Data Mining and Knowledge Discovery
https://doi.org/10.1007/s10618-022-00906-4

Mining Pareto-optimal counterfactual antecedents with a branch-and-bound model-agnostic algorithm

Marcos M. Raimundo[1,2] · **Luis Gustavo Nonato**[3,4] · **Jorge Poco**[1,5]

Received: 21 May 2021 / Accepted: 1 December 2022

Abstract

Mining *counterfactual antecedents* became a valuable tool to discover knowledge and explain machine learning models. It consists of generating synthetic samples from an original sample to achieve the desired outcome in a machine learning model, thus helping to understand the prediction. An insightful methodology would explore a broader set of counterfactual antecedents to reveal multiple possibilities while operating on any classifier. Therefore, we create a tree-based search that requires monotonicity from the objective functions (a.k.a. cost functions); it allows pruning branches that will not improve the objective functions. Since monotonicity is only required for the objective function, this method can be used for any family of classifiers (e.g., linear models, neural networks, and decision trees). However, additional classifier properties speed up the tree search when it foresees branches that will not result in feasible actions. Moreover, the proposed optimization generates a diverse set of *Pareto-optimal* counterfactual antecedents by relying on multi-objective concepts. The results show an algorithm with working guarantees that enumerates a wide range of counterfactual antecedents. It helps the decision-maker understand the machine learning decision and finds alternatives to achieve the desired outcome. The user can inspect these multi-

Responsible editor: Martin Atzmueller, Johannes Fürnkranz, Tomáš Kliegr and Ute Schmid.

✉ Marcos M. Raimundo
mraimundo@ic.unicamp.br

Luis Gustavo Nonato
gnonato@icmc.usp.br

Jorge Poco
jorge.poco@fgv.br

[1] Fundação Getúlio Vargas, Rio de Janeiro, Brazil

[2] University of Campinas, Campinas, Brazil

[3] ICMC-USP, São Carlos, Brazil

[4] New York University, New York, USA

[5] Universidad Católica San Pablo, Arequipa, Peru

ple counterfactual antecedents to find the most suitable one and better understand the prediction.

Keywords Explainability · Interpretability · Machine learning · Knowledge mining · Multi-objective optimization · Crime data · Counterfactual

1 Introduction

Given a sample with an undesired outcome from a machine learning model, a **counterfactual antecedent** is a synthetic sample that achieves the desired outcome with minimal changes compared to the original sample. It helps explain the prediction of such a sample by observing what could be changed to achieve another outcome. For instance, let us suppose that a person is diagnosed with a high risk of diabetes by a machine learning technique. The attributes involved in the decision include an `Insulin` level of 285, `BMI` (Body Mass Index) of 44, `BloodPreassure` of 80, `Skin Thickness` of 34, and `Glucose` level of 114. A **counterfactual antecedent mining** system would suggest some counterfactual antecedents to swap the outcome from high to low risk.—e.g., this person should decrease his/her `BMI` from 44 to 36 or decrease `Glucose` level from 114 to 99. Counterfactual mining is also known by other names such as Actionable knowledge (Yang et al. 2003), Inverse Classification (Yang et al. 2012), Counterfactual Explanations (Wachter et al. 2018), and other variations. People use this technique to recommend medical treatments (Yang et al. 2012; Krause et al. 2016), marketing strategies to retain customers (Yang et al. 2003), and personal profile changes to have credit approved (Ustun et al. 2019). Recently, counterfactual antecedent mining has become popular in explaining the behavior of learning models (Wachter et al. 2018).

Counterfactual mining methods explore counterfactual antecedent with a single (Krause et al. 2016) or multiple (Wachter et al. 2018) feature changes to achieve the desired outcome. However, the methods capable of finding a diverse set of counterfactual antecedents (Karimi et al. 2020; Ustun et al. 2019; Mothilal et al. 2020) rely on a single objective function. Although this function should capture the cost of making changes to create a counterfactual antecedent, it is hard to design a single objective function that fulfills the preferences of a user (Rudin 2019). To clarify the importance of having multiple counterfactual antecedents, Fig. 1 shows the original sample (see the Orig column) and a set of counterfactual antecedents (C1, . . ., C12 columns) for a diabetes example. For instance, to change the patient's model's outcome from high to low risk of diabetes:

- C1 suggests increasing `Insulin` from 285 to 468, decrease `BMI` from 44 to 42, and decrease `Glucose` from 114 to 113.
- C8 suggests increasing `Insulin` from 285 to 468, decrease `BMI` from 44 to 42, and decrease `SkinThickness` from 34 to 24.
- C10 gives another alternative, namely, to increase `Insulin` from 285 to 416, decrease `BMI` from 44 to 42, and decrease `SkinThickness` from 34 to 12.

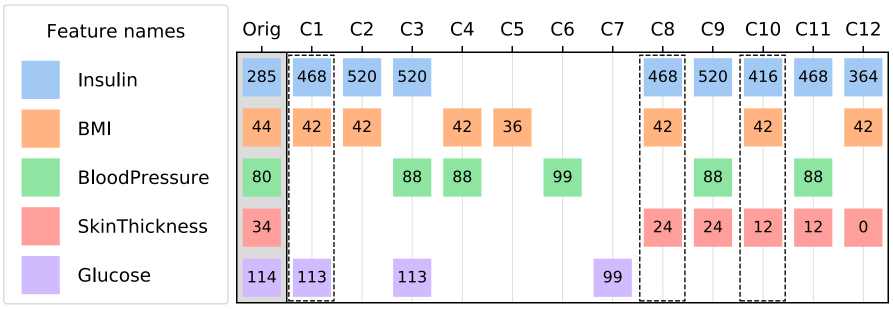


Fig. 1 A multi-objective enumeration of counterfactual antecedents that reduces the risk of diabetes. Orig column is the original sample and the columns C1 to C12 are counterfactual antecedents

This example shows that it is possible to explore changes in distinct features (C1 vs. C8) and changes in the values of the same features (C8 vs. C10). This diversity is achievable by using multiple objectives, especially using multi-objective optimization. Moreover, most existing methods usually operate on a limited family of classification methods (Ustun et al. 2019; Mothilal et al. 2020); thus, making it unfeasible to use them with some types of classifiers. Therefore, there is a need for methods to operate in any family of classifiers (model-agnostic method) and capable of mining a diverse set of counterfactual antecedents with multiple objectives. Worth mentioning that counterfactual antecedents in correlation-based machine learning models do not imply a causal relationship (Chou et al. 2022). Nevertheless, the diversity of options promoted by the proposed methodology allows experts to explore and find proper counterfactual antecedents that are the most appropriate for an application domain.

This paper's main contribution is a methodology called MAPOCAM (Model-Agnostic Pareto-Optimal Counterfactual Antecedent Mining), which relies on a tree-based search to enumerate all Pareto-optimal counterfactual antecedents using a given set of objective functions. The proposed methodology only requires the monotonicity of the objective functions to prune branches that do not improve the solution's quality. Moreover, MAPOCAM can also explore optional conditions to improve performance. For example, when the decision function (that evaluates if we achieve the result) is monotonic (an increase in the value of a feature increases the probability of achieving the target outcome), the search can be sped up. It happens because it is possible to foresee branches that take unfeasible actions; thus, they can be pruned.

In summary, the contributions of this work are:

– A model agnostic methodology that enumerates all counterfactual antecedents considering the trade-off of multiple objective functions;
– A mechanism to create an overview of how much the variables need to change to give rise to counterfactual antecedents;
– A comprehensive set of experiments that explore multiple machine learning models—logistic regression, decision trees, and neural networks—to validate our methodology on real applications;
– An in-depth experiment using urban and social-economic variables to explain how those variables can be handled to reduce crime incidents.

2 Related work

Counterfactual antecedent mining is approached in the literature under various names depending on the application: Actionable knowledge (Cui et al. 2015; Yang et al. 2007; Lv et al. 2018; Lu et al. 2017; Yang et al. 2003), Actionable Plans (Lu et al. 2017), Actionable recourse (Ustun et al. 2019), Actionable feature tweaking (Tolomei et al. 2017), Inverse Classification (Yang et al. 2012; Aggarwal et al. 2010), Counterfactual Explanations (Wachter et al. 2018), Why-not questions (Gao et al. 2015; He and Lo 2012; Chen et al. 2015). The coined name *counterfactual antecedent* tries to homogenize those terms. The term counterfactual already explains the nature of the method. We create an antecedent (distinct from the original sample) that achieves the consequent (the target outcome, which is fixed).

Actionable knowledge methods (plans/recourse/feature tweaking) usually aim to mine counterfactual antecedents for specific machine learning models, exploring their structure to find counterfactual patterns. They usually explore a single counterfactual antecedent with a single objective function, differing mainly on the optimization procedure. Heuristic approaches typically require an already fitted decision tree or ensemble of trees and use greedy algorithms to find groups of samples on which actions can change the outcome (Yang et al. 2003, 2007; Subramani et al. 2016). Heuristics can also search on every positive path of a decision tree (i.e., changes that result in a swap of outcomes), searching for positive paths that change the ensemble's outcome with the lowest cost (Tolomei et al. 2017). Another heuristic consists of making greedy moves (i.e., changes that increase the probability) on a KNN classifier (Yang et al. 2012). Other approaches explore counterfactual antecedents with more theoretical support but still use speed-up heuristics. For instance, some approaches explore A^*-like search to find an optimal change to additive tree models using a heuristic that considers the probability achieved by the action and the action's cost (Lu et al. 2017). Some works use the desired state distance as a heuristic to refine an A^*-like search (Lv et al. 2018). Mixed linear-integer modeling (enforcing optimality guarantees) has also been employed to swap between leaves of the trees, searching for a feature change with lower cost (Cui et al. 2015). Those formulations can also search through a set of actions to find a counterfactual antecedent on linear classifiers (Ustun et al. 2019).

Inverse classification defines a set of features (also interpreted as actions) that achieves the desired class. An example of inverse classification consists of searching for a set of feature changes whose correspondent subset of samples has a high Gini index (Aggarwal et al. 2010). Therefore, the inverse classification does not rely on a machine learning model; it mines sets of features that achieve the target outcome.

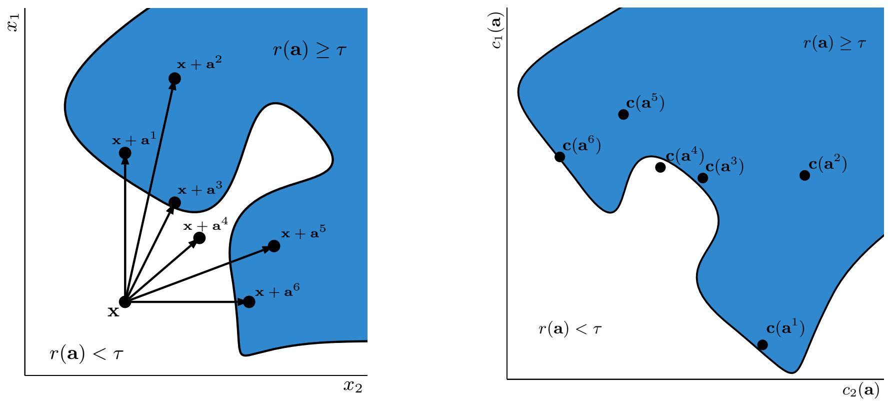
Similarly, the why-not questions approach focuses on determining the absence of an object in a database query. Knowing a user preference (usually expressed using a weight vector), a query returns a set of objects. If the user is expecting another answer, then it can use a quadratic optimization to find a weight vector with the desired object (Gao et al. 2015); creating a sampling procedure to find new weight vectors with the desired object (He and Lo 2012); or using a tree-based bound and prune algorithm (Chen et al. 2015). This family of methods is quite limited since it only uses a weight vector to classify the samples, similar to linear classifiers.

Counterfactual explanations are usually capable of exploring broader families of machine learning models. Convex optimization can be applied to linear classifiers and neural networks to find a new outcome that is as close as possible to a new target and still close enough to the original sample (Wachter et al. 2018), and add constraints to ensure plausibility (Artelt and Hammer 2020). The method that uses the Nelder-Mead optimization finds counterfactual antecedent on black-box models with an iterative approach to determine how close it is to the original sample (Grath et al. 2018). Then it uses differentiable approximations of tree ensembles to keep the convexity of the problem (Lucic et al. 2019).

Other approaches include using the density of samples to create a graph and finding a path to a counterfactual that is both feasible and actionable (Poyiadzi et al. 2020). Some methods use binary search to verify the counterfactual antecedent's satisfiability on a model-agnostic oracle of logic formula's representation of machine learning models (Karimi et al. 2020). Generative adversarial net has also been used to synthesize counterfactual antecedents (Yang et al. 2021). Some techniques rely on agnostic tree-based searches to find a single counterfactual explanation (Kaffes et al. 2021) via quadratic optimization over the nodes of an oblique decision tree (Hada and Carreira-Perpiñán 2021). Sampling and clustering the feature space to select the closest counterfactual antecedent for each cluster is also an alternative (Wellawatte et al. 2022).

There are also strategies to explore multiple counterfactual antecedents: (i) mixed-integer model can exclude the combination of features that already find a counterfactual antecedent (Ustun et al. 2019); (ii) convex optimization searches for finding multiple solutions at once combined with distance metrics between counterfactual antecedents to induce diversity (Mothilal et al. 2020); or (iii) iterative approaches that find new solutions with minimum distance from the already found solution, i.e., using l_0 larger than one to have at least one new change (Karimi et al. 2020). The main limitation of those methods is the use of distances like l_0, which does not allow exploring the same set of features but increasing one and decreasing others. Furthermore, the use of l_2 distance might induce counterfactual antecedents with many features slightly modified. Changing all features simultaneously is not desired in many applications. In contrast, multi-objective optimization allows the creation of diversity without relying on a diversity-inducing function. A proper representation of the Pareto-front promoted by multi-objective optimization will result in solutions with different trade-offs between objectives (as shown in Fig. 1), thus generating counterfactual antecedents sufficiently different to promote diversity.

Our method has a set of characteristics not concomitantly present in the methods described above: (i) it is model-agnostic; (ii) it is capable of exploring multiple objective functions; and (iii) it is capable of exploring multiple counterfactual antecedents, properties that are provided by exploring multiple trade-offs by using multi-objective optimization. Moreover, our approach has solid theoretical guarantees, and it is capable of dealing with binary features—quality not present in methods that rely on convex optimization because this type of optimization is not built to deal with integer value directly.


x_1
x_2
$r(\mathbf{a}) \geq \tau$
$r(\mathbf{a}) < \tau$
$\mathbf{x}$
$\mathbf{x}+\mathbf{a}^1$
$\mathbf{x}+\mathbf{a}^2$
$\mathbf{x}+\mathbf{a}^3$
$\mathbf{x}+\mathbf{a}^4$
$\mathbf{x}+\mathbf{a}^5$
$\mathbf{x}+\mathbf{a}^6$
$c_1(\mathbf{a})$
$c_2(\mathbf{a})$
$\mathbf{c}(\mathbf{a}^1)$
$\mathbf{c}(\mathbf{a}^2)$
$\mathbf{c}(\mathbf{a}^3)$
$\mathbf{c}(\mathbf{a}^4)$
$\mathbf{c}(\mathbf{a}^5)$
$\mathbf{c}(\mathbf{a}^6)$

(a) Feature space **(b)** Objective space

Fig. 2 Representation of feature space (**a**) and objective space (**b**), taking two features and two objectives. The blue area represents the space where the samples are classified with the target outcome. In the feature space (**a**), the goal is to find a plan $\mathbf{a}^i$ that makes $\mathbf{x}$ achieve the desired outcome $r(\mathbf{x}+\mathbf{a}^i) \geq \tau$, thus $\mathbf{a}^i$ is a feasible action (e.g., $\mathbf{a}^1$, $\mathbf{a}^2$, $\mathbf{a}^3$, $\mathbf{a}^5$, $\mathbf{a}^6$). In the objective space (**b**), the goal is to exclude all feasible actions $\mathbf{a}^i$ that have other actions $\mathbf{a}^j$ with lower (thus better) values of objectives $\mathbf{c}(\mathbf{a}^i) \geq \mathbf{c}(\mathbf{a}^j)$ (e.g., $\mathbf{a}^2$ has higher values than $\mathbf{a}^1$ and $\mathbf{a}^3$). The feasible actions with no other feasible action with higher values are called Pareto-optimal (e.g., $\mathbf{a}^1$, $\mathbf{a}^3$, $\mathbf{a}^6$)

3 Proposed method

This section defines the goals and premises of our methodology, a branch-and-bound algorithm, and details the proposed methodology's essential properties.

3.1 Background

In the following, we consider a binary classification problem where each sample comprises a vector of features $\mathbf{x} \in \mathcal{R}^d$, where d is the number of features, and a binary outcome $y \in \{0, 1\}$. We represent the classifier by a **decision function** $r(\bullet)$ that is 1 when $r(\mathbf{x}) \geq \tau$ and 0 when $r(\mathbf{x}) < \tau$, where τ is given a threshold. For the sake of simplicity, 1 will be the target outcome.

An **action** $\mathbf{a} = [a_1, \ldots, a_d]$ on a sample $\mathbf{x}$ is a set of **changes** $a_i \in \mathbb{R}$, $\forall i \in \{1, \ldots, d\}$ that creates a new synthetic sample $\mathbf{x} + \mathbf{a}$. Each change a_i can be positive (increase the value of the feature: $x_i + a_i \geq x_i$) or negative (decrease the value of the feature $x_i + a_i \leq x_i$), but not both for the same feature. Without losing generality, we will focus only on non-negative changes $a_i \geq 0$, $\forall i \in \{1, \ldots, d\}$[1] In this context, a feasible action $\mathbf{x} + \mathbf{a}$ that achieves the target outcome $r(\mathbf{x} + \mathbf{a}) \geq \tau$ is called a counterfactual antecedent. Figure 2a illustrates these concepts.

Definition 1 - *Feasible action.* An action $\mathbf{a} \in \mathbb{R}^d$ belongs to the feasible set of solutions $\mathcal{A}$ if and only if it achieves the desired outcome $r(\mathbf{x} + \mathbf{a}) \geq \tau$.

[1] As Ustun et al. (2019) presented, we can apply negative or positive changes, and any of the following properties would hold. However, we simplify the notation to aid clarity.

When an **objective function** $c(\bullet) \in \mathbb{R}$ is optimized, it makes an action $\mathbf{a}$ better fulfill the decision-maker's preferences. We consider, without losing generality, that we want to minimize an objective function $c(\bullet) \in \mathbb{R}$. Thus, we search for actions with the lowest value of the objective function. But here, we propose using **multiple objective** functions $c_1(\bullet), \ldots, c_m(\bullet)$ where each objective function $c_i(\bullet),\ i \in \{1, \ldots, m\}$ account for a specific goal defined by the user. When the objectives are conflicting, it might be impossible to find a single feasible action with the best performance since one objective's optimization might negatively impact others. However, it is possible to define a partial order in multiple objectives and define a dominant action—an action with all objectives better than other actions.

Definition 2 - *Partial order.* Ordering relation on partially ordered sets occurs when all components are ordered in the same sense. We use the symbols $\preceq$ and $\succeq$ to describe the ordering relations on partially ordered sets; for example, $\mathbf{x} \preceq \mathbf{y}$ is equivalent to $x_i \leq y_i,\ \forall i \in \{1, \ldots, m\}$.

Definition 3 - *Dominant action.* A feasible action $\mathbf{a} : r(\mathbf{x} + \mathbf{a}) \geq \tau$ dominates $\mathbf{a}'$ if and only if $\mathbf{c}(\mathbf{a}) \preceq \mathbf{c}(\mathbf{a}')$.

The concept of a dominant action used here is based on the dominance concept in multi-objective optimization where given two solutions $\mathbf{a}$ and $\mathbf{a}'$, $\mathbf{a}$ dominates $\mathbf{a}'$ iff $c_i(\mathbf{a}) \leq c_i(\mathbf{a}'), \forall i \in \{1, \ldots, m\}$ and exists j such that $c_j(\mathbf{a}) < c_j(\mathbf{a}')$ (Miettinen 1999). A feasible action is called Pareto-optimal action when no other feasible action dominates it. Figure 2b illustrates this concept mathematically defined below.

Definition 4 - *Pareto-optimal action* (Miettinen 1999). Consider an objective function vector $\mathbf{c}(\bullet) : \mathbf{R}^d \to \mathbf{R}^m$ that we want to minimize, and a feasible set of solutions $\mathcal{A}$. An action $\mathbf{a}^*$ is Pareto-optimal iff there is no action $\mathbf{a} \in \mathcal{A}$ that dominates $\mathbf{a}^*$.

Finally, another important concept in our context is monotonicity.

Definition 5 - *Monotonicity w.r.t. a partial order*. Given any two actions $\mathbf{a}$, $\mathbf{a}' \in \mathbb{R}^d$ such that $a_i \geq a'_i, \forall\ i \in \{1, \ldots, d\}$, a function vector $\mathbf{f}(\bullet) : \mathbf{R}^d \to \mathbf{R}^m$ is monotone if only if $f_j(\mathbf{a}) \geq f_j(\mathbf{a}'),\ \forall j \in \{1, \ldots, m\}$.

Monotonicity helps our branch-and-bound (B&B) algorithm to foresee if a search space is fruitful or not, avoiding unnecessary computation. Monotonicity is a requirement for all objective functions; however, it is optional for the decision functions (classifiers), although a monotone decision function improves the algorithm's performance. Also, since there are no dominance relations among Pareto-optimal actions, we propose a branch-and-bound algorithm to find all Pareto-optimal actions because we consider them equivalently good.

3.2 Model-agnostic Pareto-optimal counterfactual antecedents mining

The proposed algorithm is a tree-based branch-and-bound (Lawler and Wood 1966) algorithm called MAPOCAM (Model-Agnostic Pareto-Optimal Counterfactual Antecedents Mining); we detail it in Algorithm 1. In this recursive algorithm, each parameter's call is a node that can create other nodes with a branching procedure or avoid exploring subsequent nodes on a pruning procedure. In essence, MAPOCAM can examine every combination of changes. However, it cleverly bounds nodes if it preemptively knows that no subsequent branching will improve the optimal set of solutions. This bounding procedure is the cornerstone of the algorithm's efficiency.

Algorithm 1 Model-Agnostic Pareto-optimal Counterfactual Antecedents Mining

```
Require: A sample $\mathbf{x}$, an objective function $c(\bullet)$, a decision rule $r(\bullet)$, a threshold $\tau$, and a number of allowed changes $k$.
1: procedure ENUMERATE($\mathbf{a}$, $\mathcal{D}$)
2:     if $|i : a_i \neq 0 \; \forall i \in \mathcal{D}| > k$ or $\exists \, \mathbf{a}' \in \mathcal{A} : c(\mathbf{a}) \succeq c(\mathbf{a}')$ then
3:         return
4:     end if
5:     if $r(\bullet)$ is monotone and $r(\mathbf{x} + \overline{\mathbf{a}}^*) < \tau$ then
6:         return
7:     end if
8:     if $r(\mathbf{x} + \mathbf{a}) \geq \tau$ then
9:         $\mathcal{A} = \mathcal{A} \cup \{\mathbf{a}\}$
10:        return
11:    end if
12:    $i$ =SELECT_FEATURE($\forall i : i \notin \mathcal{D}$)
13:    for $\forall \mathbf{a}' : a_i' \geq a_i$ do
14:        ENUMERATE($\mathbf{a}'$, $\mathcal{D} \cup \{i\}$, $\mathcal{A}$)
15:    end for
16: end procedure
17: $\mathcal{A} = \{\}, \mathcal{D}^0 = \{\}$
18: $\mathbf{a}_i^0 = 0 \; \forall i \in \{1, \ldots, d\}$.
19: ENUMERATE($\mathbf{a}^0$, $\mathcal{D}^0$)
20: return $\mathcal{A}$
```

A **node** consists of an **action a** : $a_i \geq 0, \; \forall i \in \{1, \ldots, d\}$ and a set of fixed features $\mathcal{D}$ (line 1). The **branching** procedure consists of selecting a non-fixed feature $i \notin \mathcal{D}$ (line 12), calling the recursive function (line 14) with different actions' values $a_i' \geq 0$ for that feature i that now is fixed $\mathcal{D}' \equiv \mathcal{D} \cup i$ for the subsequent calls. All features with no made decision are considered null $a_i = 0, \; \forall i \notin \mathcal{D}$ and might have their value altered in the subsequent branching.

Since any recursive call (line 14) will always increase the magnitude of the action $a_i \geq 0$, we can use the monotonicity function to stop a current call to make other recursive calls. Thus, the **bounding step** consists of inspecting any branch that will not improve the optimal set of solutions; thus, the node should not go deeper. This algorithm works with three possible conditions:

1. bounding any node that has more than k changes: $|i : a_i \neq 0 \; \forall i \in \mathcal{D}| > k$ (line 2). Thus, since any other recursive call will increase the number of changes, we can stop the call at this point;
2. bounding any node that will not improve the objective functions $(\mathbf{a}', \mathcal{D}') : \mathbf{c}(\mathbf{a}) \preceq \mathbf{c}(\mathbf{a}')$, $\mathbf{a} \in \mathcal{A}$ (from Definition 3 and Corollary 1 in Appendix A.1) when we already have a set of feasible solutions $\mathcal{A}$ (line 2). Thus, since any other recursive call will increase the action size, and we impose monotonicity in the objective functions,

any recursive call in this node will worsen the objective functions, creating a dominated solution;
3. bounding nodes that any subsequent bounding will not generate a feasible solution (line 5). Since any other recursive call will increase the size of the action, if we have monotonicity in the decision function, we can follow Definition 9 and see that a maximal achievable action[2] $\overline{\mathbf{a}}^*$ is unfeasible. Any recursive call in this node will also be unfeasible.

In Appendix A.1, we show two essential properties of the bounding procedure. First, by enforcing monotonicity in the objective functions, it is possible to see that any subsequent branching of a node with dominated action will be dominated (Corollary 1). Second, by enforcing monotonicity on the classifier, it is possible to prove when all subsequent node branches are unfeasible because the maximal achievable action $\overline{\mathbf{a}}^*$ is not feasible (Theorem 1). In Appendix A.1, we also prove two properties of the algorithm: First, Algorithm A.1 finds all Pareto-optimal solutions (Theorem 2). Second, the time complexity of Algorithm 1 is $T(d, k) = \mathcal{O}((bd)^k)$ (Theorem 3), where d is the number of variables, b is the maximum number of possible states of each variable, and k is the maximal number of allowed changes.

Worth mentioning that the procedure `SELECT_FEATURE` in Algorithm 1 is responsible for indicating the next feature to be explored. Ideally, the procedure `SELECT_FEATURE` will return the feature that most likely helps to find a feasible solution. We give further details in Appendix C.

3.3 Objective functions

The monotonicity of the objective functions is the main property required by the algorithm—in Appendix A.2, we define some basic properties of these monotonic functions. To define the monotonic objective functions used in this work, we need to know that we have a set of N samples $\mathbf{x}^i \in \mathcal{R}^d,\ i \in \{1, \ldots, N\}$, where d is the number of features. We also define the working sample $\mathbf{x} \in \mathcal{R}^d$ that we want to obtain a different outcome. We present three objective functions: percentile change, feature change, and the number of changes.

Definition 6 - *Maximal percentile change (MPC).*

First, let's define the percentile for the feature value z_j:

$$l_j(\mathbf{z}) = \frac{|\{i|x_j^i \geq z_j, \forall i \in \{1, \ldots, N\}|}{N} \times 100 \tag{1}$$

where $| \bullet |$ is the cardinality of the set, x_j^i is the j-th feature of the i-th sample.

Given that, a percentile change of an action $\mathbf{a}$ for the feature j is the absolute difference in percentiles $|l_j(\mathbf{x}) - l_j(\mathbf{x}+\mathbf{a})|$. Thus, the maximal percentile change $c(\mathbf{a})$ of an action $\mathbf{a}$ is $\max(|l_j(\mathbf{x}+\mathbf{a}) - l_j(\mathbf{x})|,\ j \in \{1, \ldots, d\})$.

[2] This action is a computationally efficient way of inspecting all possible feasible actions from future branches. The mathematical concepts are better explained in Appendix A.1.

Definition 7 - *j-th feature change*. The feature change for the feature j consists of the magnitude of an action $\mathbf{a}$ for the feature j: $c_j(\mathbf{a}) = a_j$.

Definition 8 - *Number of changes*. Consists on counting the number of changes (non-zero values) of an action $\mathbf{a}$: $c(\mathbf{a}) = |\{a_j | a_j \neq 0, \forall j \in \{1, \ldots, d\}|$.

Theorem 5 in Appendix A.2 recognizes that any Pareto-optimal solution of the monotonic objective function of any nature will be a Pareto-optimal solution using the feature changes as objectives. Thus, resorting to feature changes as objectives create a poll of counterfactual antecedents that will satisfy any preference, only needing to filter other sets of objectives to find the desired antecedents.

3.4 Monotonicity on classifiers

Another characteristic that a classifier can have is its monotonicity. This property guarantees that if a feature increases, the outcome of the classifier (usually the probability) will increase or decrease, but never both simultaneously; the same occurs when the feature decreases. This property helps avoid branches with no feasible actions since we can foresee the behavior of the classifier. Despite being a useful property, it is an optional property to MAPOCAM.

In Appendix A.3, we show that some linear classifiers and ensembles of monotone classifiers are monotone. Furthermore, in Appendix B, we depict how a similar property for decision trees allows foreseeing that some decisions in the search make the goal unattainable.

3.4.1 Enforcing monotonicity

In addition to naturally monotonic methods (e.g., logistic regression), some other methods can be adapted to be monotonic, even though their families (e.g., decision trees and neural networks) are not monotonic. Those methods enforce the machine learning model to have a monotone behavior w.r.t. to any desired feature—increasing that feature will always increase (or decrease) the output. In this category, we have LightGBM[3] (Ke et al. 2017) and Gupta's research (Gupta et al. 2016).

3.4.2 Dealing with non-monotone classifiers

Despite being possible to enforce/guarantee the monotonicity of the classifiers/decision functions, it is also possible to anticipate that some high-quality classifiers cannot hold monotonicity. However, it is possible to use Algorithm 1 to find Pareto-optimal actions for non-monotone classifiers. We can see that conditions on Lines 2 and 8 hold despite the lack of monotonicity in the classifier as long as the objective function is monotone: any actions with higher costs are not interesting. However, the second condition in Line 5 cannot hold since it can exist a feasible action $\mathbf{a}' : \mathbf{a} \succeq \mathbf{a}' \succeq \overline{\mathbf{a}}^*$ in non-monotone classifiers. Given that, the first condition ($r(\bullet)$ is monotone) in Line 5 ensures the second condition ($r(\mathbf{x} + \overline{\mathbf{a}}^*) < \tau$) only being applied if the decision rule is monotonic.

[3] It can be set with the monotone_constraints parameters at lightgbm.readthedocs.io/en/latest/Parameters.html.

3.5 Counterfactual antecedent index

Let us suppose the method generated a set of solutions $\mathcal{A} \equiv \{\mathbf{a}^1, \ldots, \mathbf{a}^{|\mathcal{A}|}\}$. An exciting property would be to observe how much each feature $i \in \{1, \ldots, d\}$ helps find a counterfactual antecedent. To do that, we propose a counterfactual antecedent index (CA_i) for each feature i:

$$\text{CA}_i = \frac{1}{\nu_i} \sum_{\mathbf{a}^j \in \mathcal{A}} \frac{a_i^j - \max a_i}{\max a_i} \times \nu_j \tag{2}$$

where $\nu_i = |a_i^j : a_i^j \neq 0, \mathbf{a}^j \in \mathcal{A}|$ is the number of counterfactual antecedents with a non-zero value in the feature i, and $\nu_j = |a_i^j : a_i^j \neq 0, j \in \{1, \ldots, d\}|$ is the number of non-zero values in the action $\mathbf{a}_i$.

With this definition, it is possible to estimate how much, on average, a variable would need to change to create a counterfactual antecedent. The index penalizes features that need help from other features by multiplying by the number ν_j. If this number is bigger than 1, this variable would have to surpass its limit to create a counterfactual antecedent, indicating the need for other features to find a counterfactual antecedent.

This index helps summarize many counterfactual antecedents and aids the user in understanding the importance of each feature.

4 Experiments

The following experiments aim at evaluating MAPOCAM in two aspects: (Experiment 1) the importance of multiple antecedents to give broader reasoning of counterfactuals, as well as showing the importance of exploring counterfactuals on different machine learning models; and (Experiment 2) the correctness and computational performance of the method on monotonic and non-monotonic classifiers with single and multiple objectives.

To evaluate the capability of finding counterfactual antecedents, we consider the three credit datasets used in (Ustun et al. 2019): German Credit (labeled as **german**), Give Me Some Credit (labeled as **giveme**), and Taiwan Credit (labeled as **taiwan**); and two datasets from other domains: Student Performance (labeled as **student**) (Dua and Graff 2017) and Pima Indians Diabetes (labeled as **pima**) (Smith et al. 1988). These datasets have 1000, 120269, 30000, 395, and 769 samples and 27, 10, 17, 30, and 8 features. The credit datasets have socioeconomic and credit/bank history as independent variables, the student dataset have socioeconomic family-related and personal variables, and the pima dataset has health indicator variables. The dependent variable indicates whether a person deserves credit (credit datasets) if a student has a good grade (student dataset) and if a patient has diabetes (pima dataset). Given this, a counterfactual objective is to provide options for changing the model outcomes. That is, for a person to be classified as low-risk of default (credit dataset), suggest changes to a student's profile to be classified as a high-grade student (student dataset), and suggest modifications to a patient's profile to be classified as a low-risk diabetic (pima dataset). It is worth mentioning that it is always recommended to consult an expert to give a practical and theoretical basis that such feature changes would change the outcome. MAPOCAM implementation using the Python programming language is available at github.com/visual-ds/cfmining.

4.1 Experiment 1: Pareto-optimal counterfactual antecedents

In this experiment, we create Pareto-optimal counterfactual antecedents for a single sample of *credit* to show the capability of MAPOCAM to work on a wide range of scenarios. In Sect. 4.1.1, we evaluate how MAPOCAM behaves in extracting counterfactual antecedents on monotone classifiers., Sect. 4.1.2 discusses how some important classifiers (such as decision trees and neural networks) cannot hold the monotonicity property and how different the mined counterfactual antecedents are.

We limit the counterfactual antecedents to change at most three features for any case in this experiment. We choose value three because it can attain various antecedents without demanding excessive computational effort. Figures 3, 4, 7, 8, and 9 depict the enumeration of counterfactual antecedents. Following the same pattern of Fig. 1, the first column (Orig column) shows the original values inside the squares, and the following columns (C1, C2, and others) illustrate a set of counterfactual antecedents with the new value inside the squares. We can observe the changes in sparsity (only a few squares per column); we also use colors to differentiate the features easily. We sort the counterfactual antecedents by MPC cost: lower cost on the left and higher on the right.

4.1.1 Monotone classifiers

In this experiment, we show the algorithm's capability of enumerating a representation of all Pareto-optimal solutions. This experiment uses the *credit* dataset to train an l_2 penalized Logistic Regression using Scikit-learn (Pedregosa et al. 2011) and selects a sample to extract counterfactual antecedents using MAPOCAM. We use two multi-objective formulations to show the behavior in more than one scenario: (i) minimize the action for every feature; (ii) minimize the maximal percentile change (MPC) cost and the number of changes—Figs. 3 and 4 show the results for formulation (i) and (ii), respectively.

Figure 3 shows a vast set of counterfactual antecedents allowing multiple combinations. Despite having the lowest MPC cost, increasing the balance above \$500 in the savings account suggested in (C1) might be hard for the user. However, other antecedents might be more accessible to the user: (C9) increasing savings account balance above \$100 and closing a loan in other banks; or (C10) increasing checking account above \$200 and signing in to have a telephone. Our technique, using this formulation, works as an enumeration of every possible combination of features' changes without depicting solutions with an even more profound feature change. This modeling gives users a wide range of possibilities to find the most suitable counterfactual antecedent.

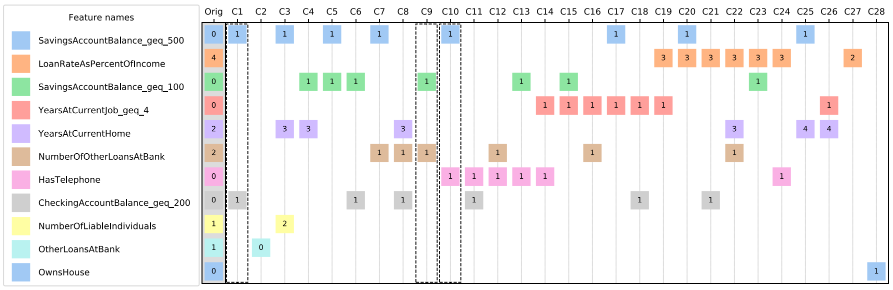
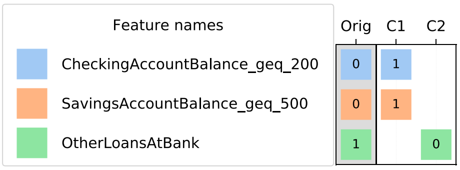


Fig. 3 Enumeration for Logistic Regression of counterfactual antecedents with Pareto-optimal values when each feature is considered as an objective function. Orig column is the original sample and the columns C1 to C28 are counterfactual antecedents

Fig. 4 Enumeration for Logistic Regression of counterfactual antecedents with Pareto-optimal values when the MPC cost and the number of changes are the objectives. Orig column is the original sample and the columns C1 and C2 are counterfactual antecedents



Using the second formulation, MAPOCAM gets two counterfactual antecedents (See Fig. 4). (C1) shows that increasing a checking balance above $200 simultaneously with increasing savings above $500 is a counterfactual antecedent with the lowest MPC cost. Meanwhile, (C2) shows that ending loans at other banks is an alternative with higher MPC costs but with a single change. Both antecedents represent the designer's preferences, also enumerated in Fig. 3. It shows that no matter user preferences, the resulting trade-offs can be filtered from the enumeration using feature changes as objectives, as said in Sect. 3.3.

4.1.2 Non-monotone classifiers

In this experiment, we explore the capability of MAPOCAM to find counterfactual antecedents on non-monotonic classifiers. Figure 5 shows the impact of increasing the variables `LoanAmount`, `LoanDuration`, and `Age` on the probability of having the credit granted for an adjusted multilayer perceptron (MLP). Here, MLP does not have monotonic behavior since the probability might increase or decrease for the same variable.

Figure 6 shows an adjusted tree with a depth of three. To understand its non-monotonicity we rely on a sample with the following attributes: `CriticalAccountOrLoansElsewhere`=1, `OtherLoansAtBank`=1, `Age`=41, `LoanDuration`=16, and `LoanAmount`=7000. If we change `CriticalAccountOrLoansElsewhere` from 1 to 0, its probability of getting credit will decrease from 45.5% to 14.3%. However, if we change `LoanAmount` from 7000 to 6000, changing `CriticalAccountOrLoansElsewhere` from 1 to 0 would cause the probability to increase from 45.5% to 76.1%. This change of behavior, depending

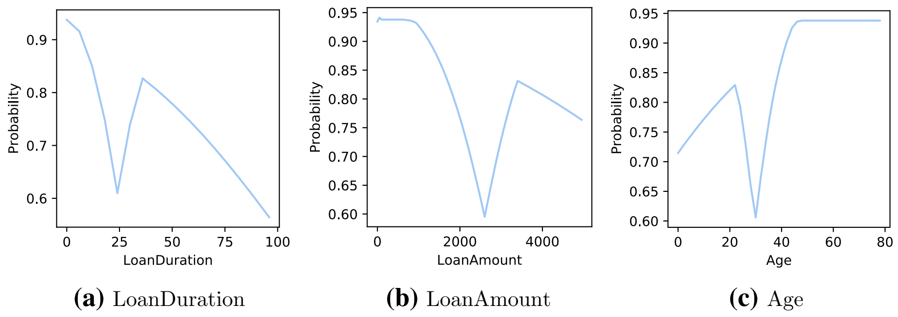


Fig. 5 Representation of impact of increasing each variable in the probability of having credit granted in a multilayer perceptron

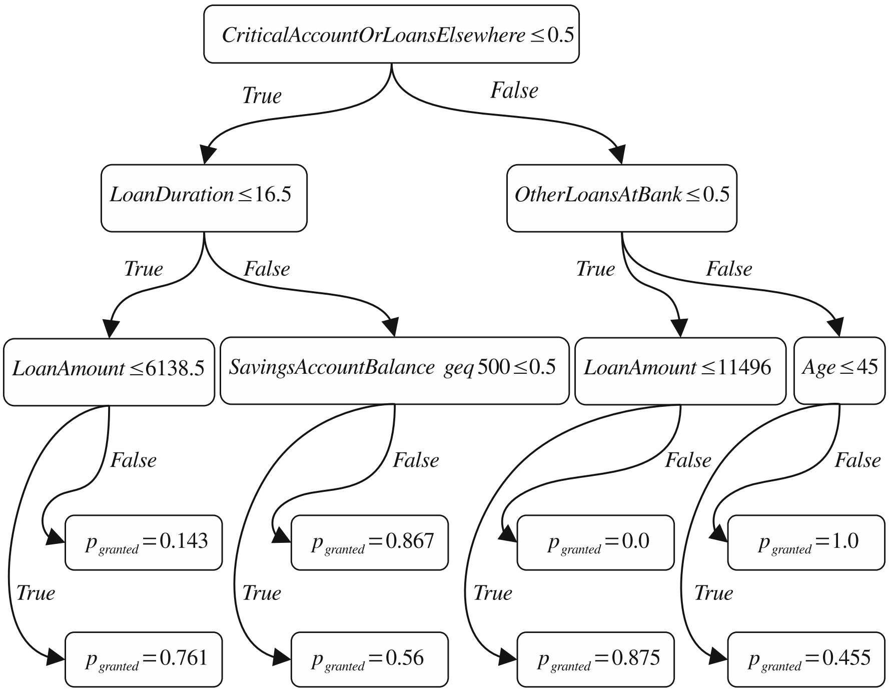


Fig. 6 Representation of a decision tree for the German dataset with a depth of 3

on other attributes, characterizes `CriticalAccountOrLoansElsewhere` as a non-monotone attribute.

Both examples show relevant machine learning models that do not preserve the monotonicity property. However, they still demand a robust methodology to find counterfactual antecedents for those complex models. Selecting the same sample investigated in Figs. 3 and 4, we used MAPOCAM to enumerate counterfactual antecedents to show how different they are and how various machine learning models uncover distinct counterfactual antecedents. We chose three machine learning models: (a) the

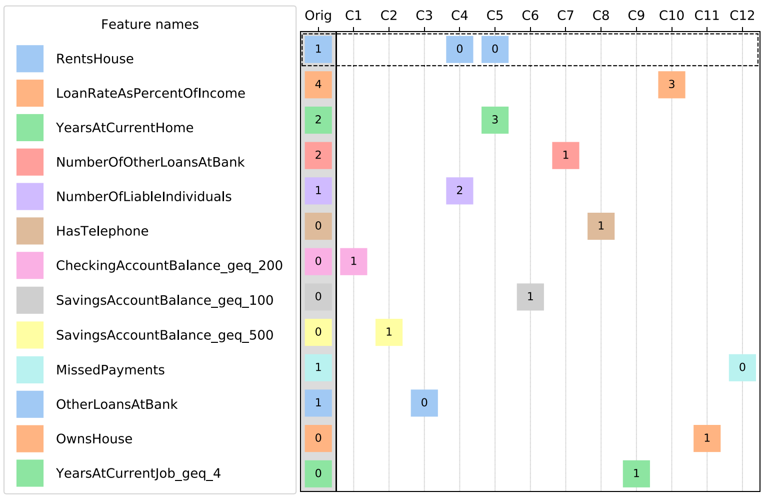
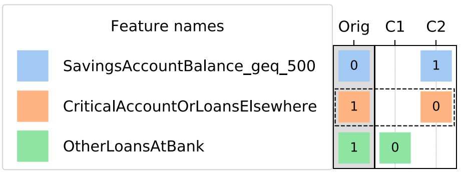


Fig. 7 Enumerations for a multilayer perceptron (whose monotonicity is depicted in Fig. 5) with Pareto-optimal values when each feature is considered as an objective function. Orig column is the original sample and the columns C1 to C12 are counterfactual antecedents

Fig. 8 Enumerations for a decision tree (depicted in Fig. 6) with Pareto-optimal values when each feature is considered as an objective function. Orig column is the original sample and the columns C1 and C2 are counterfactual antecedents



multilayer perceptron, which has the probability behavior depicted in Fig. 5; (b) the decision tree illustrated in Fig. 6; and (c) a gradient boosting trees (LightGBM) to complement the decision trees with a more robust classifier.

Figure 7, 8, and 9 shows valuable suggestions to change the prediction given by non-monotone classifiers. It is worth observing that new features are essential to the decision depending on the classifier: (1) changing `CriticalAccountOrLoans-Elsewhere` is an antecedent on Decision Trees (C1 in Fig. 8) and LightGBM (C2, C3, and others, in Fig. 9), but it is not in Logistic Regression (Fig. 3) and Neural Networks (Fig. 7). (2) changing `RentsHouse` is an antecedent on LightGBM (C3 and C7 in Fig. 9) and Neural Networks (C4 and C5 in Fig. 7), but it is not in Logistic Regression (Fig. 3) and Decision Trees (Fig. 8). The variations found by enumeration in distinct learning machines show the importance of using a model-agnostic algorithm. MAPOCAM compares the classifiers' counterfactual antecedents and promotes richer analysis to understand the problem in real life.

4.2 Experiment 2: comparing correctness and performance

In this experiment, we rely on the three datasets (german, giveme, and taiwan) to assess the capability of finding the optimal solution (on single-objective) or set of Pareto-

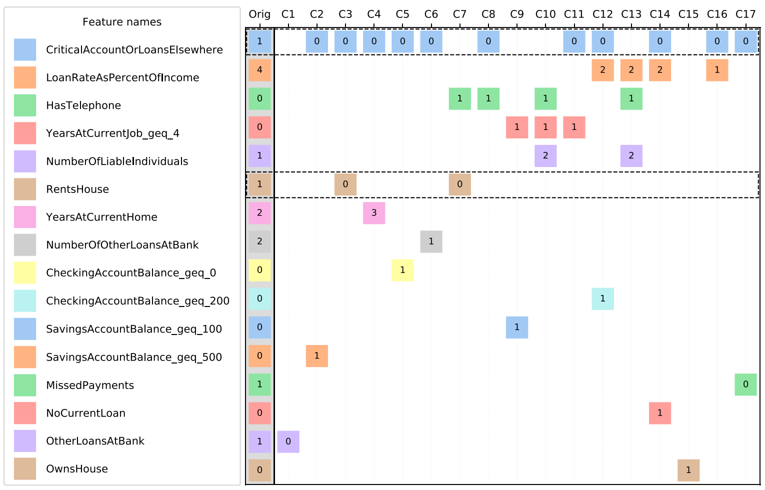


Fig. 9 Enumerations for a LightGBM with Pareto-optimal values when each feature is considered as an objective function. Orig column is the original sample, and the columns C1 to C17 are counterfactual antecedents

optimal solutions (on multi-objective). Moreover, we analyze the computational time of our approach. No matter the experiment, we first train a machine learning model (which can be Logistic Regression or lightGBM classifier) in a training set and then perform a test on a separate data set of 100 samples. Finally, we use MAPOCAM (and the baselines) to extract counterfactual antecedents from samples with an undesired outcome on the test set, allowing us to access the algorithms' correctness and performance.

4.2.1 Baselines

We have different sets of baselines for single and multi-objective experiments. For single-objective experiments (using the objective of MPC cost described in Sect. 3.3), we compare MAPOCAM with a mixed-integer formulation (to compare with an exact approach) and a greedy process (to have an algorithmic reference). The mixed-integer formulations consist of optimization models that extract properties from specific classifiers to find counterfactual antecedents, thus demanding individualized approaches: *mi-logistic* for logistic regression (based on (Ustun et al. 2019)) and *mi-trees* for the ensemble of decision trees (based on (Cui et al. 2015)). The *greedy* approach is not individualized. Given an intermediate action, the greedy algorithm finds a new action with the best performance to participate in the next iteration. This action should have a change in a single feature, and the performance is the ratio between the MPC cost of making such an action and the improvement in the decision rule. This iterative procedure continues until it finds counterfactual antecedents with the desired outcome or no other action can improve the decision rule.

For multi-objective experiments (using the objective of MPC cost vs. the number of changes and j-th feature change described in Sect. 3.3), we also compare MAPOCAM

Table 1 Average execution time, in seconds, for finding an action optimizing the MPC cost

	German	Giveme	Taiwan	Pima	Student
Mi-logistic	0.02	0.015	0.02	0.01	0.02
Greedy	0.01	0.013	0.03	0.02	0.03
MAPOCAM	0.02	0.004	0.06	0.01	0.14

with a mixed-integer formulation (to compare with an exact approach) and a brute-force approach (to have an algorithmic reference). The mixed-integer formulations in a multi-objective context consist of approaches that execute the mixed-integer optimization several times: *po-mi-logistic* for logistic regression and *po-mi-trees* for the ensemble of decision trees. At each time, we add a constraint that forbids the optimization to find counterfactual antecedents dominated by the previously found solutions—resorting to a procedure used in general mixed-integer problems (Sylva and Crema 2004) that is further detailed in Algorithm 2 in Appendix C. We proposed both approaches (*po-mi-logistic* and *po-mi-trees*) as simple extensions to (Ustun et al. 2019) and (Cui et al. 2015) because there is no Pareto-optimal approach for counterfactual antecedents in the literature. The *brute-force* approach is not individualized and is designed similarly to Algorithm 1 without the code from Line 2 to 11 to avoid any early bounding, storing every counterfactual antecedent, even the dominated ones. Thus, it enumerates all possible actions without trying to stop the explorations early. Both multi-objective formulations and methods have the number of changes limited to three to attain a good variety of antecedents without demanding excessive computational resources.

Worth mentioning that (Ustun et al. 2019) indicate a method to find multiple counterfactuals; however, we decided to promote a modification to find all Pareto-optimal counterfactual antecedents instead of the suggested method to be comparable with the proposed it method. We did not include other methods that find multiple counterfactual antecedents because they cannot find Pareto-optimal counterfactual antecedents. Also, they use different norms (to enforce diversity (Karimi et al. 2020; Mothilal et al. 2020)), generating antecedents of different natures and even generating dominated antecedents. Thus, the execution time would not be comparable to the proposed method.

4.2.2 Experiment 1: comparing performance on logistic regression

In this experiment, we consider the three credit datasets and train a l_2 penalized logistic regression using Scikit-learn (Pedregosa et al. 2011) library. After cross-validation training, we define a threshold of τ to guarantee the probability of credit granted at least 50%. We use three optimization approaches to assess the correctness and performance: MAPOCAM, *mi-logistic*, and *greedy*. Table 1 shows the time performance of such algorithms. Worth mentioning that MAPOCAM and *mixed-integer* always find the best solution while *greedy* algorithms find 79%, 25%, 3%, 27%, and 56% optimal solutions for the datasets.

Table 2 Average execution time, in seconds, for finding a representation of a Pareto-front

Conflict	Method	German	Giveme	Taiwan	Pima	Student
MPC versus #changes	Po-mi-logistic	0.01	0.01	0.02	0.01	0.02
	Brute-force	0.84	1.64	2.39	0.74	1.32
	MAPOCAM	0.09	0.01	0.16	0.02	0.18
Feature	Po-mi-logistic	0.55	0.04	2.89	0.15	2.21
	Brute-force	0.55	1.07	2.35	0.56	1.43
	MAPOCAM	0.18	0.01	0.45	0.03	0.48

This experiment verifies that the proposed method can find the best result in all situations, showing MAPOCAM's correctness. The computational times of MAPOCAM are similar in three credit datasets, being a bit costly for taiwan and student datasets, but still quite acceptable. The higher computational cost, mainly when compared against mixed-integer, can be justified by two factors. (i) *Data structure and programming language*: our approach is implemented in Python with a NumPy array as a data structure, while the mixed-integer uses a CPlex[4] package implemented in C. (ii) *The mixed-integer approach relies on more information to tackle the problem*: the mixed-integer formulation makes use of additional information in every node to find a tighter projection of the "future" of the node. In contrast, our approach does not rely on any of this information. This makes our approach more general and capable of dealing with any decision function, such as decision trees or multilayer perceptron.

4.2.3 Experiment 2: enumeration of Pareto-optimal counterfactual antecedents

In this experiment, we show the algorithm's capability of enumerating a representation of all Pareto-optimal solutions. We use the same training procedure and threshold of the previous experiment to explore two multi-objective formulations: (i) minimize the action for every feature; (ii) minimize the MPC cost and the number of changes.

Table 2 shows the time performance of the MAPOCAM, *mixed-integer*, and *brute-force* approach; we also confirm that both searches find the same number of solutions for all of the samples. The proposed method tends to perform similarly to the *mixed-integer* approach, being faster than a *brute-force* approach in both scenarios: with two objectives (MPC cost vs. # changes) and more than two objectives (features). Notice that MAPOCAM is more costly than *mixed-integer* with two objectives (MPC cost vs. # changes), but it performs better with more than two objectives (features). This difference is mainly caused by evaluating the MPC cost—while this computation is costly in our approach, it is not relevant in the mixed-integer since the model itself embeds this information—since the cost of comparing the features is low.

[4] Available at ibm.com/analytics/cplex-optimizer.

Table 3 Average execution time, in seconds, for finding solutions on a random forest (*greedy not always find the best solution)

Conflict	Method	German	Giveme	Taiwan	Pima	Student
MPC cost	mi-trees	0.02	0.02	0.04	0.02	0.02
	Greedy	0.01	0.01	0.02	0.02	0.01
	MAPOCAM-N	0.37	0.02	5.42	0.56	0.02
	MAPOCAM-P	0.04	0.01	0.49	0.12	0.01
MPC versus #changes	po-mi-trees	0.04	0.03	0.99	0.10	0.02
	Brute-force	0.71	0.21	2.41	1.45	0.62
	MAPOCAM-N	0.30	0.02	0.73	0.21	0.04
	MAPOCAM-P	0.10	0.01	0.13	0.09	0.02
Feature	po-mi-trees	0.03	0.02	0.98	0.09	0.02
	Brute-force	0.26	0.13	1.70	1.01	0.15
	MAPOCAM-N	0.17	0.06	1.28	0.45	0.05
	MAPOCAM-P	0.13	0.02	0.29	0.18	0.03

4.2.4 Experiment 3: comparing performance on non-monotonic classifiers

We showed in Sect. 4.1.2 that the proposed method works properly in some non-monotonic classifiers. This experiment aims to examine the performance of the proposed algorithm in a forest of decision trees. We consider the three credit datasets and train a Random Forest from Scikit-learn (Pedregosa et al. 2011), with five estimators, a depth of five, and 31 leaves. We define a threshold of τ to guarantee the probability of credit granted at least 50% and use three optimization approaches to assess the correctness and performance.

Table 3 shows the time performance of such algorithms with one (MPC cost), two (MPC cost vs. # changes), and multiple objectives (feature). Except for the *greedy* algorithm, all methods were capable of finding the Pareto-optimal set of solutions. In this experiment, we show two versions of the proposed algorithm: (i) MAPOCAM-N is a version with no unfeasibility branching, and (ii) MAPOCAM-P explores the possibility of evaluating all leaves from the random forest and finding the optimization tree's maximal probability, considering that it already took some actions. This second approach can behave similarly to a monotone classifier because it avoids unpromising branching, thus improving time performance.

In this experiment, the MAPOCAM can find optimal solutions in all instances. The performance is quite competitive when the predictor can foresee unfeasibility in the node (MAPOCAM-P version). However, the technique tends to be time-consuming without resorting to methods that foresee unfeasible nodes (MAPOCAM-N version).

Since it is possible to observe a gain in MAPOCAM-P performance, we investigate the monotonicity's impact on the optimization procedure. In that regard, we explore the monotonicity constraints on the LGBMClassifier class from the lightGBM[5] library. Given that, we train LGBMClassifier with ten estimators, a depth of ten, and 63 leaves.

[5] lightgbm.readthedocs.io.

Table 4 Average execution time, in seconds, for finding solutions on a monotone lightgbm (*greedy not always find the best solution)

Conflict	Method	German	Giveme	Taiwan	Pima	Student
MPC cost	mi-trees	0.03	0.03	0.07	0.03	0.02
	Greedy	0.02	0.02	0.08	0.03	0.01
	MAPOCAM-N	4.13	0.01	1.35	0.24	0.10
	MAPOCAM-P	0.02	0.01	0.05	0.02	0.01
	MAPOCAM-M	0.02	0.01	0.05	0.02	0.01
MPC versus #changes	po-mi-trees	0.07	0.04	0.94	0.05	0.02
	Brute-force	1.39	0.68	20.94	0.75	0.46
	MAPOCAM-N	0.49	0.01	0.56	0.12	0.11
	MAPOCAM-P	0.04	0.01	0.08	0.02	0.02
	MAPOCAM-M	0.04	0.02	0.09	0.02	0.01
Feature	po-mi-trees	0.07	0.03	0.94	0.05	0.02
	brute-force	0.54	0.46	17.04	0.53	0.27
	MAPOCAM-N	0.30	0.07	12.56	0.22	0.22
	MAPOCAM-P	0.04	0.01	0.94	0.03	0.01
	MAPOCAM-M	0.04	0.02	0.90	0.03	0.01

We define a threshold of τ to guarantee the probability of credit granted at least 50% and use three optimization approaches to assess the correctness and performance. We compare the same contenders from the previous comparison, but now we have three variations from the proposed algorithm. (i) MAPOCAM-N is a version with no unfeasibility branching. (ii) MAPOCAM-P explores the possibility of evaluating all leaves from the random forest and finding the maximal probability of the optimization tree, considering that it already took some actions. And (iii) MAPOCAM-M explores the monotonicity to avoid unfeasible branches. Table 4 shows the time performance of such algorithms with one (MPC cost), two (MPC cost vs. # changes), and multiple objectives (feature).

This experiment shows that MAPOCAM can find optimal solutions in all instances. MAPOCAM has a competitive performance when it explores the monotonicity property of the classifier (MAPOCAM-M version) or when it deals with other classifiers' properties to estimate the maximum probability of a counterfactual antecedent (MAPOCAM-P version). The difference in performance is due to the difference in computational effort requested by the foreseeing step on MAPOCAM-P and MAPOCAM-M in particular cases. However, the method tends to be time-consuming without resorting to methods that foresee unfeasible nodes (MAPOCAM-N version).

4.2.5 Discussion

In general, MAPOCAM achieves good performance when: (i) the classifiers are monotonic or (ii) it explores the constraints of achievable leaves in decision trees.

The execution time is also manageable without any information about the model (MAPOCAM-N)- generally taking less than 1 min. This method application leads to a flexible plug-and-play model-agnostic method capable of working with any classifier without providing any model-related procedure. Any other method in the literature needs model-related procedures to work, such as (i) creating a new mixed-integer model (Ustun et al. 2019; Cui et al. 2015), (ii) requiring differentiable functions (Mothilal et al. 2020), or (iii) transforming the model to another representation (Karimi et al. 2020).

Worth mentioning that MAPOCAM can create a diverse set of counterfactual antecedents using multi-objective optimization. Theorem 5 in Appendix A.2 shows that MAPOCAM enumerates all proper counterfactual antecedents without resorting to norms and adjusting their parameters (Mothilal et al. 2020; Karimi et al. 2020). Also, *po-mi-logistic* and *po-mi-trees* are modifications from mixed-integer models (Ustun et al. 2019; Cui et al. 2015) that we adapted to interactively find Pareto-optimal counterfactual antecedents. Moreover, this paper is a pioneer in considering multi-objective optimization in counterfactual antecedent mining.

5 Case study: enumeration of counterfactual antecedents for crimes in São Paulo

In this experiment, we consider the impact of counterfactual antecedents on policy decision-making on crime. As a case study, we investigate crime patterns in São Paulo city in Brazil. We gathered socioeconomic, urban, and crime information from census regions of São Paulo. The Center for the Study of Violence from the University of São Paulo (NEV-USP)[6] provided us with the criminal records. The Center for Metropolitan Studies (CEM)[7] provided us with data about schools, bus stops, and bars. With this data, we ranked census regions by the rate of passerby crimes normalized by each region's total population. Then, we labeled the 10% regions with higher rates as dangerous and fitted an l_2 regularized Logistic Regression to classify if an area is hazardous or not.

Given a census region classified as dangerous, MAPOCAM can create counterfactuals that give an idea of what a decision-maker could do to make that region safer. An expert can then be consulted to provide a practical and theoretical basis and assert that these feature changes would change the outcome. Figure 10 describes counterfactuals in Region 1. This region has a high incidence of violent crimes in the 2000s. The original attributes of this region are depicted in the first column (Orig) of the figure, showing an impoverished region with a relatively low `HighIncomeHolder` (percentage of householders), `PermanentHousing` (percentage), `WaterSupply` and `SewageCollection` (service coverage). The other columns (C1, C2, and others) show possible counterfactuals to change the high-criminality outcome. For instance, C2 shows that increasing `PermanentHousing` from 0.843 to 0.876 is a counterfactual. On the other hand, C5 shows that a minor increase in `PermanentHousing`

[6] nevusp.org.

[7] centrodametropole.fflch.usp.br/pt-br.

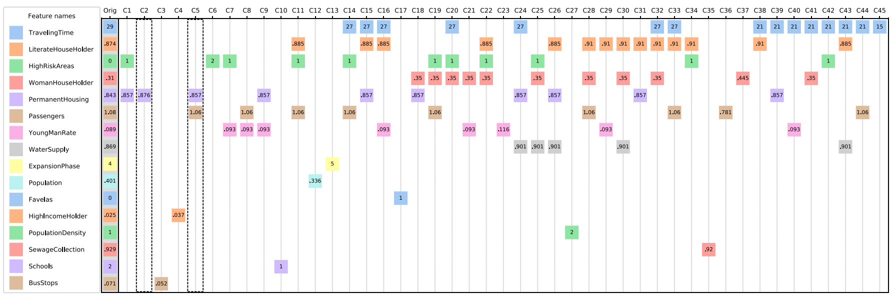


Fig. 10 A multi-objective enumeration of actions that reduce the criminality rate of a region of São Paulo. Orig column is the original sample, and the columns C1 to C45 are counterfactual antecedents

(from 0.843 to 0.857) and reducing `Passengers` from 1.08 (thousand) to 1.06 is another way to achieve a counterfactual.

In general, it is possible to observe that reducing population size, average traveling time, and the number of bus stops—lowering people volume in the region—would change outcomes related to high-criminality. Other factors, such as increasing literate householders, permanent housing, water supply, and high-income holders, might indicate a socioeconomic interpretation of reduced criminality in wealthier regions. One can correctly say that this is a direct effect of the logistic regression weights. Nevertheless, only with our methodology can we observe the intensity and combination of changes that lead to the desired safety. We also want to highlight that it is suggested to increase the number of favelas to 1 (C17). It looks counterintuitive, but we should remember that we are using correlation-based machine learning to extract the counterfactual antecedents. This correlation might come from the low notification of crimes in poorer, vulnerable regions, making the machine learning model associate sub-notification with a low-crime region. This suggestion should not be considered when proposing a public policy. Nevertheless, since the proposed method can find a diverse set of counterfactual antecedents, an expert can remove those related to critical (or sensible) variables and investigate others.

However, we can also find some undesired suggestions, such as becoming a high-risk (a favela area) or decreasing the region's sewage collection. To fix that, we can disable changes on these features or specify if those features should increase or decrease. Figure 11 depicts an enumeration of changes after the aforementioned disabling. This new scenario shows more realistic planning, such as modifying the bus routes (to reduce the number of passengers passing through this region), reducing the traveling time, improving the water supply, and incentivizing population growth. Note that we labeled the regions considering the criminality rate, so the suggestion to increase the population might reduce the rate, explaining this proposed change.

Figures 10 and 11 show a set of highly insightful counterfactual antecedents. These options help understand the prediction, but when the number of counterfactual antecedents is too high, it is hard to select the best suitable one and understand each feature's impact on the learning. To better understand the overview of counterfactual antecedents before selecting one, Fig. 12a presents the feature importance in a logistic regression (higher the value, the more important is the variable), and

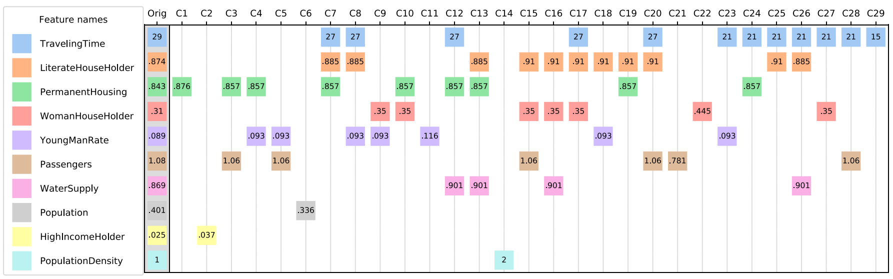


Fig. 11 A multi-objective enumeration of actions that reduce the criminality rate of a region of São Paulo. This is the same enumeration from Fig. 10, but some features were not allowed to participate in counterfactual antecedents because of undesired changes. Orig column is the original sample and the columns C1 to C29 are counterfactual antecedents

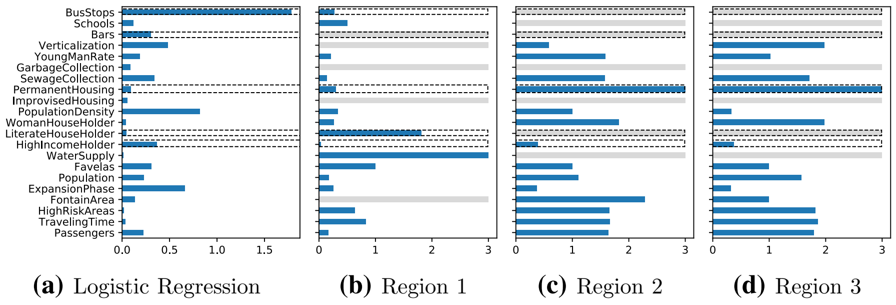


Fig. 12 This figure shows the representation of the importance of each feature. **a** represents the feature importance for the logistic regression learning process (higher the better); **b**–**d** represent how much a feature has to change to participate in a counterfactual antecedent (the lower the better). The value is normalized by how much the variable can change; greater than one means that the variable usually needs other features to participate in a counterfactual antecedent. Grey bars represent features that never participate in a counterfactual antecedent. Thus, we represented them as having the largest cost—3

Fig. 12b–d present the CA_i index described in Sect. 3.5 for three different regions in São Paulo: the already mentioned Region 1, the central and commercial Region 2, and the wealthy Region 3 (with headquarters of many corporations). The CA_i index indicates how much a feature has to change to create a counterfactual antecedent on relative average (near 0—small change, near 1—vary to its maximum, between 1 and 3—need another variable to create a counterfactual antecedent, and gray bar—unachievable.

Figure 12 shows exciting properties: (i) the CA_i index for each feature is different from the importance calculated with counterfactuals antecedents; (ii) the indexes on each region are different from each other. From these results, we can say that reducing crime requires different approaches for each region. Also, we observe that reducing the number of `BusStops` and `Bars` does not help achieve counterfactuals in Regions 2 and 3 because these regions do not have those amenities. Socio-economic variables (`PermanentHousing`, `HighIncomeHolder`, and `LiterateHouseHolding`) need more change to help Region 2 and Region 3 become safer, mainly because those regions already have better socio-economic conditions.

This case study depicts the importance of relying on counterfactual antecedents in policy decision-making. We could identify the particularities of each region, pointing to more effort in solving socio-economic issues in the most vulnerable areas, which a single explanation would not necessarily point to the whole model (e.g., feature importance in Logistic Regression). We envision a real impact on criminality control using our methodology with a richer dataset (e.g., information about road signaling, sidewalk preservation, wall painting, and urban lighting), which would allow suggestions for departments on urban administration to help with criminality control.

6 Discussion and limitations

This research's final product successfully constructs a model-agnostic algorithm that finds diverse counterfactual antecedents using multi-objective optimization concepts. However, there are some limitations and future work that can be addressed in follow-up work.

Performance. Compared to other approaches that explore mixed-integer solvers (Cui et al. 2015; Ustun et al. 2019), we notice that our method is rarely better but has competitive performance. The mixed-integer approach has more available information (e.g., linear relaxations of the problem), more computational resources (can explore parallelism), and a more cost-efficient programming language and data structure. Thus, it indicates that having a competitive performance shows our proposal's real value, which is also more general and can deal with any classifier. Therefore, the following steps can improve our approach:

- Use model information before the branching procedure on each node, such as: (1) explore the use of a linear relaxation for logistic regression; (2) use convex optimization to find heuristic solutions on neural networks; and (3) use satisfiability modulo theories (SMT) to verify the feasibility of some actions (Karimi et al. 2020).
- As a tree search, a node does not depend on information from other nodes. Thus, we could implement a parallel version that only needs to share the found counterfactual antecedents.
- As a first approach, we used the NumPy array as a data structure for counterfactual antecedents, but it can be improved using more sophisticated solutions.
- Another option to improve the performance is to use other search algorithms such as A^*. This strategy can take advantage of heuristics designed for every objective function.

Discretization. We proved that our algorithm could find every Pareto-optimal counterfactual antecedent given a set of possible actions. The optimality is globally valid for decision trees. We create a step for every interval between decisions on the features promoted by decision trees (or ensemble trees), thus covering all possible changes for that decision tree. We use a simple stepwise discretization with absolute or relative steps for numerical features for linear classifiers and neural networks. However, this decision does not guarantee the optimal result for all discretizations. One solution could be to use a binary search for further refining when a counterfactual antecedent is found (Karimi et al. 2020). Improving the discretization can also help find better-suited solutions and optimal value with lower costs. On the other hand, for non-monotone classifiers, it helps to find more counterfactual antecedents. Nevertheless, it can drastically change the search structure or require a specific design for every family of classifier methods.

Opposite direction actions. Another alternative to dealing with non-monotone classifiers is to create two variables for each feature, i.e., one for the positive and one for the negative values. This strategy removes any conflict between the negative and positive directions. Also, it allows the design of a monotone objective function to deal with negative actions. We plan to work on it as an immediate future work; however, it needs additional theoretical and experimental developments to ensure stable algorithm behavior.

Categorical variables. Another limitation in our proposal consists of dealing with categorical features. We did not consider this feature in the algorithm because it depends on the encoding strategy. However, it is possible to encode constraints in the algorithm to use categorical variables. It might be an immediate future work, but we need more theoretical and experimental developments to ensure a stable algorithm's stable behavior.

Causality. Since the investigated model is correlation-based, like most machine learning models, the generated counterfactual antecedents reveal correlations and not causal implications. This limitation is common in model-agnostic approaches (Chou et al. 2022). We believe that using causal inference methods (Judea 2010) can help experts to find more reasonable counterfactual antecedents, but dependencies between variables are hard to infer. In fact, generating counterfactual antecedents with causal relations is still an open question, we intend to approach in a follow-up work.

Visualization. All the experiments take care of choosing examples that do not have many possible actions; we did it to simplify the analysis. However, we notice cases with more than a hundred actions, which could be overwhelming to anyone analyzing the possible counterfactual antecedents. We are now working on creating a visual analytic system to explore the counterfactuals interactively. This system should also allow selecting and filtering counterfactuals with some properties, e.g., having (or not having) a feature of a specific change and showing the counterfactual antecedents with different visualization methods.

7 Conclusion

The main finding of this research is a tree-based algorithm, called MAPOCAM, that enumerates all Pareto-optimal counterfactual antecedents. Our technique helps to interpret model prediction and to mine counterfactual scenarios. Moreover, our algorithm possesses properties validated by empirical evidence. First, it is optimal; we found the same outcomes as a mixed-integer algorithm in single-objective problems and with a brute-force approach in multi-objective situations. Second, it is model-agnostic; we tested our algorithms with linear models, multilayer perceptron, and decision trees. Third, it is diverse; we explore multiple counterfactual antecedents to analyze real-world applications.

Another exciting feature of this research relies on the counterfactual antecedent index. This index helps explore real-world applications when the set of antecedents is numerous. For instance, in the case study of São Paulo, our tools gave a good overview of the usefulness of diverse counterfactual antecedents in crime analysis.

The use of multi-objective concepts by MAPOCAM diversifies the counterfactual antecedents, helping to provide information on the model's behavior for a given sample. Modeling each feature as an objective helps find counterfactual antecedents with distinct sets of features and find trade-offs among them. Both aspects and the Pareto-optimality (no other counterfactual antecedent is better in all objectives) are keys to mining counterfactual antecedents. This research makes an essential step in this matter.

Acknowledgements This work was supported by CNPq-Brazil (Grants #312483/20180) and Getúlio Vargas Foundation.

Appendices

A Proofs of the theorems

This appendix presents some theoretical proofs of properties concerning the proposed methodology.

A.1 Algorithm properties

To help on branching nodes in Algorithm 1 it is simple to see that from Definition 3 subsequent nodes of a node with dominated action will be dominated.

Corollary 1 *Let* $\mathbf{a}$ *and* $\mathbf{a}'$ *as in Definition* 3 *and consider that* $\mathbf{a}'$ *and a set of taken decisions* $\mathcal{D}'$ *defines a node. It is possible to see that no subsequent action in this node* $\bar{\mathbf{a}} \succeq \mathbf{a}'$ *will improve any monotone objective functions.*

To further improve the search, if the chosen decision rule $r(\mathbf{x})$ is monotone, we can define an **unfeasible node**.

Definition 9 - Unfeasible node. Given a node with its related actions $\mathbf{a}$ and taken decisions $\mathcal{D}$, and given a monotone decision function $r(\bullet)$; the maximal achievable action $\bar{\mathbf{a}}^*$ for this node is given by:

$$\bar{a}_i^* = \begin{cases} a_i, & \text{if } i \in \mathcal{D} \\ \max a_i, & \text{otherwise} \end{cases} \tag{3}$$

Given that, a **node is unfeasible** iff $r(\mathbf{x} + \bar{\mathbf{a}}^*) < \tau$.

Theorem 1 *Let* $\mathbf{a}$ *and* $\mathcal{D}$ *as in Definition* 9. *We want to prof that any action* $\bar{\mathbf{a}} : \bar{a}_i \geq a_i \forall i \notin \mathcal{D}$, *is infeasible* $r(\bar{\mathbf{a}}) < \tau$.

Proof Let's suppose by contradiction that $r(\overline{\mathbf{a}}) \geq \tau$, since $\overline{\mathbf{a}}^* \geq \overline{\mathbf{a}}$, we have, by monotonicity of $r(\bullet)$ that $r(\overline{\mathbf{a}}^*) \geq r(\overline{\mathbf{a}}) \geq \tau$, what contradicts $r(\overline{\mathbf{a}}^*) < \tau$. □

Notice that it is only possible to define a node as unfeasible if the decision rule respects the monotone property. But when we have this property, we can preemptively discard this node.

There are some guarantees that Algorithm 1 is capable of detecting all optimal solutions and evaluating their time complexity.

Theorem 2 *All Pareto-optimal solutions are found.*

Proof We know that Algorithm 1 would search through all possible actions if no pruning condition (lines 2, 5, and 8 in Algorithm 1) is activated. Any Pareto-optimal solution $\mathbf{a}^* : \mathbf{c}(\mathbf{a}^*) \leq \mathbf{c}(\mathbf{a}),\ \forall \mathbf{a} \in \mathcal{A}$ would not be found iff the conditions in lines 2, 5, and 8 are activated by $\mathbf{a}^*$ or a predecessor $\mathbf{a} \preceq \mathbf{a}^*$.

We can split the proof by contradiction into 3 cases:

– Line 2 would be activated if a predecessor $\mathbf{a} \preceq \mathbf{a}^*$ is dominated by a feasible solution $\mathbf{a}' \in \mathcal{A}$: $\mathbf{c}(\mathbf{a}') \preceq \mathbf{c}(\mathbf{a})$. This cannot be true since $\mathbf{c}(\mathbf{a}') \preceq \mathbf{c}(\mathbf{a}) \preceq \mathbf{c}(\mathbf{a}^*)$ would contradict the Pareto-optimally of $\mathbf{a}^*$.
– Line 5 would be activated if $r(\mathbf{x} + \overline{\mathbf{a}}^*) < \tau$. This cannot be true since $\mathbf{a}^* \preceq \overline{\mathbf{a}}^*)$ and it is a feasible solution.
– Line 8 would be activated if $r(\mathbf{x} + \mathbf{a}) \geq \tau$. This cannot be true because this condition would indicate that $\mathbf{a}$ is feasible, and $\mathbf{c}(\mathbf{a}) \preceq \mathbf{c}(\mathbf{a}^*)$ would contradict the Pareto-optimally of $\mathbf{a}^*$.

Since all conditions led to a contradiction the proof is done. □

Theorem 3 *The worst-case complexity of the algorithm is $T(d, k) = \mathcal{O}((bd)^k)$, where d is the number of variables, b is the maximum number of possible states of each variable, and k is the maximal number of allowed changes.*

Proof We can see the cost of each recurrence is given by:

$$T(d,k) = \begin{cases} dbT(d-1, k-1) + db + d^2, & k > 0 \\ 1, & k = 0 \end{cases} \tag{4}$$

By substitution we can see, for $k > 0$ that:

$$\begin{aligned} T(d,k) &\leq dbT(d-1, k-1) + db + d^2, && (5)\\ &\leq db(b(d-1))^{(k-1)} + db + d^2, && (6)\\ &\leq b^k(d^k - d) + db + d^2 && (7) \end{aligned}$$

Given that, we can see that $T(d, k) = \mathcal{O}((bd)^k)$. □

A.2 Objective function properties

Since monotonicity is a requirement for the algorithm, in this section, we discuss some properties of monotone functions and prove the monotone properties of the functions used in the experiment.

Definition 10 Negative changes do not decrease the cost.

Defining a pair of actions $\mathbf{b}$, and $\mathbf{a}$. In which $\mathbf{b}$ have negative changes and $\mathbf{a}$ is equivalent to $\mathbf{b}$ but with *null* action in such way: $\mathbf{a}, \mathbf{b} : a_j = 0, b_j < 0, \forall j \in \mathcal{J}$ and $a_j = b_j, \forall i \notin \mathcal{J}$.

The objective functions must be defined like that: $c_k(\mathbf{a}) = c_k(\mathbf{b}),\ \forall k \in \{1, \ldots, M\}$.

Theorem 4 *Any feasible solution with negative changes has a correspondent feasible action on monotone decision functions.*

Proof Let's suppose an action $\mathbf{b} : b_j < 0 \in \mathcal{J}$, that is feasible. By Definition 10, it is possible to see that exists an action $\mathbf{a} : a_j = 0, b_j < 0, \forall \in \mathcal{J}$ and $a_j = b_j, \forall i \notin \mathcal{J}$ with the same cost.

By the monotonicity of $r(\bullet)$ it is possible to see that $\mathbf{a}$ is also feasible. □

Also, we want to show that using the j-th feature change for all features leads to an enumeration of Pareto-optimal solutions that will contain any set of Pareto-optimal solutions from other objective functions.

Theorem 5 *Any Pareto-optimal solution for a vector of monotone objective functions is a Pareto-optimal solution if we optimize the feature changes.*

Proof Knowing that $\mathbf{c}(\bullet)$ is the vector of functions for the j-th feature changes, and $\mathbf{c}'(\bullet)$ is another vector of functions that are monotone.

Let's assume, by contradiction, that $\mathbf{a}^*$ is a Pareto-optimal for $\mathbf{c}'(\bullet)$, and dominates all other solutions. But $\mathbf{a}^*$ is not Pareto-optimal for $\mathbf{c}(\bullet)$.

From the lack of Pareto-optimality of $\mathbf{a}^*$ in $\mathbf{c}(\bullet)$, we assume that a feasible solution $\mathbf{a}$ dominates $\mathbf{a}^*$, thus $c_j(\mathbf{a}^*) \geq c_j(\mathbf{a}), \forall j \in \{1, \ldots, d\}$, meaning that $\mathbf{a}^* \succeq \mathbf{a}$ (because the objective are the actions it-selves).

From the Pareto-optimality of $\mathbf{a}^*$ in $\mathbf{c}'(\bullet)$ and dominates all other solutions, exists k such that $c'_k(\mathbf{a}^*) < c'_k(\mathbf{a})$. Since $c'_k(\bullet)$ is monotone $c'_k(\mathbf{a}^*) \geq c'_k(\mathbf{a})$, we have a contradiction. □

A.3 Monotone classifier's properties

Another important trait that a classifier can have is its monotonicity. It helps in avoiding branches with no feasible actions.

A.3.1 Logistic regression and SVM

The decision rules from Logistic regression and SVM can be formulated as $p(\mathbf{x}) = \frac{1}{1+e^{-\theta^\top \mathbf{x}+b}}$ (SVM probability being formulated as in Platt Scaling). Assuming that $\theta_i \geq 0$ for all i mutable (consequently $\mathbf{a}_j = 0$ for all j non-mutable).

Theorem 6 *The decision function* $p(\mathbf{x}) = \frac{1}{1+e^{-\theta^\top \mathbf{x}+b}}$ *is monotone.*

Proof Since $\mathbf{a} \succeq \mathbf{0}$:

$$\frac{1}{1+e^{-\theta^\top(\mathbf{x}+\mathbf{a})+b}} = \frac{1}{1+e^{-\theta^\top \mathbf{a}}e^{-\theta^\top(\mathbf{x})+b}} \tag{8}$$

Since $e^{-\theta^\top \mathbf{a}} \le 1$ then $e^{-\theta^\top \mathbf{a}}e^{-\theta^\top(\mathbf{x})+b} \le e^{-\theta^\top \mathbf{x}+b}$. Consequently $p(\mathbf{x}+\mathbf{a}) \ge p(\mathbf{x})$. □

A.3.2 Ensemble of monotone classifiers

Supposing an ensemble of classifiers $p^{ens}(\mathbf{x}) = \sum_{i=1}^{E} \frac{p^i(\mathbf{x})}{E}$ where $p^i(\mathbf{x})$ is monotone for all $i \in \{1, \ldots, E\}$.

Theorem 7 *The decision function* $p^{ens}(\mathbf{x})$ *is monotone.*

Proof Let's suppose $\mathbf{x}' \succeq \mathbf{x}$. Given that:

$$p^i(\mathbf{x}') \ge p^i(\mathbf{x}), \ \forall i \in \{1, \ldots, E\} \tag{9}$$

$$\sum_{i=1}^{E} \frac{1}{E} p^i(\mathbf{x}') \ge \sum_{i=1}^{E} \frac{1}{E} p^i(\mathbf{x}) \tag{10}$$

$$p^{ens}(\mathbf{x}') \ge p^{ens}(\mathbf{x}) \tag{11}$$

Proving that $p^{ens}(\bullet)$ is monotone. □

B Bounding procedure in a decision tree

The main goal of monotonicity for classifiers is to assess if the desired probability is achievable in the subsequent branching in the B&B optimization. It is possible to obtain that without enforcing monotonicity in decision trees. Each node in B&B constrains the value of actions a_i for all features i in the set of taken decisions $\mathcal{D}$. With that, it is possible to filter which leaves are reachable in the decision trees by the counterfactuals on that node. Figure 13 shows an example of a decision tree. Let's suppose that in the search process, we already have a counterfactual antecedent with $x_2 = 10, x_5 = 1$, and $x_6 = 60$ but without enforcing any other variable. In this scenario, note $\textcircled{1}$ has no decision (x_1 not chosen yet). Thus we can go to any branch. Node $\textcircled{2}$ guides to node $\textcircled{4}$ that has no decision, thus guiding to leaves $p_1 = 0.7$ and $p_2 = 0.2$. Node $\textcircled{3}$ guides to node $\textcircled{7}$, thus guiding to leaf $p_8 = 0.5$. In this context, any other decision in the counterfactual antecedents search will not find any probability higher than 0.7. If the threshold $\tau = 0.8$, no counterfactual antecedent can achieve this goal; thus we can bound the process. Thus, finding that no leaf in decision trees reaches the desired probability can rule out the correspondent node in B&B optimization. We also use this methodology to speed up the algorithm in the experiments.

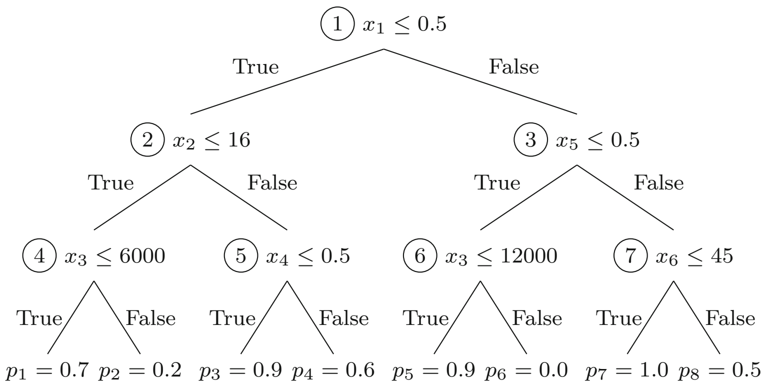


Fig. 13 An example of a decision tree. The decision process occurs in the inner nodes that channel the decision using the features **x**. The leaf nodes contain the probability of the samples that were channeled to that leaf belonging to class 1 (e.g., a sample with $x_1 = 0$, $x_2 = 20$ and $x_4 = 1$ falls into leaf 4 with a probability of 60%, or $p_4 = 0.6$)

C Implementation details

It is crucial to notice that in all formalization of the method, all actions are additive $\mathbf{a} \succeq \mathbf{0}$ to simplify the notation. However, it is possible to explore actions that can subtract a value of sample **x**, and the objective functions and the decision functions will handle only negative actions to sustain the monotone property.

Also, worth mentioning the actions must be discretized to work in the proposed method [as well as other approaches (Ustun et al. 2019; Cui et al. 2015)]. To do so, we used the same strategy as Ustun et al. (2019), where they discretize the actions by making fixed steps between the lowest and largest values. For each feature in our datasets, we use two strategies: (1) a fixed step size; and (2) a fixed number of steps. Moreover, for decision tree-based classifiers, we used the same strategy as Cui et al. (2015), where they discretize the space of a feature taking into account the splits of the decision tree classifier. Thus, there is a change value for each space between the splits for that feature, enumerating all actions to the decision trees

As previously mentioned, the procedure SELECT_FEATURE in Algorithm 1 is responsible for indicating the next feature to be explored. To do that, we resorted to variable importance from Random Forests (Breiman 2001) that randomly permute the values of a feature and estimate the impact in the prediction to evaluate the feature importance. We select an unexplored feature on a given node with higher priority to find a counterfactual antecedent quickly.

Another implementation detail involves the modification presented in this work to make mixed-integer models (Cui et al. 2015; Ustun et al. 2019) to generate Pareto-optimal counterfactual antecedents. Given a *model* that finds a counterfactual explanation (Cui et al. 2015; Ustun et al. 2019), the procedure in Algorithm 2 consists of iteratively finding counterfactual antecedents and adding constraints (lines 7 and 8) that counts (v) the number of objective components of action with a value larger than a previously found antecedent $\mathbf{a}^*$ and enforces it to be lower than the num-

Algorithm 2 Pareto-optimal procedure for mixed-integer formulations.

```
Require: A mixed-integer model builder $build$, a sample $\mathbf{x}$, a objective function $\mathbf{c}(\bullet)$, a decision rule $r(\bullet)$,
    a threshold $\tau$, and a number of allowed changes $k$.
1: procedure ENUMERATE_MIP($model$)
2:    $\mathcal{A} = \{\}$
3:    while $model.feasible$ do
4:       $\mathbf{a}^* = model.optimize()$
5:       if $model.feasible$ then
6:          $\mathcal{A} = \mathcal{A} \cup \{\mathbf{a}^*\}$
7:          $v = |i : c_i(\mathbf{a}) \geq c_i(\mathbf{a}^i),\ \forall i \in \{1, \ldots, m\}|$
8:          $model.addConstr(v \leq m - 1)$
9:       end if
10:   end while
11:   return $\mathcal{A}$
12: end procedure
13: $model = build(\mathbf{x}, \mathbf{c}, r, \tau)$
14: $\mathcal{A}$ =ENUMERATE($model$)
15: return $\mathcal{A}$
```

ber of objectives ($v \leq m - 1$). This procedure induces any subsequent optimization of the *model* not to have any new action dominated by previously found antecedents.

References

Aggarwal CC, Chen C, Han J (2010) The inverse classification problem. J Comput Sci Technol 25(3):458–468

Artelt A, Hammer B (2020) Convex density constraints for computing plausible counterfactual explanations. In: International conference on artificial neural networks, Springer, pp 353–365

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Chen L, Lin X, Hu H, Jensen CS, Xu J (2015) Answering why-not questions on spatial keyword top-k queries. In: Proceedings—international conference on data engineering, pp 279–290. https://doi.org/10.1109/ICDE.2015.7113291

Chou YL, Moreira C, Bruza P, Ouyang C, Jorge J (2022) Counterfactuals and causability in explainable artificial intelligence: theory, algorithms, and applications. Inf Fus 81:59–83

Cui Z, Chen W, He Y, Chen Y (2015) Optimal action extraction for random forests and boosted trees. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining, association for computing machinery, New York, pp 179–188

Dua D, Graff C (2017) UCI machine learning repository-student performance

Gao Y, Liu Q, Chen G, Zheng B, Zhou L (2015) Answering why-not questions on reverse Top-k queries. Proc VLDB Endow 8(7):738–749. https://doi.org/10.14778/2752939.2752943

Grath RM, Costabello L, Van CL, Sweeney P, Kamiab F, Shen Z, Lecue F (2018) Interpretable credit application predictions with counterfactual explanations. arXiv preprint 1:1–9 arXiv:1811.05245

Gupta M, Cotter A, Pfeifer J, Voevodski K, Canini K, Mangylov A, Moczydlowski W, Van Esbroeck A (2016) Monotonic calibrated interpolated look-up tables. J Mach Learn Res 17(1):3790–3836

Hada SS, Carreira-Perpiñán MÁ (2021) Exploring counterfactual explanations for classification and regression trees. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 489–504

He Z, Lo E (2012) Answering why-not questions on top-k queries. In: Proceedings—international conference on data engineering, pp 750–761. https://doi.org/10.1109/ICDE.2012.8

Judea P (2010) An introduction to causal inference. Int J Biostat 6(2):1–62

Kaffes V, Sacharidis D, Giannopoulos G (2021) Model-agnostic counterfactual explanations of recommendations. In: Proceedings of the 29th ACM conference on user modeling, adaptation and personalization, pp 280–285

Karimi AH, Barthe G, Balle B, Valera I (2020) Model-agnostic counterfactual explanations for consequential decisions. In: International conference on artificial intelligence and statistics. PMLR, pp 895–905

Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY (2017) LightGBM: a highly efficient gradient boosting decision tree. In: Advances in neural information processing systems 2017 (Nips), pp 3147–3155

Krause J, Perer A, Ng K (2016) Interacting with predictions: visual inspection of black-box machine learning models. In: Conference on human factors in computing systems—proceedings. Association for Computing Machinery, New York, pp 5686–5697

Lawler EL, Wood DE (1966) Branch-and-bound methods: a survey. Oper Res 14(4):699–719

Lu Q, Cui Z, Chen Y, Chen X (2017) Extracting optimal actionable plans from additive tree models. Front Comput Sci 11(1):160–173

Lucic A, Oosterhuis H, Haned H, de Rijke M (2019) Focus: flexible optimizable counterfactual explanations for tree ensembles. arXiv preprint arXiv:1911.12199

Lv Q, Chen Y, Li Z, Cui Z, Chen L, Zhang X, Shen H (2018) Achieving data-driven actionability by combining learning and planning. Front Comput Sci 12(5):939–949

Miettinen K (1999) Nonlinear multiobjective optimization. Springer, New York

Mothilal RK, Sharma A, Tan C (2020) Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the 2020 conference on fairness, accountability, and transparency, pp 607–617

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830

Poyiadzi R, Sokol K, Santos-Rodriguez R, De Bie T, Flach P (2020) Face: feasible and actionable counterfactual explanations. In: Proceedings of the AAAI/ACM conference on AI, ethics, and society, pp 344–350

Rudin C (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat Mach Intell 1(5):206–215

Smith JW, Everhart JE, Dickson W, Knowler WC, Johannes RS (1988) Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Proceedings of the annual symposium on computer application in medical care. American Medical Informatics Association, p 261

Subramani S, Wang H, Balasubramaniam S, Zhou R, Ma J, Zhang Y, Whittaker F, Zhao Y, Rangarajan S (2016) Mining actionable knowledge using reordering based diversified actionable decision trees. In: Web information systems engineering—WISE 2016. Springer, Cham, pp 553–560

Sylva J, Crema A (2004) A method for finding the set of non-dominated vectors for multiple objective integer linear programs. Eur J Oper Res 158(1):46–55. https://doi.org/10.1016/S0377-2217(03)00255-8

Tolomei G, Silvestri F, Haines A, Lalmas M (2017) Interpretable predictions of tree-based ensembles via actionable feature tweaking. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining part F1296, pp 465–474

Ustun B, Spangher A, Liu Y (2019) Actionable recourse in linear classification. In: FAT* 2019—proceedings of the 2019 conference on fairness, accountability, and transparency, association for computing machinery, New York, pp 10–19

Wachter S, Mittelstadt B, Russell C (2018) Counterfactual explanations without opening the black box: automated decisions and the GDPR. Harv J Law Technol 31(2):841–887

Wellawatte GP, Seshadri A, White AD (2022) Model agnostic generation of counterfactual explanations for molecules. Chem Sci 13:3697

Yang Q, Yin J, Ling CX, Chen T (2003) Postprocessing decision trees to extract actionable knowledge. In: Proceedings—IEEE international conference on data mining, ICDM 1, pp 685–688. https://doi.org/10.1109/icdm.2003.1251008

Yang Q, Yin J, Ling C, Pan R (2007) Extracting actionable knowledge from decision trees. IEEE Trans Knowl Data Eng 19(1):43–55

Yang C, Street WN, Robinson JG (2012) 10-year CVD risk prediction and minimization via inverse classification. In: IHI'12—Proceedings of the 2nd ACM SIGHIT international health informatics symposium. Association for Computing Machinery, New York, pp 603–609

Yang F, Alva SS, Chen J, Hu X (2021) Model-based counterfactual synthesizer for interpretation. In: Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining, pp 1964–1974