Prescriptive Process Monitoring based on Causal Effect Estimation

Zahra Dasht Bozorgi[1], Irene Teinemaa*[2], Marlon Dumas[3], Marcello La Rosa[1], and Artem Polyvyanyy[1]

[1]*The University of Melbourne, Parkville, VIC, 3010, Australia*
[2]*Bloomberg, 3 Queen Victoria St, EC4N 4TQ, London, United Kingdom*
[3]*University of Tartu, Narva mnt 18, 51007, Tartu, Estonia*

Abstract

Prescriptive process monitoring methods seek to control the execution of a business process by triggering interventions, at runtime, to optimize one or more performance measure(s) such as cycle time or defect rate. Examples of interventions include, for example, using a premium shipping service to reduce cycle time in an order-to-cash process, or offering better loan conditions to increase the acceptance rate in a loan origination process. Each of these interventions comes with a cost. Thus, it is important to carefully select the set of cases to which an intervention is applied. The paper proposes a prescriptive process monitoring method that incorporates causal inference techniques to estimate the causal effect of triggering an intervention on each ongoing case of a process. Based on this estimate, the method triggers interventions according to a user-defined policy, taking into account the net gain of the interventions. The method is evaluated on four real-life data sets.

Keywords: business process, prescriptive process monitoring, causal inference, heterogeneous treatment effect estimation

1 Introduction

Prescriptive process monitoring is a family of techniques that provides recommendations on which actions to take to improve a business process with respect to one or more performance metric(s). Depending on the process, there may be several *interventions* (a.k.a. *treatments*) that workers may perform to improve a certain performance metric. For example, in an application-to-approval process, giving a call to a customer to obtain missing information may speed up a case. However, such interventions come at a cost. Therefore, the prescriptive process monitoring method needs to incorporate

*This research was performed while I. Teinemaa was at Booking.com.

a decision procedure to determine which cases should be subjected to a given type of intervention, taking into account the benefits and costs of these interventions.

In this setting, this paper tackles the following problem: *Given an intervention that in general improves a process performance metric of a case, which cases should we trigger this intervention for to maximise the total net gain?* Here, the total net gain is the sum of the differences between the benefit of each intervention and its cost.

To tackle this problem, we propose a prescriptive monitoring method that triggers interventions based on the continuous observation of the process. The method relies on causal effect estimation models to estimate the effect of an intervention on a binary or real-valued process performance metric (e.g., the cycle time or the positive outcome rate). The estimated causal effect is then used to define a *policy* which maps a case prefix to a treatment assignment. An intervention is triggered if the policy determines that the case should receive it. The method incorporates an applicability testing procedure to determine if the dataset used to train the causal effect estimation model meets basic assumptions required for this purpose, as well as a procedure to determine which method to use to train the causal effect estimation model. The method also incorporates an approach to assist users in selecting a treatment policy and understanding which cases will be treated for a given policy (policy interpretation). The applicability of the proposed method is evaluated using four real-life datasets.

This paper is an extended and revised version of a conference paper [5]. With respect to the conference version, this article incorporates the following additional contributions:

- The method presented in the conference paper focuses on interventions that reduce the cycle time of a case. In this article, we extend the scope of the method to deal both with real-valued case outcomes (e.g., cycle time, cost) and with binary case outcomes (positive vs. negative case outcome).
- We extend the method with steps to evaluate the applicability of causal effect estimation methods on the input dataset and the relative performance of alternative causal effect estimation methods, prior to applying them to train a causal effect estimation model and treatment policy.
- We interpret our causal estimation models and selected policies using established methods from the explainable machine learning literature.

The next section motivates the approach via an example. Section 3 reviews related work. Section 4 introduces preliminary concepts and notations. The proposed method is described in Section 5, while the evaluation is discussed in Section 6. Section 7 draws conclusions and discusses future work directions.

2 Motivating Example

We consider a loan origination process that starts when a client submits a loan application. The submitted documents are screened. If some documents are missing, a request

for further information is sent to the customer either by email or phone. After the missing documents are received, the application is either approved or rejected. In either case, the customer is notified.

Historical data shows that customers respond faster to phone calls than to emails. Thus, one way of reducing the cycle time of the process is to call the customer about the missing documents. However, handling missing document events via email is less costly and takes less effort. Therefore, workers would ideally only call a customer if this would speed up the process considerably. Commonly, a company would leave it at the discretion of the employees to decide which customers get a phone call. This might, however, lead to a suboptimal tradeoff between the costs and the cycle time. Alternatively, one could try and define a policy for picking the cases that receive the call, making use of the historical data.

One approach to determine which customers to call (and when) is by predicting the remaining time of each case and triggering an intervention on those cases that are expected to be most delayed w.r.t. a cycle time target [34, 19, 18]. However, this approach may be ineffective. Suppose that the reason why a case is likely to have a long cycle is that the employee handling it is occupied with other cases. If so, making a phone call to obtain the missing information (the intervention) would consume more resources with little effect on cycle time. Since the phone call is an expensive intervention, we would like to reserve it for cases that benefit from it the most. So, although the above case is likely to have a long cycle time, the phone call should not be made.

A more suitable approach could be to estimate the causal effect of a phone call on the cycle time for each individual case and to direct the interventions to cases with the highest estimated effect. In the example above, the causal effect of a phone call on the cycle time would be close to zero, since this intervention does not address the actual bottleneck (the employee's time). A policy informed by this estimate would correctly suggest not making the phone call and therefore avoiding unnecessary costs. This paper pursues the causal effect estimation approach for defining intervention policies.

3 Related Work

3.1 Prescriptive Process Monitoring

Various prescriptive process monitoring methods have been proposed. In [15], the authors provide a comprehensive review of prescriptive monitoring methods. Teinemaa *et al*. [34] propose to trigger an intervention when the probability of a case leading to a negative outcome is above a threshold optimised w.r.t. a net gain function. This method is later extended in [6] to optimise the timing of firing an alarm. The method by Weinzierl *et al*. [37] uses predictive models to determine which activity, among the most likely next activities in the case, is correlated with higher values of a given KPI. Metzger *et al*. [18] use deep learning models to generate predictions about an ongoing case and feed these predictions to an online reinforcement learning technique, which triggers an intervention based on the predictions and their reliability. The above methods tackle the problem of identifying which cases need an intervention, while our approach aims to identify cases that can benefit most from a given intervention.

This problem is addressed in the conference version of this paper, where we propose a prescriptive monitoring method based on a causal estimator named orthogonal random forest (ORF) [22] to reduce the cycle time of the process. This work was extended by Shoush and Dumas [29] to address resource constraints when triggering interventions.

Resource allocation is another major focus in prescriptive monitoring. The work by Wibisono *et al*. [38] makes recommendations about which police officer should be assigned to the next task in a driving application process. Another work by Sindhgatta *et al*. [31], recommends resources to take up a task if they are predicted to finish the task within the defined time frame. While our approach is not concerned with allocating specific resources to specific tasks, it can be applied to the problem of which type of resource is assigned to a case (e.g., if there is a dedicated "experienced resource", our method can be used to identify which cases benefit the most from having this resource assigned to them).

3.2 Causality in Process Mining

In recent years, both causal discovery and inference have been gaining increasing attention in process mining applications. Hompes *et al*. [10] propose an approach to discover cause-effect relations between aggregate characteristics of a process (e.g. frequency of activity) and process performance indicators (e.g. mean cycle time). Koorn *et al*. [14] present a method to identify causal relations between a response and an effect. They use statistical tests to discover action-response-effects, where the action defines a subgroup of cases and the response is a treatment that enhances the probability of the effect. Polyvyanyy *et al*. [24] present causality mining, a systematic approach to discovering causal dependencies between events encoded in large datasets. Narendra *et al*. [20] use structural causal models to confirm potential cause-effect relations identified by analysts. Qafari *et al*. [25] use structural equation models to test for the existence of a causal relation between an attribute and an outcome. They later expand the use of SEMs in [26] for counterfactual explanation of case-level predictions. None of the above studies quantifies the effect of a treatment at the level of an individual case. Hence, they cannot be applied to prescriptive process monitoring, which is the focus of this paper.

In our previous work [4], we use action rule mining to extract candidate treatments correlated with a positive outcome. That work deals only with binary case outcomes, whereas in this paper we deal with both binary (process outcome) and numerical (cycle time) target variables. Furthermore, this work focuses on providing recommendations in an operational setting, whereas in [4], the recommendations are in the form of tactical rules.

3.3 Reinforcement Learning

Reinforcement learning (RL) is a learning paradigm to learn the effectiveness of an agent's actions through interactions with the environment [32]. An RL agent selects an action a in response to an environment s and receives a reward r for that behaviour. The goal of the agent is to maximise the reward. RL has been used for different tasks

in process mining. In [30], the authors build a business support system based on Q-Learning. Another study by [12], uses RL for resource allocation. While these methods have shown considerable success in using reinforcement learning for process mining, they require online interactions with the environment and can suffer from the cold start problem initially. Notably, RL-based methods have an advantage in dealing with non-stationarity in the data [18] and time-varying attributes. In our approach, the focus is on finding the best cases to receive the treatment. Hence, dealing with time-varying attributes is not as significant an issue as when the goal is finding the best time to treat, which is the problem addressed in [18]. Therefore, we chose causal effect modelling over RL as the backbone of our approach to take advantage of the information in the event log rather than through interactions with a live environment.

4 Preliminaries and Definitions

This section provides the background knowledge required to understand the subsequent discussions. It includes definitions from process mining and causal inference fields, focusing on the estimation of causal effects.

4.1 Event Logs and Traces

We refer to an instance of a process execution as a *case*. A case consists of a collection of *events*, where an event represents an execution of an activity. Each event has three attributes: a *case identifier* specifying which case the event belongs to, an *activity* that triggered the event, and a *timestamp* specifying when the event occurred. An event may also have further attributes, such as a resource that carried out the activity or an event type.

Definition 1 (Event, Trace, Event Log)
An *event* is a tuple $(a,c,t,(d_1,v_1),\ldots,(d_m,v_m))$, $m \in \mathbb{N}_0$, where a is an activity name (label), c is a case identifier, t is a timestamp, and $(d_1,v_1),\ldots,(d_m,v_m)$ are attribute-value pairs. A *trace* is a finite sequence $\sigma = \langle e_1,\ldots,e_n\rangle$, $n \in \mathbb{N}_0$, of events with the same case identifier in ascending order of their timestamps. An *event log*, or *log*, is a multiset of traces.

As we aim to estimate the causal effect of an intervention on the target of interest, we are interested in events that occurred before the intervention. We use the notion of a k-prefix to capture such preceding events.

Definition 2 (k-Prefix)
A *k-prefix* of a trace $\langle e_1,\ldots,e_n\rangle$, $n \in \mathbb{N}_0$, is a sequence $\langle e_1,\ldots,e_k\rangle$, $0 \leq k \leq n$.

In this paper, we train a number of machine learning models that take as input a fixed number of independent variables (herein called features). Therefore, the traces in the event log must be encoded as feature vectors.

Definition 3 (Sequence encoder)
A sequence encoder $f : S^* \to X_1 \times \cdots \times X_p$ is a function that takes a (partial) trace σ and the event log $L \subseteq S^*$ that σ belongs to and transforms σ to a feature vector X in the p-dimensional vector space $X_1 \times \cdots \times X_p$ with S^* being the universe of all possible traces over a set of events S and $X_i \subseteq \mathbb{F}$, where $\mathbb{F}$ is a set of all possible features and $X_i \cap X_j = \emptyset$, $i, j \in [1 \,..\, p]$ and $i \neq j$.

4.2 Causal Identification

The field of causal inference is concerned with determining the independent effect of a phenomenon on an outcome of interest. Two causal inference frameworks are discussed in the literature. In the *causal graphical models* framework [23], a causal graph is constructed by a domain expert. Given a causal graph, it is possible to ascertain whether a causal estimand is identifiable and if so, it can be estimated using automated methods. While this framework focuses on identifying causal effects, the *Neyman-Rubin potential outcomes* framework [27] focuses on the estimation of causal effects of interventions. Hence, in this paper, we use the *Neyman-Rubin potential outcomes* framework.

An intervention, or *treatment*, can be captured by a binary variable $T \in \{0,1\}$, where $T = 1$ indicates that the treatment is applied and $T = 0$ that it is not. Each case in a log has two potential outcomes: the outcome that would have happened if the treatment was applied (Y^1) and the outcome that would have happened under no treatment (Y^0). Then, the *conditional average treatment effect* (CATE) of a case is defined as the difference between the potential outcomes conditioned on a set of features describing the current state of a case:

Definition 4 (Conditional Average Treatment Effect)
Let X be a set of attributes that characterize a case. Then, the *conditional average treatment effect* (CATE) of the case is defined as follows:

$$CATE : \theta(x) = \mathbb{E}[Y^1 - Y^0 \,|\, X = x],$$

where $\mathbb{E}$ denotes the expected value of $Y^1 - Y^0$ and X is a feature vector of random variables derived from case prefixes as in definition 3. The details of the included features in X are explained in Section 5.1.

CATE is a causal estimand, meaning that in order to estimate it, we need to have access to both Y^0 and Y^1. Note that Y^1 and Y^0 are hypothetical quantities, not observed ones. This means that in real life, it is impossible to follow both potential realities (i.e., we cannot both apply a treatment to a case and not apply it). This is known as the "fundamental problem of causal inference" [9]. Thus, to estimate the causal estimand, we express it via statistical estimands, such as $\mathbb{E}[Y \,|\, T = 1, X]$ and $\mathbb{E}[Y \,|\, T = 0, X]$, which can be estimated from data. According to the potential outcomes framework, CATE can be expressed via statistical estimands only if the *ignorability*, *positivity*, *consistency*, and *no interference* conditions hold.

The ignorability (a.k.a. exchangeability) condition means that given the pre-treatment attributes X, treatment assignment is independent of the potential outcomes. In other words, after conditioning on X, the treatment assignment should be as good as

random, which ensures that the treated and not treated groups are exchangeable, that is:

$$Y^1, Y^0 \perp\!\!\!\perp T \,|\, X.$$

The positivity condition means that for every set of values for X, treatment assignment is not deterministic. So, every subgroup of interest has some chance of getting either treatment:

$$P(T = t \,|\, X = x) > 0, \text{ for all } t \text{ and for all } x.$$

The consistency condition states that the potential outcome under treatment $T = t$ is equal to the observed outcome if the actual treatment received is $T = t$:

$$Y = Y^t \text{ if } T = t \text{ for all } t.$$

According to no interference, the potential outcomes of one subject are not affected by the treatments received by others.

The positivity condition can be partially assessed from data, for example by modelling the propensity score. The propensity score tells us the probability of a case receiving the treatment. It can be modelled using any off-the-shelf probabilistic model. If the propensity score for all cases is high, we can verify that the positivity condition holds. However, if the propensity score for some cases is low, we cannot make a definite conclusion about the positivity condition. Thus, we need additional analysis to ensure the estimation step can deal with the violation of positivity. The consistency conditions can be verified by ensuring that there are no multiple definitions of the treatment under study, for example by domain expertise. For example, if the definition of the treatment is "skip an activity", this definition leads to a violation of the consistency assumption. Because some cases might receive the treatment "skip activity a", but another case might receive the treatment "skip activity b". These two treatments are different, but in our definition "skip an activity" are considered the same. This why a domain expert needs to ensure that the definition of treatment should refer to one specific activity and does not contain variations.

Verification of the ignorability condition is not straightforward. In observational data, such as event logs, there often exist variables that influence both the treatment assignment and the outcome. The existence of these *confounding variables* (or *controls*) creates a non-causal association between T and Y, which can invalidate the study. The best way to circumvent this problem is to conduct a randomized experiment (A/B test). As we are working with event logs, where randomization of the treatment is not ensured, we *assume* that ignorability holds to estimate CATE. Under this assumption, the observed variables X contain sufficient information needed to adjust for confounding. The adjustment can then be carried out during the estimation step. Making this assumption is commonplace in methods using the potential outcomes framework [13]. However, it is reasonable to believe that there are confounders outside of X. Therefore, in this paper, we implement a check to see how our method behaves when this assumption is violated.

In many real-life use cases, including business processes, the no interference condition can often be violated as well. For instance, skipping an activity in one case might leave the process worker available to perform activities in another case, and vice versa.

As a result, the potential outcomes of the second instance are affected by the treatment applied in the first. However, due to the work by [3, 36], who show that it is possible to make causal claims under interference, it is common to assume that a violation of the no interference condition often has only a small influence on the causal effect estimates. However, the severity of the violation of no interference depends on the chosen treatment and the outcome and care must be taken if domain experts believe that the level of interference is high.

4.3 Heterogeneity in Causal Effects

In causal estimation, the goal is to estimate two main quantities [8]: the average treatment effect (ATE) and the conditional average treatment effect (CATE). The average treatment effect provides an overall view of the treatment's impact on the outcome. For example, we might want to know what is the overall effect of decreasing interest rates on customers accepting loans. In this example, we are not interested in effect of this treatment on individual customers, rather, we are looking for an overall quantity that describes the effectiveness of the treatment.

In other studies, we might be interested in a more granular form of treatment effect. For example, instead of investigating the overall effect of increasing interest rates, we might be interested in knowing which types of customers are more impacted by the interest rate and would accept a loan because of the lower interest. In a study such as this, we assume that customers respond differently to a given treatment and we would like to estimate an individual quantity for each customer. In other words, we assume there is *heterogeneity* in the treatment effect.

4.4 Causal Estimation

Next to identification, the other core problem in causal inference is estimating CATE from observational data. A large body of works from recent years focus on using machine learning methods for CATE estimation. We henceforth refer to this method as *causal estimators*. Notably, a family of methods known as meta learners [16] combine supervised learning or regression methods as base learners. This gives them the flexibility to employ any method that fits particular data types. Here we provide a brief description of the meta-learners we use in this paper.

4.4.1 S-Learner

The S-learner, uses a single model μ to estimate the observed outcome with the treatment indicator being included as an ordinary feature:

$$\mu(x,t) = \mathbb{E}[Y|X = x, T = t].$$

Any supervised machine learning or regression method can be used to estimate μ from the entire dataset. This model is then used to estimate the outcome under treatment and under no treatment:

$$\hat{\mu}(x,1) = \mathbb{E}[Y|X = x, T = 1],$$
$$\hat{\mu}(x,0) = \mathbb{E}[Y|X = x, T = 0].$$

The treatment effect is calculated by getting the difference between our estimates of the potential outcomes:

$$\theta(x) = \hat{\mu}(x,1) - \hat{\mu}(x,0).$$

4.4.2 X-Learner

The X-learner is another meta learner that is designed to work well with imbalanced treatment groups. This method has three main stages:

1. Use any supervised learning or regression method to get estimates of the potential outcome. Note that instead of including the treatment indicator as a feature as in the S-learner, two separate models are trained for estimating μ under treatment and no treatment:

$$\mu_1(x) = \mathbb{E}[Y(1)|X = x],$$
$$\mu_0(x) = \mathbb{E}[Y(0)|X = x].$$

2. Use $\hat{\mu_0}(x)$ to impute the treatment effect for the treated cases, and $\hat{\mu_1}(x)$ for the control (not treated) cases:

$$D_{1,i}(x) = Y_i(1) - \hat{\mu_0}(x_i),$$
$$D_{0,i}(x) = \hat{\mu_1}(x_i) - Y_i(0).$$

We call the quantities above the imputed treatment effects. Next, using the imputed treatment effects as the response variable (what we are trying to estimate), we train two models using any supervised machine learning or regression method, one for the treated cases and another for the control cases:

$$\theta_1(x) = \mathbb{E}[D(1)|X = x],$$
$$\theta_0(x) = \mathbb{E}[D(0)|X = x].$$

3. In the third stage, the CATE estimate is calculated as the weighted average of the two estimates in stage two:

$$\hat{\theta}(x) = g(x)\hat{\theta}_0(x) + (1 - g(x))\hat{\theta}_1(x),$$

where $g(x)$ is a weight function. In the paper proposing X-learner, the propensity scores are used as weights.

4.4.3 Causal Forest

Another popular method is the work by Athey *et al*. [2]. which is a flexible non-parametric estimation method based on *generalized random forests*. To understand this method, we need to understand the base learners of causal forests, namely causal trees. Causal trees work very similarly to regression trees. They divide the feature space $\mathbb{X}$ recursively into a deep partition Π using a greedy algorithm. In a regular regression tree, the splits are selected to minimise the mean squared error (MSE) of the outcomes. But in a causal tree, they are selected to minimise our estimate of the expected mean squared error (EMSE) of the treatment effect. Another splitting criterion in causal trees is that all nodes after the split should contain both treated and control samples. Pruning the partitions is done using cross-validation similar to regression trees. The treatment effect of each partition Π is estimated using the samples in that partition:

$$\hat{\theta}(\Pi) = \mathbb{E}[Y(1,\Pi) - Y(0,\Pi)].$$

One important difference between Causal trees and regular regression trees is *honest splitting*. Honest splitting means that the data is divided into two sets: one for tree-building and one for estimation of the treatment effects. This ensures that the tree's estimates are unbiased within a particular sub-group [35].

Causal forest are constructed by averaging the results of many honest causal trees. In the original Causal Forest method, the trees are constructed by sub-sampling the data instead of bagging, which is more commonly used in regression trees.

4.5 Evaluating Causal Estimation Models

As mentioned before, real-world data sets such as event logs do not have the ground truth CATEs due to the fundamental problems of causal inference. This means that causal estimators cannot be evaluated by methods like train-test split or cross validation like predictive models. So, it is common practice in the causal inference literature to evaluate causal estimators with simulated data [1]. Here, we give a brief overview of different ways data can be simulated for evaluating causal models.

The simplest and most common way is to create fully synthetic data from a set of known confounders. This will give us access to the true treatment selection and outcome mechanisms, that can be used for computing propensity scores and true causal effects. The main disadvantage of this approach is that the results from fully synthetic data might not generalize to real data.

An improvement over the previous approach is to use semi-synthetic data where the features come from real data, but the treatment selection and outcome come from arbitrary stochastic functions [21]. The main advantage of this approach is that the treatment selection and outcome functions can be designed to have varying properties while keeping the feature set realistic. However, this approach still suffers from unrealistic treatment and outcome mechanisms.

The problem of unrealistic selection and outcome mechanisms can be addressed by fitting them to real data. In this approach, we assume the pre-treatment features X are a complete set of confounders. Then, we fit a generative model to model treatment selection and outcome that closely match the real mechanisms $P(T|X)$ and $P(Y|T,X)$. We can then sample from this generative model to simulate the outcome under treatment and no-treatment conditions, which will provide us data with the ground truth CATE while being statistically indistinguishable from the real data. Since our aim is to assess causal estimators under realistic conditions, we use this approach in this paper.

4.6 Realcause

Realcause is a method to generate synthetic data that is fit to real data [21]. This approach involves training models for treatment selection $P(T|X)$ and the outcome mechanism $P(Y|T,X)$. In Realcause, neural networks, specifically multi-layer perceptrons, are used to model these two mechanisms. For treatment selection, the input of the neural network is X and the output of the network is the probability of receiving

the treatment. For the outcome, the structure of the neural network follows the one proposed in TARNet [28].

First, all of the data regardless of what treatment group they belong to are used as input of the neural network to get a treatment agnostic representation of the data. The network then branches off into two sections, one for the treatment group and one for the control. These two sections are represented by two additional neural networks. Both take the treatment agnostic representation of the data as input, but one network only uses the data from the treatment group and the other the data from the control group. In Realcause, all three networks use the same architecture for simplicity. The output of this network is used to parameterise the distribution of the data. For binary variables, such as treatment or binary outcomes, the distribution is assumed as the Bernoulli distribution. The logistic sigmoid activation function is applied to the last layer of the networks to parameterise the mean parameter for Bernoulli. For real-valued variables, the output of the network is fed into a generative model called Sigmoidal Flow [11] to derive a more flexible distribution specific to the data. These distributions are sampled under both treatment and control conditions to get data with two potential outcomes $Y(1)$ and $Y(0)$.

5 Prescriptive Monitoring Approach

We aim to design a recommender system that seeks to optimise a goal in a cost-aware manner. This goal can be binary (e.g., process outcome) or continuous (e.g., cycle time). At the core of our system is a causal estimation module that estimates the effect of a given treatment on a target metric. In this method, we assume that the treatments are binary. For instance, suppose that in the loan origination process discussed in Section 2, the default way of handling missing documents is by sending emails to clients to request additional documents, while some customers get phone calls instead. In this scenario, the binary treatment options are sending emails ($T = 0$) or making phone calls ($T = 1$). We refer to cases that got a phone call as the treatment group and the cases that got an email as the control group. Making a phone call should, in general, speed up the process. However, it is not feasible to call every client because of the associated additional costs. Thus, we want to find the best policy for deciding which clients should get phone calls.

Our approach consists of four phases, as depicted in Fig. 1. In the first phase, *data pre-processing*, case prefixes are extracted from the log and encoded into feature vectors. Next, in the *applicability test and method selection phase*, the feasibility of using a causal estimator is assessed by simulating realistic artificial data that has the same data generating distribution as the real event log. The causal model is first trained and evaluated on the artificial data. If the results of the applicability test suggests that there is benefit in applying a causal estimator, the best causal estimator is selected and it is trained on the real event log. The applicability test and method selection phase is one of the main contributions of this paper. While methods for assessing the performance of causal estimators exist in the causal inference literature, our checks have been designed specifically for the prescriptive monitoring task and with process improvement as the goal. In the third phase, *training and model selection*, the chosen causal estimator is

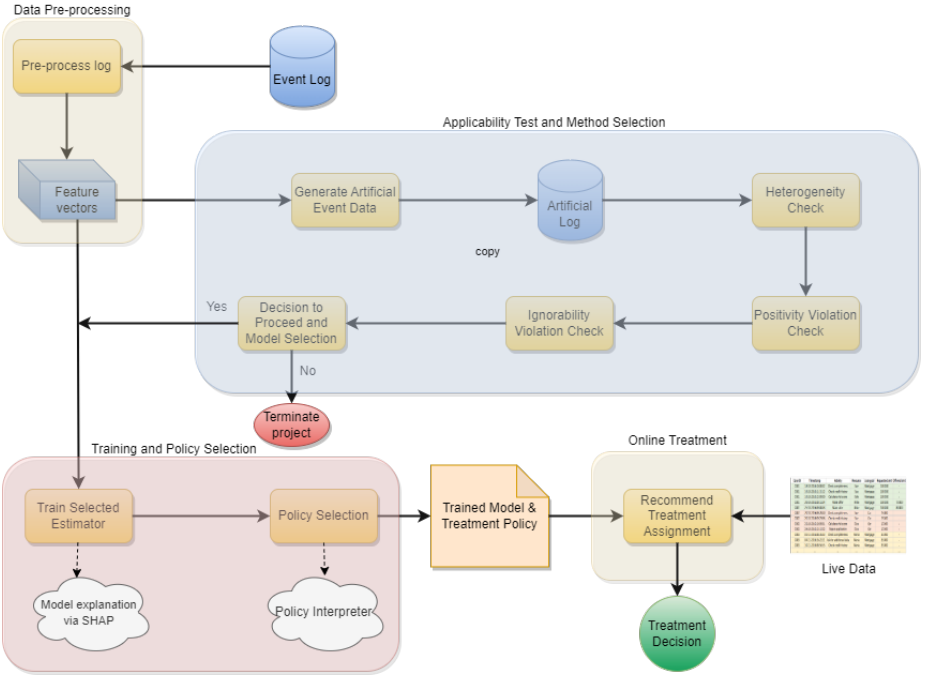


Figure 1: Overview of the proposed approach

trained on the real event log and the best treatment policy is selected. This policy is used to define a threshold for treatment decision. In the *online treatment* phase, the causal estimator and the selected policy are used to recommend whether to treat or not to treat an ongoing case based on its estimated treatment effect and the selected treatment policy.

Next, we describe the steps of the approach, namely *data pre-processing*, *applicability test & method selection*, *training and policy selection*, followed by the *online treatment*.

5.1 Data Pre-processing

This section elaborates on preparing the input data for training the causal model. To this end, we extract k-prefixes from the event log and encode their associated data to capture the heterogeneity of the treatment effects and the potential confounders.

5.1.1 Data Cleaning

In the offline phases of the approach, we train models based on completed historical cases. First, we pre-process the log to remove incomplete cases. We then repair missing attribute values. For a numeric attribute, we set missing values to the median value of that attribute. For a categorical attribute, we set missing values to the most observed value for that attribute (the mode). We also exclude cases that appear to have incorrect timestamps.

5.1.2 k-Prefix Extraction

The time when the treatment occurs in a case might be determined by a number of factors specific to a given process. It may be that, for example, a call to a customer would only be effective towards the end of the case, once a customer has had time to think about the initial loan offer. Accordingly, we select prefixes for training in a way that reflects the time points the treatments are observed in the data, as follows. First, for each case that was treated in the training set, we select the prefix ending at the event where the treatment occurred. For instance, given a trace $\sigma = \langle e_1, \ldots, e_n \rangle$, suppose that e_{k+1} indicates the treatment event where $k+1 \leq n$. In this case, the selected prefix is $\langle e_1, \ldots, e_k \rangle$. Thus, k can be different for different cases.

For cases where no treatment occurs, we replicate the same distribution of prefix lengths as that of cases that were treated. To do so, we construct the histogram of distributions of prefix lengths where a treatment occurred (using the cases where the treatment did occur). We then scale this histogram in such a way that its total mass is equal to the number of cases where a treatment did not occur. Finally, we use the scaled histogram to select the prefix lengths for the untreated cases. For example, let's consider the example where 15 cases in the training set were treated at prefix length 3, 20 cases at prefix length 4, and 10 case at prefix length 5. This leads us to a histogram [(3,15), (4, 20), (5,10)]. We then randomly draw from this histogram. The reason we select the prefix lengths in this way is that we can be confident that the treatment can happen at these points in the process, since we can observe them in the log. Also, we ensure that there is a balance in the amount of information present in both treatment and control groups. If it happens that a case is shorter than the prefix length we need to extract from it, then we do retain the full case instead of extracting a prefix of it.

5.1.3 Prefix Encoding

To apply the machine learning method for estimating treatment effects, we need to encode trace prefixes (ongoing cases) as fixed-sized feature vectors. Encoding static case attributes is straightforward since their values do not change during the execution of a case. However, traces often contain dynamic event attributes whose values change as the case unfolds. Various methods have been proposed to encode dynamic attributes, including *aggregation encoding*, *last-state encoding*, and *index-based encoding*. According to the results in the survey by Teinemaa *et al* [33], index-based encoding does not lead to higher performance, and aggregation encoding generally yields the best classifiers. So, we use aggregation encoding for the activity type and resource attributes and other event attributes. Hence, the feature vector constructed from a k-prefix contains numeric features for each activity type, resource, or event attribute A, the number of times A appears in the k-prefix. If the attribute is numeric, we use the mean as our aggregation function as we have found it to be the best performing. We use last-state encoding for temporal features. This is because some of the temporal features we use are inherently aggregates (e.g., time since case start) as will be explained in the next section. So, they do not need to be aggregated again.

If an encoded case attribute is a categorical attribute, we apply one-hot encoding to represent it as a numeric feature vector – a common practice when using machine

learning models.

5.1.4 Feature Extraction

The two essential features needed for our method are the treatment and outcome. We start by flagging the cases that include an activity indicative of the treatment. Then we create a binary feature vector where the flagged cases get the value of one and other cases get the value of zero. If the target of improvement is cycle time, we create the target vector by taking the difference between the timestamp of the last event and the timestamp of the first event. If the target is the case outcome, we either look for a single feature that already represents the outcome or we look for an activity or a group of activities that indicate the success or failure of a case. In this case, the target feature can be engineered based on this information.

Given that we seek to estimate treatment effects on process outcome or cycle time, we include temporal information in the feature vectors. Specifically, we include the month, weekday, and hour of the timestamp of the last event in the prefix, the time between the first and the last event in the prefix, and the time between the last two events in the prefix (i.e. the inactivity period prior to the most recent event).

We also include the number of active cases as a feature to act as a proxy for the current workload in the process. We do so because workload is a potential confounder, influencing both the estimation target and the treatments (during high workloads, the treatment might not be applied as often as usual and also, cases may be delayed or result in a negative outcome due to customer dissatisfaction more often, creating a spurious correlation between the treatment and the outcome). Finally, we encode the difference between the first event in the prefix (the start time of the case) and the first event in the log (the start time of the log's timeframe) since the process might behave differently at different points in time; hence, this feature can be a confounder.

5.2 Applicability Test and Method Selection

Before applying the causal estimator to the real-life dataset, its performance should be assessed. Due to the fundamental problem of causal inference, we do not observe the ground truth causal effect in the real-life event log. Recent advances in generative machine learning allow us to learn a highly accurate distribution of the data generating process. We can treat this empirical distribution as the true data generating distribution and generate data under both control and treatment conditions, yielding two outcomes under each condition. We can then calculate the treatment effect by simply taking the difference between the two generated outcomes. The result of this step is that we will have a data set that is statistically indistinguishable from the real data, but with the added benefit of having access to the treatment effects. Using the artificial data, we can answer the following question: *How good is the estimator $\hat{\theta}$ when the data generating distribution is F, where F is any arbitrary distribution?* We can use the artificially generated data to answer questions about the causal estimator we plan to use.

Therefore, using generative machine learning models, we generate an artificial log that is similar to the real-life one, but contains both potential outcomes. To do this, we use Realcause [21], a method for generating data that is both realistic and contains the

ground truth causal effect. The advantage of using Realcause over other data simulation methods is we can generate artificial data that is statistically indistinguishable from our real-life event log, and also, change some of the characteristics of the data generating process using tunable knobs. Specifically, we can use the positivity/overlap knob to test the robustness of our model to the violation of the positivity assumption.

We start by fitting the Realcause method to the pre-processed event log. We then create an artificial log by sampling from this generative flow model. In the next step, we train the chosen causal model on the artificial log.

Once we have the artificial log, we use it to train one or several causal estimators. Since there is no causal estimator that outperforms the rest on all datasets [40], we use this step to select the best causal estimation method for each event log.

Since we have the ground truth in the artificial dataset, we can compute metrics such as the mean squared error (MSE) or the mean absolute percentage error (MAPE). But, these metrics are not suited for computing the net gain from applying the model. This is because MSE and MAPE capture the deviation of the estimated treatment effect from the ground truth treatment effect. This does not directly provide us with any useful insights about how good our decisions are and how much we gain from a treatment policy. Therefore, we use the Qini metric which is easier to visualize and can be adjusted to evaluate treatment policies as we will discuss later in this section. Traditionally, *Qini curves* and the closely related *uplift curves* provide a way to evaluate causal estimators when the ground truth treatment effect is not available, which is the case in all real-world datasets [40]. However, since we do have the ground-truth in this phase of our approach, we use a modified version of the Qini curve presented below.

First, let us introduce the necessary notations. Coming back to the example of the loan origination process, let us suppose that we used a causal estimator $\hat{\theta}$ and for every customer represented by k-prefixes σ_i (with the treatment decision happening at the k-th event), we have an estimated treatment effect for each customer. Let π be the ascending ordering of the customers according to their estimated treatment effect, i.e., $\hat{\theta}^{\pi}(\sigma_i) \leq \hat{\theta}^{\pi}(\sigma_j), \forall\, i < j$, if the target is cycle time reduction. For outcome improvement, π denotes the descending ordering of the traces, i.e., $\hat{\theta}^{\pi}(\sigma_i) \geq \hat{\theta}^{\pi}(\sigma_j), \forall\, i < j$. In addition, $\pi(n)$ is used to denote the first n percent of traces from the ordering. In other words, $\pi(n)$ denotes the customers whose treatment effect is in the top n percent of the ordering.

Furthermore, let $R^{Y(1)}_{\pi(n)}$ and $R^{Y(0)}_{\pi(n)}$ be, respectively, the sums of the potential outcomes of cases in under treatment and control conditions in $\pi(n)$:

$$R^{Y(1)}_{\pi(n)} = \sum_{k=1}^{|\pi(n)|} Y_k(1),$$

$$R^{Y(0)}_{\pi(n)} = \sum_{k=1}^{|\pi(n)|} Y_k(0),$$

where $|\pi(n)|$ is the number of cases in $\pi(n)$.

We now define the Qini curve for cycle time reduction as:

$$Qini(n) = R^{Y(0)}_{\pi(n)} - R^{Y(1)}_{\pi(n)}.$$

For outcome improvement, the Qini curve is defined as:

$$Qini(n) = R^{Y(1)}_{\pi(n)} - R^{Y(0)}_{\pi(n)}.$$

$Qini(n)$ is the expected total improvement in the target variable (reduction of cycle time or the number of positive outcomes), given that the top n percent of cases selected by the causal model are treated (e.g. we make phone calls to the top n percent of cases). The Qini curve can be plotted by computing $Qini(n)$ for different values of n.

5.2.1 Heterogeneity Check

Having the artificial version of our event log, we would like to assess the applicability of causal modelling. We perform various types of analysis using the Qini curve, to assess whether or not the event log contains enough signal to perform causal analysis. We start by plotting the Qini curve for various causal estimation methods. These can be tree or forest-based methods, meta-learners, or doubly robust methods. In this paper, we report the results for X-Learner, S-Learner, and Causal Forest as we have found them to be the highest performing. We can assess the performance of these methods by comparing their area under the Qini curves. We only proceed to the next steps if the event log contains enough signal about causal effect heterogeneity. This is because the main assumption that we make for our prescriptive monitoring method to provide gain is that the treatment has varying causal effects on the target variable across different cases. If this variation is not observed, we terminate the prescriptive monitoring procedure.

5.2.2 Positivity Violation Check

The next step of the analysis is to determine the robustness of the selected methods to violations of the positivity assumption. Because event logs are observational data, this assumption, along with ignorability, are the most likely to be violated. For example, in our loan origination process, let us suppose that the phone call is only made to customers with very high credit scores. In this case, the treatment assignment is deterministic and the positivity assumption is violated. With this check, we would like to see how the causal estimator behaves when this violation occurs. To perform this check, we use the Positivity/Overlap knob in Realcause. This knob can take any value β between 0 and 1, with $\beta = 0$ corresponding to a setting where treatment assignment is fully deterministic, and $\beta = 1$ corresponding the treatment assignment observed in the real data. All other values $0 < \beta < 1$ correspond to somewhere in between. We plot the Qini curves for different values of β to see how each method responds to the violation of the positivity assumption. A method which is robust to the violation of positivity will maintain high area under the Qini curve with low values of β.

5.2.3 Ignorability Violation Check

Next, we assess sensitivity of the selected methods to the violation of the ignorability assumption. Recall that the existing causal estimation methods work under the assumption of no unobserved confounding. This means that for these methods to work well, we must assume that the observed features are sufficient to adjust for treatment selection bias. However, in practice, it is possible that there are confounding features which

influence both treatment and outcome, and are not captured by the available data. To simulate such a condition, we train the Realcause model to use all the available features in the event log, but we exclude a set of features C when training the causal estimator. Recall that Realcause models $P(T|X)$ and $P(Y|T,X)$. So, if we generate the artificial log using the complete feature set X, all features in X will participate in modelling the treatment selection and the potential outcomes. But if we have a feature set C such that $C \subset X$ and we train the causal estimator using $X - C$, then C will be a set of unobserved confounding features because C participated in the data generating process, but was not adjusted for in the causal estimation step. We then proceed to plot the Qini curves.

To choose C, we categorise the features into three main groups: case features, resource features, and temporal features. For each causal estimator, we train three additional models, each time excluding one group of features. To assess the robustness of each causal estimator to unobserved confounding, we compare four models. The first is made when the estimator is trained with all the features ($C = \emptyset$). Let us call it model $M1$. The second model is trained with C=case features (model $M2$), the third when C=resource features (model M3), and the fourth for when C=temporal features (model M4). If we see a significant drop in the area under the qini curve of either $M2$, $M3$, or $M4$ compared to $M1$, then we conclude the causal estimator is sensitive to the violation of the ignorability assumption, and great care must be taken to ensure that all confounding features are present in the real-life event log.

Finally, we select the model that performs well in terms of area under the Qini curve, and is robust to the violation of both positivity and ignorability conditions to be used in the policy selection phase on the real-life event log.

5.3 Training and Policy Selection

5.3.1 Training

This step of the offline phase takes as input the encoded k-prefixes and learns a function that, given a k-prefix $\langle \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)} \rangle$, returns a point estimate of the treatment effect θ. The best causal estimator is selected based on the results of the Applicability Test and Method Selection step.

The causal estimator requires four inputs: the target variable Y, the treatment indicator T, features capturing heterogeneity X, and the confounding variables W. Y is a vector containing the process performance indicator we are trying to optimise, for example, cycle time or outcome. We assume that a binary treatment is previously identified, which is hypothesised to increase the performance. In the loan origination example in Section 2, this is a phone call.

As the proposed approach can be applied at operational level, we would like to know the treatment effect when it is time for the process worker to decide whether to treat a case or not. Therefore, effect heterogeneity is captured via static case-level attributes and all event attributes (including the activity name and the resource), and the inter-case features available at the decision time. This means that X is a feature vector that captures all of this information about the k-prefix after the suitable encodings have been applied, as described in Section 5.1. In this study, we assume that all of the available features derived from the event log are also potential confounders, meaning

that all the features present in X are included in W. However, in general, X and W do not need to be the same. A domain expert can remove some features from W if they believe these features do not influence the treatment decision or outcome.

Armed with these definitions of Y, T, X, and W, the next step is to train the estimator. The input data is temporally split into the train and test sets. A separate test set is required for the evaluation of the trained model, and for selecting the best policy.

If the selected model is not readily explainable (such as meta-learners or forest-based methods), Shapley values can be used with causal models. Similar to predictive models, SHAP [17] can explain the heterogeneity of the estimated effects, that is, what features lead the model to produce higher or smaller effects for different subgroups of cases. In other words, it tells which features were the most important in determining treatment effect heterogeneity. If we think of different features as players contributing to the treatment effect estimation, SHAP provides scores for each player's contribution. Our approach is similar to the work by Galanti *et al* [7]. They use SHAP to provide explanations in a predictive monitoring setting, so, they compute Shapley values for each prefix σ_k, for all $k \leq n$ where n is the length of the entire trace. In this paper, however, we only compute the Shapley values for one prefix. Recall that in Section 5.1.2, we select one prefix length k where k is the number of events that occur before the treatment decision time. Using the methods proposed in Sections 5.1.3 and 5.1.4, we obtain the feature vector $X = [x_1, \ldots x_n]$ where x_i can either denote a case attributes, or an encoded event attributes, or an engineered feature (e.g., inter-case or temporal features). Unlike in the work by Galanti *et al*, the feature set in X is fixed and does not vary from prefix to prefix. This is because the way they construct feature vectors is by appending the features of each event in the prefix. In contrast, we use the same combination of aggregation and last state encoding as described in Section 5.1.3, which results in the same features for all cases.

We use SHAP explanations in an offline manner, meaning that we seek to explain which features are important to the trained model when making its causal effect estimation. We compute Shapley values for each feature in X using the methodology described in [17] and display the features with the highest Shapley values using a beeswarm plot. We chose the beeswarm plot because it can display the most important features and their distribution of Shapley values based on their feature value. For example, Figure 11a shows a list of the most important features. Each dot in the beeswarm plot corresponds to one data point (one prefix, encoded as a feature vector). The color of each data point denotes the value of that feature, with red denoting high values and blue denoting low values. The x-axis shows the Shapley values. Positive or negative Shapley values indicate whether that feature contributed to increasing or decreasing the treatment effect, respectively. Using the beeswarm plot, we can visualise how values of important features affect the estimation of treatment effects.

5.3.2 Policy Selection

Oftentimes, applying treatments to cases comes at a cost. Making phone calls to customers is an example of this. Having a treatment effect estimate for every running case can help decision makers separate the cases that would benefit from the treatment from those that would not be affected or would be negatively affected by it. Particularly, if

the cost of the treatment is not far below from its benefit, it is important to carefully select a policy to determine which cases to treat. Let us first define what we mean by a policy. A policy is a function Π that maps a feature set $X_i \in \mathscr{X}$ describing a case to a treatment decision, $\Pi : \mathscr{X} \rightarrow \{0,1\}$. There are many ways to select a policy. For example, we can decide to implement the random policy, meaning that for every case, we randomly decide whether to treat them or not. The random policy is not the best way of deciding treatment assignment. One way to improve on that is to decide treatment assignment based on prediction. For example, in the loan origination example, we can use predictive monitoring to predict the cycle time of each case. If the predicted cycle time is higher than a threshold, we treat that case by giving that customer a call. In this paper, we propose to select the best policy for treatment based on causal effect estimates. Accordingly, in this step, we propose a policy selection module that maximises the net benefit given the cost of the treatment and the treatment effect estimates. Furthermore, our proposed policy selection approach provides a way to evaluate the model that is used to estimate the treatment effects.

The first step in the policy selection module is to build a *Qini curve*. The Qini curve in this step is similarly built as the one in the method selection phase. The only difference here is that the curve is constructed using the real event log and we do not have $Y(1)$ and $Y(0)$ for every case. So, in the final step of calculating the Qini score for each percentage, we take the difference in the outcomes of the treatment and control group in lieu of the potential outcomes. Also, because the number cases in the treatment and control groups are usually different, the sum of the outcomes in the control group are multiplied by a scale factor. So, the Qini scores are computed the following way for cycle time reduction and outcome improvement respectively.

$$Qini(n) = R^{T=0}_{\pi(n)} \times \frac{N^{T=1}_{\pi(n)}}{N^{T=0}_{\pi(n)}} - R^{T=1}_{\pi(n)}.$$

$$Qini(n) = R^{T=1}_{\pi(n)} - R^{T=0}_{\pi(n)} \times \frac{N^{T=1}_{\pi(n)}}{N^{T=0}_{\pi(n)}}.$$

where $N^{T=1}_{\pi(n)}$ and $N^{T=0}_{\pi(n)}$ respectively denote the number of traces in the treated and control groups in $\pi(n)$. Note that we can evaluate the causal estimator's performance on the original event log by looking at its area under the Qini curve, similar to the method selection phase.

In addition to evaluating our causal estimator, we use the Qini curve for policy selection by incorporating the cost and the benefit multipliers to create *net value curves*. Suppose v is the value of reducing the cycle time by one unit of time (or improving the outcome of one case) and c is the cost of applying the treatment to one case. Then, the net gain of applying the treatment to $\pi(n)$ is defined as follows:

$$gain(n) = v \times Qini(n) - c \times N^{T=1}_{\pi(n)}.$$

Similar to the Qini curve, the net value curve can be drawn by computing $gain(n)$ for different values of n. The user has the option to view the curve and select the percentage n based on organizational constraints. For example, suppose the organization has a target of achieving a net benefit of x. In that case, the net value curve will provide the minimum proportion of cases that need to be treated to achieve that goal. If there

are no constraints, the percentage n that yields the highest net value can be selected automatically.

In the last step of this phase, we select a threshold for triggering the treatment. We do this by looking at the treatment effect of the last case in gain(n) for the n that was selected previously (recall that the cases in Gain(n) are ordered according to their treatment effects and the target). For example, the process owner of the loan origination process has looked that the net-value curve and decided that treating the top 30% of cases lead to the net-benefit required by the bank ($n = 30$). Suppose that the treatment effect of the last case observed in gain(30) is 0.25 ($\theta = 0.25$). Then the policy is defined as the following: For every new case, if the estimated treatment effect is higher than 0.25 ($\theta \geq 0.25$) apply the treatment, otherwise, do not apply.

Similar to the estimation phase, the selected policy can be interpreted. This is done by simplifying the causal model in the form of a tree. We build a policy tree in the following way: First we select the policy that we would like to have explanations for. For example, suppose the chosen policy is treating 30% of cases with the highest treatment effect. We label the data in such a way that the cases which are in the top 30% in terms of treatment effect receive the label "treat", and the rest of the cases receive the label "do no treat". We then proceed to build a decision tree on top of this labelled data. This tree tries to separate the subgroups with the label "treat", from the subgroups with the label "do not treat". By traversing the tree, explanations for both "treat" and "do not treat" can be shown.

5.4 Online Treatment

In the online phase, the applicability of the treatment for an ongoing case with an observed k-prefix is assessed. If the treatment is not applicable at that point, the assessment is repeated after the next event. If the treatment is applicable, the k-prefix of the case is encoded as a feature vector using the same approach as in the offline phase. Then, the treatment effect is estimated using the pre-trained causal model. If the estimated net gain is sufficiently high according to the pre-selected treatment policy, the prescriptive monitoring system recommends applying the treatment.

Let us come back to the example loan origination process discussed in Section 2. First, we take a log of this process, clean it, encode the case-level and event-level features, and create a treatment attribute, where the value of this attribute is zero for cases that got an email and 1 for cases that got a phone call. We then divide this preprocessed dataset into training and testing sets. We then create an artificial version of the event log with both potential outcomes. We select the best causal estimators which has a high area under the Qini curve, and is robust to the violation of the positivity and ignorability conditions. We then use the training set of the real data to train the selected estimator. This trained estimator can then take an incomplete case as input and returns, as a number, the estimated effect of calling the customer. We then estimate the effect of the phone call on all the cases in the test set. We use these estimates to create a net-value curve. The policy maker uses the curve to decide the percentage of future cases that will get the phone call based on organizational constraints.

In the online phase, a new application is created by the customer. After the execution of each activity, the applicability of the phone call at that stage is assessed. In the

initial stages of document processing, there is no need for the phone call, so we move on to the next activity. If we encounter missing documents in this case, we encode the information about this case using the same method as in the offline phase and estimate the treatment effect using our trained model. Suppose that the selected policy is to treat half of the cases. If the estimated effect for this new case is above the observed CATE in the top 50% of case, we call this customer; otherwise, we send an email.

6 Evaluation

We demonstrate our approach by conducting experiments on four real-life logs. The underlying assumption for the usefulness of our approach is the heterogeneity of the causal effect across different cases, meaning that each case with its unique set of conditions has a different response to the proposed treatment. Hence the most basic check that needs to be done before the prescriptive monitoring method is applied to a given process is to check if the event log contains enough signal about the heterogeneity of causal effects. If signal about heterogeneity is not present in the event log, then assigning a treatment based on the estimated causal effects will not provide more benefit than assigning the treatment randomly. Moreover, we would like to find the best causal estimator for each event log, since there is no single estimator that can outperform all the other estimators for every data set.

As discussed in Section 4, in observational studies the violation of positivity and ignorability assumptions are among the threats that can invalidate the study. Therefore, we need to check how the chosen causal estimator behaves if these assumptions are violated. These questions cannot be answered using the real-life event logs at our disposal, because the ground truth causal effect is not available for such logs. Therefore we answer these questions using our designed applicability test and method selection phase, which use simulated data. To evaluate the performance of our checks, we apply them to a fully simulated event log. In this log, we have full control of how the treatment and outcome variables are generated. We simulate scenarios where the positivity and ignorability conditions are severely violated to observe if our checks can capture these conditions.

Once the above issues are addressed, the chosen estimator can be applied to a real-life event log. The policy for applying treatment at run-time will be chosen based on the estimated causal effects on the real data. In addition, we incorporate an explainability module that provides explanations for both the causal estimator and the selected policy.

We translate the above issues to eight main research questions which we use to guide our evaluation:

- RQ1: How realistic are the artificial logs we generate using Realcause?
- RQ2: Can the positivity and ignorability violations checks successfully capture the violation of these assumption?
- RQ3: Do the selected datasets contain enough effect heterogeneity to benefit from the prescriptive monitoring approach?
- RQ4: Which causal estimator performs best for each dataset?

- RQ5: Which causal estimator is the least sensitive to the violation of the *positivity* assumption?
- RQ6: Which causal estimator is the least sensitive to the violation of the *ignorability* assumption?
- RQ7: How do the selected causal estimators perform when used on the original event logs?
- RQ8: How can the causal model and the policy be explained?

To answer these questions, we follow the pipeline in the proposed approach (see Fig. 1). Therefore, we conduct four main experiments: "statistical test on enhanced logs", which is designed to answer research question 1, "experiment on fully simulated log", which is designed to answer research question 2, "experiment on enhanced logs", which is designed to answer research questions 3-6, and "experiment on original logs", which is designed to answer research questions 7-8. The experiment on the enhanced log will follow the steps in the blue box in Fig. 1, and the experiment on the original logs will follow the steps in the red box.

6.1 Datasets

The data we used are the event logs from the Business Process Intelligence Challenge in 2016, 2017, 2019, and 2020, available from the 4TU Center for Research Data.[1] We used these datasets because they contain interventions that could possibly have a causal effect on the process outcome or the cycle time. Moreover, each log shows unique features. For example, BPI17 is characterised by a combination of case-level and event-level attributes, while most of the attributes in the BPI19 log are case-level. In addition to the BPIC logs above, we use a fully simulated claims management event log where we have full control of the treatment and outcome variables.

Claims Management: This log contains traces of a claims management process. This log contains various case attributes, resource information, and dynamic event attributes. We create an outcome variable called "satisfied" to indicate whether the customer was satisfied with the process of their claim. As a treatment, we add an activity "Call the customer" at various points in time between length two and six (for cases that are longer than six). We then create two versions of this event log. In the first version, we place the treatment in such a way that the treated cases are unrelated to the untreated ones (violation of positivity). In the second version of the log, we generate the treatment and outcome in such a way that both are highly influenced by resource attributes (strong confounding).

BPI16: This is the log of a call center from a Dutch autonomous administrative authority. The log contains information about the calls that were made by customers and the questions that they asked. In this log, it can be seen that sometimes customers call the call center multiple times to ask the same question. This can create extra work for the employees and keep the line busy, resulting in extra waiting times for other customers who are first time callers. One way to potentially alleviate this problem is to

[1] https://data.4tu.nl

spend more time with the customer when they call the first time to ensure that they have completely understood the answer to their question. So in this process, the treatment is increasing the duration of the call to be higher than average. We consider a case to have a negative outcome ($Y = 0$) if the customer calls again within a month after their first call, and a positive outcome otherwise.

BPI17: This log contains traces of a loan application process of a Dutch financial institute. The data contains attributes about the applications and the loan offers made by the bank. This log shows a number of cases with multiple offers made to the same customer. We observe that in such cases the rate of acceptance among customers is higher than in those cases where only a single offer is sent to the customer. However, sending multiple offers requires additional work for the resources. So in this process we aim to find a data-driven policy for sending multiple offers only to the customers which are highly likely to be responsive to it. Accordingly, the treatment is sending multiple offers, and the outcome of interest is whether the customer accepts the final offer.

BPI19: This log contains traces from a purchase-to-pay process of a Dutch multinational company. Each case describes a purchase order item from its creation to payment. This log also records activities such as changes, cancellations, and message exchanges related to purchase orders. In this log, we observe some activities that indicate a change in the purchase order items. Since changes might lead to re-work, we hypothesised that avoiding changes leads to lower cycle times. Specifically, we considered skipping of activity *Change Price* as our treatment for this experiment. However, fixing the price of purchase orders all at the same time for each case without considering the specific context of each case leads to a rigid order placement procedure and might require extra work at the beginning of the process. Thus, rather than fixing the price at the beginning of every case, we propose a targeted approach. Specifically, we provide a recommendation for each case indicating whether a price change should be permitted or not. In this way, we can avoid the rework in cases where this change is harmful and increases cycle time while preserving flexibility in other cases that are not highly affected by price change.

BPI20: This log contains the data of a university travel reimbursement process at TU Eindhoven. The process starts when a declaration is submitted by an employee and ends when the payment is handled. In this process, the activity *declaration APPROVED by BUDGET OWNER* is an extra check that could be skipped to reduce the cycle time. We observe that when this activity is present in the trace, the cycle time is longer. However, we do not know if the delay is caused by this activity or because managers who have a budget owner are in general busier and that is the cause of the longer cycle time in such cases. So, in this process the treatment is skipping the activity *declaration APPROVED by BUDGET OWNER* and the target of interest is reducing the cycle time.

6.2 Statistical Tests on Enhanced Logs

In this experiment, we aim to answer RQ1. To evaluate how realistic the enhanced logs are compared to their original counterpart, we run a series of statistical tests. They test the hypothesis that two samples come from different distributions. We use both

univariate and multivariate tests. We apply the univariate tests to the treatment and outcome variables and the multivariate test to treatment and outcome with and without the prefix attributes. For univariate testing, we use the Kolmogorov-Smirnov (KS) test and the Epps-Singleton (ES) test. The ES test is more suitable for discrete distributions (such as binary treatment or outcome). The multivariate tests we use are the Friedman-Rafsky test, k-nearest neighbor (kNN) test, and energy test. We also run permutation tests with Wasserstein-1 and Wasserstein-2 distance metrics. Table 1 shows the p-values of these tests. The null hypothesis is that the enhanced data and the original data come from the same distribution. Large p-values (e.g., $p > 0.05$) indicates that there is no statistically significant evidence that the enhanced data and the original data come from different distributions. All the p-values in Table 1 are above the commonly used threshold of 0.05. The answer to RQ1 is that the enhanced logs are statistically indistinguishable from the original event logs.

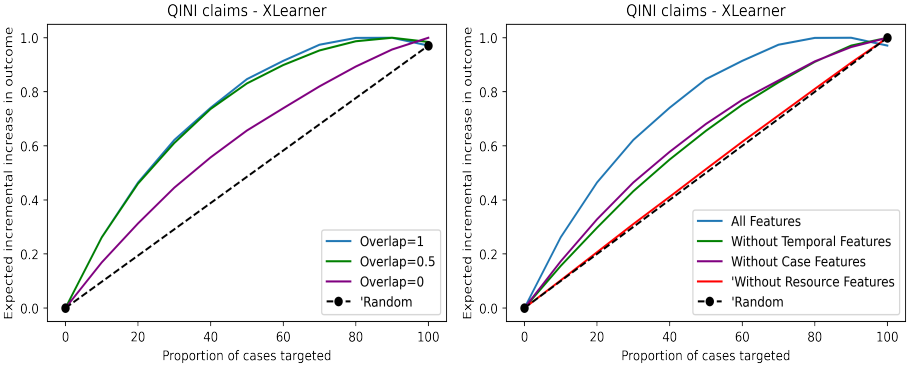
Test	BPI16	BPI17	BPI19	BPI20	Claims V1	Claims V2
T KS	0.0941	0.7605	0.6762	0.1359	0.4477	0.9325
T ES	0.1919	0.6567	0.879	0.2492	0.3557	09523
Y KS	0.9999	1.0	0.079	0.1	0.6534	0.1234
Y ES	0.8024	0.9396	0.1156	0.071	0.2862	0.1749
(T,Y) Wass1	0.528	0.432	0.609	0.052	0.751	0.422
(T,Y) Wass2	0.525	0.346	0.747	0.044	0.496	0.654
(T,Y) FR	0.587	1.0	0.144	0.089	0.805	0.054
(T,Y) kNN	0.878	0.292	0.651	0.081	0.731	0.059
(T,Y) Energy	0.521	0.454	0.622	0.44	0.738	0.276
(X,T,Y) Wass1	0.733	0.089	0.394	0.668	0.684	0.364
(X,T,Y) Wass2	0.698	0.24	0.519	0.554	0.645	0.363
(X,T,Y) FR	0.38	0.598	0.065	0.499	0.841	0.865
(X,T,Y) kNN	0.377	0.797	0.099	0.474	0.857	0.901
(X,T,Y) Energy	0.703	0.169	0.393	0.787	0.584	0.439
|X| (n attributes)	25	178	482	26	93	93

Table 1: Table of p-values for the various statistical tests on all the logs used in our experiments.

6.3 Experiment on Fully Simulated Logs

In this experiment, we aim to answer RQ2. We first created two versions of the claims management log. Version1 has unrelated treatment and control groups (positivity violation) and version2 has resource information as confounding variables. We applied Realcause to enhance both versions of the log with both potential outcomes. We then applied the positivity violation check to version1, and the ignorability violation check to version2 of the log. The treatment and outcome generation was done using the CausalML library, which allowed us to purposefully violate positivity and ignorability. We only report the Qini curves for X-Learner because we achieved almost identical results with other causal estimators. Figure 2 shows the results:

The positivity check was applied to version1 of the log. In version1, we have unrelated treatment and control groups, meaning that some cases have no chance of being selected for treatment. In Figure 2a we can see a big drop in performance when



(a) Positivity Check

(b) Ignorability Check

Figure 2: Positivity and Ignorability violation checks for Claims Management

the overlap knob is set to zero. Recall that the overlap knob adjusts the distribution of treatment assignment. The fact that the area under the Qini curve drops significantly with $overlap = 0$ means that the generated causal effects of Realcause were realistic enough to reflect the data generating distribution (in this case, unrelated treatment and control groups).

We applied the ignorability check to version2 of the claims management log. In this version, we generated the treatment and outcome columns with resource features as confounding variables. We can observe in Figure 2b that when we train the causal estimator without the resource features (resource features are hidden from the causal estimator), performance drops to the level of random. This shows that our check successfully identified the confounding present in the data and that the Realcause generation of true treatment effect reflects this. So the answer to RQ2 is that the positivity and ignorability checks can successfully detect these violations.

6.4 Experiment on Enhanced Logs

In this experiment, we aim to answer RQ1–4 and to choose the best causal estimator for each log. We trained a generative model of Realcause to create artificial versions of our four logs which are statistically indistinguishable from the real ones. To answer RQ1 and RQ2, we measured the estimated causal effects using various causal estimation methods and reported the findings for three methods which performed well on these logs. We then proceeded to the positivity violation check to answer RQ3 and the ignorability violation check to answer RQ4. If the results from the artificial logs were satisfactory, we went on to apply the selected estimation method on the original logs.

6.4.1 Evaluation measures:

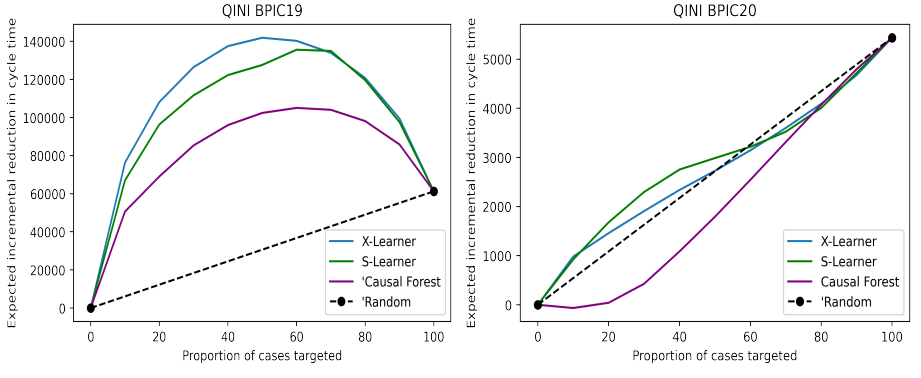
In this experiment, we have access to both potential outcomes, therefore, we can directly measure the benefit of applying the treatment without using the treatment and

control group as a proxy. We use the area under the Qini curve to compare the different estimators.

6.4.2 Training settings

Generative model for creating the enhanced log. For training the generative sigmoidal flow model, we use a 50%–10%–40% split for training, validation, and testing, respectively. We also standardise the data to have zero mean and unit variance to get maximum performance. We use the default optimization and model selection procedure in Realcause, that is, we use an Adam optimiser to maximise the likelihood on the training set, perform hyperparameter tuning using grid search, and select the model with the best validation likelihood and a p-value passing 0.05 on the validation set.

Causal estimator on the enhanced logs. To train the causal estimators on the enhanced logs, we sample from the generative model, which results in an event log with the same number of cases as the original log. We sample a separate test set to evaluate the model. Each simulation is run 100 times, and the results of the simulations for each estimator are averaged. For the positivity violation check, we use the Overlap knob in Realcause with $0, 0.5, 1$ for the β value. For the ignorability violation check we train the causal estimation multiple times, each time by excluding a set of features, namely: i) case features, ii) resource features, and iii) time features.

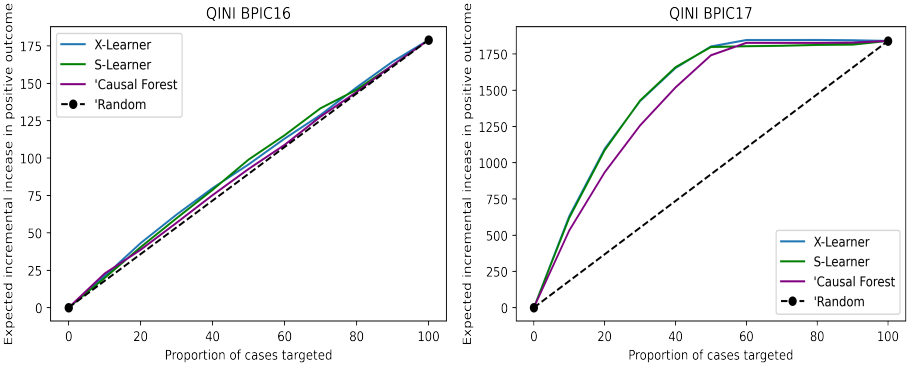


(a) BPI 2019 (b) BPI 2020

Figure 3: Qini curves for cycle time datasets.

6.4.3 Results

It is clear from Qini curves for BPI16 and BPI20 that applying a causal estimation method on these two logs results in very minimal improvement over treating the cases randomly (see Figures 3 & 4). This is because in these two datasets, there is not enough signal for capturing the heterogeneity of treatment effects. Therefore, defining a policy based on ordering cases by their estimated CATE does not lead to any benefit. Also, we



(a) BPI 2016

(b) BPI 2017

Figure 4: Qini curves for outcome datasets.

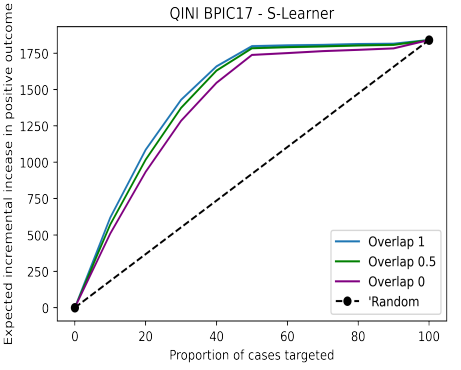
observe that these treatments do not lead to much benefit even if all the cases receive the treatment. This is because in general, the causal effect of the chosen treatments are not very high. Recall that in the BPIC20 event log one reason for a long cycle time when the event *Declaration APPROVED by BUDGET OWNER* is present is that a supervisor with a separate budget owner may in general be busier than a supervisor without one. So, skipping the activity involving the budget owner may not be a helpful treatment, and hence, we observe a low area under the Qini curve. Based on these results, we stop the analysis for these two event logs here.

For the BPI17 and BPI19 logs, we can see a significant improvement over the random treatment policy, suggesting that these logs contain information about the heterogeneity of the CATE among the cases. So, we are able to estimate an individual CATE score for each case, and doing so will lead to some benefit. But, these conclusions are made under the positivity and ignorability assumptions. In observational studies such as the study of event logs, these two assumptions are often violated. In the next step, we look at how these models behave under positivity violation.

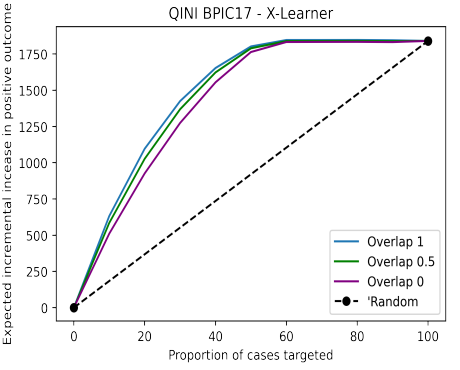
In summary, the answer to RQ1 is that BPI17 and BPI19 have enough signal about effect heterogeneity to benefit from our prescriptive monitoring approach, but BPI16 and BPI20 do not. The answer to RQ2 is that the S-Learner, X-Learner and Causal Forest are all high performing causal estimators for the BPI17 and BPI19 logs. The two meta-learners slightly outperform Causal forest, and in BPIC19, the X-Learner outperforms the S-Learner.

Positivity violation check: Here, we train each selected causal estimator with a different degree of overlap. Figures 5 and 6 show the Qini curves for S-Learner, X-Learner, and Causal Forest with overlap values 0, 0.5 and 1. It can be seen that these models are highly robust to positivity violation as the decrease in the area under the Qini curve is minimal for all three estimators. In fact, for the BPI19 event log, there is no change in the Qini score for the top 10% of cases, meaning that we are still able to identify the cases benefiting most from the treatment, even if treatment assignment

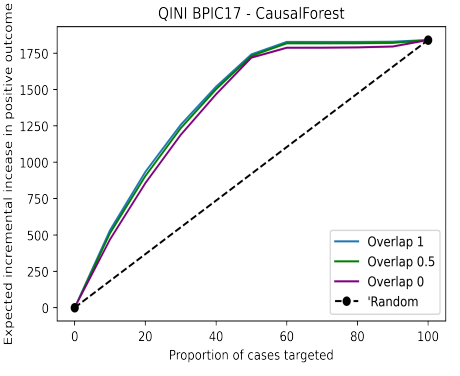
is deterministic. So, the answer to RQ3 is that all three estimators are robust to the violation of the positivity assumption.



(a) BPI 2017 SLearner



(b) BPI 2017 XLearner

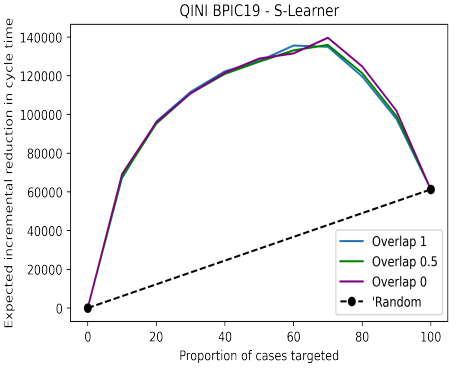


(c) BPI 2017 Causal Forest

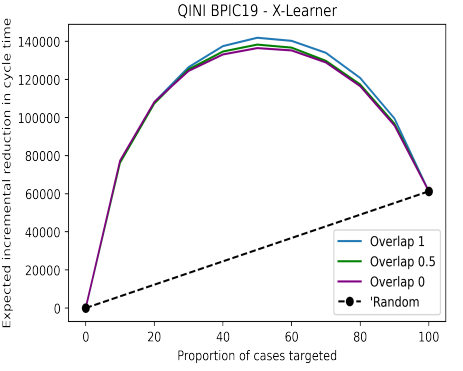
Figure 5: Qini curves for outcome dataset.

Ignorability violation check: This is the last step of this experiment to answer RQ4. Here, we re-train the estimators each time with one set of features removed. The results of this experiment can be found in Figures 7 and 8. For BPI17, the case attributes are the strongest confounders because the area under the Qini curve reduces a lot more than other features when it is removed. Also, comparing the three chosen estimators, for this dataset, Causal Forest seems to be most sensitive to confounding. This is apparent from the amount of reduction in the area under the curve. For this event log, the S-Learner is the most robust to confounding, and since its area under the Qini curve is similar to the other two estimators, it is the chosen method to be applied to the real-life BPI challenge event log.

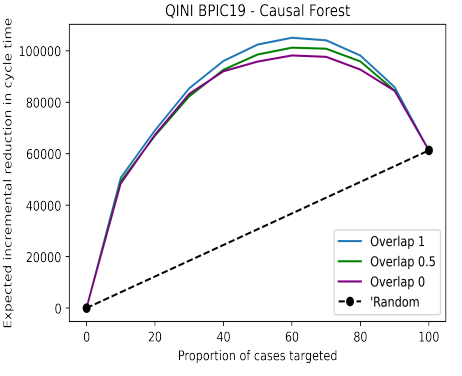
In the BPI19 dataset, Causal Forest seems to be the most sensitive again. But, this time it is the time features which are the strongest confounders, which is expected



(a) BPI 2019 SLearner



(b) BPI 2019 XLearner



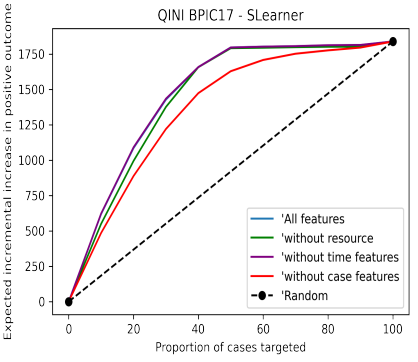
(c) BPI 2019 Causal Forest

Figure 6: Qini curves for cycle time dataset.

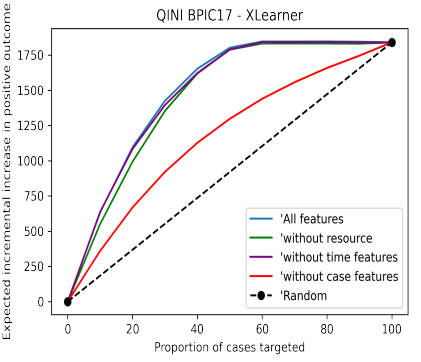
because the target variable in this event log is the case cycle time. Both S-Learner and X-Learner perform very well even under strong confounding. The S-Learner struggles on the lower percentages, but recovers at around 60%. The X-Learner's curve under strong confounding is consistent with the curve under no confounding, which leads us to believe that that X-Learner is a more stable estimator for this dataset, and is chosen for the next phase of the approach. In summary, the answer to RQ4 is that the S-Learner is the most robust to the violation of ignorability for BPI17 and the X-Learner for BPI19.

6.5 Experiment on Original Logs

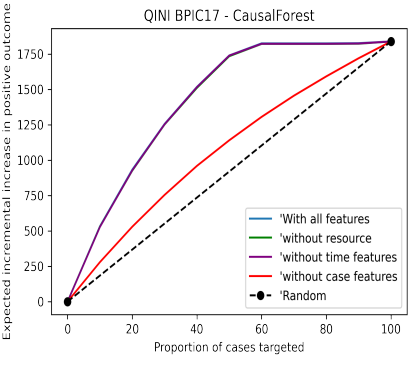
While the purpose of the previous experiment was to select the best causal estimator for each dataset, in this experiment we assess the performance of the selected estimator on the original logs. We will also discuss the benefit of different treatment policies.



(a) BPI 2017 SLearner



(b) BPI 2017 XLearner



(c) BPI 2017 Causal Forest

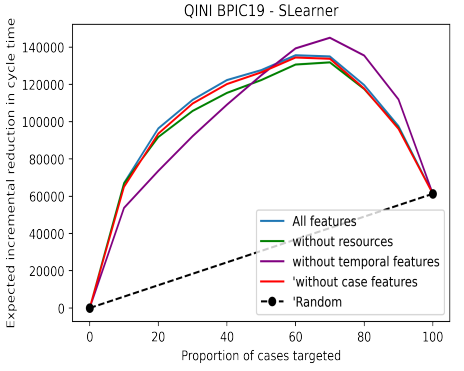
Figure 7: Qini curves for outcome dataset.

6.5.1 Evaluation measures

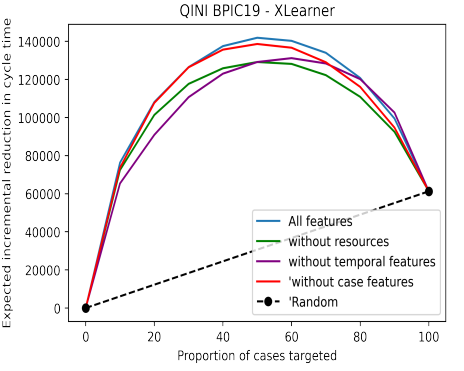
Unlike the enhanced logs, the original logs do not have both potential outcomes. So, the last stage of computing the Qini score will be the difference in target values in the control and treatment groups. The underpinning intuition is that if the CATE is estimated accurately, the cases with a positive outcome in the treated group would have a higher estimated CATE than those in the same group with negative outcomes. Also, cases in the control group with a negative outcome should have a higher CATE than the positive outcome cases in the same group. Thus, a desirable causal model has a Qini curve above the random curve. Similar to the previous experiment, we will use the area under the Qini curve as the main performance metric.

6.5.2 Training setting

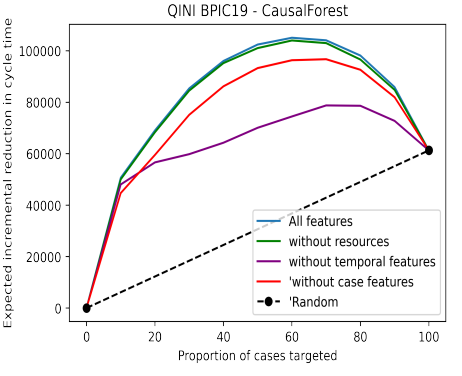
Here, we split the data into 60%–20%–20% for training, validation, and testing, respectively, by preserving the temporal order between cases. We used the training set to



(a) BPI 2019 SLearner



(b) BPI 2019 XLearner



(c) BPI 2019 Causal Forest

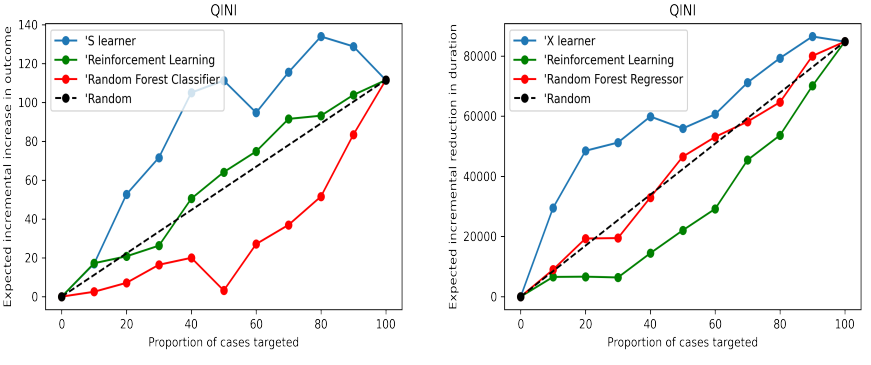
Figure 8: Qini curves for cycle time dataset.

train the causal model, the validation set to tune the hyperparameters, and the test set to provide an unbiased evaluation of the model. We standardise the data similar to the artificial log generation step.

6.5.3 Results

Here we answer RQ5. The plots in Figure 9 show the Qini curves of the models constructed in our experiments. The black dashed line shows the expected improvement if a random policy was used to treat a certain percentage of the cases. The blue line shows the improvement if the policy was based on the selected causal estimator, the green line if the policy was determined by the reinforcement learning method proposed in [18] and the red line if the policy was determined by a random forest predictor. It can be seen that in both datasets, the expected improvement is higher if the CATE-based policy is followed rather than policies based on non-causal predictive models. Notably, the curves for the predictive model are close to the random policy line indicating that *while predictive models are good at identifying which cases will be problematic in the*

future, they are not necessarily good at identifying which cases should be targeted with the chosen treatments. The reason that the reinforcement learning method does not perform well in this task is that while RL is well suited to determine treatment policies, the method proposed by [18] relies on a predictive model and is designed to find cases that need *a treatment*, not the cases that need the specific treatments we have chosen. An interesting line for further investigation is to combine reinforcement learning (e.g., the method proposed by [18]) with causal estimation to get treatment policies for specific treatments. Both the S-Learner for BPI17 and the X-Learner for BPI19 have Qini curves well above random. In BPI17, the curve shows that treating 80% of cases will have more benefit than treating every case. For BPI19, the X-Learner is very good at identifying top cases. According to the Qini curve, treating only 40% of cases will lead to more than half the benefit of applying the treatment to everybody. Also, treating 90% of cases has more benefit than treating every case. Note that this curve is not monotonically increasing as usual cumulative gain charts because the quantity on the y-axis is the *difference* between treated and non-treated cases, which in some segments can be negative since the treatment hurts the outcome in these cases.

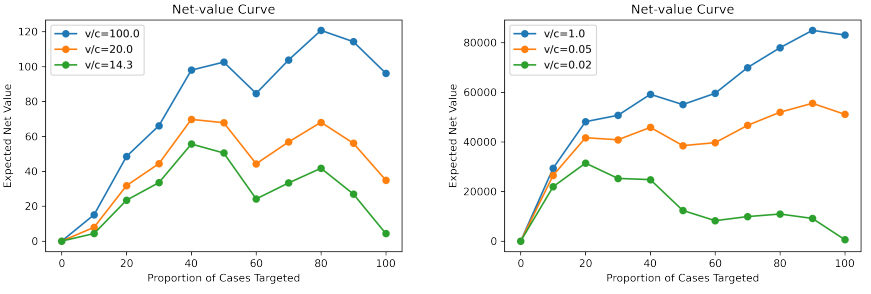


(a) BPI 2017 (b) BPI 2019

Figure 9: Qini curves for original logs.

The Qini curves are without cost and benefit multipliers. So, we proceeded to plot the net value curves for both models with varying values of v/c, where v is the value of improving the target variable (reducing the cycle time by one day or getting a positive outcome for one case), and c is the cost of applying the treatment to one case. Rather than the absolute values of v and c, it is the ratio v/c that affects the shape of the curve. Fig. 10 shows the net value curves with different values of v/c for the two logs. We observe that as the ratio v/c decreases — i.e. the treatment becomes more expensive relative to the benefit it provides — the net value of treating the cases decreases, and so, it becomes more important to apply the treatment with a more targeted approach. Fig. 10 shows the net-values for three values of v/c. For the curve in blue, v/c is chosen in such a way that it produces the same shape as the Qini curve without the cost and benefit multipliers, and for the green curve, v/c is chosen in such a way that the net-value of applying the treatment to all cases is zero. The orange curve describes a

situation between these two extremes. The way we choose v/c is that we suppose $v = 1$ and we change the value c to produce curves with different v/c. For BPI17, the value of applying the treatment should be 100 times more than the cost of treatment for it to produce the same net-value as the Qini curve suggests, while in BPI19, this is the case if the value is only equal to the cost. Also, for the treatment to provide no net-value when applied to all cases, the ratio v/c needs to be 14.3 in BPI17 and 0.02 in BPI19. This leads us to believe that the chosen treatment in BPI19 is a more value-adding treatment than the one chosen for BPI17.



(a) BPI 2017 (b) BPI 2019

Figure 10: Net value curves

Explainability: Many heterogeneous treatment effect estimation models, such as the ones used in this evaluation (i.e., S-Learner and X-Learner) are black-box models. Yet, these black-box models can be explained similar to predictive models via the SHAP method [17]. We use the EconML implementation of SHAP. These SHAP values explain why the model produces a large or small CATE for a particular subgroup. We plot the feature importance for the causal models that we trained in this experiment to provide insight as to why our model behaves the way it does. Figure 11 shows the most important features for each dataset. For BPI17, the customer's credit score is an important feature. In particular, the treatment seems to have a negative effect on customers with very high or very low credit scores, while cases with average credit score benefit from the treatment. The application type is another important feature, with new credit applications benefiting from the treatment in contrast to limit raise applications. In the BPI19 dataset, the most important feature is the presence of the activity *Create Purchase Order Item*. This explanation is expected, because many cases in the event log are purchase requisition items that do not lead to a purchase order. Since our treatment is about allowing price changes, it is expected that we do not recommend this treatment if no purchase order is created first. Another important feature is the number of open cases. When the number of open cases is large, the treatment is recommended. This explanation is intuitive. If there are a large number of active cases, allowing price changes adds to the workload of the resources leading to delays in cycle time.

Let us now compare these explanations with the most important features used by the random forest predictors. Figure 12 shows the most important features for the predictive model. For BPI17, Credit Score is the most important feature, similar to the

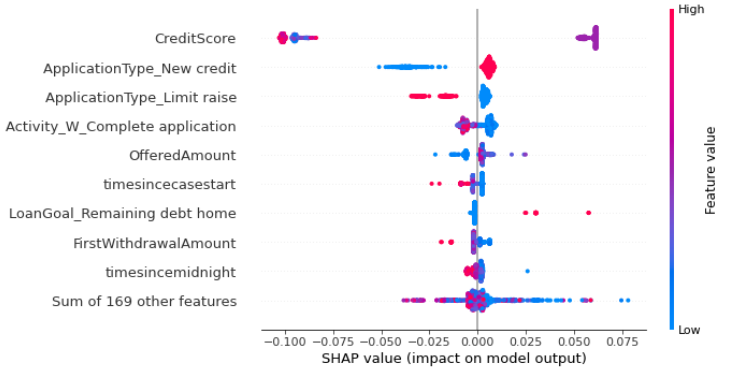
causal model. But, the way this feature influences the model is very different. For the predictive model, high credit score leads to a high outcome, but for the causal model, high credit score does not lead to a high treatment effect. Instead, the customers with average credit scores get a high treatment effect, but the average credit score is not indicative of a successful outcome. Other differences between the causal model and the predictive model include the features which are most important. For the causal model, application type is an important feature. But this feature does not appear in the most important features of the predictive model. Instead, the predicitve model relies on the presence of certain activities. Also, not surprisingly, the offered amount is a high predictor of case success, with high offered amounts being predictors of a successful case. But this attribute is not among the most important features of the causal model.

For the BPI19 dataset, we also observe some features that are important both for the causal model and the predictive model. For example, time since first case is a highly predictive feature for cycle time prediction. We can see that low values of this attribute indicates a high cycle time. This means that earlier cases in the event log take longer. We also observe that this feature indicates the effectiveness of the treatment. An important difference between the causal model and the predictive model is the importance the case attributes in each model. For the predictive model, the cumulative net worth is an important feature. More expensive purchase items have a longer cycle time. But this features is not important in causal estimation of the chosen treatment. Instead, the case attributes related to the type of purchase order, e.g., Spend area test determine are important indicators of causal effects.

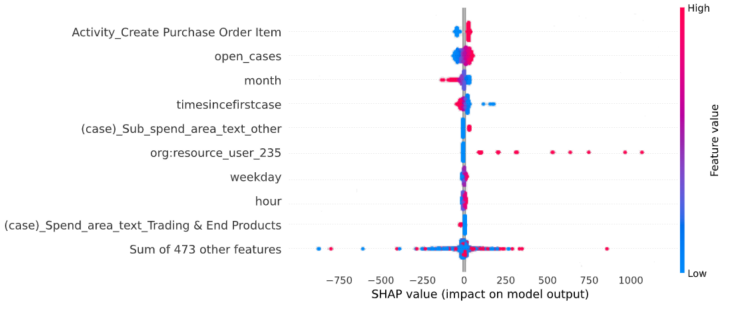
We also provide explanations for the treatment policy. The policy we select is the one that provides the highest net-value in the green coloured curve in Fig. 10. We chose this curve as it is a scenario describing a situation when the treatment cost is high relative to the value it provides. For the BPI17 event log the chosen policy is treating 40% of cases with the highest estimated causal effect and for BPI19, the chosen cases are in the top 20%. Figure 13 shows a fragment of the policy tree for BPI17. We only show a fragment of the policy tree for readability purposes.[2] The orange nodes contain the cases that should not receive the treatment, while the blue nodes contain cases recommended for treatment. Each node contains the splitting criterion, the number of samples in that node, the number of samples for each label (treat or not treat), and the dominant class label. We can see that a number of features which are important according to the SHAP methodology, such as "Credit Score", "Application Type New Credit", and "Loan Goal Remaining Debt Home", also appear in the policy tree, reinforcing their importance.

Figure 14 shows a fragment of the policy tree for BPI19. In this tree, we can also see some common features with the SHAP plot of Fig. 11b. For instance, features such as "time since first case", "Activity Create Purchase Order Item", and "Open Cases" appear in both SHAP plot and policy tree.

[2]The full tree can be found in the GitHub repository mentioned in the reproducability section of this paper.



(a) BPI 2017

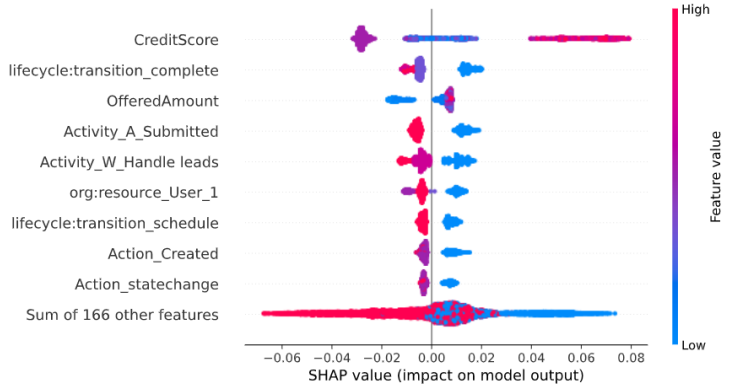


(b) BPI 2019

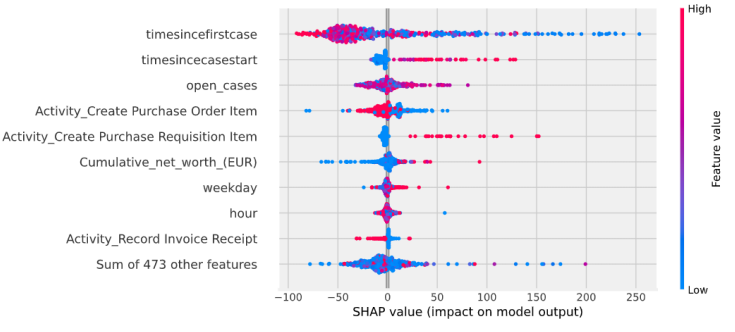
Figure 11: Feature importance using SHAP for S and X Learners

6.6 Threats to Validity

The evaluation comes with a threat to external validity (lack of generalizability) as it relies on only four event logs. This can be addressed by conducting further experiments with logs of different characteristics and from different domains. Since we used four real-life event logs in our evaluation, a threat to internal validity is posed by potential data quality issues. To mitigate this threat, we performed data cleaning steps on all four logs. A threat to construct validity is coming from the fact that we use simulations to enhance the event logs with alternative outcomes. For these simulations, we use established methods from the fields of causal inference and machine learning, but it is possible that the assumptions upon which these methods rely are not fulfilled in some real-life situations. Specifically, using the ignorability violation check, we attempt to ensure that the chosen method is robust to unobserved confounding. However, if the actual unobserved confounders are stronger than the confounders present in the data, this check is not sufficient to analyze the behaviour of the estimator under stronger confounding. So, a rigorous A/B test, where the randomization of the treatment assignment is guaranteed, should be conducted before deploying the recommendations of our method in an operational setting. Finally, due to the sharing of resources between



(a) BPI 2017



(b) BPI 2019

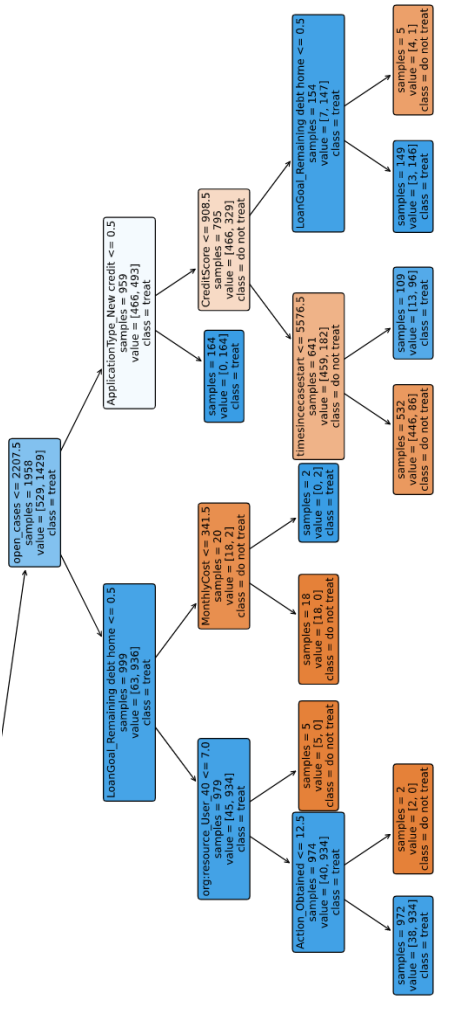
Figure 12: Feature importance using SHAP for Random Forest Predictors

process cases, we expect that the no-interference assumption to be violated. However, the impact of the violation of no-interference is dependent on the specific process. So a domain expert needs to assess the severity of this violation before our method is applied.

7 Conclusion and Future Work

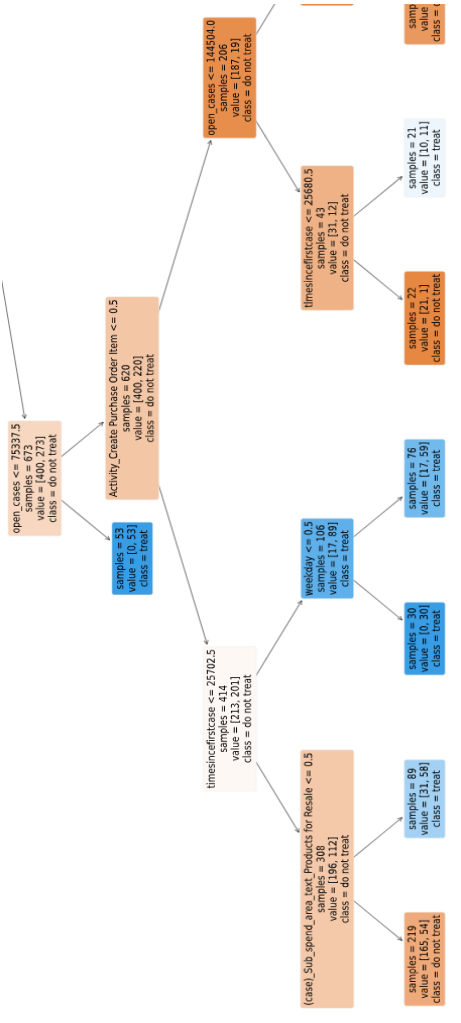
We proposed a prescriptive monitoring method that recommends whether or not to apply an intervention (treatment) to an ongoing case, to improve a given process performance metric. The method uses causal inference methods on historical traces, to estimate the causal effect (the treatment effect) given the current state of a case. Based on this estimate, the method calculates the expected gain of the treatment given a cost function and generates recommendations based on a user-defined policy. The method incorporates steps for checking the applicability of the supported causal effect estimation methods on the input data, and for selecting a causal estimation method specifically for a given dataset.

We demonstrated the applicability of the proposed method on four real-life logs.



(a) BPI 2017

Figure 13: Excerpt from policy tree for treating the top 30% of cases for BPIC17



(a) BPI 2019

Figure 14: Excerpt from policy tree for treating the top 20% of cases for BPIC19

We showed that the method is able to evaluate causal models in terms of robustness to confounding effects and the violation of the positivity assumption. We also showed that the proposed approach is able to identify policies that yield higher net-gain than treatment policies based on non-causal predictive approaches, as previously proposed in the literature.

The proposed approach tells us whether a given case should be treated or not, but it does not deal with the question of what is the optimal time to trigger an intervention during the lifetime of a case. An avenue for future work is to optimize the time of the treatment (i.e. "when to treat?") based on historical data.

The approach assumes that there is a single type of treatment and that this treatment is binary (i.e. it is either applied to a case or not). Another direction for future work is extending this method to accommodate multiple treatment types (choosing between one type of treatment or another) as well as treatments with continuous values (e.g. offering a discount of X% to a customer where X is a number).

Another limitation of the proposed approach is our assumption that there is no unobserved confounding. We address this limitation by performing an ignorability violation check. However, this check only shows the behaviour of the causal estimator for known confounders. If there are unknown confounders with higher strength than the known ones, or if their strength is unknown, other methods for addressing unobserved confounding (e.g., instrumental variable methods, front door adjustment, etc.) need to be employed. We leave that for future work.

We use SHAP and policy tree methods to explain the causal model and the selected policy. Another interesting area of further exploration is using other explainability methods, particularly the ones providing a comprehensive framework for business processes e.g., [39, 7]

Finally, the approach requires the treatment type to be given as input by the user. Another direction for future work is to design methods to automatically discover candidate treatments from a historical event log.

Reproducibility The source code of our tool, the datasets used in our evaluation and all the results from the experiments can be found at https://github.com/zahradbozorgi/CausalPrescriptiveMonitoring.

Acknowledgments Research funded by the Australian Research Council (grant DP180102839), the European Research Council (PIX Project), and the Estonian Research Council (grant PRG1226). Thanks to Simon Remy, Kiarash Diba, and Luise Pufahl for providing preprocessed BPIC datasets.

References

[1] Susan Athey, Guido W. Imbens, Jonas Metzger, and Evan Munro. Using wasserstein generative adversarial networks for the design of monte carlo simulations. *Journal of Econometrics*, 2021.

[2] Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized random forests. *The Annals of Statistics*, 47(2):1148 – 1178, 2019.

[3] Jake Bowers, Mark M. Fredrickson, and Costas Panagopoulos. Reasoning about interference between units: A general framework. *Political Analysis*, 21(1):97–124, 2013.

[4] Zahra Dasht Bozorgi, Irene Teinemaa, Marlon Dumas, Marcello La Rosa, and Artem Polyvyanyy. Process mining meets causal machine learning: Discovering causal rules from event logs. In *2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4-9, 2020*, pages 129–136. IEEE, 2020.

[5] Zahra Dasht Bozorgi, Irene Teinemaa, Marlon Dumas, Marcello La Rosa, and Artem Polyvyanyy. Prescriptive process monitoring for cost-aware cycle time reduction. In *3rd International Conference on Process Mining, ICPM 2021, Eindhoven, Netherlands, October 31 - Nov. 4, 2021*, pages 96–103. IEEE, 2021.

[6] Stephan A. Fahrenkrog-Petersen, Niek Tax, Irene Teinemaa, Marlon Dumas, Massimiliano de Leoni, Fabrizio Maria Maggi, and Matthias Weidlich. Fire now, fire later: alarm-based systems for prescriptive process monitoring. *Knowledge and Information Systems*, 64(2):559–587, 2022.

[7] Riccardo Galanti, Bernat Coma-Puig, Massimiliano de Leoni, Josep Carmona, and Nicolò Navarin. Explainable predictive process monitoring. In *2020 2nd International Conference on Process Mining (ICPM)*, pages 1–8. IEEE, 2020.

[8] Justin Grimmer, Solomon Messing, and Sean J. Westwood. Estimating heterogeneous treatment effects and the effects of heterogeneous treatments with ensemble methods. *Political Analysis*, 25(4):413–434, 2017.

[9] Paul W Holland. Statistics and causal inference. *Journal of the American statistical Association*, 81(396):945–960, 1986.

[10] Bart F. A. Hompes, Abderrahmane Maaradji, Marcello La Rosa, Marlon Dumas, Joos C. A. M. Buijs, and Wil M. P. van der Aalst. Discovering causal factors explaining business process performance variation. In *Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings*, volume 10253 of *Lecture Notes in Computer Science*, pages 177–192. Springer, 2017.

[11] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron C. Courville. Neural autoregressive flows. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2083–2092. PMLR, 2018.

[12] Zhengxing Huang, W.M.P. van der Aalst, Xudong Lu, and Huilong Duan. Reinforcement learning based resource allocation in business process management. *Data & Knowledge Engineering*, 70(1):127–145, 2011.

[13] Guido W. Imbens. Potential outcome and directed acyclic graph approaches to causality: Relevance for empirical practice in economics. *Journal of Economic Literature*, 58(4):1129–79, December 2020.

[14] Jelmer Jan Koorn, Xixi Lu, Henrik Leopold, and Hajo A. Reijers. Looking for meaning: Discovering action-response-effect patterns in business processes. In *Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings*, volume 12168 of *Lecture Notes in Computer Science*, pages 167–183. Springer, 2020.

[15] Kateryna Kubrak, Fredrik Milani, Alexander Nolte, and Marlon Dumas. Prescriptive process monitoring: Quo vadis?, 2021.

[16] Sören R. Künzel, Jasjeet S. Sekhon, Peter J. Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4156–4165, 2019.

[17] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc.

[18] Andreas Metzger, Tristan Kley, and Alexander Palm. Triggering proactive business process adaptations via online reinforcement learning. In *Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings*, volume 12168 of *Lecture Notes in Computer Science*, pages 273–290. Springer, 2020.

[19] Andreas Metzger, Adrian Neubauer, Philipp Bohn, and Klaus Pohl. Proactive process adaptation using deep learning ensembles. In *Advanced Information Systems Engineering - 31st International Conference, CAiSE 2019, Rome, Italy, June 3-7, 2019, Proceedings*, volume 11483 of *Lecture Notes in Computer Science*, pages 547–562. Springer, 2019.

[20] Tanmayee Narendra, Prerna Agarwal, Monika Gupta, and Sampath Dechu. Counterfactual reasoning for process optimization using structural causal models. In *Business Process Management Forum - BPM Forum 2019, Vienna, Austria, September 1-6, 2019, Proceedings*, volume 360 of *Lecture Notes in Business Information Processing*, pages 91–106. Springer, 2019.

[21] Brady Neal, Chin-Wei Huang, and Sunand Raghupathi. Realcause: Realistic causal inference benchmarking, 2021.

[22] Miruna Oprescu, Vasilis Syrgkanis, and Zhiwei Steven Wu. Orthogonal random forest for causal inference. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 4932–4941. PMLR, 2019.

[23] Judea Pearl. *Causality*. Cambridge university press, 2009.

[24] Artem Polyvyanyy, Anastasiia Pika, Moe T. Wynn, and Arthur H.M. ter Hofstede. A systematic approach for discovering causal dependencies between observations and incidents in the health and safety domain. *Safety Science*, 118:345–354, 2019.

[25] Mahnaz Sadat Qafari and Wil M. P. van der Aalst. Root cause analysis in process mining using structural equation models. In *Business Process Management Workshops - BPM 2020 International Workshops, Seville, Spain, September 13-18, 2020, Revised Selected Papers*, volume 397 of *Lecture Notes in Business Information Processing*, pages 155–167. Springer, 2020.

[26] Mahnaz Sadat Qafari and Wil M. P. van der Aalst. Case level counterfactual reasoning in process mining. In *Intelligent Information Systems - CAiSE Forum 2021, Melbourne, VIC, Australia, June 28 - July 2, 2021, Proceedings*, volume 424 of *Lecture Notes in Business Information Processing*, pages 55–63. Springer, 2021.

[27] Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688, 1974.

[28] Uri Shalit, Fredrik D. Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3076–3085. PMLR, 06–11 Aug 2017.

[29] Mahmoud Shoush and Marlon Dumas. Prescriptive process monitoring under resource constraints: A causal inference approach. In *Second International Workshop on Leveraging Machine Learning in Process Mining*, 2021.

[30] Johan Silvander. Business process optimization with reinforcement learning. In Boris Shishkov, editor, *Business Modeling and Software Design*, pages 203–212, Cham, 2019. Springer International Publishing.

[31] Renuka Sindhgatta, Aditya K. Ghose, and Hoa Khanh Dam. Context-aware analysis of past process executions to aid resource allocation decisions. In *Advanced Information Systems Engineering - 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings*, volume 9694 of *Lecture Notes in Computer Science*, pages 575–589. Springer, 2016.

[32] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[33] Irene Teinemaa, Marlon Dumas, Marcello La Rosa, and Fabrizio Maria Maggi. Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Trans. Knowl. Discov. Data*, 13(2):17:1–17:57, 2019.

[34] Irene Teinemaa, Niek Tax, Massimiliano de Leoni, Marlon Dumas, and Fabrizio Maria Maggi. Alarm-based prescriptive process monitoring. In *Business Process Management Forum - BPM Forum 2018, Sydney, NSW, Australia, September 9-14, 2018, Proceedings*, volume 329 of *Lecture Notes in Business Information Processing*, pages 91–107. Springer, 2018.

[35] Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

[36] Stefan Wager and Kuang Xu. Experimenting in equilibrium. *Management Science*, 67(11):6629–7289, 2021.

[37] Sven Weinzierl, Sebastian Dunzer, Sandra Zilker, and Martin Matzner. Prescriptive business process monitoring for recommending next best actions. In *Business Process Management Forum - BPM Forum 2020, Seville, Spain, September 13-18, 2020, Proceedings*, volume 392 of *Lecture Notes in Business Information Processing*, pages 193–209. Springer, 2020.

[38] Arif Wibisono, Amna Shifia Nisafani, Hyerim Bae, and You-Jin Park. On-the-fly performance-aware human resource allocation in the business process management systems environment using naïve bayes. In *Asia Pacific Business Process Management - Third Asia Pacific Conference, AP-BPM 2015, Busan, South Korea, June 24-26, 2015, Proceedings*, volume 219 of *Lecture Notes in Business Information Processing*, pages 70–80. Springer, 2015.

[39] Bemali Wickramanayake, Chun Ouyang, Catarina Moreira, and Yue Xu. Generating purpose-driven explanations: The case of process predictive model inspection. In Jochen De Weerdt and Artem Polyvyanyy, editors, *Intelligent Information Systems*, pages 120–129, Cham, 2022. Springer International Publishing.

[40] Weijia Zhang, Jiuyong Li, and Lin Liu. A unified survey of treatment effect heterogeneity modelling and uplift modelling. *ACM Computing Surveys*, 54(8):36, 2022.