

GLANCE: Global Actions in a Nutshell for Counterfactual Explainability

Ioannis Emiris^{1,2}, Dimitris Fotakis^{3,4}, Giorgos Giannopoulos²,
 Dimitrios Gunopulos¹, Loukas Kavouras^{2,*}, Kleopatra Markou¹,
 Eleni Psaroudaki^{3,2}, Dimitrios Rontogiannis², Dimitris Sacharidis⁵,
 Nikolaos Theologitis², Dimitrios Tomaras⁶, and Konstantinos
 Tsopelas²

¹*Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Athens, Greece*

²*Institute for the Management of Information Systems, Athena Research Center, Athens, Greece*

³*Department of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece*

⁴*Archimedes Research Unit, Athena Research Center, Athens, Greece*

⁵*Université Libre de Bruxelles, Bruxelles, Belgium*

⁶*Department of Informatics, Athens University of Economics and Business, Athens, Greece*

* *Corresponding author: Loukas Kavouras, kavouras@athenarc.gr*

May 30, 2024

Abstract

Counterfactual explanations have emerged as an important tool to understand, debug, and audit complex machine learning models. To offer global counterfactual explainability, state-of-the-art methods construct summaries of local explanations, offering a trade-off among conciseness, counterfactual effectiveness, and counterfactual cost or burden imposed on instances. In this work, we provide a concise formulation of the problem of identifying global counterfactuals and establish principled criteria for comparing solutions, drawing inspiration from Pareto dominance. We introduce innovative algorithms designed to address the challenge of finding global counterfactuals for either the entire input space or specific partitions, employing clustering and decision trees as key components. Additionally, we conduct a comprehensive experimental evaluation, considering various instances of the problem and comparing our proposed algorithms with state-of-the-art methods. The results highlight the consistent capability of our algorithms to generate meaningful and interpretable global counterfactual explanations.

1 Introduction

Complex machine learning models are used in critical real-world decision-making applications, such as loan approvals or hiring. This creates the need to explain these models and provide users with insights into how they should act to obtain the desired prediction results [15]. Counterfactual explanations have gathered extensive attention as they can be interpretable [22], actionable [20], and suitable as a means to achieve algorithmic recourse [9] and audit for model fairness [19]. Put simply in the context of a classification task, a counterfactual explanation describes changes to the feature values, which we hereafter call an *action*, that one needs to apply to reverse an unfavorable decision. We hereafter simply consider the positive as the favorable class.

By definition, a counterfactual explanation applies for a particular negative instance and thus offers *local* explainability. However, there are several use cases that would benefit from a *global counterfactual explainability* method, meaning that it can counterfactually explain any negative instance from the model’s feature space. Trivially, the set of all local counterfactual explanations can serve this function, but at the cost of being non-interpretable and thus self-defeating. We define global counterfactual explainability as a small set of *global actions* such that they collectively cover the entire feature space.

Global counterfactual explainability can assist for example in: (1) model understanding and debugging, by presenting contrastive evidence and highlighting the main causal associations encoded by the model [18, 12]; (2) policy making, by suggesting horizontal actions or interventions towards recourse [8]; (3) fairness auditing, by comparing recourse options for subpopulations [20, 19, 10].

In this work, we further distinguish between two types of global counterfactual explainability. In a *counterfactual summary*, there is no explicit connection between instances and actions, implying that each instance can achieve recourse (if possible) through any global action. The goal here is to condense all local counterfactual explanations down to a small set of actions. In a *counterfactual segmentation*, the feature space is segmented into interpretable partitions and each global action applies to all instances within a partition, creating thus an explicit connection between a partition and a global action. The goal here is to find actions that lead to recourse for negative instances and best segment the feature space.

Finding a compact set of actions is a challenging problem that inherently involves trade-offs. In addition to small cardinality, the selected action sets are ideally characterized by (1) high *effectiveness*, i.e., act as counterfactual explanations for numerous instances (also known as correctness in [18], and coverage in [12]), and (2) low *cost*, i.e., entail few and small changes to instances. At one extreme, the set of local counterfactuals theoretically¹ achieves the best cost and effectiveness, at the expense of having cardinality proportional to the data size. For global counterfactual explainability, concessions are necessary.

¹In practice, model agnostic counterfactuals are discovered by heuristic search and may not be cost-optimal.

ARES [18] sets as its objective a weighted combination of cardinality, cost, and effectiveness, and asks the user to decide what are meaningful weights. As a result, the global counterfactual explanations derived by ARES are highly sensitive to these weights, and their cost and effectiveness vary greatly. Similarly, CET [8] defines a linear combination of cost and effectiveness, while constraining cardinality, and implements a heuristic search to identify a good segmentation of the feature space. Consequently, the quality of the action set varies greatly. GLOBE-CE [12] aims for high effectiveness primarily and low cost secondarily, as it finds a few highly effective actions, and then scales them down to minimize cost. GLOBE-CE trades off cardinality for cost, since scaling down a single effective action results in a large number of micro-actions.

We introduce a novel approach to global counterfactual explainability, termed Global Actions in a Nutshell for Counterfactual Explainability (GLANCE). Figure 1 presents a counterfactual summary consisting of three global actions derived by GLANCE for a Logistic Regression classifier trained over the HELOC dataset. Table 1 quantitatively compares the action sets derived by GLANCE with those from local counterfactual explainability (Local CE) and by GLOBE-CE. Note that Local CE returns an action for each instance and achieves perfect effectiveness and good cost. GLOBE-CE can improve on the cost with a small drop in effectiveness while bringing down the number of actions to half. Nonetheless, the action set of GLOBE-CE is long and unsuitable for inspection and interpretation by humans. In contrast, GLANCE offers an easily interpretable action set of cardinality three (Figure 1) that has optimal effectiveness with a negligible hit in cost.

Contribution On the conceptual side, the main contribution of this work is that we provide a concise formulation for the problem of finding explainable, acceptable in terms of cost, and highly effective global counterfactuals considering (1) the entire input space (2) predefined partitions of the input space. Additionally, we establish principled criteria for comparing solutions to this problem, based on Pareto dominance. These criteria provide a systematic approach for evaluating and comparing different solutions.

On the technical side, in GLANCE framework, we propose two algorithms for finding counterfactual summaries. These algorithms build on two simple observations, (a) similar instances can be explained by similar actions, and (b) similar actions can be replaced by a single action with minimal impact on cost and effectiveness. Therefore, the algorithms perform agglomerative clustering considering distances both in the feature space and the action space. Furthermore, we propose an algorithm for counterfactual segmentation. The idea is to iteratively divide the space, finding a split that improves the aggregate effectiveness.

The benefits of GLANCE over the state-of-the-art are experimentally validated by considering four widely used datasets and three different classification models.

ACTIONS:

- (1) NumInqLast6Mexcl7days+9.1
- (2) NumInqLast6Mexcl7days+13.87 & NumSatisfactoryTrades -0.75
- (3) NumInqLast6Mexcl7days+10.27

Figure 1: Suggested actions (Algorithm 1) for HELOC dataset-LR model.

Table 1: Cardinality, cost, and effectiveness of the action sets generated by different methods for the HELOC dataset. "LOCAL CE" refers to local counterfactuals generated from DiCE. [16]

METHOD	ACTIONS	COST	EFFECTIVENESS
ALGORITHM 1	3	1.38	100.00%
GLOBE-CE	588	0.41	99.90%
LOCAL CE	1045	1.16	100.00 %

Structure Section 2 overviews related work. Section 3 formulates global counterfactual explainability, and Sections 4 and 5 introduce our algorithms for counterfactual summaries and segmentation, respectively. Section 6 presents an evaluation framework, and Section 7 presents experimental results. Section 8 draws a conclusion, and Section 9 discusses the impact of our work.

2 Related Work

Local Counterfactual Explanations In the last years, there has been a plethora of work in explainable machine learning (XAI) [2], with an emphasis on counterfactual explanations [21]. An overview of methods on algorithmic recourse, which aims to provide explanations and recommendations to individuals unfavorably treated by automated decision-making systems can be found in the survey by Karimi et al. [9]. These methods can be model-agnostic or model-specific (e.g., for trees [4]); they may focus on different properties (e.g., the diversity of explanations [16], the feasibility of explanations [20], etc.); and can be either local or global.

Global Counterfactual Explanations Various frameworks have been proposed to offer global explanations. A common approach provides synthesizing or aggregating local explanations to generate global explanations [17, 13, 6]. Lakkaraju et al. [11] aims to explain how a model behaves in subspaces characterized by certain features of interest.

Building on this work, Rawal and Lakkaraju [18] introduced a model agnostic framework called Actionable Recourse Summaries (ARes) to construct global counterfactual explanations that provide an interpretable and accurate summary

of recourses for the entire population, utilizing a two-level decision set description. Additionally, Kanamori et al. [8] proposed a framework called Counterfactual Explanation Tree (CET) that assigns effective actions with decision trees, ensuring transparency through the interpretability of decision trees and consistency through the discrete partitions of the decision tree. Despite their interpretability, both AReS and CET fail to address the trade-off between cost and effectiveness.

To overcome this trade-off, Ley et al. [12] defines global counterfactual explanations as a comprehensive direction in which a cluster of inputs can collectively adjust their predictions. They suggest a framework called Global and Efficient Counterfactual Explanations (GLOBE-CE), which is the first framework to provide global explanations in a computationally efficient manner. Although it generally achieves high effectiveness (coverage) and low cost, the global explanation consists of a high number of micro-actions, one for each individual, potentially failing the requirement for global counterfactuals to consist of a limited number of actions. Finally, Kavouras et al. [10] suggests auditing subgroup fairness through counterfactual explanations, proposing a framework called Fairness Aware Counterfactuals for Subgroups (FACTS), which formulates the difference in the difficulty of groups to achieve recourse as a metric of bias.

3 Preliminaries and Problem Formulation

We first provide basic definitions and then formulate the two global counterfactual explainability problems.

3.1 Preliminaries

We consider a **feature space** $F^d = F_1 \times \dots \times F_d$ and a binary **classifier** $h : F^d \rightarrow \{-1, 1\}$ where the positive outcome is the favorable one. For a given h , we focus on the dataset $X_{\text{aff}} \subseteq F^d$ of **adversely affected individuals**, i.e., those who receive the unfavorable outcome.

We denote as \mathcal{A} the set of all possible actions, where an **action** $a \in \mathcal{A}$ is a set of changes to feature values, e.g., $a = \{\text{country} \rightarrow \text{US}, \text{education-num} \rightarrow +2\}$, which, when applied to an instance $x \in X_{\text{aff}}$, results in a **counterfactual** instance $x' = a(x)$. Every action a has a cost, denoted as $\text{cost}(a, x)$, and is effective for an instance x if $h(a(x)) = h(x') = 1$. The recourse cost $\text{rc}(A, x)$ of an instance x is the minimum cost incurred from an effective action in $A \subseteq \mathcal{A}$:

$$\text{rc}(A, x) = \min\{\text{cost}(a, x) \mid a \in A : h(a(x)) = 1\}$$

Let $X = \{x \in X_{\text{aff}} \mid h(a(x)) = 1, a \in A\}$ be the set of instances that flip their prediction by using one of the actions in A , then the effectiveness of the actions for the affected instances is defined as:

$$\text{eff}(A, X_{\text{aff}}) = \frac{|X|}{|X_{\text{aff}}|},$$

and their cost as:

$$\text{avc}(A, X_{\text{aff}}) = \frac{\sum_{x \in X} \text{rc}(A, x)}{|X|}.$$

Finally, let $\text{size}(A)$ be a function returning the cardinality of a set A .

3.2 Problem Formulation

Initially, we define the challenge of identifying global counterfactuals, anchoring the definition in the essential properties demanded by existing state-of-art works, such as Rawal and Lakkaraju [18], Kanamori et al. [8], Ley et al. [12].

Problem 1 (Global Counterfactuals). *A global counterfactual for X_{aff} is a set $S \subseteq \mathcal{A}, S \neq \emptyset$ that represents a solution to the following multi-objective optimization problem:*

1. minimize $\text{size}(S)$
 $S \subseteq \mathcal{A}$
2. minimize $\text{avc}(S, X_{\text{aff}})$
 $S \subseteq \mathcal{A}$
3. maximize $\text{eff}(S, X_{\text{aff}})$
 $S \subseteq \mathcal{A}$

A good solution to Problem 1 is a small set of actions that, when employed collectively, have maximum effectiveness and minimal cost. The requirement for a small set of actions is to enhance the interpretability of the explanation. This can be also expressed as a constraint to the set size that we can afford. In this case, the first objective of Problem 1 is replaced by the constraint

$$\text{size}(S) \leq t,$$

where t is a small positive integer.

Nevertheless, numerous practical scenarios demand tailored actions for each subpopulation. Motivated by this fact, we introduce a specialization of the problem, aiming to identify impactful solutions for distinct segments within the affected population.

Problem 2 (Global Counterfactuals for Partition). *Given a subset of the features F , a set of global counterfactuals for a partition is a tuple (P, A) , where $P = \{P_1, P_2, \dots, P_t\}$ partitions X_{aff} based on the features in F and $A = \{a_1, a_2, \dots, a_t\}$ is a set of t counterfactual actions, where each a_i is uniquely assigned to the partition P_i and represents a solution to the following multi-objective optimization problem:*

1. minimize $\text{avc}(a, P_i)$
 $a \subseteq \mathcal{A}$
2. maximize $\text{eff}(a, P_i)$
 $a \subseteq \mathcal{A}$

Given the emphasis of Problem 2 on specific subgroups within the affected population, it is crucial to grant the company the flexibility to determine the targeted features. Therefore, we provide the user with the ability to select the desired features F , which will define the partition. Finally, in accordance with [8], each subgroup (partition) should be linked to a single action, promoting both consistency and transparency.

4 Algorithms for Problem 1

We address Problem 1 employing an agglomerative approach. The core idea is to establish multiple small clusters and determine representative actions for each. The initial clusters will undergo merging using one of our two proposed procedures until the predetermined number of desired clusters is achieved. Ultimately, from each cluster, a single optimal action will be extracted with universal applicability across all instances. Intuitively, we want to exploit the idea that “*similar individuals are expected to achieve recourse with similar counterfactual actions*”, and use the notion of proximity in both the original feature space and the action space.

The first algorithm, denoted as **Iterative Merges** and outlined in Algorithm 1, follows a procedure primarily designed to enhance the effectiveness metric. Conversely, the second algorithm, named **Augmented Space** and described in Algorithm 2, adopts a strategy focused on optimizing the cost metric. Both algorithms undergo two phases. The initial phase, common to both methods, involves the generation of multiple small clusters and corresponding counterfactual actions for each cluster. The subsequent phase consists of merging the initial clusters down to a predefined small number and extracting a single optimal action from each of the merged clusters.

Phase 1: Initial clusters and actions generation. The primary objectives of the first phase, which is common in Algorithm 1 and Algorithm 2, are twofold: (1) to initiate actions from various widely distributed points within the input space, and (2) to guide these actions towards diverse directions leading to the decision boundary. To realize these objectives, the input space is partitioned into k clusters (line 2, Algorithms 1 and 2), and a local counterfactual method is employed to generate m counterfactual actions for each cluster, resulting in a total of km actions. This process can be executed by either *sampling* m instances from each cluster and applying the local counterfactual method to each sample within the cluster, or by computing the *centroid* of each cluster and generating m counterfactuals for each centroid (as described in line 4, Algorithms 1 and 2).

Phase 2: Merging and final actions extraction. The goal of this phase is to identify a set of effective actions with low average cost, and a small predefined size as a solution to Problem 1, based on the output of Phase 1. To achieve this, two strategies are employed, each using a distinct approach to merge the initial k clusters into a smaller set of t clusters, from which a single optimal action is extracted for each cluster. Both methods aim to generate solutions with optimal effectiveness and average cost, with the **Iterative Merges** strategy prioritizing effectiveness, and the **Augmented Space** strategy prioritizing average cost.

Given the initial k clusters and the set of km actions, the **Iterative Merges** strategy combines clusters based on both their inherent proximity and the proximity of their respective actions. In each iteration, we calculate, for every pair of clusters, the sum of the distance between their centroids and the distance

Algorithm 1 Iterative Merges

- 1: **Input:** X_{aff} , number of local counterfactuals m , number of initial clusters k , number of final clusters t
 - 2: **Output:** t global counterfactuals
 - 3: Cluster X_{aff} into k clusters.
 - 4: For each cluster, generate m counterfactuals from the cluster centroid.
 - 5: While the number of clusters exceeds t :
 - 6: For each pair of clusters:
 - 7: Compute the distance d between their centroids and the distance d' between the centroids of their associated counterfactual instances.
 - 8: Merge the two clusters exhibiting the minimum distance sum $d + d'$ and inherit their counterfactual actions to the merged cluster.
 - 9: Return the single optimal action from each of the t clusters.
-

of the centroids of their associated counterfactuals (which are generated from the cluster centroids) (lines 6–7, Algorithm 1). Subsequently, the two clusters that minimize this combined distance metric are merged, concluding the iteration (line 8, Algorithm 1). The algorithm continues until the initial k clusters are merged into t clusters. Subsequently, a single optimal action is extracted from each of these clusters, and the algorithm returns the set of these t counterfactual actions (line 9, Algorithm 1).

The second algorithm, named **Augmented Space**, transforms the initial k clusters utilizing their respective m actions through the following procedure: For each instance, it identifies the single action from its cluster that flips its class to the positive and has the minimum cost² (lines 6–7, Algorithm 2). The feature vector of the instance is then concatenated with the feature vector of its action (line 8, Algorithm 2), resulting in a new, augmented space. Subsequently, a clustering algorithm is applied to the augmented space, producing t clusters (line 9, Algorithm 2). These t clusters consist of instances that include their minimal cost-effective action in their feature vector. Consequently, we can extract the unique actions from each cluster and select the most effective for each cluster (line 10, Algorithm 2).

Iterative Merges prioritizes effectiveness by merging clusters that are close in the input space and possess close counterfactual actions. This ensures that when two clusters are merged, their combined action sets will retain the most effective actions for the merged cluster. On the other hand, **Augmented Space** prioritizes average cost, as each original instance is mapped to a new instance incorporating the minimum cost action in its feature vector. Therefore, the initial set of km actions (m per cluster) will be filtered to arrive at a smaller set of $l < km$ lower-cost actions and the selection of each cluster action will be among these filtered actions.

²The feature vector of the action has d entries, where each entry represents the value of the change at the respective feature, e.g., $(0, +1, 0)$ means increasing the second feature by one. If no cluster action flips the class of the instance, then the feature vector of the action for this instance has zero values in all entries.

Algorithm 2 Augmented Space

- 1: **Input:** X_{aff} , number of local counterfactuals m , number of initial clusters k , number of final clusters t
 - 2: **Output:** t global counterfactuals
 - 3: Cluster X_{aff} into k clusters.
 - 4: For each cluster, generate m counterfactuals from the cluster centroid.
 - 5: For each instance x_i :
 - 6: Find the cluster actions A_i that flip its prediction.
 - 7: Find minimum cost action a_i from A_i .
 - 8: Concatenate x_i and a_i .
 - 9: Cluster the augmented data into t clusters.
 - 10: Return the single optimal action from each of the t clusters.
-

5 Algorithm for Problem 2

We address the special case of Problem 2 using a divisive approach, named **Counterfactual Tree** and outlined in Algorithm 3. This algorithm starts with a single cluster containing all affected instances and recursively divides it into smaller clusters defined by subsets of the input features. The decision to split is determined by comparing the actions of the parent node with those of the child nodes.

Actions are generated at each node of the tree, starting with the root. Actions for the root are generated using either the sampling or the centroid approach, with the best action retained based on the selected criterion (line 3, Algorithm 3). The root then becomes the parent node, and each feature’s splits are examined: for every feature, the node is divided into subgroups based on the feature’s values. Actions are generated for each subgroup, and their average score, based on a predefined criterion, is calculated (line 4, Algorithm 3). The subgroups associated with the feature yielding the maximum average score are retained. If this maximum score is greater than the score of their parent node, these subgroups are added as nodes to the tree (line 6, Algorithm 3). The new nodes become the next parent nodes, and the feature used to produce them is removed (line 8, Algorithm 3). The process continues until a node has no more features to examine or until the maximum score of the actions from all remaining features is less than or equal to the score of its parent node (line 7, Algorithm 3).

Notice that, from the description above, the split does not continue if it can not find better (based on the criterion) counterfactual actions for the child nodes. This informs us that a more refined partition does not offer better performance based on the criterion. However, this is not a limitation for our algorithm since it is straightforward to continue the splits at each level by just asking to split based on the feature with the maximum score even if it does not surpass the score of the parent node. Moreover, given the speed of our method compared to similar methods [8], as seen in Section 7.3, we can also compute the optimal tree based on the performance of the leaves towards the criterion.

Algorithm 3 Counterfactual Tree

- 1: **Input:** X_{aff} , a set of features for splitting F , number of local counterfactuals m .
 - 2: **Output:** A counterfactual tree where each node has an action assigned.
 - 3: Generate m actions for the root and keep the one maximizing the score of a selected criterion.
 - 4: Explore all candidate splits in the feature set F .
 - 5: Select the split that produces the maximum average score, surpassing that of the root.
 - 6: If no such split is identified, return the root and the maximum score action.
 - 7: Otherwise, recursively generate counterfactual trees for subgroups derived from the chosen split, excluding the selected split feature from F .
 - 8: Consider the newly constructed counterfactual trees as children of the root node.
 - 9: Return the counterfactual tree.
-

6 Solution Evaluation

Finding a solution to Problem 1 or Problem 2 that optimizes all objectives simultaneously is not always feasible. The effectiveness and average cost of an action set may only improve with the addition of new actions, resulting in an increase in the set size, which is unfavorable. Moreover, instances that are far away from the decision boundary need costly actions to flip their class, so an increase in effectiveness in order to flip these points will also increase average cost (and set size). Therefore, an optimal solution to these problems refers to a set of values that represents the best possible compromise among these conflicting objectives of the problem, i.e., Pareto optimal solutions, where no other solution is better in all objectives simultaneously. The next two subsections formulate definitions that compare action sets and determine their quality as a solution to Problem 1 and Problem 2.

6.1 Evaluation for Problem 1

We use the following definitions to evaluate Algorithm 1 and Algorithm 2 towards GLOBE-CE and dGLOBE-CE in Section 7.2.

Definition 3 (Strong dominance). An action set A *strongly dominates* an action set A' if and only if:

1. $\text{size}(A) < \text{size}(A')$
2. $\text{eff}(A, X_{\text{aff}}) > \text{eff}(A', X_{\text{aff}})$
3. $\text{avc}(A, X_{\text{aff}}) < \text{avc}(A', X_{\text{aff}})$

Definition 3 is straightforward; we will choose A over A' if it is better on all metrics.

Definition 4 (Size dominance). An action set A *size dominates* an action set A' if and only if:

1. $\text{size}(A) < \text{size}(A')$
2. $\text{eff}(A, X_{\text{aff}}) \geq \text{eff}(A', X_{\text{aff}})$

$$3. \text{avc}(A, X_{\text{aff}}) \leq \text{avc}(A', X_{\text{aff}})$$

Definition 4 states that if an action set A is at least as good as A' in terms of effectiveness and average cost and is smaller, then A is considered superior. It is worth noting that if A *strongly dominates* A' , then A also *size dominates* A' .

Definition 5 (Dominance). An action set A *dominates* an action set A' if and only if $\text{size}(A) = \text{size}(A')$ and:

1. $\text{eff}(A, X_{\text{aff}}) > \text{eff}(A', X_{\text{aff}})$
2. $\text{avc}(A, X_{\text{aff}}) < \text{avc}(A', X_{\text{aff}})$

The last definition applies to action sets of the same size. If an action set A is better in terms of effectiveness and average cost against an other set A' of the same size, then one should prefer A over A' .

Definition 6 (Weak Dominance). An action set A *weakly dominates* another action set A' if and only if A it is at least as good as A' in two objectives and exhibits a strictly superior performance in the third one.

Observe, that if an action set A *strongly dominates*, or *size dominates* or *dominates*, an action set A' then A *weakly dominates* A' . If none of the definitions above applies, then we say that A is *competitive* with A' .

6.2 Evaluation for Problem 2

We will use the following definition to evaluate Algorithm 3 towards CET in Section 7.2. Let (P, A) be a solution for Problem 2, with $P = (P_1, P_2, \dots, P_t)$ and $A = \{a_1, a_2, \dots, a_t\}$. Then, the weighted average effectiveness of A is

$$\bar{\text{eff}}(A, P) = \frac{\sum_{i=1}^t |P_i| \cdot \text{eff}(a_i, P_i)}{|X_{\text{aff}}|}$$

and the weighted average cost is

$$\text{a}\bar{\text{v}}\text{c}(A, P) = \frac{\sum_{i=1}^t \text{avc}(a_i, P_i) \cdot |P_i| \cdot \text{eff}(a_i, P_i)}{\sum_{i=1}^t |P_i| \cdot \text{eff}(a_i, P_i)}$$

Utilizing the notation mentioned above, we derive a criterion to assess the quality of our solution, focusing on the performance of our algorithm within each block of the partition. The weighted average effectiveness and weighted average cost aggregate the quality of the solution at each leaf of the partition, enabling comparison across solutions with different numbers of leaves and different segments of affected instances.

Definition 7 (Tree Dominance). Let (P, A) and (P', A') be two solutions for Problem 2. Then, (P, A) *tree dominates* (P', A') if

$$\bar{\text{eff}}(A, P) \geq \bar{\text{eff}}(A', P), \text{a}\bar{\text{v}}\text{c}(A, P) < \text{a}\bar{\text{v}}\text{c}(A', P)$$

or if

$$\bar{\text{eff}}(A, P) > \bar{\text{eff}}(A', P), \text{a}\bar{\text{v}}\text{c}(A, P) \leq \text{a}\bar{\text{v}}\text{c}(A', P)$$

Definition 7 deems an action set A superior to A' if A is at least as good as A' in one objective and outperforms A' in the other objective.

7 Experimental Evaluation

Section 7.1 describes the experimental setup, while Sections 7.2 and 7.3 present experimental results for the two global counterfactual explainability problems.

7.1 Experimental Setting

In our experimental evaluation, we employ four publicly accessible and widely used **benchmark datasets** that focus on binary classification problems: COMPAS [1], German Credit [5], Default Credit [23], and HELOC [3]. These datasets were utilized in previous research by Rawal and Lakkaraju [18] and Ley et al. [12]. Further details on the datasets and the preprocessing can be found in Appendix A.

In the context of model training, we opted for three different model types: XGBoost (XGB), Logistic Regression (LR), and Deep Neural Network (DNN). Throughout all conducted experiments, our approach strictly followed the dataset preprocessing methodology outlined by Ley et al. [12]. Moreover, we maintained consistency in experimental conditions by employing the same hyperparameters for model training as used by Ley et al. [12]. The hyperparameters utilized during model training and statistics on the training accuracy are provided in Appendix B.

Table 2: Subset of the comparative results of our Algorithms 1 and 2 (ITERATIVE MERGES and AUGMENTED SPACE) with various configurations of GLOBE-CE, i.e., the default version of GLOBE-CE (GLOBE-CE), GLOBE-CE with a maximum of three micro-actions (scalars) (GLOBE-CE), dGLOBE-CE with a maximum of three directions (dGLOBE-CE), and dGLOBE-CE with a maximum of three directions and a maximum of three scalars per direction (dGLOBE-CE (MAX 3 SCALARS)). For all the model-dataset combinations see Tables 9 to 12.

METHOD	HELOC DNN				German Credit LR				Default Credit XGB			
	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME
GLOBE-CE	100.00 %	7.75	403	4.62 SEC	72.41 %	1.28	3	3.44 SEC	62.82 %	0.42	14	19.63 SEC
GLOBE-CE (3 SCALARS)	100.00 %	18.58	3	9.47 SEC	72.41 %	1.28	3	0.65 SEC	51.99 %	42.6	3	21.68 SEC
dGLOBE-CE	100.00 %	2.93	893	8.68 SEC	79.31 %	1.19	6	6.32 SEC	89.46 %	0.43	128	44.74 SEC
dGLOBE-CE (3 SCALARS)	99.90 %	5.98	4	6.59 SEC	75.86 %	2.06	2	2.22 SEC	51.99 %	64	1	11.2 SEC
AUGMENTED SPACE	99.81 %	4.94	3	71.49 SEC	98.25 %	1.82	3	24.12 SEC	90.31 %	1.81	3	212.13 SEC
ITERATIVE MERGES	99.91 %	8.55	3	38.96 SEC	100.00 %	1.21	3	9.64 SEC	95.44 %	3.17	3	102.48 SEC

For the initialization of Algorithms 1 and 2 we use k -means with an initial number of clusters appropriate for each dataset depending on the affected population, and generate actions employing DiCE [16]. Therefore our cost is highly dependent on the cost of the actions produced by DiCE. Regarding the partitioning of Algorithm 3, it is worth noting that the flexibility inherent in our

Table 3: Subset of the comparative results of our Algorithm 3 (CT) with CET. We run both algorithms with the same limitations on the maximum number of leaves. For Algorithm 3 the maximum number of features came naturally from the pre-determined features that were used for splits, 4 (CF TREE - 4) and 8 (CF TREE - 8) leaves accordingly. For CET we applied the restriction for a maximum number of leaves to be in accordance with Algorithm 3 (CET-4 and CET-8). For the model-dataset combinations see Tables 14 to 17.

METHOD	HELOC DNN				German Credit LR				Default Credit XGB			
	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME
CET - 4	47.34%	12.69	3	13662.29 SEC	86.21%	2.32	3	91.91 SEC	45.16%	9.56	2	5324.21 SEC
CF TREE - 4	97.61 %	10.88	4	156.10 SEC	100.00 %	5.39	4	18.17 SEC	79.00 %	3.34	4	257.94 SEC
CET - 8	32.03%	15.90	2	11347.89 SEC	91.38%	2.52	6	72.28 SEC	39.17%	9.00	1	3119.20 SEC
CF TREE - 8	90.12 %	12.78	6	277.75 SEC	100.00 %	5.52	8	20.16 SEC	87.75 %	3.76	6	180.95 SEC

methodology allows for the unrestricted selection of features for splitting. For the hyperparameters, chosen features for splitting, and additional details, we direct the reader to Appendix C.

As for the computation of an **action’s cost**, we follow the paradigm outlined in Ley et al. [12], by binning continuous features into 10 equal intervals post-training and scaling the cost of each change proportionally to the bin length. We also set the cost of moving from one category to another for the categorical features to be 1. To ensure the **reproducibility** of our results, a consistent random seed of 13 is applied across all models and algorithms. The code for the reproduction of our results can be found in the supplementary material.

7.2 Results for Problem 1

For Problem 1 we evaluate our algorithms in comparison with GLOBE-CE, the state-of-the-art method for generating global counterfactuals. Specifically, we access Algorithms 1 and 2 against various configurations of GLOBE-CE, including the default version of GLOBE-CE, GLOBE-CE with a maximum of three micro-actions (scalars), dGLOBE-CE with a maximum of three directions, and dGLOBE-CE with a maximum of three directions and a maximum of three scalars per direction. Compared to GLOBE-CE, our algorithms demonstrate slightly extended runtimes, however they remain computationally efficient. Our primary evaluation criteria encompass the assessment of cost, effectiveness, and the number of actions.

Table 2 presents only a subset of our experimental results. For a comprehensive overview of results across all dataset-model combinations, we refer the reader to Appendix D. In the case of the HELOC dataset with the DNN model, we observe a notable improvement in cost for both of our algorithms compared to the vanilla version of GLOBE-CE, which utilizes 403 micro-actions in a single direction. Moreover, GLOBE-CE’s average cost tends to rise as the number of micro-actions decreases, since the magnitude of each action increases. Contrasting this, dGLOBE-CE achieves a reduced cost by employing 893 micro-actions in

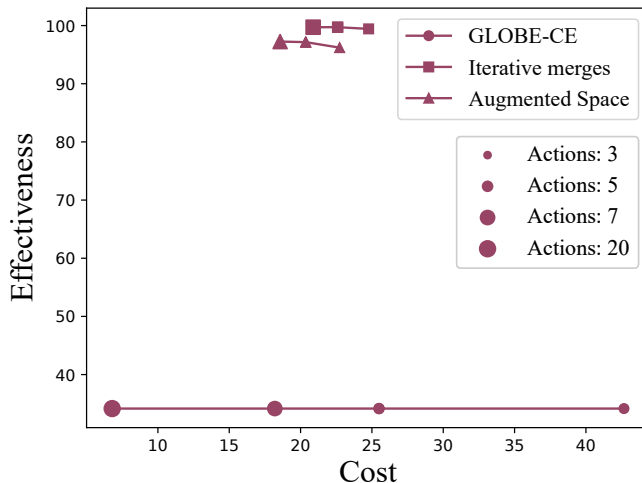


Figure 2: Effectiveness - Cost plot for XGBoost-HELOC combination. We present the results for Algorithms 1 and 2 (3, 5, and 7 actions) and for GLOBE-CE with a constrained number of micro-actions, (3, 5, 7, and 20, where 20 corresponds to the unconstrained version for the specific dataset-model combination).

three directions, whereas our algorithms demonstrate comparable performance with merely three actions.

When considering the German Credit dataset with the LR model, it becomes evident that when only a small number of actions are required for both GLOBE-CE and dGLOBE-CE, our algorithms can achieve similar costs with GLOBE-CE and significantly larger levels of effectiveness.

In the Default Credit dataset - XGB model combination, GLOBE-CE exhibits suboptimal effectiveness when employing a single direction, despite achieving a modest cost reduction. With only three micro-actions in this configuration, the cost increases from 0.42 to 42.6. In contrast, our proposed solutions in Algorithm 1 and Algorithm 2 exhibit significantly reduced costs, specifically 13 and 23 times lower, respectively. In the case of dGLOBE-CE utilizing three directions, we attain a similar level of effectiveness with only 3 actions (vs 128), while restricting dGLOBE-CE to a maximum of three scalars per direction leads to 20 to 35 times higher cost than our proposed solutions.

In Figure 2, the distinction between our methodologies and GLOBE-CE becomes evident within the HELOC dataset using the XGBoost model, when both approaches utilize an identical number of actions. Similar behavior is observed when limiting dGLOBE-CE to a small number of scalars, leading to a notable increase in cost in most cases.

Finally, we use the criteria outlined in Section 6.1 to access our algorithms in comparison with GLOBE-CE and dGLOBE-CE. As depicted in Figure 3, it becomes evident that both Algorithms 1 and 2 successfully dominate GLOBE-CE as per the definitions of dominance (Definitions 3 to 6), achieving dominance 6

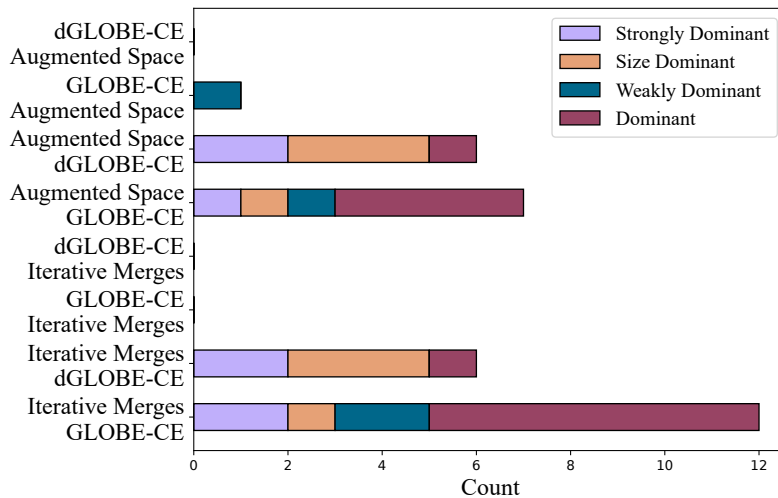


Figure 3: Visual representation of the evaluation of Algorithms 1 and 2 vs GLOBE-CE and dGLOBE-CE versions. These results correspond to the experimental results presented in Table 2 and in Appendix Tables 9 to 12.

times each. Additionally, they successfully dominate dGLOBE-CE 12 times and 7 times, respectively. On the contrary, GLOBE-CE weakly dominates Algorithm 2 in solely one dataset-model combination, while dGLOBE-CE fails to dominate both of our algorithms across all model-dataset combinations.

7.3 Results for Problem 2

In evaluating the proposed algorithm (Algorithm 3) for addressing Problem 2 we employ the CET framework. Both algorithms are executed under identical constraints concerning the maximum number of leaves (4 and 8 leaves). In the case of Algorithm 3, the predetermined features used for splits naturally dictate the maximum number of leaves as 4 (CF TREE - 4) and 8 (CF TREE - 8). For CET we directly restrict the maximum number of leaves by adjusting the relevant hyperparameters (CET-4 and CET-8).

In the examples presented in Table 3 it is evident that CET in HELOC dataset with DNN model and in DEFAULT CREDIT dataset with XGB model, proves to be inferior when compared to our proposed method. CET faces challenges in identifying a good partition of the space with high effectiveness and small cost. This struggle persists despite CET’s inherent flexibility to perform splits across the entire feature space. While CET demonstrates better performance with Logistic Regression models compared to other models, leveraging model internals, our proposed solutions still outperform CET in terms of overall performance, with higher effectiveness and competitive costs.

In Appendix E, we provide comprehensive results for all model-dataset combinations. Our findings indicate that Algorithm 3 consistently outperforms

CET in both parameterizations, considering scenarios with a maximum of 4 and 8 leaves.

8 Conclusions

This paper presents GLANCE, a novel method to provide global counterfactual explainability. GLANCE finds a small set of counterfactual actions, which collectively act as explanations to a large number of instances at a low counterfactual cost. Extensive experimental evaluation demonstrates that GLANCE is a practical method that consistently finds interpretable global counterfactual summaries of high effectiveness and low cost.

9 Impact Statement

In the field of explainability, especially when counterfactuals are produced, there are numerous challenges when someone attempts to define the “best” actions in terms of cost. This is a known and intricate problem, which is previously recognized in the bibliography [12, 10] since the cost definition is dataset- and individual-specific, giving rise to concerns of privacy breaches or manipulation of the explainability of the model. We acknowledge these difficulties, but we believe that they are out of the scope of our paper.

We also recognize the importance of global counterfactual explanations and their potential as a horizontal intervention. Therefore, we firmly believe, that work in producing globally fair actions, especially in terms of Problem 2, could be very beneficial in the advancement of the field and left for future work.

Acknowledgments and Disclosure of Funding

This work has been funded by the European Union’s Horizon Europe research and innovation programme under Grant Agreement No. 101070568 (AutoFair).

References

- [1] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. *Ethics of Data and Analytics*, pages 254–264, 5 2016. doi: 10.1201/9781003278290-37.
- [2] Francesco Bodria, Fosca Giannotti, Riccardo Guidotti, Francesca Naretto, Dino Pedreschi, and Salvatore Rinzivillo. Benchmarking and survey of explanation methods for black box models. *Data Mining and Knowledge Discovery*, pages 1–60, 2023.

- [3] Kyle Brown, Derek Doran, Ryan Kramer, and Brad Reynolds. HELOC applicant risk performance evaluation by topological hierarchical decomposition. *CoRR*, abs/1811.10658, 2018. URL <http://arxiv.org/abs/1811.10658>.
- [4] Miguel Á Carreira-Perpiñán and Suryabhan Singh Hada. Counterfactual explanations for oblique decision trees: Exact, efficient algorithms. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6903–6911, 2021.
- [5] Dheeru Dua and C Graff. Uci machine learning repository. university of california, school of information and computer science, irvine, ca (2019), 2019.
- [6] Jingyue Gao, Xiting Wang, Yasha Wang, Yulan Yan, and Xing Xie. Learning groupwise explanations for black-box models. In *IJCAI*, pages 2396–2402, 2021.
- [7] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL <https://www.gurobi.com>.
- [8] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Yuichi Ike. Counterfactual explanation trees: Transparent and consistent actionable recourse with decision trees. In *International Conference on Artificial Intelligence and Statistics*, pages 1846–1870. PMLR, 2022.
- [9] Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050*, 2020.
- [10] Loukas Kavouras, Konstantinos Tsopelas, Giorgos Giannopoulos, Dimitris Sacharidis, Eleni Psaroudaki, Nikolaos Theologitis, Dimitrios Rontogiannis, Dimitris Fotakis, and Ioannis Emiris. Fairness aware counterfactuals for subgroups. *arXiv preprint arXiv:2306.14978*, 2023.
- [11] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Faithful and customizable explanations of black box models. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 131–138, 2019.
- [12] Dan Ley, Saumitra Mishra, and Daniele Magazzeni. GLOBE-CE: A translation based approach for global counterfactual explanations. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19315–19342. PMLR, 23–29 Jul 2023. URL <https://proceedings.mlr.press/v202/ley23a.html>.
- [13] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and

- Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67, 2020.
- [14] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.
- [15] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.
- [16] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 607–617, 2020.
- [17] Dino Pedreschi, Fosca Giannotti, Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, and Franco Turini. Meaningful explanations of black box ai decision systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9780–9784, Jul. 2019. doi: 10.1609/aaai.v33i01.33019780. URL <https://ojs.aaai.org/index.php/AAAI/article/view/5050>.
- [18] Kaivalya Rawal and Himabindu Lakkaraju. Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses. *Advances in Neural Information Processing Systems*, 33:12187–12198, 2020.
- [19] Shubham Sharma, Jette Henderson, and Joydeep Ghosh. CERTIFAI: counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. *CoRR*, abs/1905.07857, 2019. URL <http://arxiv.org/abs/1905.07857>.
- [20] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 10–19, 2019.
- [21] Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan E Hines, John P Dickerson, and Chirag Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.
- [22] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.
- [23] Ivy Yeh and Che-Hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36:2473–2480, 03 2009. doi: 10.1016/j.eswa.2007.12.020.

Appendix

The appendix is formatted as follows.

1. In Appendix A we discuss the *Datasets* used in our experimental evaluation and the *Preprocessing* performed in each of them.
2. In Appendix B we discuss the *Models* used in our experimental evaluation.
3. In Appendix C we discuss all the *Hyperparameters* used in our experimental evaluation, both in our methods as well as in the state-of-art methods we utilize for comparison.
4. In Appendix D we present *Further Results on Problem 1*.
5. In Appendix E we present *Further Results on Problem 2*.

A Datasets & Preprocessing

We use four publicly available datasets as benchmarks. Our choice is based on their established use in previous works. A short description follows. Table 4 summarizes the datasets’ information, including the number of instances, the number of categorical and continuous features, the input dimensions (i.e., the number of continuous features plus the number of categorical after a preprocessing step using one-hot-encoding), and the number of instances used for train and test after the 80:20 split.

Once again, it should be mentioned that the whole of the preprocessing of the datasets (as well as the models with all their hyperparameters) are exactly the same as in Ley et al. [12], taken from the repository the authors have made public in <https://github.com/danwley/GLOBE-CE/>. Nonetheless, we provide here a brief description of each dataset, for completeness.

COMPAS The COMPAS dataset (Correctional Offender Management Profiling for Alternative Sanctions) [1] available at <https://github.com/propublica/compas-analysis/blob/master/compas-scores-two-years.csv>. Detailed description and information on the dataset can be found at <https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm>. It categorizes recidivism risk based on several factors, including race.

For the preprocessing of this dataset, we drop the “days_b_screening_arrest” feature, as it contains missing values. We also turn jail-in and jail-out dates to durations and turn negative durations to 0. Some additional filters are taken from the COMPAS analysis by ProPublica. Finally, the target variable’s values are transformed into the canonical 0 for the negative class and 1 for the positive class.

German Credit The German Credit dataset [5] classifies people described by a set of attributes as good or bad credit risks. A detailed description and the dataset can be found in [https://archive.ics.uci.edu/ml/datasets/statlog+\(german+credit+data\)](https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)).

The only preprocessing step we performed for this dataset was the transformation of the target variable’s values into 0 - 1.

Default Credit The Default Credit dataset [23] is designed to classify the risk of default on customer payments, aiming to support the development and assessment of models for predicting creditworthiness and the likelihood of loan default. It can be obtained at <https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients>.

To properly work with this dataset, we needed to drop the “ID” feature, since it holds no useful information, and transform the target labels into the canonical 0 - 1 values.

HELOC The HELOC (Home Equity Line of Credit) dataset [3] contains anonymized information about home equity line of credit applications made by real homeowners, classifying credit risk. It is available at <https://community.fico.com/s/explainable-machine-learning-challenge>. All the features on this dataset are numeric.

A substantial percentage of these features’ values are missing, so the main preprocessing step we performed here was to remove rows where all values are missing, and then replace all remaining missing values with the median of the respective feature. Other than that, we only needed to transform target labels to 0-1.

Table 4: Summary of the datasets used in our experiments. Specifically, we list the number of instances, input dimensions (i.e., the number of continuous features plus the number of categorical after a preprocessing step using one-hot-encoding), the number of categorical and continuous features, and the number of instances used for train and test after the 80:20 split.

DATASET	NO. INSTANCES	INPUT DIM.	CATEGORICAL	CONTINUOUS	TRAIN	TEST
COMPAS	6172	15	4	2	4937	1235
GERMAN CREDIT	1000	71	17	3	800	200
DEFAULT CREDIT	30000	91	9	14	24000	6000
HELOC	9871	23	0	23	7896	1975

B Models

In our experimental framework, three distinct models are employed: XGBoost (XGB), Logistic Regression (LR), and Deep Neural Networks (DNNs). We train

these models with *the same* 80:20 train-test split as Ley et al. [12]; the distinctive hyperparameters for each model were also obtained from Ley et al. [12], facilitating a standardized basis for the comparative analysis of our methodologies.

XGBoost (XGB) Implementation from the common `xgboost`³ library. Hyperparameter values for each dataset and the model’s accuracy on the test set are shown in Table 5.

Table 5: XGBoost Hyperparameter Configurations.

DATASET	DEPTH	ESTIMATORS	γ, α, λ	TEST ACCURACY
COMPAS	4	100	1,0,1	68%
GERMAN CREDIT	6	500	0,0,1	74%
DEFAULT CREDIT	10	200	2,4,1	83%
HELOC	6	100	4,4,1	74%

Logistic Regression (LR) Implementation from the common `sklearn`⁴ library. Hyperparameter values for each dataset and the model’s accuracy on the test set are shown in Table 6.

Table 6: Logistic Regression Hyperparameter Configurations.

DATASET	MAX ITER.	CLASS WEIGHTS(0:1)	TEST ACCURACY
COMPAS	1000	1:1	65%
GERMAN CREDIT	1000	1:1	76%
DEFAULT CREDIT	2000	0.65:0.35	83%
HELOC	2000	1:1	75%

Deep Neural Network (DNN) Implementation using `pytorch`⁵ library. Hyperparameter values for each dataset and the model’s accuracy on the test set are shown in Table 7.

C Hyperparameters & Implementation Details

In this section, we discuss in detail the set of parameters for each one of the algorithms examined in this work and provide the values we have used for

³<https://xgboost.readthedocs.io/en/stable/>

⁴https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

⁵<https://pytorch.org/docs/stable/index.html>

Table 7: Deep Neural Network Hyperparameter Configurations.

DATASET	WIDTH	DEPTH	DROPOUT	TEST ACCURACY
COMPAS	30	5	0.4	65%
GERMAN CREDIT	50	10	0.3	78%
DEFAULT CREDIT	80	5	0.3	81%
HELOC	50	5	0.3	74%

these parameters in our experiments. First, we describe the parameters of our proposed algorithms (Iterative Merges, Augmented Space, and Counterfactual Tree), and then we proceed with the rest of the state-of-the-art algorithms examined (GLOBE-CE and CET), as well as the algorithms we integrated into our approach (k -means and DiCE).

C.1 Iterative Merges

Our *Iterative Merges* algorithm requires the clustering of individuals in feature space and for each cluster to find some representative counterfactuals. Therefore, the first two parameters we need to provide to our algorithm are `CLUSTERS`, which denotes the number of initial clusters to be produced, and `CENTROID_CFS_NUM`, which represents the number of counterfactual explanations to be generated per cluster. To generate the counterfactuals, two sampling methods can be utilized, “`sampling`” or “`centroid`”, and this can be defined by the `SAMPLING_METHOD` parameter. Our algorithm proceeds by using a distance function to merge the clusters up to the specified number of final clusters. The parameter `THRESHOLD` sets up the number of final clusters after the merges. We also need to define the function that will be used to measure the distance from one instance to another, which can be set up by the `DIST_FUNC` parameter. To enable the reproduction of the results we use the `RANDOM_STATE` parameter.

For our experiments, we have used `CLUSTERS = 100` and `CENTROID_CFS_NUM = 10` for all datasets, except for the German Credit dataset, which contained a very small number of affected individuals, and as such the experiments were run with `CLUSTERS = 20` and `CENTROID_CFS_NUM = 50` (so that the total number of generated local counterfactuals is constant for all datasets). For the `SAMPLING_METHOD` parameter, in the experiments presented in both the main paper and the appendix, we have consistently used “`centroid`”. As for the `THRESHOLD`, it is set to 3 (which can also be observed in Appendix D since it coincides with the number of explanations given in the output). The `DIST_FUNC` is always set to the cost function we discussed in detail in Section 7.1 and `RANDOM_STATE` is always set to 13.

C.2 Augmented Space

Our *Augmented Space* algorithm shares a similar initial phase with the *Iterative Merges*, as we have also described in the main body of the paper. This involves the clustering of individuals in feature space into numerous clusters and for each cluster, selecting a limited number of representative counterfactuals. These counterfactuals are then translated into actions and applied to all the individuals in the cluster. Therefore, the first two parameters we need to provide to our algorithm are (again) `N_INITIAL_CLUSTERS`, which denotes the number of initial clusters to be produced, and `N_CLUSTER_SAMPLES`, which stands for the number of counterfactual explanations to be generated per cluster. Two sampling methods, “`sampling`” or “`centroid`”, can be utilized to generate the counterfactuals, and this can be defined by the `SAMPLING_METHOD` parameter. Our algorithm then proceeds by concatenating each instance with its cost-efficient effective action and subsequently by re-clustering these “augmented” instances. The `N_FINAL_CLUSTERS` parameter determines the number of sets of individuals for this phase. The algorithm’s output comprises these sets of individuals, in the form of clusters, and the best action (in terms of effectiveness) together with its effectiveness and cost, for each of these clusters. Additionally, the `DIST_FUNC` parameter is used to define the function measuring the cost to move from one instance to another. To enable the reproduction of the results we use the `RANDOM_STATE` parameter.

For our experiments, we have used `N_INITIAL_CLUSTERS = 100` and `N_CLUSTER_SAMPLES = 10` for all datasets, except for the German Credit dataset, where the experiments were run with `N_INITIAL_CLUSTERS = 20` and `N_CLUSTER_SAMPLES = 50` (so that the total number of generated local counterfactuals is constant for all datasets). For the `SAMPLING_METHOD` parameter, in the experiments presented in both the main paper and the appendix, we have consistently used “`centroid`”. As for the `N_FINAL_CLUSTERS`, it is once again consistently set to 3. The `DIST_FUNC` is always set to the cost function we discussed in detail in Section 7.1 and `RANDOM_STATE` is always set to 13.

C.3 Counterfactual Tree

Our *Counterfactual Tree* algorithm requires to begin with a node of all the affected individuals and produce a counterfactual for them using a set of candidate features. The output of the algorithm will be an interpretable tree, which, based on this set of candidate features, will provide recourses for the group of individuals along with the recourses’ effectiveness and cost. Therefore, the first parameter we need to provide to our algorithm is `SAMPLING_METHOD` which denotes the method to generate counterfactuals, “`sampling`” or “`centroid`”, as well as the set of candidate features to provide recourse with, which can be done by providing the `CANDIDATE_FEATS` parameter. To tune the number of random samples to choose, when using the “`sampling`” method, the parameter `SAMPLE_SIZE` must be provided. Similarly, when using the “`centroid`” method, the parameter `CENTROID_CFS_NUM` should be provided. Finally, at any given

node within the tree, when partitioning the data on a feature, our method divides it into all unique values of that feature by default. To regulate this process and consequently limit the number of children a node may have, we introduce the `CHILD_COUNT` parameter. When equal to -1, it keeps the default behavior, otherwise, each node will have `CHILD_COUNT` children.

For the experiments presented in both the main paper and the appendix the “centroid” was used as the `SAMPLING_METHOD`. The `CENTROID_CFS_NUM` parameter is always set to 30. The `CHILD_COUNT` parameter is always set to 2, to be able to produce trees with at most 4 or 8 leaves (which is how we run our experiments in Appendix E, for comparison purposes with Kanamori et al. [8]).

Finally, the `CANDIDATE_FEATS` parameter is chosen on a per-dataset basis as follows:

Table 8: COMPAS Dataset

COMPAS		GERMAN CREDIT	DEFAULT CREDIT	HELOC
MAX # LEAVES = 4	SEX, C_CHARGE_DEGREE	TELEPHONE, FOREIGN-WORKER	AGE, SEX	NUMSATISFACTORYTRADES, NUMTOTALTRADES
MAX # LEAVES = 8	SEX, C_CHARGE_DEGREE	TELEPHONE, PROPERTY	MARRIAGE, SEX	NUMTRADES60EVER2DEROGPubREC, NUMSATISFACTORYTRADES, NUMTOTALTRADES

C.4 GLOBE-CE

The state-of-the-art algorithm GLOBE-CE [12] requires the definition of several parameters, such as the number of different directions sampled, which is defined by the `N_SAMPLE` parameter; the magnitude of all those initial direction vectors is determined by the `MAGNITUDE` parameter. The number of features changed by all generated actions is defined by the `N_FEATURES` parameter.

A parameter of great importance is `N_DIV`, the number of best directions selected from the initially sampled ones. For the vanilla GLOBE-CE, this is equal to 1, i.e. it picks the direction with the highest effectiveness. For dGLOBE-CE, $d > 1$ directions are chosen.

These directions (either 1 or d) are subsequently explored by scaling each one with many scalar values. The number of different scalars to use can be defined by the `N_SCALARS` parameter.

In our experimental setup, designed to ensure a fair comparison, we standardized `N_SAMPLE` and `N_SCALARS` to 1000. In addition, we conducted experiments varying `SPARSITY_POWER` (evaluated at 1 and 5), `MAGNITUDE` (evaluated at 1 and 2), and `N_FEATURES` (examined at 5 and 2), mirroring the parameter choices found in GLOBE-CE’s source code examples. It should be stressed that the parameters were heavily experimented with and the results presented showcase the configurations that achieve *optimal cost-effectiveness* across our experiments.

The same parameter configurations were used for the dGLOBE-CE experiments with `N_DIV` set to 3.

C.5 CET

The tree-based state-of-the-art algorithm CET [8] requires the definition of several parameters such as the maximum number of iterations of the algorithm, which is defined by the `MAX_ITERATION` parameter, the maximum number of leaves, defined by the `MAX_LEAF_SIZE` parameter and the maximum number of features to change, which can be defined by the `MAX_CHANGE_NUM` parameter. The algorithm also requires the definition of two additional parameters, `GAMMA` and `LAMBDA`, which are scalar variables used in the objective function of the optimization problem CET defines and solves (as a subroutine). We thought it significant to mention that tweaking these parameters appears to be essential for achieving satisfactory results.

In our experiments, we have used `MAX_ITERATION = 100`, `MAX_CHANGE_NUM = 4`, `GAMMA = 1` and `LAMBDA = 0.02`. All experiments were run twice, once with `MAX_LEAF_SIZE = 4` and once with `MAX_LEAF_SIZE = 8`.

C.6 *k*-means

As part of our proposed algorithms, we employed the well-known *k*-means algorithm [14]. The *k*-means algorithm requires the definition of three parameters: the `N_CLUSTERS`, which denotes the number of clusters to form as well as the number of centroids to generate, the `N_INIT`, which denotes the number of times the *k*-means algorithm is run with different centroid seeds and finally, the `RANDOM_SEED` which indicates the random seed we mentioned before to have reproducible results.

We used the `scikit-learn` implementation for *k*-means, and details can be found in <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>. In our experiments, the `N_CLUSTERS` is naturally set to the number of desired final clusters/explanations, as described above in each of our methods. `N_INIT` is set to 10 and `RANDOM_SEED` to 13.

C.7 DiCE

Finally, as part of our proposed algorithms, we employed the DiCE algorithm [16] for the generation of local counterfactuals. The algorithm requires a set of parameters to define before execution, such as the `DATASET` parameter to define the dataset we want to use, the `MODEL` parameter which incorporates our trained ML model, and also which of the features of the dataset are continuous, which can be defined by the `CONTINUOUS_FEATURES` parameter. It is also important to define the name of the target variable, denoted by the `OUTCOME_NAME` parameter. Additionally, we need to define through the `BACKEND` parameter the backend of the model implementation. Lastly, it is important to define the method to use when generating counterfactuals, through the `METHOD` parameter.

We used the `dice-ml` python library, which can be found <https://interpret.ml/DiCE/>. In our experiments, we set the `BACKEND` parameter to “`sklearn`”, since the models used follow the `scikit-learn` paradigm. `METHOD` is set to

“random”, for Randomized Sampling. To DATASET and MODEL we pass the dataset and model with which we want to perform each run, and CONTINUOUS_FEATURES and OUTCOME_NAME are set on a per-dataset basis.

D Additional Results for Problem 1

We evaluate our algorithms in comparison with GLOBE-CE, which is the state-of-the-art method for producing global counterfactuals. Specifically, we compare Algorithms 1 and 2 with default GLOBE-CE, GLOBE-CE with a constrained number of micro-actions (scalars) to a maximum of 3, dGLOBE-CE with a maximum of 3 directions, and dGLOBE-CE with a maximum of 3 directions and a maximum of three scalars per direction. Our algorithms are slower than all versions of GLOBE-CE. However, they remain computationally efficient. We primarily assess our algorithms in terms of cost, effectiveness, and the number of actions. Tables 9 to 12 summarize the results for each dataset-model combination.

Table 9: COMPAS Dataset

method / model	DNN				LR				XGB			
	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME
GLOBE-CE	98.75 %	2.83	182	9.45 SEC	100.00 %	1.78	78	3.7 SEC	85.4 %	0.97	25	10.5 SEC
GLOBE-CE (SCALARS 3)	98.75 %	7.27	3	4.24 SEC	100.00 %	5.12	3	1.43 SEC	85.00 %	4.19	3	7.48 SEC
dGLOBE-CE	98.75 %	2.43	291	19.36 SEC	100.00 %	1.42	280	9.48 SEC	100.00 %	0.93	111	21.83 SEC
dGLOBE-CE (SCALARS 3)	98.75 %	10.9	2	3.94 SEC	100.00 %	7.13	6	1.85 SEC	19.2 %	25.77	3	3.69 SEC
AUGMENTED SPACE	92.85 %	4.13	3	20.11 SEC	100.00 %	7.04	3	13.14 SEC	98.40 %	4.95	3	16.08 SEC
ITERATIVE MERGES	95.96 %	3.32	3	25.04 SEC	100.00 %	2.77	3	13.48 SEC	99.60 %	1.99	3	21.96 SEC

Table 10: German Credit Dataset

method / model	DNN				LR				XGB			
	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME
GLOBE-CE	96.49 %	1.09	6	8.31 SEC	72.41 %	1.28	3	3.44 SEC	70.27 %	1.23	4	13.66 SEC
GLOBE-CE (SCALARS 3)	94.73 %	1.17	3	5.76 SEC	72.41 %	1.28	3	0.65 SEC	67.56 %	1.17	3	12.16 SEC
dGLOBE-CE	100.00 %	1.42	55	18.58 SEC	79.31 %	1.19	6	6.32 SEC	81 %	1.19	21	31.4 SEC
dGLOBE-CE (SCALARS 3)	100.00 %	3.49	2	4.25 SEC	75.86 %	2.06	2	2.22 SEC	78.37 %	6.4	3	6.2 SEC
AUGMENTED SPACE	100.00 %	1.56	3	68.59 SEC	98.25 %	1.82	3	24.12 SEC	97.30 %	1.06	3	22.49 SEC
ITERATIVE MERGES	100.00 %	1.47	3	41.95 SEC	100.00 %	1.21	3	9.64 SEC	100.00 %	1.08	3	22.72 SEC

Table 11: Default Credit Dataset

method / model	DNN				LR				XGB			
	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME
GLOBE-CE	99.82 %	1.06	4	20.83 SEC	100.00 %	1.06	10	7.37 SEC	62.82 %	0.42	14	19.63 SEC
GLOBE-CE (SCALARS 3)	99.65 %	3.41	3	17.44 SEC	100.00 %	1.08	3	2.69 SEC	51.99 %	42.6	3	21.68 SEC
dGLOBE-CE	100.00 %	0.97	253	44.97 SEC	100.00 %	1.11	7	12.96 SEC	89.46 %	0.43	128	44.74 SEC
dGLOBE-CE (SCALARS 3)	100.00 %	5.5	6	11.22 SEC	100.00 %	6.4	2	4.49 SEC	51.99 %	64	1	11.2 SEC
AUGMENTED SPACE	100.00 %	1.13	3	733 SEC	100.00 %	1.00	3	82.94 SEC	90.31 %	1.81	3	212.13 SEC
ITERATIVE MERGES	100.00 %	1.00	3	410 SEC	100.00 %	1.06	3	119.79 SEC	95.44 %	3.17	3	102.48 SEC

Figure 3 of the main paper provides a summary in a stacked bar plot form regarding the dominance for specific method combinations, following the

Table 12: HELOC Dataset

method / model	DNN				LR				XGB			
	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME
GLOBE-CE	100.00 %	7.75	403	4.62 SEC	99.90 %	0.41	588	1.00 SEC	34.82 %	1.57	20	3.66 SEC
GLOBE-CE (SCALARS 3)	100.00 %	18.58	3	9.47 SEC	99.09 %	0.66	3	1.63 SEC	34.15 %	42.66	3	7.68 SEC
DGLOBE-CE	100.00 %	2.93	893	8.68 SEC	99.90 %	0.41	588	1.38 SEC	47.91 %	1.68	70	7.91 SEC
DGLOBE-CE (SCALARS 3)	99.90 %	5.98	4	6.59 SEC	99.90 %	1.27	6	2.09 SEC	42.97 %	64.7	5	5 SEC
AUGMENTED SPACE	99.81 %	4.94	3	71.49 SEC	100.00 %	1.37	3	96.82 SEC	96.20 %	22.74	3	53.79 SEC
ITERATIVE MERGES	99.91 %	8.55	3	38.96 SEC	100.00 %	1.42	3	93.26 SEC	99.43 %	24.77	3	59.41 SEC

Definitions 3 to 6. As we can see our methods dominate GLOBE-CE variations in many of the dataset-model combinations.

However, there are dataset-model combinations that the solutions proposed by Algorithms 1 and 2 are deemed to be competitive with GLOBE-CE variations, since none dominates the other. Thus we want to make the following observations. (1) In some cases GLOBE-CE achieves a very small cost with a high number of scalars (micro-actions), while our methods achieve comparable cost and efficiency but with only 3 actions (e.g., Compas Dataset-DNN model combination). Limiting the scalars of GLOBE-CE leads to higher costs. (2) GLOBE-CE may achieve a small cost but at the same time fail to achieve high efficiency, while our algorithms aim to achieve high efficiency (e.g., Default Credit -XGB) even at the expense of the cost (e.g., HELOC XGB). We notice that in such cases, limiting the scalars of GLOBE-CE variations may lead to very large costs. (3) Scaling up our algorithms (i.e., an increase in our number of counterfactuals) may also lead to domination following one or more of the Definitions 3 to 6, especially when the two methods are comparable (see the results in Table 13).

Table 13: Additional results of our methods

METHOD	HELOC DNN				German Credit DNN				Compas LR				Compas DNN			
	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME	EFF	COST	# ACTIONS	RUNTIME
GLOBE-CE	100.00 %	7.75	403	4.62 SEC	96.49 %	1.09	6	8.31 SEC	100.00 %	1.78	78	3.7 SEC	98.75 %	2.83	182	9.45 SEC
GLOBE-CE (3 SCALARS)	100.00 %	18.58	3	9.47 SEC	93.73 %	1.17	3	5.76 SEC	100.00 %	5.12	3	1.43 SEC	98.75 %	7.27	3	4.24 SEC
DGLOBE-CE	100.00 %	2.93	893	8.68 SEC	100.00 %	1.42	55	4.25 SEC	100.00 %	1.42	280	9.48 SEC	98.75 %	2.43	291	19.36 SEC
DGLOBE-CE (3 SCALARS)	99.90 %	5.98	4	6.59 SEC	100.00 %	3.49	2	4.25 SEC	100.00 %	7.13	6	1.85 SEC	98.75 %	10.9	2	3.94 SEC
AUGMENTED SPACE (SCALED)	100.00 %	2.64	50	53.15 SEC	100.00 %	1.05	25	177.434 SEC	100.00 %	1.39	100	18.17 SEC	93.93 %	1.49	50	48.65 SEC
ITERATIVE MERGES (SCALED)	100.00 %	3.58	50	50.21 SEC	100.00 %	1.07	25	136.24 SEC	100.00 %	1.1	50	21.32 SEC	96.27 %	1.95	50	40.75 SEC

E Results for Problem 2

In Tables 14 to 17, we present the results of our experiments for Problem 2. Specifically, we show the 3 main objectives we have discussed in the main body of the paper, i.e. effectiveness, cost, and number of leaves. Additionally, we show the runtime (in seconds) of each experiment.

Figure 4 provides a summary in a bar plot form regarding the dominance for specific method combinations, following the Definition 7. As we can see our methods dominate CET variations in some dataset-model combinations, while CET fails to dominate us in any combination.

Table 14: German Credit Dataset

method / model	DNN				LR				XGB			
	EFF	COST	# LEAVES	RUNTIME	EFF	COST	# LEAVES	RUNTIME	EFF	COST	# LEAVES	RUNTIME
CET - 4	92.98%	1.17	2	374.60 SEC	86.21%	2.32	3	91.91 SEC	94.59%	3.29	3	379.25 SEC
CF TREE - 4	100.00 %	1.99	3	39.13 SEC	100.00 %	5.39	4	18.17 SEC	100.00 %	4.20	4	22.28 SEC
CET - 8	89.47%	1.00	1	219.71 SEC	91.38%	2.52	6	72.28 SEC	100.00%	3.15	6	254.19 SEC
CF TREE - 8	100.00 %	1.85	8	44.52 SEC	100.00 %	5.52	8	20.16 SEC	100.00 %	3.49	7	22.51 SEC

Table 15: COMPAS Dataset

method / model	DNN				LR				XGB			
	EFF	COST	# LEAVES	RUNTIME	EFF	COST	# LEAVES	RUNTIME	EFF	COST	# LEAVES	RUNTIME
CET - 4	39.81%	1.18	2	899.37 SEC	69.46%	0.96	4	132.42 SEC	76.20%	1.03	2	990.70 SEC
CF TREE - 4	89.64 %	3.42	4	42.21 SEC	100.00 %	6.61	4	32.06 SEC	98.36 %	5.10	4	46.07 SEC
CET - 8	58.79%	2.13	3	575.16 SEC	86.61%	1.70	6	89.47 SEC	80.60%	1.28	4	718.57 SEC
CF TREE - 8	86.16 %	5.00	8	83.26 SEC	100.00 %	5.85	8	60.50 SEC	100.00 %	5.51	8	89.45 SEC

Table 16: Default Credit Dataset

method / model	DNN				LR				XGB			
	EFF	COST	# LEAVES	RUNTIME	EFF	COST	# LEAVES	RUNTIME	EFF	COST	# LEAVES	RUNTIME
CET - 4	65.71%	9.48	2	7981.46 SEC	99.34%	2.40	1	3324.19 SEC	45.16%	9.56	2	5324.21 SEC
CF TREE - 4	100.00 %	3.57	4	316.31 SEC	100.00 %	10.43	4	123.31 SEC	79.00 %	3.34	4	257.94 SEC
CET - 8	67.80%	6.71	3	4203.78 SEC	99.34%	2.40	1	2368.79 SEC	39.17%	9.00	1	3119.20 SEC
CF TREE - 8	100.00 %	4.32	6	332.47 SEC	100.00 %	5.42	6	124.36 SEC	87.75 %	3.76	6	180.95 SEC

Table 17: HELOC Dataset

method / model	DNN				LR				XGB			
	EFF	COST	# LEAVES	RUNTIME	EFF	COST	# LEAVES	RUNTIME	EFF	COST	# LEAVES	RUNTIME
CET - 4	47.34%	12.69	3	13662.29 SEC	98.08%	4.18	2	2018.43 SEC	36.53%	14.10	2	13668.97 SEC
CF TREE - 4	97.61 %	10.88	4	156.10 SEC	100.00 %	8.50	4	188.47 SEC	94.66 %	21.36	4	150.82 SEC
CET - 8	32.03%	15.90	2	11347.89 SEC	99.90%	1.66	1	1319.07 SEC	28.46%	6.21	4	10842.03 SEC
CF TREE - 8	90.12 %	12.78	6	277.75 SEC	100.00 %	8.83	6	344.05 SEC	96.76 %	24.39	6	272.02 SEC

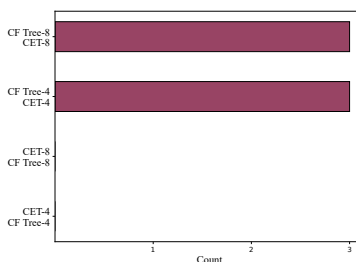


Figure 4: Visual representation of the evaluation of Algorithm 3 vs CET. These results correspond to the experimental results presented in the Tables 14 to 17.

We want to make the following observations. (1) We achieve higher effectiveness in all cases, while CET’s cost is lower only in cases where the difference

in effectiveness is particularly significant; which is to be expected. The final number of actions has more variance, however, we believe that this is less significant, given the large difference in effectiveness. (2) We can see that in all cases our experiments’ runtime is at least one order of magnitude lower, which is a very important part of our contribution. (3) Moreover, there are notable caveats with CET that we thought were important to highlight since they considerably hindered the performance and ease of use of the method⁶. (4) Our framework exposes the selection of features by which to construct the tree as a hyperparameter to the user. As we have mentioned in the main body, the user may wish to extract counterfactuals for a segmentation based on specific features of the dataset. (5) The comparison with CET in specific models is not entirely fair to our algorithm, since CET takes advantage of model internals to calculate counterfactuals for the populations on the leaves (they solve an LP which depends on the model). Specifically, they use model internals for Logistic Regression, Random Forests, and MLPs, and employ a LIME approximation for any other model. Incidentally, this also reinforces our algorithm’s dominance on the results for the Logistic Regression, since CET uses internal information, which our framework does not.

Additionally, it should be mentioned that CET is not entirely black-box. To calculate counterfactuals for the populations on the leaves, they solve an LP which depends on the model. They use model internals for Logistic Regression, Random Forests, and MLPs, and employ a LIME approximation for any other model. Incidentally, this also reinforces our algorithm’s dominance on the results for the Logistic Regression, since CET uses internal information, which our framework does not.

⁶We needed a medium-effort experimentation with different MILP solvers before we could run the framework in a sensible amount of time. The openly available solvers we tested took a very long time to run, while other options, including the default, seemed to be available only via a license. In the end, we are using, for our experiments, the well-known gurobi solver [7], with an academic license. Moreover, CET optimizes the whole structure of the tree. However, we believe this turns out to be a disadvantage because it outputs a single “optimal” solution (which turns out to be unsatisfactory, as we mentioned earlier).