Case Level Counterfactual Reasoning in Process Mining

Mahnaz Sadat Qafari(✉) and Wil M. P. van der Aalst

Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Aachen, Germany
{m.s.qafari,wvdaalst}@pads.rwth-aachen.de

Abstract. Process mining is widely used to diagnose processes and uncover performance and compliance problems. It is also possible to see relations between different behavioral aspects, e.g., cases that deviate more at the beginning of the process tend to get delayed in the later part of the process. However, correlations do not necessarily reveal causalities. Moreover, standard process mining diagnostics do not indicate how to improve the process. This is the reason we advocate the use of *structural equation models* and *counterfactual reasoning*. We use results from causal inference and adapt these to be able to reason over event logs and process interventions. We have implemented the approach as a ProM plug-in and have evaluated it on several data sets.

Keywords: Process mining · Counterfactual statement · Structural equation model

1 Introduction

Humans tend to learn from the past (their experiences) by analyzing possible alternatives of what happened in the reality and reflecting on their findings aiming for better results in future similar cases (e.g., not doing the same mistakes). Thinking about possible alternatives to what happened in the past is called *counterfactual thinking*.

The information systems of companies save data about the process instances (cases) in their event logs. Process mining extracts knowledge from the event logs for discovering the process model, monitoring process KPIs, and improving processes. Process improvement requires a deep comprehension of the process behavior and its cases. In this paper, we tailor the concept of *counterfactual thinking* to process mining and explain why a specific situation has a special outcome. Given an instance with an undesirable outcome, we aim at providing a set of counterfactual statements (we call them *explanations*) to explain such an outcome.

Companies can boost customer satisfaction and build trust by providing explanations for their specific cases without putting other people's rights and privacy in danger [7]. Case level explanation can be used to explain why a customer has received a particular result, was it fair, or how to approach to get a better result. Moreover, the process manager can benefit from this method as it can be used to explain why something happens in a specific case and how to act differently to get different results in the future.

S. Nurcan and A. Korthaus (Eds.): CAiSE Forum 2021, LNBIP 424, pp. 55–63, 2021.
https://doi.org/10.1007/978-3-030-79108-7_7

Two important aspects of an explanation are accuracy and applicability. Both of them can be amended by distinguishing between correlation and causation among the process features, which prevents misleading explanations that recommend altering features with non-causal relationships with the result. For this matter, we propose using the *structural equation model (SEM)* of the features in the procedure of generating explanations.

Case level explanations are case dependent, which means, an explanation that is useful for a customer may not be favored by another customer with the same undesirable outcome. To overcome this issue, in the proposed method, we present a set of diverse explanations (i.e. explanations that differ from the given instance in different features) to the user such that the user can decide which one to apply. Moreover, as the explanations are meant to be used by the human, the readability and understandability of the explanations are important. Therefore those explanations with a smaller number of features with different values from the given instance, are preferred [10].

The rest of the paper is organized as follows. In Sect. 2, a brief overview of the related work is presented. In Sect. 3, the method is presented. The experimental results are presented in Sect. 4. Finally, in Sect. 5 the conclusion is presented.

2 Related Work

There are already several approaches in the domain of process mining that deal with root cause analysis using the findings of a classification techniques [2,9]. The drawback of these methods is that the classification techniques are based on correlation and not causal relationships. Also, there are several works considering causal relationships among different process features at the process level [3,4,6]. Moreover, in [1] a method for generating case-level recommendations of treatments that maximize the probability of a given outcome is proposed. In this method a subset of candidate treatments that are most correlated with the outcome is extracted by applying an association rule mining technique. Then the subgroups with causal relation between treatment and outcome are identified using uplift tree. Finally, the subgroups are sorted by the ratio of the score associated to them by the uplift trees and their cost.

It is worth noting that counterfactual reasoning for explainability has been studied extensively in the field of data mining and machine learning (e.g., [8,10]).

3 Method

The general overview of the proposed method is presented in Fig. 1. First, we enrich the event log. Then, several random counterfactual instances similar to the current instance are generated. Among them, those that have a desirable outcome regarding a given threshold are selected and optimization techniques are used to make them as close as possible to the current instance. The resulting desirable counterfactual instances are ordered according to their distance with the current instance, and finally, converted into a set of explanations and presented to the people involved.

In the following, we first explain how we extract the data from the event log and then we describe the explanation generation method.

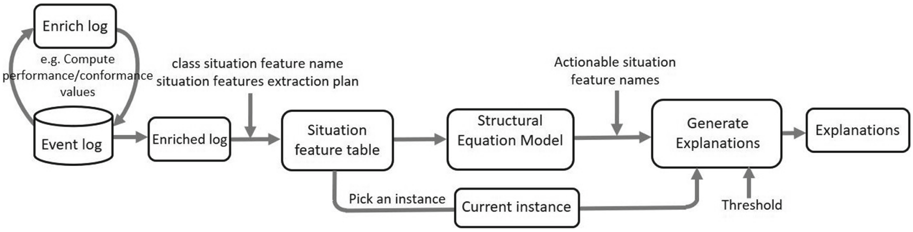
Enrich log
e.g. Compute performance/conformance values
Event log
Enriched log
class situation feature name
situation features extraction plan
Situation feature table
Pick an instance
Current instance
Structural Equation Model
Actionable situation feature names
Generate Explanations
Threshold
Explanations

Fig. 1. The general overview of the proposed method.

3.1 Situation Feature Table Extraction

Here, we mention how to extract features from the event logs of processes. An event log is a collection of traces where each trace is a collection of events. Each event indicates that specific activity has happened at a specific time for a specific case. A trace is a sequence of chronologically ordered events that belong to a specific case. Both traces and events may have several attributes. Here we assume the uniqueness of the events in the event log. We define an event log as follows:

Definition 1 (Event Log). *An event log is a set of traces where each trace is composed of a chronologically ordered sequence of events and trace attributes (if applies). Moreover, each event refers to a case, an activity, a timestamp, and event attributes (if applicable).*

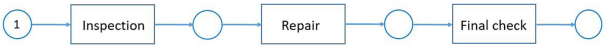

Fig. 2. The process of the repair company.

Through this paper, we consider a company that repairs a specific product as the running example. The Petri-net model of this repair company is shown in Fig. 2. Each trace in the event log is corresponding to the process of repairing one product. In the "inspection" activity, several tests are done to determine the defects of a product. Then it is repaired, and afterward "final check" is done. We know that the newer products are more complicated and harder to deal with. In this example, "model" is a trace level attribute where newer products have higher model numbers. "team size" is another trace-level attribute that indicates the number of resources involved in repairing the product. "num test" is an event-level attribute indicating the number of tests that have been done in the "inspection" activity. A snapshot of an event log is shown in Table 1. The manager of the repair company believes that in the trace with "case id" = $c1$ the "repair" activity (event $e2$) was too long and should have taken at most 500 h. He/She needs to know if it was the case, and if so, how they could had prevented it.

When we are interested in features not existing in the event log, we need to enrich the event log by adding new derived features from the event log or possibly other sources to its traces and events. For example, we can enrich the repair company event log by adding an attribute named “duration” to its events indicating the duration of that event in hours. In the repair company example, the value of the “duration” attribute can be computed by subtracting the timestamp of two consecutive events in each trace.

With respect to the time precedence of the cause and effect, we consider just the features that have been extracted from that part of a trace that has been recorded before a specific feature as possible causes of it. For example, in the repair company, extracting the data from the “final check” activity is meaningless when we want to find the features that causally influence the “duration” of the “repair” activity. So we need to extract the data from a proper prefix of a trace, which we call a *situation*. Also, we define the *situation set* of an event log as the set of all situations generated using its traces. Some of the subsets of the situation set of a given event log are more meaningful. For example the set of all the situations that end with an event with a specific activity name or the set of all traces. In the repair company example, we can have a subset of situations that end with an event whose activity name is “repair”. In this case, the situation subset would include all the prefixes of traces which include the events with the activity name “inspection” and “repair”. The situation extracted from the first trace would include the two events with “event id” $e1$ and $e2$. Let’s call this situation s_1.

An event log may have both event and trace level attributes. Moreover, it is possible to have the same attribute name in both levels. To concretely determine the attributes that we are interested in their values, we use *situation feature* notion. A situation feature refers to an attribute and an activity name (or possibly “trace”). For example in the repair company, $sf_{teamsize}$ and sf_{model} are two situation features indicating “team size” and “model” attributes in the trace level. While, $sf_{inspDuration}$ and $sf_{inspNumTest}$ are the situation features referring to the “duration” and “num test” in the “inspection” activity. Also, $sf_{repairDuration}$ refers to the “duration” of the “repair” activity. The situation feature value extraction mechanism from a given situation is as follows:

- If the situation feature refers to an attribute name and “trace”, then the value of that attribute in the trace-level is assigned to the situation feature.
- If the situation feature refers to an attribute name and an activity name, then the value of that attribute from an event with the given activity name, with the maximum timestamp is assigned to the situation feature.

For example, for the situation s_1, the value assigned to $sf_{inspDuration}$ is 71 (computed using timestamps) and the value assigned to sf_{model} is 7.

To generate explanations, we need to know the situation feature that identifies the problem (we call it *target situation feature*) and a set of descriptive situation features that are those features that may have causal effect on the problem. We call the set including the descriptive situation features and the target situation feature a *situation feature extraction plan* and denote it by $\boldsymbol{SF}$. We can look at the $\boldsymbol{SF}$ as the schema in a tabular data. For example in the repair company, as the manager believes that the duration of “repair” activity for some cases should have been shorter, the target situation feature is $sf_{repairDuration}$. Also he has considered sf_{model}, $sf_{teamsize}$, $sf_{inspNumTest}$, and $sf_{inspDuration}$ as descriptive situation features. So, in this example we have $\boldsymbol{SF}_{repair} = \{sf_{model}, sf_{teamsize}, sf_{inspNumTest}, sf_{inspDuration}, sf_{repairDuration}\}$.

Table 1. A snapshot of the event log of the repair company.

Event id	Case id	Activity name	Timestamp	Team size	Num test	Model
e1	c1	Inspection	01-04-2020T08:00:00	2	42	7
e2	c1	Repair	04-04-2020T07:00:00	2	42	7
e3	c1	Final test	28-04-2020T08:00:00	2	42	7
e4	c2	Inspection	01-05-2020T08:00:00	3	26	5
e5	c2	Repair	03-05-2020T11:00:00	3	26	5
e6	c2	Final test	19-05-2020T20:00:00	3	26	5
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Given a situation feature extraction plan, $\boldsymbol{SF}$ we can map each situation to a data point by simply extracting the values of situation features in $\boldsymbol{SF}$ using the proper mechanism. We call such a data point an *instance*. Moreover, we can define a target-dependent tabular data, called *situation feature table*, extracted from a given situation subset, as the bag of the instances extracted from the situations in a given situation subset. As an example, using $\boldsymbol{SF}_{repair}$ instance $i_{repair} = \{(sf_{model}, 7), (sf_{teamsize}, 2), (sf_{inspNumTest}, 42), (sf_{inspDuration}, 71), (sf_{repairDuration}, 577)\}$ is generated from s_1.

3.2 Explanation Generation Method

Consider an instance i in the situation feature table with an undesirable target situation feature value regarding a threshold t. For example, in the repair company the threshold is 500. W.l.o.g., in this paper, we always assume that the values lower than the given threshold are desirable. Explanations are diverse instances which are close to i and have a desirable target situation feature value. As it is vain to investigate the effect of intervention on those situation features that their value can not be altered by the user, we study the effect of changing the value of those situation features that are modifiable by the user. We call the set of modifiable situation features *actionable situation features* and denote it with $\boldsymbol{ASF}$. We define a set of counterfactual explanations for a given instances as follows.

Definition 2 (A Set of Counterfactual Explanation). *Let i be an instance for which the target situation feature value is undesirable. A set of explanations for i is a set of diverse instances that are close to i and yet differ from i in a subset of* $\boldsymbol{ASF}$ *and have a desirable result for the target situation feature.*

To generate the set of counterfactual explanations, we take the following three steps:

1. Generating candidates. We generate several candidates for the values that could had been assigned to the actionable situation features. Each candidate is a value assignment to a subset of situation features in $\boldsymbol{ASF}$. We generate candidates such that for half of them the situation feature values are selected from their distribution in the situation feature table and for the other half, they are selected randomly from their domain.

2. Predicting the value of the target situation feature. In the second step, we compute the effect of replacing the values of the situation features in the given instance with those in the generated candidates on the value of target situation feature using the SEM of the situation features. The SEM of the situation features of a situation feature table can be provided by a customer who possesses the process domain knowledge or can be inferred in a data-driven manner using several methods that already exist in the literature (e.g., [4,6]). Loosely speaking, a SEM is a set of equations that determine how to generate the observational and interventional distributions. More formally:

Definition 3 (Structural Equation Model (SEM)). *Let $\boldsymbol{SF}$ be a situation feature extraction plan, the SEM of $\boldsymbol{SF}$ is defined as $\mathcal{EQ} \in \boldsymbol{SF} \rightarrow Expr(\boldsymbol{SF})$ where for each $sf \in SF$, $Expr(\boldsymbol{SF})$ is an expression over the situation features in $\boldsymbol{SF}$ and possibly some noise N_{sf}. Moreover, the noise distributions of N_{sf} for all $sf \in \boldsymbol{SF}$ have to be mutually independent.*

We assume that $\boldsymbol{SF}$ includes all relevant situation features and there is no common hidden confounder for the situation features in $\boldsymbol{SF}$. Also, we assume that the SEM does not include any loop. In Table 2, a possible SEM for the repair company is presented.

Table 2. A possible SEM for the repair company.

$sf_{model} = N_{sf_{model}}$	$N_{sf_{model}} \sim Uniform(1, 10)$
$sf_{teamsize} = N_{sf_{teamsize}}$	$N_{sf_{teamsize}} \sim Uniform(1, 3)$
$sf_{repairDuration} = 10sf_{model} + N_{sf_{repairDuration}}$	$N_{sf_{repairDuration}} \sim Uniform(-2, 4)$
$sf_{inspNumTest} = 5sf_{model} + 3sf_{teamsize} + N_{sf_{inspNumTest}}$	$N_{sf_{inspNumTest}} \sim Uniform(-1, 2)$
$sf_{repairDuration} = 50sf_{model} + 5sf_{inspNumTest} + N_{sf_{repairDuration}}$	$N_{sf_{repairDuration}} \sim Uniform(10, 20)$

Using SEM $\mathcal{EQ}$, prediction of the class situation feature value for each candidate involves three steps *abduction*, *action*, and *prediction* [5]. We explain these steps using the repair company example.

- **Abduction.** First we need to incorporate the observed data, instance i, into the model, $\mathcal{EQ}$, and generate a *counterfactual SEM* that explains the conditions and the behavior of the system and the environment when i was happening. A *counterfactual SEM*, $\mathcal{EQ}'$, is obtained by replacing the distribution of noise terms in $\mathcal{EQ}$ with the corresponding noise distributions condition on $\boldsymbol{SF} = i$. Considering the SEM in Table 2 and i_{repair}, the equations of the counterfactual SEM $\mathcal{EQ}'_{repair}$ are: $sf_{model} = 7$, $sf_{inspNumTest} = 2$, $sf_{inspDuration} = 10sf_{model}+1$, $sf_{inspNumTest} = 5sf_{model}+3sf_{teamSize}+1$, and $sf_{repairDuration} = 50sf_{model} + 5sf_{inspNumTest} + 17$.
- **Action.** The second step is taking action toward enforcing changes in the counterfactual SEM $\mathcal{EQ}'$, regarding candidate c. The result is a SEM $\mathcal{EQ}''$ where $sf = c_{sf}$

where c_{sf} is the value assigned to sf by c if $sf \in dom(c)$ and $sf = \mathcal{EQ}'(sf)$ where sf is not in the domain of c. As an example, suppose that we are interested in predicting the value of $sf_{repairDuration}$ for the candidate $\{(sf_{teamSize}, 3)\}$. Intervention on the counterfactual SEM $\mathcal{EQ}'_{repair}$, results in replacing $sf_{teamSize} = 2$ with $sf_{teamSize} = 3$.

- **Prediction.** The third step involves using the modified SEM to predict the counterfactual value of the target situation feature by simply computing the value of targer situation feature (or its distribution) in the counterfactual SEM under the intervention. In this step, we remove those situation features from the domain of c that do not affect the target situation feature value. In the above example, computing the values of the situation features we have: $\{((sf_{model}, \perp), 7), (sf_{teamSize}, 3), (sf_{inspNumTest}, 45), (sf_{inspDuration}, 71), (sf_{repairDuration}, 592)\}$. We call such an instance a *counterfactual instance*.

3. Selecting a subset of candidates. We want explanations to be a set of diverse candidates with a small domain and a desirable predicted target situation feature value. Also we want them to be close to the given instance. To compute the distance between instances, we use L_1 metric on the normalized situation features. As mentioned in [10], using L_1 metric, more sparse explanations would be generated. For the diversity, we partition candidates with desirable predicted outcome based on their domain and then sort them in each partition according to their distance from the given instance. A set of these candidates are selected one by one from different partitions, with the priority of those partitions that have a smaller domain.

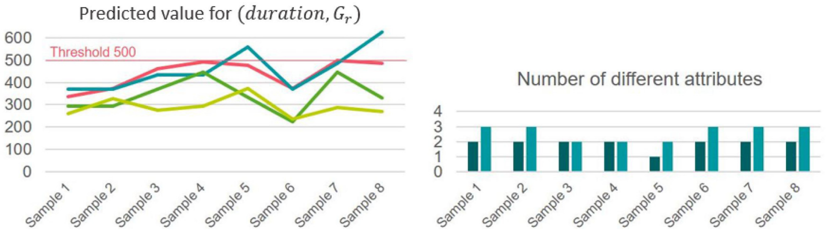

Fig. 3. The result of applying the implemented method on the synthetic event logs. (Color figure online)

4 Experimental Results

The implemented plugin is available in ProM nightly build under the name *counterfactual explanation*. In the implemented plugin, we can apply several classifiers (including Regression Tree (RT), Locally Weighted Learning (LWL), Multi-layer perceptron (NN)), as well as SEM, to predict the target situation feature value of candidates.

We applied the implemented plugin on a synthetic event log to see how different might be the explanations generated by the SEM and by a machine learning technique with the highest accuracy in terms of predicted target situation feature values and the number of situation features with different values in the given instance and the explanations. So, we did not use optimization on the selected desirable counterfactual instances.

For the synthetic event log, we have used the repair company example and i_{repair} as the instance with the undesirable target situation feature. Also, the values lower than the given threshold 500 were desirable. We considered all the descriptive situation features as actionable. We have generated 1000 traces such that the SEM of its situation feature values is the one in Table 2. Then, we generate a set of 8 explanations by generating several candidates and using the SEM in Tabble 2 to evaluate them.

We have used the classifier with the highest accuracy for predicting the value of $sf_{rapairDuration}$ on the selected candidates in the previous step. The accuracy of RT, LWL, and NN on the data were 0.818, 0.990, and 0.984, respectively. But their accuracy reduced on the counterfactual instances to 0.74, 0.77, and 0.76, respectively.

The results of applying the proposed method using SEM and three mentioned machine learning techniques are presented in Fig. 3. In left part of Fig. 3, the predicted $sf_{rapairDuration}$ of the selected desirable candidates using SEM (red line), RT (blue line), LWL (green line), and NN (light green line) are presented. In the right side of Fig. 3, the size of the domain of the selected candidates is demonstrated.

Discussion. As demonstrated in Fig. 3, there is a gap between the values predicted by the machine learning techniques and by SEM. Also, the accuracy of the classifiers predicting the value of the counterfactual instances drops dramatically. This phenomenon can be explained by the difference in their mechanism of predicting counterfactual values. Using a machine learning technique, neither the behavior of the environment nor the effect of an intervention is considered; but, the generated instance is regarded as a new instance, which may result in wrong predictions.

The difference in the number of effective situation features with different values between the given and explanations comes from the fact that machine learning techniques do not distinguish among the situation features with causal and mere correlation relationship with the target situation feature. On the other hand, using SEM the changes in the values of the situation features that have no causal relationships with the target situation feature in the counterfactual instances are simply ignored.

5 Conclusion

We have presented a method that can be used by companies to explain to their customers why they have received a specific outcome in a case-specific manner and help them to prevent the same outcome in the future. As a result, the interpretability and accountability of the companies would be boosted.

The results of the evaluations have shown that ignoring the causal relationships among the situation features may end up in explanations that suggest changing situation features with no causal effect on the class situation feature. Moreover, using a machine learning technique, regardless of its accuracy, for predicting the value of the target situation feature may result in wrong explanations or missing some of the good explanations.

References

1. Bozorgi, Z.D., Teinemaa, I., Dumas, M., Rosa, M.L., Polyvyanyy, A.: Process mining meets causal machine learning: discovering causal rules from event logs. In: ICPM (2020)
2. Ferreira, D.R., Vasilyev, E.: Using logical decision trees to discover the cause of process delays from event logs. Comput. Ind. **70**, 194–207 (2015)
3. Hompes, B.F.A., Maaradji, A., La Rosa, M., Dumas, M., Buijs, J.C.A.M., van der Aalst, W.M.P.: Discovering causal factors explaining business process performance variation. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 177–192. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_12
4. Narendra, T., Agarwal, P., Gupta, M., Dechu, S.: Counterfactual reasoning for process optimization using structural causal models. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNBIP, vol. 360, pp. 91–106. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26643-1_6
5. Pearl, J., et al.: Models, Reasoning and Inference. Cambridge University Press, Cambridge (2000)
6. Qafari, M.S., van der Aalst, W.: Root cause analysis in process mining using structural equation models. In: Del Río Ortega, A., Leopold, H., Santoro, F.M. (eds.) BPM 2020. LNBIP, vol. 397, pp. 155–167. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-66498-5_12
7. Reddix-Smalls, B.: Credit scoring and trade secrecy: an algorithmic quagmire or how the lack of transparency in complex financial models scuttled the finance market. UC Davis Bus. LJ **12**, 87 (2011)
8. Russell, C.: Efficient search for diverse coherent explanations. In: Proceedings of the Conference on Fairness, Accountability, and Transparency, pp. 20–28 (2019)
9. Suriadi, S., Ouyang, C., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Root cause analysis with enriched process logs. In: La Rosa, M., Soffer, P. (eds.) BPM 2012. LNBIP, vol. 132, pp. 174–186. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36285-9_18
10. Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: automated decisions and the GDPR. Harv. JL Tech. **31**, 841 (2017)