher $a\text{-}acc$, $a\text{-}F1$, and $t\text{-}acc$ scores compared to the model discovered from the desirable event log. Cluster 2 shows a comparable trend, though the improvements are less substantial.

Depending on the interdependencies among features and their discriminative strength, the number of important features identified by the sparse regression model trained on desirable and undesirable event logs may vary. Features identified as important and clustered together often exhibit consistent directional influence, generally highlighting either desirable or undesirable behavior. However, this consistency does not always hold. For instance, in the BPIC 2017 event log, we observe that rule 15 and rule 6, although referring to similar groups of traces, receive coefficients with opposite signs. This discrepancy may stem from subtle behavioral differences that are particularly relevant for the classification task. Alternatively, it may be a consequence of multicollinearity among features, which warrants further investigation.

A concrete example of this can be seen in cluster 3 of Fig. 5c and Table 3. While the most important feature in this cluster has a relatively high positive coefficient, a very similar feature in the heatmap exhibits a strong negative coefficient. This contrast suggests potential multicollinearity that could destabilize the interpretation of individual

coefficients. Notably, the conformance checking results for models discovered using this cluster across different discovery techniques indicate that the resulting models better capture undesirable behavior, despite the most important rule suggesting desirability.

This observation underscores the need for caution when interpreting feature importance in isolation. We therefore recommend focusing primarily on clusters where the important features have a consistent directional interpretation, such as cluster 1 and cluster 2, which offer clearer insights into the behavioral differences between trace groups. In contrast, cluster 4 includes a rule with a strong positive coefficient that does not align closely with others in the heatmap. However, conformance checking suggests that this rule still corresponds well to traces exhibiting desirable behavior, indicating that the differences between the traces corresponding to this feature and other features within this group is contributing to the discriminative power of the models.

Based on the experimental results across scenarios with both desirable and undesirable event logs, discovering process models that consider only a single aspect is less effective in explaining the differences between the logs. In contrast, the framework proposed in this paper facilitates the identification of the aspects that drive these distinctions. The process models subsequently discovered for these aspects are better suited to represent the distinctive behaviors within each event log.

7 Conclusion

The proposed method leverages the strengths of supervised learning to identify key differences between desirable and undesirable event logs. These discriminative features enable the discovery of process models that are more focused on representing the underlying factors that drive this distinction. The results are promising and introduce a novel perspective for working with labeled event logs, supporting interpretable and outcome-aware process discovery.

Nonetheless, the approach also presents opportunities for further refinement. In particular, multicollinearity among features may lead to instability in the regression coefficients, an issue that warrants further investigation and can be mitigated through preprocessing techniques. Using discriminative patterns to first filter the event log and then discover imperative models, which are subsequently evaluated for their relevance to desirable and undesirable traces, represents an indirect approach to identifying discriminative process models. Future work could explore more direct methods for deriving such subprocess models.

References

1. Augusto, A., Conforti, R., Dumas, M., Rosa, M.L., Polyvyanyy, A.: Split miner: automated discovery of accurate and simple business process models from event logs. Knowl. Inf. Syst. **59**(2), 251–284 (2019)
2. Bolt, A., de Leoni, M., van der Aalst, W.M.P.: Process variant comparison: using event logs to detect differences in behavior and business rules. Inf. Syst. **74**, 53–66 (2018)
3. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99414-7

4. Chesani, F., et al.: Shape your process: discovering declarative business processes from positive and negative traces taking into account user preferences. In: Almeida, J.P.A., Karastoyanova, D., Guizzardi, G., Montali, M., Maggi, F.M., Fonseca, C.M. (eds.) Enterprise Design, Operations, and Computing, EDOC 2022. LNCS, vol. 13585, pp. 217–234. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-17604-3_13
5. Chesani, F., et al.: Process discovery on deviant traces and other stranger things. IEEE Trans. Knowl. Data Eng. **35**(11), 11784–11800 (2023)
6. Ciccio, C.D., Mecella, M.: On the discovery of declarative control flows for artful processes. ACM Trans. Manag. Inf. Syst. **5**(4), 24:1–24:37 (2015)
7. Cohen, W.W.: Fast effective rule induction. In: ICML, pp. 115–123. Morgan Kaufmann (1995)
8. Francescomarino, C.D., Donadello, I., Ghidini, C., Maggi, F.M., Puura, J.: Business process deviance mining with sequential and declarative patterns. Bus. Inf. Syst. Eng. **67**, 877–894 (2025)
9. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust process discovery with artificial negative events. J. Mach. Learn. Res. **10**, 1305–1340 (2009)
10. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013. LNBIP, vol. 171, pp. 66–78. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06257-0_6
11. de León, H.P., Nardelli, L., Carmona, J., vanden Broucke, S.K.L.M.: Incorporating negative information to process discovery of complex systems. Inf. Sci. **422**, 480–496 (2018)
12. Norouzifar, A., van der Aalst, W.M.P.: Discovering process models that support desired behavior and avoid undesired behavior. In: SAC, pp. 365–368. ACM (2023)
13. Norouzifar, A., Dees, M., van der Aalst, W.M.P.: Imposing rules in process discovery: an inductive mining approach. In: Araújo, J., de la Vara, J.L., Santos, M.Y., Assar, S. (eds.) Research Challenges in Information Science, RCIS 2024. LNBIP, vol. 513, pp. 220–236. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-59465-6_14
14. Norouzifar, A., Rafiei, M., Dees, M., van der Aalst, W.M.P.: Process variant analysis across continuous features: a novel framework. In: van der Aa, H., Bork, D., Schmidt, R., Sturm, A. (eds) Enterprise, Business-Process and Information Systems Modeling, BPMDS EMMSAD 2024. LNBIB, vol. 511, pp. 129–142. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-61007-3_11
15. Rafiei, M., et al.: Extracting rules from event data for study planning. In: De Smedt, J., Soffer, P. (eds) Process Mining Workshops, ICPM 2023. LNBIP, vol. 503, pp. 361–374. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-56107-8_28
16. Sato, D.M.V., Freitas, S.C.D., Barddal, J.P., Scalabrin, E.E.: A survey on concept drift in process mining. ACM Comput. Surv. **54**(9), 189:1–189:38 (2022)
17. Slaats, T., Debois, S., Back, C.O., Christfort, A.K.F.: Foundations and practice of binary process discovery. Inf. Syst. **121**, 102339 (2024)
18. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: review and benchmark. ACM Trans. Knowl. Discov. Data **13**(2), 17:1–17:57 (2019)
19. Weerdt, J.D., vanden Broucke, S.K.L.M., Vanthienen, J.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12), 2708–2720 (2013)