Learning Decision Trees and Forests with Algorithmic Recourse

Kentaro Kanamori [1] **Takuya Takagi** [1] **Ken Kobayashi** [2] **Yuichi Ike** [3]

Abstract

This paper proposes a new algorithm for learning accurate tree-based models while ensuring the existence of recourse actions. Algorithmic Recourse (AR) aims to provide a recourse action for altering the undesired prediction result given by a model. Typical AR methods provide a reasonable action by solving an optimization task of minimizing the required effort among executable actions. In practice, however, such actions do not always exist for models optimized only for predictive performance. To alleviate this issue, we formulate the task of learning an accurate classification tree under the constraint of ensuring the existence of reasonable actions for as many instances as possible. Then, we propose an efficient top-down greedy algorithm by leveraging the adversarial training techniques. We also show that our proposed algorithm can be applied to the random forest, which is known as a popular framework for learning tree ensembles. Experimental results demonstrated that our method successfully provided reasonable actions to more instances than the baselines without significantly degrading accuracy and computational efficiency.

1. Introduction

Algorithmic decision-making with machine learning models has been applied to various tasks in the real world, such as loan approvals. In such critical tasks, the predictions made by a model might have a significant impact on individual users (Rudin, 2019). Consequently, decision-makers need to explain how individuals should act to alter the undesired decisions (Miller, 2019; Wachter et al., 2018). *Algorithmic Recourse (AR)* aims to provide such information (Ustun et al., 2019). For a classifier $h\colon \mathcal{X} \to \mathcal{Y}$, a desired class $y^* \in \mathcal{Y}$, and an instance $\boldsymbol{x} \in \mathcal{X}$ such that $h(\boldsymbol{x}) \neq y^*$, AR provides a perturbation $\boldsymbol{a}$ that flips the prediction result into the desired class, i.e., $h(\boldsymbol{x}+\boldsymbol{a}) = y^*$, with a minimum effort measured by some cost function c. The user can regard the perturbation $\boldsymbol{a}$ as a *recourse action* for obtaining the desired outcome y^* from the classifier h (Karimi et al., 2022).

To provide affected individuals with recourse certainly, we need to guarantee the existence of actions $\boldsymbol{a}$ that are executable with small costs. In general, however, such reasonable actions do not always exist (Ross et al., 2021). Figure 1 demonstrates this observation on a synthetic loan approval task. Consider a situation where a user $\boldsymbol{x}$ in Figure 1(a) is denied one's loan application by classification trees h_0 and h_1 in Figures 1(b) and 1(c). By changing "#ExistingLoans," the user can get the loan approved by h_0. In contrast, for the case of h_1, the user must change "Purpose" or "Education," which are inappropriate recourse actions because changing these features is difficult in practice. It suggests that the user $\boldsymbol{x}$ cannot get one's loan approved as long as h_1 is deployed, and that h_0 is more desirable as it can ensure a reasonable action for the user $\boldsymbol{x}$ (Sullivan & Verreault-Julien, 2022; Venkatasubramanian & Alfano, 2020).

In this paper, we aim to learn a classifier h that ensures the existence of executable actions $\boldsymbol{a}$ for input instances $\boldsymbol{x}$ with high probability while keeping accuracy. Specifically, we focus on the *tree-based models*, such as *decision trees* (Breiman et al., 1984) and *random forests* (Breiman, 2001). Recently, Ross et al. (2021) proposed a gradient-based method for learning differentiable classifiers such as deep neural networks while guaranteeing the existence of actions. However, their method cannot be directly applied to the tree-based models because these models are not differentiable. Tree-based models are known to perform well on tabular datasets (Grinsztajn et al., 2022), which often appear in the areas where AR is required (e.g., finance and justice) (Verma et al., 2020). Therefore, we need a new learning algorithm designed for tree-based models while ensuring the existence of recourse actions.

1.1. Our Contributions

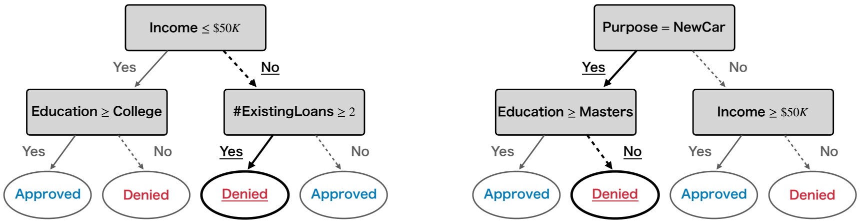
In this paper, we propose *Recourse-Aware Classification Tree (RACT)*, a new framework for learning tree-based classifiers that make accurate predictions and guarantee recourse actions. Our contributions are summarized as follows:

[1] Fujitsu Limited, Japan [2] Tokyo Institute of Technology, Japan [3] Kyushu University, Japan. Correspondence to: Kentaro Kanamori <k.kanamori@fujitsu.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Features	Values
Income	$70K
Purpose	NewCar
Education	College
#ExistingLoans	2

(a) Instance $\boldsymbol{x}$



(b) Classification tree h_0

(c) Classification tree h_1

Figure 1. Examples of an input instance $\boldsymbol{x}$ and classification trees h_0, h_1. To get the loan approved from h_0, the instance $\boldsymbol{x}$ needs to just reduce "#ExistingLoans." In contrast, for the case of h_1, $\boldsymbol{x}$ needs to change "Purpose" or "Education," which are not executable easily.

- We introduce the task of learning a classification tree under the constraint on the ratio of instances having at least one valid action. Then, we propose a specialized top-down greedy algorithm by leveraging the adversarial training techniques (Guo et al., 2022). Our algorithm is simple yet efficient, as we show that its computational complexity is equivalent to that of the standard learning algorithm for classification trees, such as CART (Breiman et al., 1984).

- We introduce a post-processing task of modifying a learned tree so as to satisfy the constraint on recourse guarantee, and prove that the task can be efficiently solved with an approximation guarantee by reducing it to the minimum set cover problem (Kearns, 1990). We also show that our learning algorithm can be easily extended to the random forest (Breiman, 2001), which is a popular framework for learning tree ensembles.

- Our numerical experiments on real datasets demonstrated that RACT successfully provided reasonable recourse actions to more individuals than the baselines while keeping comparable predictive accuracy and computational efficiency. We also confirmed that we could control the trade-off between keeping accuracy and guaranteeing recourse by tuning the hyperparameter of RACT.

1.2. Related Work

Algorithmic Recourse (AR) (Ustun et al., 2019), also referred to as *Counterfactual Explanation* (Wachter et al., 2018), has attracted increasing attention in recent years (Verma et al., 2020; Karimi et al., 2022). While almost all of the previous papers focused on extracting actions from a learned machine learning model (Cui et al., 2015; Tolomei et al., 2017; Pawelczyk et al., 2020; Kanamori et al., 2020; 2021; Parmentier & Vidal, 2021; Lucic et al., 2022; Carreira-Perpiñán & Hada, 2023), there exist few studies on learning models while ensuring the existence of recourse actions (Ross et al., 2021; Dominguez-Olmedo et al., 2022). However, because the existing methods are based on gradient descent, we cannot directly apply them to learning tree-based models.

Learning *tree-based models*, such as decision trees (Breiman et al., 1984), random forests (Breiman, 2001), and gradient boosted trees (Friedman, 2000; Chen & Guestrin, 2016; Ke et al., 2017), has been widely studied due to their interpretability, accuracy, and efficiency (Hu et al., 2019; Grinsztajn et al., 2022). There also exist several methods for learning tree-based models under some constraints, such as fairness (Kamiran et al., 2010) and stability (Hara & Yoshida, 2023). Our method is most related to the algorithms for learning robust decision trees (Chen et al., 2019; Vos & Verwer, 2021; 2022; Guo et al., 2022). To learn a robust decision tree, these methods determine the split condition in each node by optimizing the worst-case information gain under adversarial perturbations. In contrast, we aim to ensure the existence of executable actions, i.e., some kind of perturbations, for obtaining a desired outcome. Thus, it is not trivial to apply these methods to our task.

2. Problem Statement

For a positive integer $n \in \mathbb{N}$, we write $[n] := \{1, \dots, n\}$. As with the previous studies (Ross et al., 2021), we consider a binary classification task between undesired and desired classes. Let $\mathcal{X} \subseteq \mathbb{R}^D$ and $\mathcal{Y} = \{\pm 1\}$ be input and output domains, respectively. We assume that all the categorical features are encoded by one-hot encoding. We call a vector $\boldsymbol{x} = (x_1, \dots, x_D) \in \mathcal{X}$ an *instance*, and a function $h \colon \mathcal{X} \to \mathcal{Y}$ a *classifier*. We denote the 0-1 loss function by $l_{01}(y, \hat{y}) = \mathbb{I}[y \neq \hat{y}]$. Without loss of generality, we assume that $h(\boldsymbol{x}) = +1$ is a desirable result (e.g., loan approval).

2.1. Algorithmic Recourse

For an instance $\boldsymbol{x} \in \mathcal{X}$, we define an *action* as a perturbation vector $\boldsymbol{a} \in \mathbb{R}^D$ such that $\boldsymbol{x} + \boldsymbol{a} \in \mathcal{X}$. Let $\mathcal{A}(\boldsymbol{x})$ be a set of feasible actions for $\boldsymbol{x}$ such that $\boldsymbol{0} \in \mathcal{A}(\boldsymbol{x})$ and $\mathcal{A}(\boldsymbol{x}) \subseteq \{\boldsymbol{a} \in \mathbb{R}^D \mid \boldsymbol{x} + \boldsymbol{a} \in \mathcal{X}\}$. For simplicity, we assume that we can write as $\mathcal{A}(\boldsymbol{x}) = [l_1, u_1] \times \cdots \times [l_D, u_D]$ with lower and upper bounds $l_d, u_d \in \mathbb{R}$ for $d \in [D]$. For a classifier h, an action $\boldsymbol{a}$ is *valid* for $\boldsymbol{x}$ if $\boldsymbol{a} \in \mathcal{A}(\boldsymbol{x})$ and $h(\boldsymbol{x} + \boldsymbol{a}) = +1$. For $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{a} \in \mathcal{A}(\boldsymbol{x})$, a *cost function* $c \colon \mathcal{A}(\boldsymbol{x}) \to \mathbb{R}_{\geq 0}$

measures the required effort of $\boldsymbol{a}$ with respect to $\boldsymbol{x}$. For a given classifier $h\colon \mathcal{X} \to \mathcal{Y}$ and an instance $\boldsymbol{x} \in \mathcal{X}$, the aim of *Algorithmic Recourse (AR)* is to find an action $\boldsymbol{a}$ that is valid for $\boldsymbol{x}$ with respect to h and minimizes its cost $c(\boldsymbol{a} \mid \boldsymbol{x})$. This task can be formulated as follows (Karimi et al., 2022):

$$\min_{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{x})} c(\boldsymbol{a} \mid \boldsymbol{x}) \quad \text{s.t.} \quad h(\boldsymbol{x} + \boldsymbol{a}) = +1. \tag{1}$$

For the cost function c, we assume that it satisfies the following properties: (i) $c(\boldsymbol{0} \mid \boldsymbol{x}) = 0$; (ii) $c(\boldsymbol{a} \mid \boldsymbol{x}) = \max_{d \in [D]} c_d(a_d \mid x_d)$, where c_d is the cost of the action a_d for a feature d; (iii) $\forall \beta \geq 0 : c_d(a_d \mid x_d) \leq c_d(a_d \cdot (1+\beta) \mid x_d)$. Note that several major cost functions, including the weighted ℓ_∞-norm (Ross et al., 2021) and max percentile shift (Ustun et al., 2019), satisfy the above properties.

2.2. Tree-Based Model

A *classification tree* $h\colon \mathcal{X} \to \mathcal{Y}$ is a classifier that consists of a set of if-then-else rules expressed as a binary tree structure (Breiman et al., 1984). For a given input $\boldsymbol{x} \in \mathcal{X}$, it makes a prediction according to the predictive label $\hat{y}$ of the leaf that $\boldsymbol{x}$ reaches. The corresponding leaf is determined by traversing the tree from the root depending on whether the *split condition* $x_d \leq b$ is true or not, where $(d, b) \in [D] \times \mathbb{R}$ is a pair of a feature and threshold of each internal node. Figure 1 presents examples of classification trees.

For a classification tree h, we denote the total number of its leaves by $I \in \mathbb{N}$. Let $r_i \subseteq \mathcal{X}$ and $\hat{y}_i \in \mathcal{Y}$ be the subspace and predictive label corresponding to a leaf $i \in [I]$. Each subspace r_i is determined by the split conditions on the path from the root to the leaf i, and can be expressed as an axis-aligned rectangle $r_i = (l_{i,1}, u_{i,1}] \times \dots (l_{i,D}, u_{i,D}]$. The set of such subspaces $\{r_1, \dots, r_I\}$ gives a partition of the input space $\mathcal{X}$, and h can be expressed as $h(\boldsymbol{x}) = \sum_{i=1}^{I} \hat{y}_i \cdot \mathbb{I}\left[\boldsymbol{x} \in r_i\right]$ (Freitas, 2014; Guidotti et al., 2018).

A *tree ensemble* is a popular machine learning model for tabular datasets (Grinsztajn et al., 2022). It makes a prediction by combining the outputs of T trees $h_1, \dots, h_T$. In this paper, we focus on *random forests* (Breiman, 2001) as a framework for learning tree ensembles. Each tree h_t of a random forest classifier is trained on a bootstrap sample of a training sample, and a split condition (d, b) in each node is determined among a random subset of the features $[D]$.

2.3. Problem Formulation

The aim of this paper is to learn an accurate tree-based classifier while guaranteeing the existence of a valid action with a low cost for any instance. For that purpose, we need to learn a classifier h under the constraint that, for any instance $\boldsymbol{x}$ in a given sample, there exists an action $\boldsymbol{a} \in \mathcal{A}(\boldsymbol{x})$ such that $h(\boldsymbol{x} + \boldsymbol{a}) = +1$ and $c(\boldsymbol{a} \mid \boldsymbol{x}) \leq \varepsilon$ for some budget parameter $\varepsilon > 0$. In practice, however, previous studies have empirically demonstrated that it is often difficult to guarantee valid actions for all instances without degrading accuracy (Levanon & Rosenfeld, 2021; Olckers & Walsh, 2023). This means that imposing the constraint of ensuring valid actions for all instances is too strict to keep accuracy.

To balance such a trade-off, we will relax the constraint on recourse. For a cost budget parameter ε, let $\mathcal{A}_\varepsilon(\boldsymbol{x}) := \{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{x}) \mid c(\boldsymbol{a} \mid \boldsymbol{x}) \leq \varepsilon\}$ be the set of feasible actions whose costs are lower than ε. Given a sample $S = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, our *empirical recourse risk* $\hat{\Omega}_\varepsilon$ is defined as

$$\hat{\Omega}_\varepsilon(h \mid S) := \frac{1}{N} \sum_{n=1}^{N} l_{\mathrm{rec}}(\boldsymbol{x}_n; h),$$

where $l_{\mathrm{rec}}(\boldsymbol{x}; h) := \min_{\boldsymbol{a} \in \mathcal{A}_\varepsilon(\boldsymbol{x})} l_{01}(+1, h(\boldsymbol{x} + \boldsymbol{a}))$ is the *recourse loss*, which corresponds to the relaxed version of the original hard constraint $\exists \boldsymbol{a} \in \mathcal{A}_\varepsilon(\boldsymbol{x}) : h(\boldsymbol{x} + \boldsymbol{a}) = +1$. By definition, this risk is equivalent to the ratio of instances $\boldsymbol{x}$ in the sample S that do not have any action $\boldsymbol{a}$ such that $h(\boldsymbol{x} + \boldsymbol{a}) = +1$ and $c(\boldsymbol{a} \mid \boldsymbol{x}) \leq \varepsilon$.

Using the empirical recourse risk $\hat{\Omega}_\varepsilon$, we formulate our learning task as follows.

Problem 2.1. Given a sample $S = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N} \subseteq \mathcal{X} \times \mathcal{Y}$ and parameters $\varepsilon, \delta > 0$, find an optimal solution to the following optimization problem:

$$\underset{h \in \mathcal{H}}{\text{minimize}} \quad \hat{R}(h \mid S) \quad \text{subject to} \quad \hat{\Omega}_\varepsilon(h \mid S) \leq \delta,$$

where $\mathcal{H}$ is the set of classification trees $h\colon \mathcal{X} \to \mathcal{Y}$ and $\hat{R}(h \mid S) := \frac{1}{N} \sum_{n=1}^{N} l_{01}(y_n, h(\boldsymbol{x}_n))$ is the empirical risk.

By solving Problem 2.1, we can obtain a classification tree h^* that makes accurate predictions while guaranteeing the existence of valid actions for at least $100 \cdot (1 - \delta)$ % instances. We are also expected to adjust the trade-off between the predictive accuracy and recourse guarantee by tuning δ.

3. Learning Algorithm

This section presents an efficient algorithm for Problem 2.1. Even for the case without the constraint $\hat{\Omega}_\varepsilon(h \mid S) \leq \delta$, efficiently finding an exact optimal solution to Problem 2.1 is computationally challenging (Hyafil & Rivest, 1976; Hu et al., 2019). In addition, the existing gradient-based methods, such as (Ross et al., 2021), cannot be directly applied to Problem 2.1 due to the combinatorial nature of tree-based models. To avoid these difficulties, we propose an efficient learning algorithm by extending the standard greedy approach like CART (Breiman et al., 1984). Our framework, named *recourse-aware classification tree (RACT)*, consists of the following two steps: (i) growing a classification tree h by recursively determining the split condition of each internal node based on the empirical risk $\hat{R}$ and recourse risk $\hat{\Omega}_\varepsilon$; (ii) modifying the predictive label of each leaf in the learned tree h so as to satisfy the constraint $\hat{\Omega}_{\varepsilon}(h \mid S) \leq \delta$. We also show that our algorithm can be applied to the random forest (Breiman, 2001), which is one of the most popular frameworks for learning tree ensembles. Note that our learning algorithm does not require a specific oracle to obtain an action (i.e., solve the problem (1) for each instance $\boldsymbol{x}$) because our algorithm itself does not depend on the construction process of actions, and thus, we can employ any existing AR method for tree-based models (e.g., (Cui et al., 2015; Tolomei et al., 2017; Kanamori et al., 2020)).

3.1. Top-Down Greedy Splitting

We first propose an efficient learning algorithm, inspired by the well-known top-down greedy approach to learning classification trees. For each node in a current tree h, it determines a split condition (d, b) that minimizes some impurity criterion (Breiman et al., 1984; Rokach & Maimon, 2005). Following this approach, we first formulate the task of determining a split condition designed for our learning problem, and propose an efficient algorithm for solving it.

We consider adding a new split condition (d, b) to a leaf $i \in [I]$ of a current tree h with I leaves. Such a new tree h' can be expressed as follows:

$$h'(\boldsymbol{x}) = \mathbb{I}\left[\boldsymbol{x} \in r_i\right] \cdot h(\boldsymbol{x}; d, b, \hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}}) + \mathbb{I}\left[\boldsymbol{x} \notin r_i\right] \cdot h(\boldsymbol{x}),$$

where $h(\boldsymbol{x}; d, b, \hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}}) = \hat{y}_{\mathrm{L}} \cdot \mathbb{I}\left[\boldsymbol{x}_d \leq b\right] + \hat{y}_{\mathrm{R}} \cdot \mathbb{I}\left[\boldsymbol{x}_d > b\right]$ is a decision stump with the split condition (d, b) and predictive labels $\hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}} \in \mathcal{Y}$ corresponding to the left and right children of the node i, respectively. Our task can be formulated as finding a best split condition (d, b) and predictive labels $\hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}}$. We assume that we are given a set $B_d = \{b_{d,1}, \dots, b_{d,J_d}\}$ of thresholds for each feature $d \in [D]$ in advance. As with the previous studies, we also assume $b_{d,1} < \dots < b_{d,J_d}$ and $J_d = \mathcal{O}(N)$. To relax the constraint $\hat{\Omega}_{\varepsilon}(h \mid S) \leq \delta$ of Problem 2.1, we define a penalized objective function as

$$\Phi_{\lambda}(d, b, \hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}}) \coloneqq \hat{R}(h' \mid S) + \lambda \cdot \hat{\Omega}_{\varepsilon}(h' \mid S),$$

where $\lambda \geq 0$ is a hyper-parameter that balances the trade-off between the empirical risk $\hat{R}$ and recourse risk $\hat{\Omega}_{\varepsilon}$. Then, our task is formulated as the following problem:

$$\min_{d \in [D], b \in B_d} \min_{\hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}} \in \mathcal{Y}} \Phi_{\lambda}(d, b, \hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}}). \tag{2}$$

When $\lambda = 0$, the problem (2) is roughly equivalent to the standard learning problem for a classification tree, and can be solved in $\mathcal{O}(D \cdot N)$ if we have a permutation $\sigma_d \colon [N] \to [N]$ such that $x_{\sigma_d(1),d} \leq \dots \leq x_{\sigma_d(N),d}$ for each $d \in [D]$ in advance (Rokach & Maimon, 2005). In contrast, if $\lambda > 0$, efficiently solving the problem (2) is not trivial. In the following, we show that the problem (2) with $\lambda > 0$ can be solved in $\mathcal{O}(D \cdot N)$ as well by leveraging the adversarial training techniques for classification trees (Guo et al., 2022).

Algorithm 1 Algorithm for the problem (2).

```
Input: a sample $S = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, parameter $\lambda > 0$, current tree $h$ with $I$
    leaves, current leaf $i \in [I]$, set of sorted thresholds $B_d$ for $d \in [D]$, set of
    permutations $\sigma_d$ such that $x_{\sigma_d(1),d} \leq \dots \leq x_{\sigma_d(N),d}$ for $d \in [D]$, and
    global variables $f_{n,i}, v_{n,i}, V_n$ for $n \in [N]$ and $i \in [I]$.
Output: a best split condition $(d^*, b^*)$ and predictive labels $\hat{y}_{\mathrm{L}}^*, \hat{y}_{\mathrm{R}}^*$.
/* Initialize Terms for Computing Objective Value */
$\bar{N} \leftarrow \sum_{n=1}^{N} (1 - f_{n,i}) \cdot \mathbb{I}\left[h(\boldsymbol{x}_n) \neq y_n\right]$;
$N^{+} \leftarrow \sum_{n=1}^{N} f_{n,i} \cdot \mathbb{I}\left[y_n = +1\right]$; $N^{-} \leftarrow \sum_{n=1}^{N} f_{n,i} \cdot \mathbb{I}\left[y_n = -1\right]$;
$M \leftarrow \sum_{n=1}^{N} \omega_{n,i}$; $\bar{M} \leftarrow \sum_{n=1}^{N} \omega_{n,i} \cdot v_{n,i}$;
$\omega_{n,i} \leftarrow \mathbb{I}\left[V_n - v_{n,i} \cdot \mathbb{I}\left[\hat{y}_i = +1\right] = 0\right]$ $(\forall n \in [N])$;
/* Search All Features and Thresholds */
for $d = 1, 2, \dots, D$ do
    $N_{\mathrm{L}}^{+} \leftarrow 0$; $N_{\mathrm{R}}^{+} \leftarrow N_{+}$; $N_{\mathrm{L}}^{-} \leftarrow 0$; $N_{\mathrm{R}}^{-} \leftarrow N_{-}$; $M_{\mathrm{L}} \leftarrow 0$; $M_{\mathrm{R}} \leftarrow \bar{M}$;
    $n, n_{\mathrm{L}}, n_{\mathrm{R}} \leftarrow \sigma_d(1)$; $m, m_{\mathrm{L}}, m_{\mathrm{R}} \leftarrow 1$;
    for $b \in B_d$ do
        /* Update Terms for Computing Empirical Risk */
        while $x_{n,d} \leq b$ and $m \leq N$ do
            $N_{\mathrm{L}}^{+}, N_{\mathrm{L}}^{-} \leftarrow N_{\mathrm{L}}^{+} + f_{n,i} \cdot \mathbb{I}\left[y_n = +1\right], N_{\mathrm{L}}^{-} + f_{n,i} \cdot \mathbb{I}\left[y_n = -1\right]$;
            $N_{\mathrm{R}}^{+}, N_{\mathrm{R}}^{-} \leftarrow N_{\mathrm{R}}^{+} - f_{n,i} \cdot \mathbb{I}\left[y_n = +1\right], N_{\mathrm{R}}^{-} - f_{n,i} \cdot \mathbb{I}\left[y_n = -1\right]$;
            $m \leftarrow m + 1$; $n \leftarrow \sigma_d(m)$;
        end while
        /* Update Terms for Computing Empirical Recourse Risk */
        while $g_{n_{\mathrm{L}}}(d, b) = 1$ and $m_{\mathrm{L}} \leq N$ do
            $M_{\mathrm{L}} \leftarrow M_{\mathrm{L}} + \omega_{n_{\mathrm{L}},i} \cdot v_{n_{\mathrm{L}},i}$; $m_{\mathrm{L}} \leftarrow m_{\mathrm{L}} + 1$; $n_{\mathrm{L}} \leftarrow \sigma_d(m_{\mathrm{L}})$;
        end while
        while $\bar{g}_{n_{\mathrm{R}}}(d, b) = 0$ and $m_{\mathrm{R}} \leq N$ do
            $M_{\mathrm{R}} \leftarrow M_{\mathrm{R}} - \omega_{n_{\mathrm{R}},i} \cdot v_{n_{\mathrm{R}},i}$; $m_{\mathrm{R}} \leftarrow m_{\mathrm{R}} + 1$; $n_{\mathrm{R}} \leftarrow \sigma_d(m_{\mathrm{R}})$;
        end while
        /* Optimize Predictive Labels using Equations (3) and (4) */
        $\hat{y}_{\mathrm{L}}(d, b), \hat{y}_{\mathrm{R}}(d, b) \leftarrow \arg\min_{\hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}} \in \mathcal{Y}} \Phi_{\lambda}(d, b, \hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}})$;
    end for
end for
/* Determine Best Split Condition */
$(d^*, b^*) \leftarrow \arg\max_{d \in [D], b \in B_d} \Phi_{\lambda}(d, b, \hat{y}_{\mathrm{L}}(d, b), \hat{y}_{\mathrm{R}}(d, b))$;
/* Update Global Variables */
for $n = 1, 2, \dots, N$ do
    $f_{n,\mathrm{L}} \leftarrow f_{n,i} \cdot \mathbb{I}\left[x_{n,d^*} \leq b^*\right]$; $f_{n,\mathrm{R}} \leftarrow f_{n,i} \cdot \mathbb{I}\left[x_{n,d^*} > b^*\right]$;
    $v_{n,\mathrm{L}} \leftarrow v_{n,i} \cdot g_n(d^*, b^*)$; $v_{n,\mathrm{R}} \leftarrow v_{n,i} \cdot \bar{g}_n(d^*, b^*)$;
    $V_n \leftarrow V_n - v_{n,i} \mathbb{I}\left[\hat{y}_i = +1\right] + v_{n,\mathrm{L}} \mathbb{I}\left[\hat{y}_{\mathrm{L}}^* = +1\right] + v_{n,\mathrm{R}} \mathbb{I}\left[\hat{y}_{\mathrm{R}}^* = +1\right]$;
end for
return $d^*, b^*, \hat{y}_{\mathrm{L}}(d^*, b^*), \hat{y}_{\mathrm{R}}(d^*, b^*)$;
```

Let $v_n(r) = \mathbb{I}\left[\exists \boldsymbol{a} \in \mathcal{A}_{\varepsilon}(\boldsymbol{x}_n) : \boldsymbol{x}_n + \boldsymbol{a} \in r\right]$ be the indicator whether the instance $\boldsymbol{x}_n$ can reach the subspace r by some action $a \in \mathcal{A}_{\varepsilon}(\boldsymbol{x}_n)$. We also denote the set of leaves that $\boldsymbol{x}_n$ can reach their corresponding subspaces by $\mathcal{I}_n = \{i' \in [I] \mid v_n(r_{i'}) = 1\}$. Recall that our recourse loss is defined by $l_{\mathrm{rec}}(\boldsymbol{x}; h) = \min_{\boldsymbol{a} \in \mathcal{A}_{\varepsilon}(\boldsymbol{x})} l_{01}(+1, h(\boldsymbol{x} + \boldsymbol{a}))$. Using $\mathcal{I}_n$, we can express the recourse loss of h for $\boldsymbol{x}_n$ as $l_{\mathrm{rec}}(\boldsymbol{x}_n; h) = \min_{i' \in \mathcal{I}_n} l_{01}(+1, \hat{y}_{i'})$; that is, if there is no leaf $i' \in [I]$ such that $i' \in \mathcal{I}_n$ and $\hat{y}_{i'} = +1$, our recourse loss takes 1; otherwise, it takes 0. Based on this observation, we can express the recourse loss after splitting at i as

$$\begin{aligned} l_{\mathrm{rec}}(\boldsymbol{x}_n; h') = \min\{\omega_{n,i}, & 1 - v_n(r_{\mathrm{L}}) \cdot \mathbb{I}\left[\hat{y}_{\mathrm{L}} = +1\right], \\ & 1 - v_n(r_{\mathrm{R}}) \cdot \mathbb{I}\left[\hat{y}_{\mathrm{R}} = +1\right]\}, \end{aligned}$$

where $\omega_{n,i} = \mathbb{I}\left[\forall i' \in \mathcal{I}_n \setminus \{i\} : \hat{y}_{i'} \neq +1\right]$ and $r_{\mathrm{L}} = \{\boldsymbol{x} \in \mathcal{X} \mid \boldsymbol{x} \in r_i \wedge x_d \leq b\}$ (resp. $r_{\mathrm{R}} = \{\boldsymbol{x} \in \mathcal{X} \mid \boldsymbol{x} \in r_i \wedge x_d > b\}$) is the subspace of the left (resp. right) child node of i.

To efficiently evaluate $\hat{\Omega}_{\varepsilon}(h' \mid S)$ for each (d, b), we introduce some global variables and manage them, as with (Guo et al., 2022). Let $f_{n,i} = \mathbb{I}[\boldsymbol{x}_n \in r_i]$ and $v_{n,i} = v_n(r_i)$. We

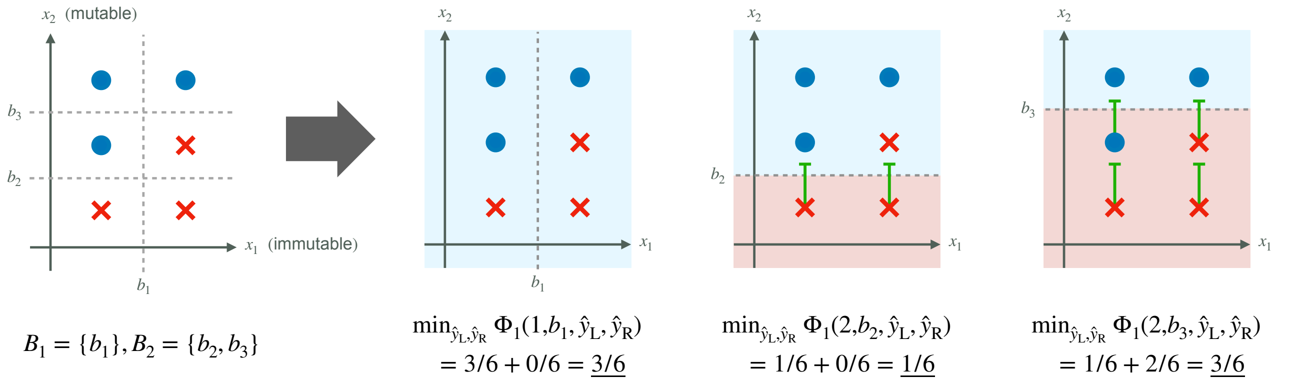

x_2 (mutable)
b_3
b_2
x_1 (immutable)
b_1
$B_1 = \{b_1\}, B_2 = \{b_2, b_3\}$
x_2
x_1
b_1
$\min_{\hat{y}_L, \hat{y}_R} \Phi_1(1, b_1, \hat{y}_L, \hat{y}_R) = 3/6 + 0/6 = \underline{3/6}$
x_2
b_2
x_1
$\min_{\hat{y}_L, \hat{y}_R} \Phi_1(2, b_2, \hat{y}_L, \hat{y}_R) = 1/6 + 0/6 = \underline{1/6}$
x_2
b_3
x_1
$\min_{\hat{y}_L, \hat{y}_R} \Phi_1(2, b_3, \hat{y}_L, \hat{y}_R) = 1/6 + 2/6 = \underline{3/6}$

Figure 2. Running example of our top-down greedy splitting algorithm on $\mathcal{X} = \mathbb{R}^2$. Here, we have six labeled instances $(\boldsymbol{x}, y)$ as a training sample S, and blue (resp. red) indicates the desired class $y = +1$ (reps. undesired class $y = -1$). We assume that while the feature x_1 is immutable (i.e., can not be changed by an action), the feature x_2 is mutable (i.e., can be changed by an action). We also assume thresholds $B_1 = \{b_1\}$ and $B_2 = \{b_2, b_3\}$ for the features x_1 and x_2, respectively, and set the trade-off parameter as $\lambda = 1$. For each instance $\boldsymbol{x}$ that is located in the red region corresponding to $\hat{y} = -1$, the green line stands for the range where $\boldsymbol{x}$ can reach by some action $\boldsymbol{a}$ within a pre-defined cost budget ε. For each split condition (d, b), we first determine predictive labels $\hat{y}_\mathrm{L}$ and $\hat{y}_\mathrm{R}$ by solving the inner problem of (2), and then find a best split condition that minimizes our objective function Φ_λ. Note that the objective value of each split condition (d, b) can be computed in amortized constant time.

denote the total number of leaves such that $v_{n,i'} = 1$ and $\hat{y}_{i'} = +1$ by $V_n = \sum_{i' \in [I]} v_{n,i'} \cdot \mathbb{I}\left[\hat{y}_i = +1\right]$. By definition, $\omega_{n,i} = 1$ holds if and only if $V_n - v_{n,i} \cdot \mathbb{I}\left[\hat{y}_i = +1\right] = 0$. Using these global variables, we define four terms

$$\begin{aligned}
&M = \sum\nolimits_{n=1}^{N} \omega_{n,i}, \quad \bar{M} = \sum\nolimits_{n=1}^{N} \omega_{n,i} \cdot v_{n,i}, \\
&M_\mathrm{L} = \sum\nolimits_{n=1}^{N} \omega_{n,i} \cdot v_{n,i} \cdot g_n(d, b), \\
&M_\mathrm{R} = \sum\nolimits_{n=1}^{N} \omega_{n,i} \cdot v_{n,i} \cdot \bar{g}_n(d, b),
\end{aligned}$$

where $g_n(d, b) = \mathbb{I}[x_{n,d} + \min_{a \in \mathcal{A}_\varepsilon(\boldsymbol{x}_n)} a_d \leq b]$ and $\bar{g}_n(d, b) = \mathbb{I}[x_{n,d} + \max_{a \in \mathcal{A}_\varepsilon(\boldsymbol{x}_n)} a_d > b]$. From our assumption on the cost function c and the definitions of r_L and r_R, we have $v_n(r_\mathrm{L}) = v_{n,i} \cdot g_n(d, b)$ and $v_n(r_\mathrm{R}) = v_{n,i} \cdot \bar{g}_n(d, b)$ (Guo et al., 2022). Note that we can compute $g_n(d, b)$ and $\bar{g}_n(d, b)$ as constants in advance when $\boldsymbol{x}_n, \mathcal{A}_\varepsilon(\boldsymbol{x}_n)$, and B_d are given. Using $M, \bar{M}, M_\mathrm{L}, M_\mathrm{R}$, we can write the empirical recourse risk of h' as

$$\hat{\Omega}_\varepsilon(h' \mid S) = \begin{cases} \frac{M}{N} & (\hat{y}_\mathrm{L} = -1 \land \hat{y}_\mathrm{R} = -1), \\ \frac{M - M_\mathrm{L}}{N} & (\hat{y}_\mathrm{L} = +1 \land \hat{y}_\mathrm{R} = -1), \\ \frac{M - M_\mathrm{R}}{N} & (\hat{y}_\mathrm{L} = -1 \land \hat{y}_\mathrm{R} = +1), \\ \frac{M - \bar{M}}{N} & (\hat{y}_\mathrm{L} = +1 \land \hat{y}_\mathrm{R} = +1), \end{cases} \tag{3}$$

which can be verified by the definition of $\hat{\Omega}_\varepsilon$ and h'. By definition, M_L and M_R depend on the split condition (d, b). While a naive computation of M_L and M_R requires $\mathcal{O}(N)$ time for each (d, b), we can compute them in amortized constant time using the monotonicity of $g_n(d, b)$ and $\bar{g}_n(d, b)$ with respect to n and b, which is described in Appendix A.1.

Algorithm 1 presents an algorithm for the problem (2). It first determines predictive labels $\hat{y}_\mathrm{L}$ and $\hat{y}_\mathrm{R}$ by solving its inner problem for each (d, b). As with the empirical recourse risk, we can also express the empirical risk of h' by

$$\hat{R}(h' \mid S) = \begin{cases} \frac{N^+ + \bar{N}}{N} & (\hat{y}_\mathrm{L} = -1 \land \hat{y}_\mathrm{R} = -1), \\ \frac{N_\mathrm{L}^- + N_\mathrm{R}^+ + \bar{N}}{N} & (\hat{y}_\mathrm{L} = +1 \land \hat{y}_\mathrm{R} = -1), \\ \frac{N_\mathrm{L}^+ + N_\mathrm{R}^- + \bar{N}}{N} & (\hat{y}_\mathrm{L} = -1 \land \hat{y}_\mathrm{R} = +1), \\ \frac{N^- + \bar{N}}{N} & (\hat{y}_\mathrm{L} = +1 \land \hat{y}_\mathrm{R} = +1), \end{cases} \tag{4}$$

and compute the terms for each (d, b) in amortized constant time. After optimizing $\hat{y}_\mathrm{L}$ and $\hat{y}_\mathrm{R}$ for all (d, b), Algorithm 1 determines a best split condition (d^*, b^*) that minimizes Φ_λ. In Proposition 3.1, we show that our algorithm for solving the problem (2) runs in the same time complexity as the standard learning algorithm for classification trees.

Proposition 3.1. *Algorithm 1 solves the problem* (2) *in* $\mathcal{O}(D \cdot N)$.

Figure 2 presents a running example of Algorithm 1. After determining a best split condition (d^*, b^*) and predictive labels $\hat{y}_\mathrm{L}^*, \hat{y}_\mathrm{R}^*$ by Algorithm 1, we apply Algorithm 1 to the left and right children with the updated global variables. We grow a tree by recursively repeating this procedure until some termination condition is met (e.g., maximum depth). Note that we need to compute the permutations σ_d and indicators $g_n(d, b)$ and $\bar{g}_n(d, b)$ for $n \in [N], d \in [D], b \in B_d$, which roughly take $\mathcal{O}(D \cdot N \cdot \log N)$ and $\mathcal{O}(D \cdot N^2)$, respectively. However, we can compute them once as preprocessing, and the computation time of growing a tree was often more dominant than that of such preprocessing in practice.

3.2. Set-Cover Relabeling

While minimizing the objective function Φ_λ of the problem (2) encourages the existence of recourse actions, the learned classification tree $\hat{h}$ does not necessarily satisfy the constraint $\hat{\Omega}_\varepsilon(\hat{h} \mid S) \leq \delta$ of Problem 2.1. We now introduce a post-processing task, called *relabeling* (Kamiran et al., 2010), that modifies the predictive labels of leaves in $\hat{h}$ so as to satisfy the constraint. We also show that the task can be reduced to a variant of the *minimum set cover* problem, which is known to be efficiently solved by a greedy algorithm with an approximation guarantee (Chvátal, 1979; Kearns, 1990).

Let $\hat{I}$ be the total number of leaves in a learned classification tree $\hat{h}$. To satisfy the constraint, a trivial solution is to set all the predictive labels of $\hat{h}$ to $+1$. However, it means to make $\hat{h}$ a constant classifier, i.e., $\hat{h}(\boldsymbol{x}) = +1$ for any $\boldsymbol{x} \in \mathcal{X}$, and may significantly degrade the predictive performance. Thus, our aim is to determine the predictive label $\hat{y}_i$ of each leaf $i \in [\hat{I}]$ so as to satisfy the constraint $\hat{\Omega}_\varepsilon(\hat{h} \mid S) \leq \delta$ without increasing the empirical risk $\hat{R}(\hat{h} \mid S)$ as much as possible.

We can minimize $\hat{R}$ by setting $\hat{y}_i$ as the majority class among the instances $\boldsymbol{x}_n$ such that $\boldsymbol{x}_n \in r_i$. In addition, our key observation on $\hat{\Omega}_\varepsilon$ is that we can express it as $\hat{\Omega}_\varepsilon(h \mid S) = 1 - \frac{1}{N} |\bigcup_{i \in [I] : \hat{y}_i = +1} \mathcal{N}_i|$, where $\mathcal{N}_i := \{n \in [N] \mid v_n(r_i)\}$ is the set of instances $\boldsymbol{x}_n$ that can reach the leaf i by some action $\boldsymbol{a} \in \mathcal{A}_\varepsilon(\boldsymbol{x}_n)$. It implies that we can reduce $\hat{\Omega}_\varepsilon(h \mid S)$ only by changing the predictive labels of the leaves i with $\hat{y}_i = -1$ to $+1$. Based on these observations, we first initialize each predictive label as $\hat{y}_i = 1 - 2 \cdot \mathbb{I}\left[N_i^+ < N_i^-\right]$, where $N_i^+ = |\{n \in [N] \mid \boldsymbol{x}_n \in r_i \land y_n = +1\}|$ and $N_i^- = |\{n \in [N] \mid \boldsymbol{x}_n \in r_i \land y_n = -1\}|$. Then, we select leaves $\mathcal{I} \subseteq [\hat{I}]$ that should be changed from $\hat{y}_i = -1$ to $\hat{y}_i = +1$ so as to minimize the increase of $\hat{R}(\hat{h} \mid S)$ under the constraint $\hat{\Omega}_\varepsilon(\hat{h} \mid S) \leq \delta$. By ignoring constant terms, this task can be formulated as

$$\min_{\mathcal{I} \subseteq \mathcal{I}^-} \frac{1}{N} \sum\nolimits_{i \in \mathcal{I}} c_i \quad \text{s.t.} \quad \frac{1}{N} \left| \bigcup\nolimits_{i \in \mathcal{I}^+ \cup \mathcal{I}} \mathcal{N}_i \right| \geq 1 - \delta, \quad (5)$$

where $\mathcal{I}^- = \{i \in [\hat{I}] \mid \hat{y}_i = -1\}$, $\mathcal{I}^+ = [\hat{I}] \setminus \mathcal{I}^-$, and $c_i = N_i^- - N_i^+$. By modifying the predictive labels of the leaves in a feasible solution $\mathcal{I}$ to the problem (5) as $+1$, we can obtain a classification tree h^* that satisfies $\hat{\Omega}_\varepsilon(h^* \mid S) \leq \delta$. In Proposition 3.2, we show that the problem (5) belongs to a class of a well-known combinatorial optimization problem that can be efficiently solved with an approximation guarantee (Chvátal, 1979).

Proposition 3.2. *The problem* (5) *is reduced to the weighted partial cover problem.*

While the weighted partial cover problem is NP-hard, there exist polynomial-time approximation algorithms (e.g., a greedy algorithm achieves $(2 \cdot H(N) + 3)$-approximation, where $H(N) = \sum_{n=1}^{N} 1/n = \Theta(\ln N)$ (Kearns, 1990)).

PAC-Style Guarantee. By solving the problem (5), we can obtain a classifier h^* that can provide valid actions to at least $100 \cdot (1 - \delta)$ % instances in a given training sample S. However, since our empirical recourse risk $\hat{\Omega}_\varepsilon$ is the probability of the existence of valid actions estimated over S, there is no guarantee for unseen test instances. To analyze this risk, we show a PAC-style bound on the estimation error of $\hat{\Omega}_\varepsilon$ in Proposition 3.3 (Mohri et al., 2012).

Proposition 3.3. *For a classifier $h \in \mathcal{H}$, let $\Omega_\varepsilon(h) := \mathbb{P}_x[\exists \boldsymbol{a} \in \mathcal{A}_\varepsilon(\boldsymbol{x}) : h(\boldsymbol{x}+\boldsymbol{a}) = +1]$ be the expected recourse risk of h. Given a sample S and $\alpha > 0$, the following inequality holds with probability at least $1 - \alpha$:*

$$\Omega_\varepsilon(h) \leq \hat{\Omega}_\varepsilon(h \mid S) + \sqrt{\frac{\ln |\mathcal{H}| - \ln \alpha}{2 \cdot |S|}}.$$

Proposition 3.3 implies that the estimation error of $\hat{\Omega}_\varepsilon$ depends mainly on the sizes of $\mathcal{H}$ and S. Since the problem (5) considers only classification trees that are same with $\hat{h}$ except for the predictive labels of the leaves in $\mathcal{I}^-$, we have $|\mathcal{H}| = 2^{|\mathcal{I}^-|}$. Thus, by replacing δ in the problem (5) with $\delta' = \delta - \sqrt{(|\mathcal{I}^-| \cdot \ln 2 - \ln \alpha)/(2 \cdot N)}$, we can obtain a classification tree h^* that satisfies $\Omega_\varepsilon(h^*) \leq \delta$ with probability at least $1 - \alpha$. Note that this condition is equivalent to the *probably approximately has recourse (PARE)* condition proposed by (Ross et al., 2021).

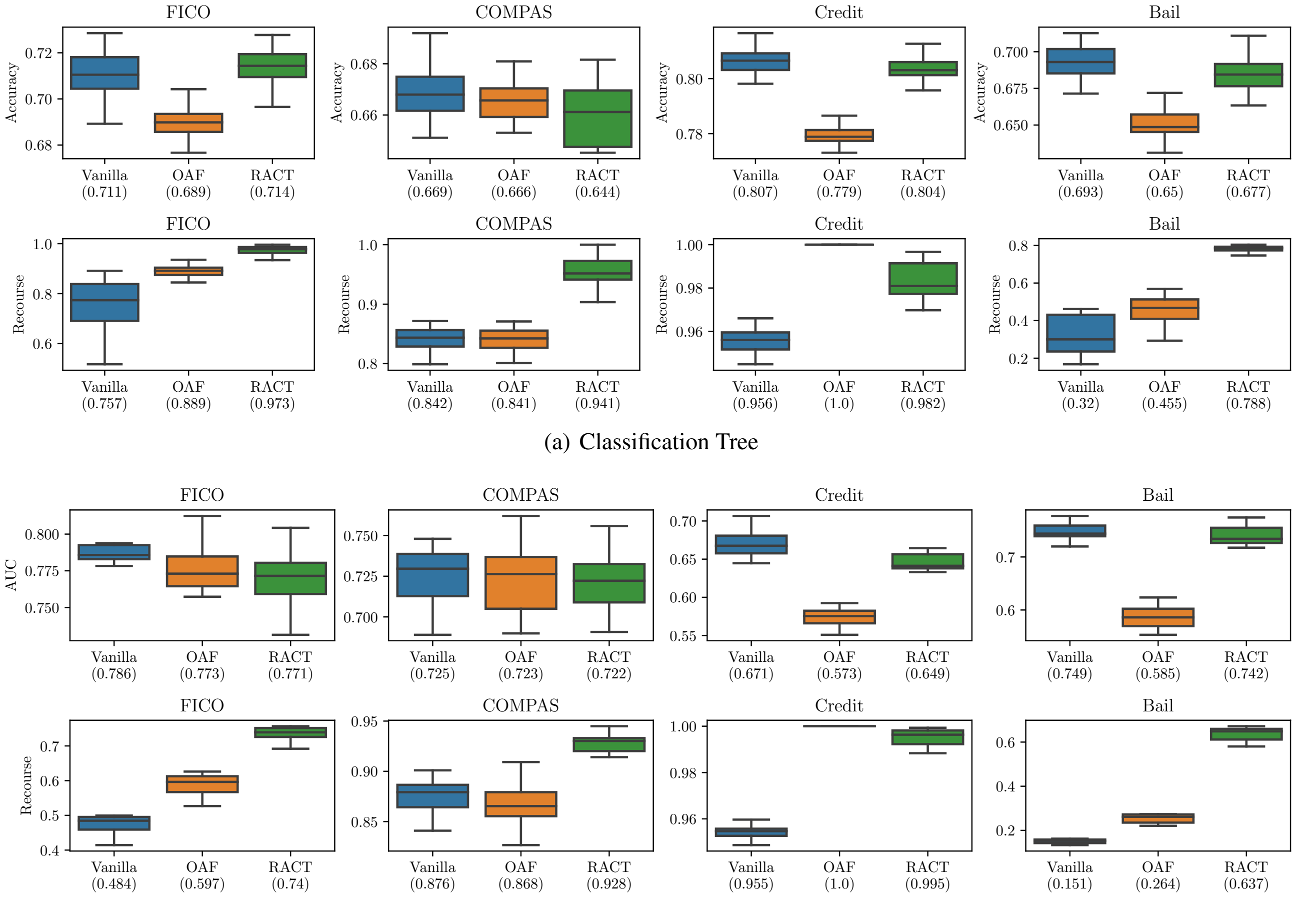
3.3. Extension to Random Forest

To learn a tree ensemble classifier, we employ our RACT algorithm as a base learner of the random forest (Breiman, 2001). A random forest classifier h makes predictions by majority voting of T classification trees $h_1, \dots, h_T$, i.e., $h(\boldsymbol{x}) = \mathrm{sgn}(\sum_{t=1}^{T} h_t(\boldsymbol{x}))$. To apply RACT to the random forest framework, we can simply replace the algorithm for finding a best split condition of each h_t with Algorithm 1.

4. Experiments

To investigate the performance of our RACT, we conducted experiments on real datasets. All the code was implemented in Python 3.10 with Numba 0.56.4 and is available at `https://github.com/kelicht/ract`. All the experiments were conducted on macOS Monterey with Apple M1 Pro CPU and 32 GB memory.

Our experimental evaluation aims to answer the following questions: (i) How are the predictive accuracy and recourse guarantee of tree-based models learned by our RACT compared to those by the baselines? (ii) How is the quality of the recourse actions extracted from the tree-based models in the sense of their practicality? (iii) Can our RACT balance the trade-off between accuracy and recourse? Due to page limitations, the complete results are shown in Appendix C.



(a) Classification Tree

(b) Random Forest

Figure 3. Experimental results of baseline comparison. Figures in the top (resp. bottom) row present predictive accuracy (resp. recourse ratio). Our RACT attained higher recourse ratio than the baselines while keeping comparable accuracy on almost all the datasets.

Experimental Settings. We used four real datasets: FICO ($N = 9871, D = 23$) (FICO et al., 2018), COMPAS ($N = 6167, D = 14$) (Angwin et al., 2016), Credit ($N = 30000, D = 16$) (Yeh & hui Lien, 2009), and Bail ($N = 8923, D = 16$) (Schmidt & Witte, 1988). As a cost function, we used the *max percentile shift (MPS)* (Ustun et al., 2019) defined as

$$c(\boldsymbol{a} \mid \boldsymbol{x}) = \max_{d \in [D]} |Q_d(x_d + a_d) - Q_d(x_d)|,$$

where Q_d is the cumulative distribution function of a feature d. To optimize an action $\boldsymbol{a}$ for each instance $\boldsymbol{x}$ by solving the problem (1), we employed the *feature tweaking algorithm* (Tolomei et al., 2017), which is a fast heuristic algorithm for tree-based models. Note that we also employed the exact method based on integer optimization (Cui et al., 2015; Kanamori et al., 2020), and its results are presented in Appendix C. To the best of our knowledge, there is no existing method for learning tree-based models while guaranteeing the existence of actions. Thus, we compared our RACT with two baselines: learning with no constraint on recourse (*Vanilla*) and learning with only actionable features (*OAF*), following (Dominguez-Olmedo et al., 2022).

4.1. Performance Comparison

First, we evaluate the performance of classification trees and random forest classifiers learned by our RACT in comparison with the baselines. We conducted 10-fold cross validation, and measured (i) the average accuracy and AUC on the test set, (ii) the average recourse ratio, which is defined as the ratio of the test instances that are guaranteed valid actions whose costs are less than $\varepsilon = 0.3$, and (iii) the average running time. For the baselines and our RACT, we trained classification trees with a maximum depth of 64 and random forest classifiers with $T = 200$ classification trees. For each dataset, we determined the hyper-parameters δ and λ of our RACT based on the results of our trade-off analyses in Section 4.3. Specifically, we first drew scatter plots between the average accuracy and recourse ratio of

Table 1. Experimental results on the average running time [s] for random forest classifiers. There is no significant difference between Vanilla and our RACT on almost all the datasets.

Dataset	Vanilla	OAF	RACT
FICO	42.09 ± 1.8	35.25 ± 1.52	48.07 ± 2.33
COMPAS	2.31 ± 0.32	1.76 ± 0.14	2.07 ± 0.13
Credit	292.53 ± 21.6	268.41 ± 18.9	277.40 ± 25.8
Bail	24.84 ± 1.93	3.26 ± 0.35	23.04 ± 1.64

Table 2. Average cost of extracted actions (lower is better). We used the MPS (Ustun et al., 2019) as a cost function c. Our RACT attained lower costs than the baselines regardless of the datasets.

Dataset	Vanilla	OAF	RACT
FICO	0.447 ± 0.05	0.407 ± 0.03	$\mathbf{0.283 \pm 0.01}$
COMPAS	0.298 ± 0.02	0.28 ± 0.01	$\mathbf{0.232 \pm 0.02}$
Credit	0.293 ± 0.02	N/A	$\mathbf{0.166 \pm 0.04}$
Bail	0.763 ± 0.03	0.525 ± 0.04	$\mathbf{0.419 \pm 0.05}$

the model trained with each δ and λ, and selected one of the Pareto-optimal solutions that achieves a good balance between the accuracy and recourse ratio as δ and λ.

Figure 3 presents the results on the predictive accuracy and recourse ratio. From Figure 3, we observed that (i) our RACT attained comparable predictive accuracy to the baselines on almost all the datasets, and (ii) our RACT achieved significantly higher recourse ratios than the baselines except for OAF on the Credit dataset[1]. These results suggest that our RACT achieved higher recourse ratios than the baselines while keeping comparable accuracy. Table 1 shows the average running time for random forests. We can see that there is no significant difference in the running time between Vanilla and RACT, which indicates that our algorithm that considers the additional constraint on the empirical recourse risk performed as fast as the existing one without the constraint. In summary, we have confirmed that *our method succeeded in guaranteeing the existence of recourse actions for more instances than the baselines without compromising predictive accuracy and computational efficiency*.

4.2. Recourse Quality Analysis

Next, we examine the quality of extracted actions from tree-based models learned by our RACT. To assess the practicality of the actions, we evaluate (i) their average *cost* (Ustun et al., 2019), (ii) their *plausibility* (Parmentier & Vidal, 2021), and (iii) their ratio that they are valid, which we call validity, in the *causal recourse* setting (Karimi et al., 2021). We report the results for random forest classifiers here.

Table 3. Average plausibility of extracted actions (lower is better). Following (Parmentier & Vidal, 2021), we measured the outlier score estimated by isolation forests. There is no significant difference in the plausibility between the baselines and our RACT.

Dataset	Vanilla	OAF	RACT
FICO	0.456 ± 0.0	0.446 ± 0.0	$\mathbf{0.437 \pm 0.0}$
COMPAS	$\mathbf{0.44 \pm 0.01}$	0.447 ± 0.01	0.453 ± 0.01
Credit	0.526 ± 0.01	N/A	$\mathbf{0.523 \pm 0.01}$
Bail	$\mathbf{0.504 \pm 0.01}$	0.507 ± 0.0	0.512 ± 0.01

Table 4. Average validity of extracted actions under the causal recourse constraint proposed by Karimi et al. (2021) (higher is better). Our RACT attained higher validity than the baselines even in the causal recourse setting.

Dataset	Vanilla	OAF	RACT
FICO	0.316 ± 0.04	0.502 ± 0.03	$\mathbf{0.648 \pm 0.03}$
COMPAS	0.599 ± 0.02	0.642 ± 0.04	$\mathbf{0.721 \pm 0.02}$
Credit	0.669 ± 0.05	N/A	$\mathbf{0.964 \pm 0.03}$
Bail	0.174 ± 0.04	0.201 ± 0.04	$\mathbf{0.539 \pm 0.03}$

Cost. Table 2 shows the average cost $c(\boldsymbol{a} \mid \boldsymbol{x})$ of the obtained valid actions $\boldsymbol{a}$ for test instances $\boldsymbol{x}$. From Table 2, we see that RACT attained the lowest costs regardless of the datasets, which suggests that our RACT succeeded in providing affected individuals with more efficient and executable actions than those of the baselines.

Plausibility. Table 3 shows the average plausibility of the obtained valid actions $\boldsymbol{a}$ for test instances $\boldsymbol{x}$. To evaluate the plausibility of $\boldsymbol{a}$ for $\boldsymbol{x}$, the previous studies often use the outlier score of $\boldsymbol{x}+\boldsymbol{a}$ (Parmentier & Vidal, 2021). Following the previous study, we employed isolation forests (Liu et al., 2008) for estimating the outlier score. From Table 3, we observed that there is no significant difference in plausibility between the baselines and RACT. These results indicate that our RACT did not harm the plausibility of actions, and its provided actions are as reasonable as those of the baselines.

Causal Recourse. Table 4 presents the average validity of the obtained actions for test instances in the causal recourse setting (Karimi et al., 2021). For each dataset, we estimated causal DAGs by DirectLiNGAM (Shimizu et al., 2011), which is a popular method for estimating causal DAGs, and optimized actions under the causal recourse constraint with the estimated causal DAGs. From Table 4, we see that our RACT attained higher validity than the baselines. These results indicate that our RACT succeeded in guaranteeing the existence of valid actions for more individuals than the baselines even in the causal recourse setting.

To conclude, we have confirmed that *our method could provide affected individuals with practical recourse actions in the sense of their cost, plausibility, and causality*.

[1] We would like to note that OAF on the Credit dataset obtained a constant classifier that predicts any input instance as the desired class for all the folds, which is the reason why OAF achieved 100% recourse ratio on the Credit dataset in Figure 3.

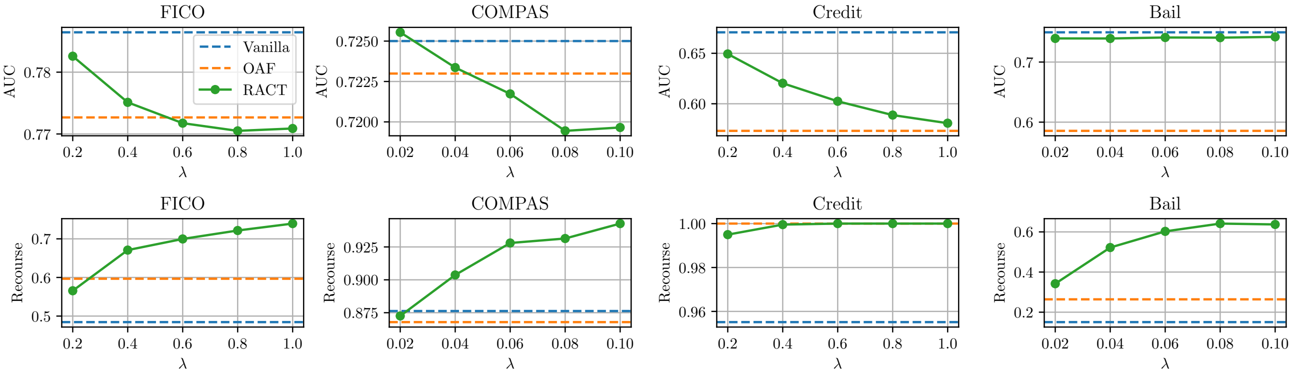


Figure 4. Sensitivity analyses of the trade-off parameter λ with respect to the average AUC and recourse ratio. While the value ranges of λ are different across the datasets, we can see that the recourse ratio (resp. AUC) was improved as λ increased (resp. decreased) overall.

4.3. Trade-Off Analysis

Finally, we analyze the trade-off between the predictive accuracy and recourse guarantee of our RACT by varying its trade-off parameters. As with Section 4.2, we report the results for random forest classifiers here. We trained random forest classifiers by varying the trade-off parameter λ, and measured their average AUC and recourse ratio.

Figure 4 shows the results for each λ. From Figure 4, we see that the recourse ratio (resp. AUC) was improved by increasing (resp. decreasing) λ on almost all the datasets. These results suggest not only that we can attain a desired trade-off by tuning λ, but also that we have a chance to achieve a higher recourse ratio than the baselines while keeping comparable accuracy by choosing an appropriate value of λ. In summary, we have confirmed that *our method can balance the trade-off between the predictive accuracy and recourse guarantee by tuning the parameter λ.*

5. Conclusion

In this paper, we proposed a new framework for learning tree-based models, named recourse-aware classification tree (RACT), that can provide both accurate predictions and executable recourse actions. We first introduced the recourse risk to evaluate the ratio of instances that are not guaranteed to have recourse actions, and formulated a task of learning a classification tree with the constraint on the recourse risk. Then, we proposed an efficient top-down greedy learning algorithm by leveraging adversarial training techniques, and showed that we can modify a learned tree so as to satisfy the constraint by solving a well-known variant of the minimum set-cover problem. We also showed that our algorithm can be easily applied to the random forest, which is one of the popular frameworks for learning tree ensembles. Experimental results demonstrated that our RACT succeeded in guaranteeing recourse actions for more individuals than the baselines while keeping comparable accuracy and efficiency.

Limitations and Future Work. There are several directions to improve our RACT. First, it is important to extend our RACT to the popular frameworks of gradient boosted trees, such as XGBoost (Chen & Guestrin, 2016) and LightGBM (Ke et al., 2017). For that purpose, we need to extend our algorithm to learning regression trees (Andriushchenko & Hein, 2019). However, it is not trivial to develop such methods without degrading computational efficiency. In addition, the computational efficiency of our proposed algorithm relies on our assumption of the ℓ_∞-type cost function c. While we can easily decide whether the budget constraint is violated or not by checking each feature independently for the ℓ_∞-type cost functions, such a property does not hold for general cost functions, including ℓ_1- or ℓ_2-type cost functions. We expect that when we introduce some heuristic strategies, such as changing the budget parameter ε for each depth of tree (Wang et al., 2020), our algorithm can be extended to deal with the ℓ_1- and ℓ_2-type cost functions with keeping efficiency. Finally, in our experiments of Section 4.3, we individually set the range of our trade-off parameter λ to be searched for each dataset. Because such a procedure is costly in practice, we need to further analyze its sensitivity to determine its default range automatically.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work was supported in part by JST ACT-X JPMJAX23C6.

Impact Statement

Positive Impacts. Our proposed method, named RACT, is a new framework that aims to learn accurate tree-based models while guaranteeing the existence of recourse actions. As demonstrated in our experiments in Section 4, tree-based models trained by our method can provide executable recourse actions to more individuals than the existing methods without degrading predictive accuracy. Thus, our method enables us to learn tree-based models that make accurate predictions and guarantee recourse actions, which improves the trustworthiness of algorithmic decision-making for critical tasks in the real world such as loan approvals and judicial decisions (Karimi et al., 2022; Ross et al., 2021). In addition, our method gives decision-makers an insight into the trade-off between the predictive accuracy and recourse ratio by tuning the hyper-parameter λ (Levanon & Rosenfeld, 2021; Olckers & Walsh, 2023).

Negative Impacts. While our method has several positive impacts, there may exist some negative impacts as well. First, in practice, it is sometimes undesirable for decision-makers to guarantee the existence of executable actions for all individuals. For example, providing easy actions for granting loans to applicants who do not have the capacity to repay might cause a serious financial crisis in the future. However, since our method can adjust the ratio of individuals who are guaranteed to have executable recourse actions by tuning its hyper-parameter λ, it helps decision-makers provide recourse actions to appropriate individuals while keeping accurate decisions (Levanon & Rosenfeld, 2021; Olckers & Walsh, 2023). Second, there is a risk that our method may be used maliciously to train a model that provides specific actions for occuring some undesired situations, such as discrimination. To avoid this risk, we may need to check the actions provided to affected individuals before deploying the models (e.g., using the existing frameworks for obtaining the global summary of recourse actions (Rawal & Lakkaraju, 2020; Kanamori et al., 2022)).

References

Andriushchenko, M. and Hein, M. Provably robust boosted decision stumps and trees against adversarial attacks. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, pp. 13017–13028, 2019.

Angwin, J., Larson, J., Mattu, S., and Kirchner, L. Machine Bias. *ProPublica*, 2016.

Breiman, L. Random forests. *Machine Learning*, 45(1): 5–32, 2001.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Wadsworth, 1984.

Carreira-Perpiñán, M. A. and Hada, S. S. Very fast, approximate counterfactual explanations for decision forests. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, pp. 6935–6943, 2023.

Chen, H., Zhang, H., Boning, D., and Hsieh, C.-J. Robust decision trees against adversarial examples. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 1122–1131, 2019.

Chen, T. and Guestrin, C. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

Chvátal, V. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

Cui, Z., Chen, W., He, Y., and Chen, Y. Optimal action extraction for random forests and boosted trees. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 179–188, 2015.

Dominguez-Olmedo, R., Karimi, A. H., and Schölkopf, B. On the adversarial robustness of causal algorithmic recourse. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 5324–5342, 2022.

FICO, Google, Imperial College London, MIT, University of Oxford, UC Irvine, and UC Berkeley. Explainable Machine Learning Challenge, 2018. URL https://community.fico.com/s/explainable-machine-learning-challenge. Accessed: 2024-05-24.

Freitas, A. A. Comprehensible classification models: A position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):1–10, 2014.

Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. Why do tree-based models still outperform deep learning on typical tabular data? In *Proceedings of the 36th Conference on Neural Information Processing Systems*, pp. 507–520, 2022.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., and Pedreschi, D. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5):1–42, 2018.

Guo, J.-Q., Teng, M.-Z., Gao, W., and Zhou, Z.-H. Fast provably robust decision trees and boosting. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 8127–8144, 2022.

Hara, S. and Yoshida, Y. Average sensitivity of decision tree learning. In *Proceedings of the 11th International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=boik01yhssB.

Hu, X., Rudin, C., and Seltzer, M. Optimal sparse decision trees. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, pp. 7265–7273, 2019.

Hyafil, L. and Rivest, R. L. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

Kamiran, F., Calders, T., and Pechenizkiy, M. Discrimination aware decision tree learning. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, pp. 869–874, 2010.

Kanamori, K., Takagi, T., Kobayashi, K., and Arimura, H. DACE: Distribution-aware counterfactual explanation by mixed-integer linear optimization. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pp. 2855–2862, 2020.

Kanamori, K., Takagi, T., Kobayashi, K., Ike, Y., Uemura, K., and Arimura, H. Ordered counterfactual explanation by mixed-integer linear optimization. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pp. 11564–11574, 2021.

Kanamori, K., Takagi, T., Kobayashi, K., and Ike, Y. Counterfactual explanation trees: Transparent and consistent actionable recourse with decision trees. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, pp. 1846–1870, 2022.

Karimi, A.-H., Schölkopf, B., and Valera, I. Algorithmic recourse: From counterfactual explanations to interventions. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 353–362, 2021.

Karimi, A.-H., Barthe, G., Schölkopf, B., and Valera, I. A survey of algorithmic recourse: Contrastive explanations and consequential recommendations. *ACM Computing Surveys*, 55(5):1–29, 2022.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pp. 3149–3157, 2017.

Kearns, M. J. *The computational complexity of machine learning*. The MIT Press, 1990.

Levanon, S. and Rosenfeld, N. Strategic classification made practical. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 6243–6253, 2021.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. Isolation forest. In *Proceedings of the 2008 IEEE International Conference on Data Mining*, pp. 413–422, 2008.

Lucic, A., Oosterhuis, H., Haned, H., and de Rijke, M. FOCUS: Flexible optimizable counterfactual explanations for tree ensembles. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pp. 5313–5322, 2022.

Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of Machine Learning*. The MIT Press, 2012.

Olckers, M. and Walsh, T. Incentives to offer algorithmic recourse. *arXiv,* arXiv:2301.12884, 2023.

Parmentier, A. and Vidal, T. Optimal counterfactual explanations in tree ensembles. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 8422–8431, 2021.

Pawelczyk, M., Broelemann, K., and Kasneci, G. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference 2020*, pp. 3126–3132, 2020.

Rawal, K. and Lakkaraju, H. Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, pp. 12187–12198, 2020.

Rokach, L. and Maimon, O. Top-down induction of decision trees classifiers - a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):476–487, 2005.

Ross, A., Lakkaraju, H., and Bastani, O. Learning models for actionable recourse. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pp. 18734–18746, 2021.

Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215, 2019.

Schmidt, P. and Witte, A. D. Predicting recidivism in north carolina, 1978 and 1980. *Inter-university Consortium for Political and Social Research,*, 1988.

Shimizu, S., Inazumi, T., Sogawa, Y., Hyvärinen, A., Kawahara, Y., Washio, T., Hoyer, P. O., and Bollen, K. Directlingam: A direct method for learning a linear non-gaussian structural equation model. *Journal of Machine Learning Research*, 12:1225–1248, 2011.

Sullivan, E. and Verreault-Julien, P. From explanation to recommendation: Ethical standards for algorithmic recourse.

In *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 712–722, 2022.

Tolomei, G., Silvestri, F., Haines, A., and Lalmas, M. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 465–474, 2017.

Ustun, B., Spangher, A., and Liu, Y. Actionable recourse in linear classification. In *Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency*, pp. 10–19, 2019.

Venkatasubramanian, S. and Alfano, M. The philosophical basis of algorithmic recourse. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pp. 284–293, 2020.

Verma, S., Boonsanong, V., Hoang, M., Hines, K. E., Dickerson, J. P., and Shah, C. Counterfactual explanations and algorithmic recourses for machine learning: A review. *arXiv,* arXiv:2010.10596, 2020.

Vos, D. and Verwer, S. Efficient training of robust decision trees against adversarial examples. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 10586–10595, 2021.

Vos, D. and Verwer, S. Adversarially robust decision tree relabeling. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 203–218, 2022.

Wachter, S., Mittelstadt, B., and Russell, C. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31:841–887, 2018.

Wang, Y., Zhang, H., Chen, H., Boning, D., and Hsieh, C.-J. On lp-norm robustness of ensemble decision stumps and trees. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 10104–10114, 2020.

Yeh, I.-C. and hui Lien, C. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2, Part 1):2473–2480, 2009.

A. Omitted Proofs

A.1. Proof of Proposition 3.1

Proof of Proposition 3.1. To prove Proposition 3.1, we first prove the correctness of Algorithm 1. Then, we show the complexity of Algorithm 1.

Correctness. At first, we show that Algorithm 1 computes the empirical recourse risk $\hat{\Omega}_{\varepsilon}(h' \mid S)$ for each split condition (d, b) and predictive labels $\hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}}$ correctly. Recall that we define $\omega_{n,i} = \mathbb{I}\left[\forall i' \in \mathcal{I}_n \setminus \{i\} : \hat{y}_{i'} \neq +1\right]$ and that we have $v_n(r_{\mathrm{L}}) = v_{n,i} \cdot g_n(d, b)$ and $v_n(r_{\mathrm{R}}) = v_{n,i} \cdot \bar{g}_n(d, b)$. From the definitions of our recourse loss l_{rec} and decision stump h', we have

$$
\begin{aligned}
\hat{\Omega}_{\varepsilon}(h' \mid S) &= \frac{1}{N} \sum_{n=1}^{N} \min\{\omega_{n,i}, 1 - v_n(r_{\mathrm{L}}) \cdot \mathbb{I}\left[\hat{y}_{\mathrm{L}} = +1\right], 1 - v_n(r_{\mathrm{R}}) \cdot \mathbb{I}\left[\hat{y}_{\mathrm{R}} = +1\right]\} \\
&= \begin{cases}
\frac{1}{N} \sum_{n=1}^{N} \omega_{n,i}, & (\hat{y}_{\mathrm{L}} = -1 \wedge \hat{y}_{\mathrm{R}} = -1) \\
\frac{1}{N} \sum_{n=1}^{N} \omega_{n,i} \cdot (1 - v_n(r_{\mathrm{L}})) & (\hat{y}_{\mathrm{L}} = +1 \wedge \hat{y}_{\mathrm{R}} = -1) \\
\frac{1}{N} \sum_{n=1}^{N} \omega_{n,i} \cdot (1 - v_n(r_{\mathrm{R}})) & (\hat{y}_{\mathrm{L}} = -1 \wedge \hat{y}_{\mathrm{R}} = +1) \\
\frac{1}{N} \sum_{n=1}^{N} \omega_{n,i} \cdot (1 - v_n(r_i)) & (\hat{y}_{\mathrm{L}} = +1 \wedge \hat{y}_{\mathrm{R}} = +1)
\end{cases} \\
&= \begin{cases}
\frac{1}{N} \sum_{n=1}^{N} \omega_{n,i}, & (\hat{y}_{\mathrm{L}} = -1 \wedge \hat{y}_{\mathrm{R}} = -1) \\
\frac{1}{N} \sum_{n=1}^{N} \omega_{n,i} \cdot (1 - v_{n,i} \cdot g_n(d, b)) & (\hat{y}_{\mathrm{L}} = +1 \wedge \hat{y}_{\mathrm{R}} = -1) \\
\frac{1}{N} \sum_{n=1}^{N} \omega_{n,i} \cdot (1 - v_{n,i} \cdot \bar{g}_n(d, b)) & (\hat{y}_{\mathrm{L}} = -1 \wedge \hat{y}_{\mathrm{R}} = +1) \\
\frac{1}{N} \sum_{n=1}^{N} \omega_{n,i} \cdot (1 - v_{n,i}) & (\hat{y}_{\mathrm{L}} = +1 \wedge \hat{y}_{\mathrm{R}} = +1)
\end{cases} \\
&= \begin{cases}
\frac{M}{N} & (\hat{y}_{\mathrm{L}} = -1 \wedge \hat{y}_{\mathrm{R}} = -1) \\
\frac{M - M_{\mathrm{L}}}{N} & (\hat{y}_{\mathrm{L}} = +1 \wedge \hat{y}_{\mathrm{R}} = -1) \\
\frac{M - M_{\mathrm{R}}}{N} & (\hat{y}_{\mathrm{L}} = -1 \wedge \hat{y}_{\mathrm{R}} = +1) \\
\frac{M - \bar{M}}{N} & (\hat{y}_{\mathrm{L}} = +1 \wedge \hat{y}_{\mathrm{R}} = +1),
\end{cases}
\end{aligned}
$$

which shows the correctness of Equation (3). Thus, we can compute $\hat{\Omega}_{\varepsilon}(h' \mid S)$ correctly if we have computed the terms $M, \bar{M}, M_{\mathrm{L}}, M_{\mathrm{R}}$ for each (d, b). Since M and $\bar{M}$ is independent to (d, b), we can compute them before searching (d, b) (line 4 of Algorithm 1). In contrast, since M_{L} and M_{R} depends on (d, b), we need to compute them for each (d, b).

Here, we show that we can compute M_{L} and M_{R} for a j-th split condition (d, b_j) using the results M'_{L} and M'_{R} for the previous split condition (d, b_{j-1}). For notational simplicity, we assume $x_{1,d} < \cdots < x_{N,d}$. From the definition of g_n, we can see the two monotonic properties on $n \in [N]$ and $j \in [J_d]$: $g_n(d, b_j) \geq g_{n'}(d, b_j)$ for any $n' > n$ and $g_n(d, b_j) \geq g_n(d, b_{j'})$ for any $j' < j$. While the former implies that $g_n(d, b_j) = 0 \implies g_{n'}(d, b_j) = 0$ holds for any $n' > n$, the latter implies that $g_n(d, b_j) = 1 \implies g_n(d, b_{j'}) = 1$ holds for any $j' > j$. Let $m = \min_{n \in [N]: g_n(d, b_{j-1}) = 0} n$ and $m' = \max_{n \in [N]: g_n(d, b_j) = 1} n$. Using the above properties, we have $M_{\mathrm{L}} - M'_{\mathrm{L}} = \sum_{n=m}^{m'} \omega_{n,i} \cdot v_{n,i} \iff M_{\mathrm{L}} = M'_{\mathrm{L}} + \sum_{n=m}^{m'} \omega_{n,i} \cdot v_{n,i}$. Furthermore, we can see the similar monotonic properties of $\bar{g}_n(d, b)$ as well, and we have $M_{\mathrm{R}} = M'_{\mathrm{R}} - \sum_{n=\bar{m}}^{\bar{m}'} \omega_{n,i} \cdot v_{n,i}$, where $\bar{m} = \min_{n \in [N]: \bar{g}_n(d, b_{j-1}) = 1} n$ and $\bar{m}' = \max_{n \in [N]: \bar{g}_n(d, b_j) = 0} n$. In lines 18–20 and lines 21–23, Algorithm 1 updates the terms M_{L} and M_{L} using these facts.

Similarly, we can show that Algorithm 1 computes the empirical risk $\hat{R}(h' \mid S)$ for each split condition (d, b) and predictive labels $\hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}}$ correctly. For Equation (4), we define as

$$
\begin{aligned}
&\hat{N} = \sum\nolimits_{n=1}^{N} (1 - \mathbb{I}\left[\boldsymbol{x}_n \in r_i\right]) \cdot l(y_n, h(\boldsymbol{x}_n))), N^{+} = \sum\nolimits_{n=1}^{N} f_{n,i} \cdot \mathbb{I}\left[y_n = +1\right], N^{-} = \sum\nolimits_{n=1}^{N} f_{n,i} \cdot \mathbb{I}\left[y_n = +1\right], \\
&N^{+}_{\mathrm{L}} = \sum\nolimits_{n=1}^{N} f_{n,i} \cdot \mathbb{I}\left[x_{n,d} \leq b\right] \cdot \mathbb{I}\left[y_n = +1\right], N^{-}_{\mathrm{L}} = \sum\nolimits_{n=1}^{N} f_{n,i} \cdot \mathbb{I}\left[x_{n,d} \leq b\right] \cdot \mathbb{I}\left[y_n = -1\right], \\
&N^{+}_{\mathrm{R}} = \sum\nolimits_{n=1}^{N} f_{n,i} \cdot \mathbb{I}\left[x_{n,d} > b\right] \cdot \mathbb{I}\left[y_n = +1\right], N^{-}_{\mathrm{R}} = \sum\nolimits_{n=1}^{N} f_{n,i} \cdot \mathbb{I}\left[x_{n,d} > b\right] \cdot \mathbb{I}\left[y_n = -1\right],
\end{aligned}
$$

where $f_{n,i} = \mathbb{I}\left[\boldsymbol{x}_n \in r_i\right]$. The terms $N^{+}_{\mathrm{L}}, N^{-}_{\mathrm{L}}, N^{+}_{\mathrm{R}}, N^{-}_{\mathrm{R}}$ depend on (d, b). As with the empirical recourse risk, Algorithm 1 computes them using the results of the previous split condition in lines 12–16, which leverages the monotonic properties of

$\mathbb{I}[x_{n,d} \leq b]$ and $\mathbb{I}[x_{n,d} > b]$. Using the above terms, we have

$$\begin{aligned}
\hat{R}(h' \mid S) &= \frac{1}{N} \sum_{n=1}^{N} (\mathbb{I}[\boldsymbol{x}_n \in r_i] \cdot l(y_n, h(\boldsymbol{x}_n; d, b, \hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}})) + (1 - \mathbb{I}[\boldsymbol{x}_n \in r_i]) \cdot l(y_n, h(\boldsymbol{x}_n))) \\
&= \frac{1}{N} \sum_{n=1}^{N} (f_{n,i} \cdot \mathbb{I}[x_{n,d} \leq b] \cdot l(y_n, \hat{y}_{\mathrm{L}}) + f_{n,i} \cdot \mathbb{I}[x_{n,d} > b] \cdot l(y_n, \hat{y}_{\mathrm{R}})) + \hat{N} \\
&= \begin{cases}
\frac{1}{N} \sum_{n=1}^{N} f_{n,i} \cdot \mathbb{I}[y_n = +1] + \hat{N}, & (\hat{y}_{\mathrm{L}} = -1 \wedge \hat{y}_{\mathrm{R}} = -1) \\
\frac{1}{N} \sum_{n=1}^{N} (f_{n,i} \cdot \mathbb{I}[x_{n,d} \leq b] \cdot \mathbb{I}[y_n = -1] + f_{n,i} \cdot \mathbb{I}[x_{n,d} > b] \cdot \mathbb{I}[y_n = +1]) + \hat{N} & (\hat{y}_{\mathrm{L}} = +1 \wedge \hat{y}_{\mathrm{R}} = -1) \\
\frac{1}{N} \sum_{n=1}^{N} (f_{n,i} \cdot \mathbb{I}[x_{n,d} \leq b] \cdot \mathbb{I}[y_n = +1] + f_{n,i} \cdot \mathbb{I}[x_{n,d} > b] \cdot \mathbb{I}[y_n = -1]) + \hat{N} & (\hat{y}_{\mathrm{L}} = -1 \wedge \hat{y}_{\mathrm{R}} = +1) \\
\frac{1}{N} \sum_{n=1}^{N} f_{n,i} \cdot \mathbb{I}[y_n = -1] + \hat{N} & (\hat{y}_{\mathrm{L}} = +1 \wedge \hat{y}_{\mathrm{R}} = +1)
\end{cases} \\
&= \begin{cases}
\frac{N^{+} + \hat{N}}{N} & (\hat{y}_{\mathrm{L}} = -1 \wedge \hat{y}_{\mathrm{R}} = -1) \\
\frac{N_{\mathrm{L}}^{-} + N_{\mathrm{R}}^{+} + \hat{N}}{N} & (\hat{y}_{\mathrm{L}} = +1 \wedge \hat{y}_{\mathrm{R}} = -1) \\
\frac{N_{\mathrm{L}}^{+} + N_{\mathrm{R}}^{-} + \hat{N}}{N} & (\hat{y}_{\mathrm{L}} = -1 \wedge \hat{y}_{\mathrm{R}} = +1) \\
\frac{N^{-} + \hat{N}}{N} & (\hat{y}_{\mathrm{L}} = +1 \wedge \hat{y}_{\mathrm{R}} = +1),
\end{cases}
\end{aligned}$$

which shows the correctness of Equation (4). Thus, we can compute the empirical risk $\hat{R}(h' \mid S)$ correctly using the terms $\hat{N}, N^{+}, N^{-}, N_{\mathrm{L}}^{+}, N_{\mathrm{L}}^{-}, N_{\mathrm{R}}^{+}, N_{\mathrm{R}}^{-}$.

From the above results, Algorithm 1 can compute $\hat{R}(h' \mid S)$ and $\hat{\Omega}_{\varepsilon}(h' \mid S)$ for a fixed (d, b) and $\hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}}$. Furthermore, we can solve the inner optimization problem of Equation (2) by enumerating four patterns of $\hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}} \in \{\pm 1\}$. Since Algorithm 1 searches all the pairs (d, b) of a feature $d \in [D]$ and threshold $b \in B_d$ and computes the optimal value of the inner optimization problem of Equation (2) for each (d, b) in lines 7–27, it can solve the problem (2) in line 29, which concludes the proof of the correctness of Algorithm 1.

Complexity. Next, we prove that Algorithm 1 runs in $\mathcal{O}(D \cdot N)$. Our complexity analysis of Algorithm 1 can be divided into the following five parts:

- In lines 2–5, Algorithm 1 initializes the terms in $\mathcal{O}(N)$.
- In the for-loop of lines 10–26, each while-loop runs at most N times through the for-loop of a fixed B_d. In addition, the optimization task in line 25 can be solved by enumerating four patterns of $\hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}}$, which can be regarded as a constant time. Since we assume $|B_d| = \mathcal{O}(N)$, the overall complexity of lines 10–26 is $\mathcal{O}(N)$.
- Since the inner for-loop of lines 10–26 takes $\mathcal{O}(N)$, the outer for-loop of lines 7–27 takes $\mathcal{O}(D \cdot N)$.
- The optimization task in line 29 can be solved in $\mathcal{O}(D \cdot N)$ because the objective value of each (d, b) has been computed in lines 7-27.
- In lines 31–35, Algorithm 1 updates the terms in $\mathcal{O}(N)$.

In summary, the overall complexity of Algorithm 1 is $\mathcal{O}(D \cdot N)$, which concludes the proof. □

A.2. Proof of Proposition 3.2

We show that the problem (5) is an instance of the weighted partial cover problem defined as follows (Chvátal, 1979; Kearns, 1990).

Problem A.1 (Weighted Partial Cover)**.** For a positive integer $I, M \in \mathbb{N}$, let $S_1, \dots, S_I \subseteq [M]$ be I subsets of $[M]$ with positive real weights $w_1, \dots, w_I \geq 0$, and we assume that $\bigcup_{i=1}^{I} S_i = [M]$. Given $p \in (0, 1]$, find an optimal solution to the following problem:

$$\underset{\mathcal{I} \subseteq [I]}{\text{minimize}} \sum_{i \in \mathcal{I}} w_i \quad \text{subject to} \quad \frac{1}{M} \left| \bigcup_{i \in \mathcal{I}} S_i \right| \geq p. \tag{6}$$

Proof of Proposition 3.2. Let $\tilde{\mathcal{N}} = \{n \in [N] \mid \exists i' \in \mathcal{I}^+, \exists \boldsymbol{a} \in \mathcal{A}_\varepsilon(\boldsymbol{x}_n) : \boldsymbol{x}_n + \boldsymbol{a} \in r_{i'}\}$ be the set of instances that have at least one reachable leaf i' with $\hat{y}_{i'} = +1$. Using $\tilde{\mathcal{N}}$, we have $\left|\bigcup_{i \in \mathcal{I}^+ \cup \mathcal{I}} \mathcal{N}_i\right| = \tilde{N} + \left|\bigcup_{i \in \mathcal{I}} \tilde{\mathcal{N}_i}\right|$, where $\tilde{N} = |\tilde{\mathcal{N}}|$ and $\tilde{\mathcal{N}_i} = \{n \in [N] \mid n \notin \tilde{\mathcal{N}} \wedge \exists a \in \mathcal{A}_\varepsilon(x_n) : x_n + a \in r_i\}$. Then, we can express the constraint of Equation (5) as

$$\frac{1}{N} \left| \bigcup_{i \in \mathcal{I}^+ \cup \mathcal{I}} \mathcal{N}_i \right| \geq 1 - \delta \iff \frac{1}{N} \left| \bigcup_{i \in \mathcal{I}} \tilde{\mathcal{N}_i} \right| \geq 1 - \delta - \frac{\tilde{N}}{N}.$$

Thus, we can see that the problem of Equation (5) is an instance of the weighted partial cover problem defined by Problem A.1 by replacing (i) p with $1 - \delta - \frac{\tilde{N}}{N}$, (ii) S_i with $\tilde{\mathcal{N}_i}$, (iii) M with N, (iv) w_i with c_i, and (v) $[I]$ with $\mathcal{I}^-$, respectively. □

A.3. Proof of Proposition 3.3

Proof of Proposition 3.3. Recall that our empirical recourse risk is defined as $\hat{\Omega}(h \mid S) = \frac{1}{N} \sum_{n=1}^{N} l_{\mathrm{rec}}(\boldsymbol{x}_n; h)$. Let $\mathcal{D}$ be a distribution over the input domain $\mathcal{X}$. By definition of our recourse loss l_{rec}, we have

$$\begin{aligned} \Omega(h) &= \mathop{\mathbb{P}}_{\boldsymbol{x} \sim \mathcal{D}}[\exists \boldsymbol{a} \in \mathcal{A}_\varepsilon(\boldsymbol{x}) : h(\boldsymbol{x} + \boldsymbol{a}) = +1] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}}[\mathbb{I}[\exists \boldsymbol{a} \in \mathcal{A}_\varepsilon(\boldsymbol{x}) : h(\boldsymbol{x} + \boldsymbol{a}) = +1]] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}}[\min_{\boldsymbol{a} \in \mathcal{A}_\varepsilon(\boldsymbol{x})} l_{01}(+1, h(\boldsymbol{x} + \boldsymbol{a}))] \\ &= \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathcal{D}}[l_{\mathrm{rec}}(\boldsymbol{x}; h)]. \end{aligned}$$

It implies that Ω is the expected value of the recourse loss l_{rec} over the distribution $\mathcal{D}$ and that $\hat{\Omega}$ is the empirical average of l_{rec} over a finite set of N instances $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$ when they are i.i.d. samples drown from $\mathcal{D}$.

Here, we consider the probability of $\Omega(h) - \hat{\Omega}(h \mid S) > \theta$ for some $\theta > 0$ and a fixed classifier $h \in \mathcal{H}$. Recall that our recourse loss l_{rec} is bounded in $[0, 1]$. By applying the Hoeffding's inequality (Mohri et al., 2012), we have

$$\mathbb{P}[\Omega(h) - \hat{\Omega}(h \mid S) > \theta] \leq \exp\left(-2 \cdot N \cdot \theta^2\right).$$

Next, we consider bounding the probability that there exists $h \in \mathcal{H}$ such that $\Omega(h) - \hat{\Omega}(h \mid S) > \theta$ by $\alpha > 0$. Using the union bound, we have

$$\mathbb{P}[\forall h \in \mathcal{H} : \Omega(h) - \hat{\Omega}(h \mid S) > \theta] \leq \sum_{h \in \mathcal{H}} \mathbb{P}[\Omega(h) - \hat{\Omega}(h \mid S) > \theta] \leq |\mathcal{H}| \cdot \exp\left(-2 \cdot N \cdot \theta^2\right).$$

Therefore, we obtain

$$|\mathcal{H}| \cdot \exp\left(-2 \cdot N \cdot \theta^2\right) = \alpha \iff \theta = \sqrt{\frac{\ln |\mathcal{H}| - \ln \alpha}{2N}}.$$

In summary, the following inequality holds with probability $1 - \alpha$:

$$\Omega(h) - \hat{\Omega}(h \mid S) \leq \theta \iff \Omega(h) \leq \hat{\Omega}(h \mid S) + \sqrt{\frac{\ln |\mathcal{H}| - \ln \alpha}{2N}},$$

which concludes the proof. □

B. Implementation Details

B.1. Baseline Methods

Vanilla. To the best of our knowledge, there is no study on learning tree-based models while guaranteeing the existence of recourse actions. Thus, as baseline approaches, we employed *classification and regression trees (CART)* (Breiman et al., 1984) and *random forest* (Breiman, 2001), which are standard learning frameworks for classification trees and tree ensembles.

Algorithm 2 Actionable feature tweaking algorithm for extracting actions from tree-based models (Tolomei et al., 2017).

```
Input: a tree ensemble classifier $h$ with $T$ classification trees $h_1, \dots, h_T$, the region $r_{t,i}$ and predictive label $\hat{y}_{t,i}$ of each leaf $i \in [I_t]$ in
    each tree $h_t$, and an instance $\boldsymbol{x}$ such that $h(\boldsymbol{x}) \neq +1$.
Output: an action $\boldsymbol{a}^*$.
$\boldsymbol{a}^* \leftarrow 0$; $c^* \leftarrow \infty$;
/* Enumerate All Leaves in Tree Ensemble */
for $t = 1, \dots, T$ do
    for $i = 1, \dots, I_t$ do
        /* Find Leaf with Desired Class Label */
        if $\hat{y}_{t,i} = y^*$ then
            /* Optimize Action for Leaf */
            $\hat{\boldsymbol{a}} \leftarrow \arg\min_{\boldsymbol{a} \in \mathcal{A}(\boldsymbol{x})} c(\boldsymbol{a} \mid \boldsymbol{x})$ s.t. $\boldsymbol{x} + \boldsymbol{a} \in r_{t,i}$;
            if $h(\boldsymbol{x} + \hat{\boldsymbol{a}}) = +1$ and $c(\hat{\boldsymbol{a}} \mid x) < c^*$ then
                $\boldsymbol{a}^* \leftarrow \hat{\boldsymbol{a}}$; $c^* \leftarrow c(\hat{\boldsymbol{a}} \mid \boldsymbol{x})$;
            end if
        end if
    end for
end for
return $\boldsymbol{a}^*$;
```

Only Actionable Features (OAF). As another baseline, we employed a modified version of CART and random forest, named *only actionable features (OAF)*, that uses only features that can be changed by actions. Specifically, this approach trains classification trees and tree ensembles using only features that are not specified as immutable in each dataset. This idea is based on the observation of the existing study that relying on actionable features facilitates the existence of recourse actions (Dominguez-Olmedo et al., 2022). Our setting of immutable features in each dataset is shown in Appendix C.1.

B.2. Algorithms for Extracting Actions from Tree-Based Models

To extract actions from tree-based models, there exist algorithms based on several mathematical techniques, such as integer optimization (Cui et al., 2015; Kanamori et al., 2020; Parmentier & Vidal, 2021), probabilistic approximation (Lucic et al., 2022), and heuristic methods (Tolomei et al., 2017; Carreira-Perpiñán & Hada, 2023). In this paper, we employed *actionable feature tweaking algorithm* (Tolomei et al., 2017) which is a fast heuristic method. The reason why we used this method is that it can handle additional constraints, such as plausibility (Parmentier & Vidal, 2021) and causality (Karimi et al., 2021).

Algorithm 2 presents a pseudo-code of the actionable feature tweaking algorithm. Algorithm 2 consists of the following three steps: (i) for each classification tree h_t in the ensemble h, enumerating all the leaves i whose predictive label is the desired class (i.e., $\hat{y}_{t,i} = +1$); (ii) computing an optimal action $\hat{\boldsymbol{a}}$ to the region $r_{t,i}$ corresponding to the leaf i; (iii) finding the minimum cost action $\boldsymbol{a}^*$ among ones altering the prediction results of h into the desired class (i.e., $h(\boldsymbol{x} + \boldsymbol{a}) = +1$). Note that we can easily compute an optimal action $\hat{\boldsymbol{a}} = (\hat{a}_1, \dots, \hat{a}_D)$ to an instance $\boldsymbol{x}$ and a region $r = [l_1, u_1) \times \cdots \times [l_D, u_D)$ as $\hat{a}_d = \mathrm{median}(x_d, l_d, u_d) - x_d$ for $d \in [D]$ (Wang et al., 2020; Carreira-Perpiñán & Hada, 2023). In addition, our implementation is parallelized using Numba[2] and runs faster than the existing public implementations.

B.3. Greedy Algorithm for Set-Cover Relabeling

While the problem (5) is NP-hard because it is an instance of the weighted partial cover problem, there exist several polynomial time algorithms with approximation guarantees (Chvátal, 1979). Algorithm 3 presents a greedy approximation algorithm for the problem. It has a $(2 \cdot H(N) + 3)$-approximation guarantee, where $H(N) = \sum_{n=1}^{N} 1/n = \Theta(\ln N)$ (Kearns, 1990).

C. Detailed Experimental Settings and Additional Results

C.1. Complete Results

Table 5 shows how we determined the hyper-parameters δ and λ of our RACT. Tables 6 to 9 present the details on the value type, minimum value, maximum value, immutability, and constraint of each feature of the datasets that we used in our experiments.

[2] https://numba.pydata.org/

Algorithm 3 Greedy approximation algorithm for the weighted partial cover problem (Kearns, 1990).

```
Input: sets of leaves $\mathcal{I}^+, \mathcal{I}^- \subseteq [I]$ whose predictive labels are $+1$ and $-1$, a set of instances $\mathcal{N}_i$ that can reach the leaf $i$ by some action for each $i \in [I]$, weight values $c_i = N_i^- - N_i^+$ for each $i \in [I]$, and a parameter $\delta \geq 0$.
Output: A set $\mathcal{I}$ of leaves that should be modified as $\hat{y}_i = +1$.
Define $g(\mathcal{I}) \coloneqq \frac{1}{N} \left| \bigcup_{i \in \mathcal{I}^+ \cup \mathcal{I}} \mathcal{N}_i \right|$;
$\mathcal{I} \leftarrow \emptyset$;
/* Greedy Optimization */
while $g(\mathcal{I}) < 1 - \delta$ do
    $\hat{i} \leftarrow \arg\max_{i \in \mathcal{I}^-} \frac{g(\mathcal{I} \cup \{i\}) - g(\mathcal{I})}{c_i}$;
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{\hat{i}\}$;
    $\mathcal{I}^- \leftarrow \mathcal{I}^- \setminus \{\hat{i}\}$;
end while
return $\mathcal{I}$;
```

Table 5. Details of hyper-parameter tuning for our RACT. We varied the values of δ and λ in the predefined ranges (**Range**), and determined each value (**Select**) based on the results.

(a) Classification Tree

	δ		λ	
Dataset	**Range**	**Select**	**Range**	**Select**
FICO	$\{0.2, 0.25, 0.3, 0.35, 0.4\}$	0.3	$\{0.025, 0.05, 0.075, 0.1\}$	0.1
COMPAS	$\{0.2, 0.25, 0.3, 0.35, 0.4\}$	0.3	$\{0.025, 0.05, 0.075, 0.1\}$	0.05
Credit	$\{0.2, 0.25, 0.3, 0.35, 0.4\}$	0.3	$\{0.025, 0.05, 0.075, 0.1\}$	0.1
Bail	$\{0.2, 0.25, 0.3, 0.35, 0.4\}$	0.3	$\{0.025, 0.05, 0.075, 0.1\}$	0.05

(b) Random Forest

	λ	
Dataset	**Range**	**Select**
FICO	$\{0.2, 0.4, 0.6, 0.8, 1.0\}$	1.0
COMPAS	$\{0.02, 0.04, 0.06, 0.08, 0.1\}$	0.06
Credit	$\{0.2, 0.4, 0.6, 0.8, 1.0\}$	0.2
Bail	$\{0.02, 0.04, 0.06, 0.08, 0.1\}$	0.1

C.1.1. Performance Comparison (Section 4.1)

Table 10 presents the average running time of each method in 10-fold cross validation. For both classification trees and random forests, we can see that there is no significant difference in running time between the baselines and RACT on almost all the datasets.

Figure 5 shows the average feature importance of random forests leaned by each method in the performance comparison. We measured the importance score of each feature by averaging the number of times the feature is used among trees, as with the "split" importance score of LightGBM (Ke et al., 2017), and normalized the scores so that the sum equals to 1.

C.1.2. Recourse Quality Analysis (Section 4.2)

Tables 11 to 13 show the average cost, the average plausibility, and the average validity under the causal recourse constraint, respectively. For both classification trees and random forests, we observed that our RACT attained (i) lower costs than the baselines; (ii) comparable plausibility to the baselines; (iii) higher validity than the baselines even in the causal recourse setting.

C.1.3. Trade-Off Analysis (Section 4.3)

Figure 6 presents the sensitivity analyses of the parameter δ for classification trees. For each $\lambda \in \{0.0, 0.025, 0.05, 0.075, 0.1\}$, we measured the average accuracy and recourse ratio by varying $\delta \in \{0.2, 0.25, 0.3, 0.35, 0.4\}$. As with the results of random forests in the main paper, we can see that the recourse ratio (resp. accuracy) was improved by increasing (resp. decreasing) δ on almost all the datasets. We also observed that, for the FICO and Bail datasets, the average accuracy of $\lambda = 0.0$ was significantly worse than the others. This result suggests that applying our set-cover relabeling to a classification tree trained without our recourse risk $\hat{\Omega}_{\varepsilon}$ often results in degrading accuracy, and that our set-cover relabeling can keep accuracy by combining our greedy splitting algorithm with $\hat{\Omega}_{\varepsilon}$.

C.2. Sensitivity Analysis of Model Complexity T

Figure 7 presents the experimental results of our performance comparison with different model complexities for random forests. For each $T \in \{100, 200, 400\}$, we measured the average AUC and recourse ratio of each method, where we used the same value for λ with Section 4.1. Figure 8 presents the sensitivity analyses of the trade-off parameter λ with different model complexities for random forests. As with Section 4.3, we measured the average AUC and recourse ratio of each method by varying λ.

C.3. Sensitivity Analysis of Cost Budget ε

Figures 9 and 10 present the sensitivity analyses of the budget parameter ε for classification trees and random forests, respectively. For each λ, we measured the average predictive performance and recourse ratio by varying $\varepsilon \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$.

C.4. Comparison using Exact AR Algorithm based on Integer Optimization

To investigate whether an action extraction algorithm affects the results, we also employed the exact AR method based on integer optimization (Cui et al., 2015; Kanamori et al., 2020) to extract actions from tree ensembles trained by each method. As with our experiment of Section 4.1, we conducted 10-fold cross-validation and extracted actions for test instances predicted as the undesired class in each fold. As an off-the-shelf solver, we used Gurobi 10.0.3[3], which is one of the state-of-the-art commercial solvers. Due to computational cost, we randomly picked 50 test instances in each fold, and a 60 second time limit was imposed on optimizing an action for each instance.

Tables 14 and 15 presents the average validity and cost of the extracted actions. As with our results using the heuristic algorithm shown in Section 4.1, we can see that our RACT achieved higher validity and lower cost than the baselines regardless of the datasets.

D. Additional Comments on Existing Assets

Numba 0.56.4[4] is publicly available under the BSD-2-Clause license. All the scripts and datasets used in our experiments are available in our GitHub repository at `https://github.com/kelicht/ract`.

All datasets used in Section 4 are publicly available and do not contain any identifiable information or offensive content. As they are accompanied by appropriate citations in the main body, see the corresponding references for more details.

[3] `https://www.gurobi.com/`

[4] `https://numba.pydata.org/`

Table 6. Details of each feature of the FICO dataset (FICO et al., 2018).

Feature	Type	Min	Max	Immutable	Constraint
ExternalRiskEstimate	Integer	0.0	94.0	Yes	Fix
MSinceOldestTradeOpen	Integer	0.0	803.0	Yes	Fix
MSinceMostRecentTradeOpen	Integer	0.0	383.0	Yes	Fix
AverageMInFile	Integer	4.0	383.0	Yes	Fix
NumSatisfactoryTrades	Integer	0.0	79.0	No	Nothing
NumTrades60Ever2DerogPubRec	Integer	0.0	19.0	Yes	Fix
NumTrades90Ever2DerogPubRec	Integer	0.0	19.0	Yes	Fix
PercentTradesNeverDelq	Integer	0.0	100.0	No	Nothing
MSinceMostRecentDelq	Integer	0.0	83.0	No	Nothing
MaxDelq2PublicRecLast12M	Integer	0.0	9.0	No	Nothing
MaxDelqEver	Integer	2.0	8.0	No	Nothing
NumTotalTrades	Integer	0.0	104.0	Yes	Fix
NumTradesOpeninLast12M	Integer	0.0	19.0	Yes	Fix
PercentInstallTrades	Integer	0.0	100.0	No	Nothing
MSinceMostRecentInqexcl7days	Integer	0.0	24.0	No	Nothing
NumInqLast6M	Integer	0.0	66.0	No	Nothing
NumInqLast6Mexcl7days	Integer	0.0	66.0	No	Nothing
NetFractionRevolvingBurden	Integer	0.0	232.0	No	Nothing
NetFractionInstallBurden	Integer	0.0	471.0	No	Nothing
NumRevolvingTradesWBalance	Integer	0.0	32.0	No	Nothing
NumInstallTradesWBalance	Integer	0.0	23.0	No	Nothing
NumBank2NatlTradesWHighUtilization	Integer	0.0	18.0	No	Nothing
PercentTradesWBalance	Integer	0.0	100.0	No	Nothing

Table 7. Details of each feature of the COMPAS dataset (Angwin et al., 2016).

Feature	Type	Min	Max	Immutable	Constraint
age	Integer	18.0	96.0	No	Increasing only
juv_fel_count	Integer	0.0	20.0	No	Nothing
juv_misd_count	Integer	0.0	13.0	No	Nothing
juv_other_count	Integer	0.0	9.0	No	Nothing
priors_count	Integer	0.0	38.0	No	Nothing
race:African-American	Binary	0.0	1.0	Yes	Fix
race:Asian	Binary	0.0	1.0	Yes	Fix
race:Caucasian	Binary	0.0	1.0	Yes	Fix
race:Hispanic	Binary	0.0	1.0	Yes	Fix
race:Native American	Binary	0.0	1.0	Yes	Fix
race:Other	Binary	0.0	1.0	Yes	Fix
c_charge_degree:F	Binary	0.0	1.0	No	Nothing
c_charge_degree:M	Binary	0.0	1.0	No	Nothing
gender	Binary	0.0	1.0	Yes	Fix

Table 8. Details of each feature of the Credit dataset (Yeh & hui Lien, 2009).

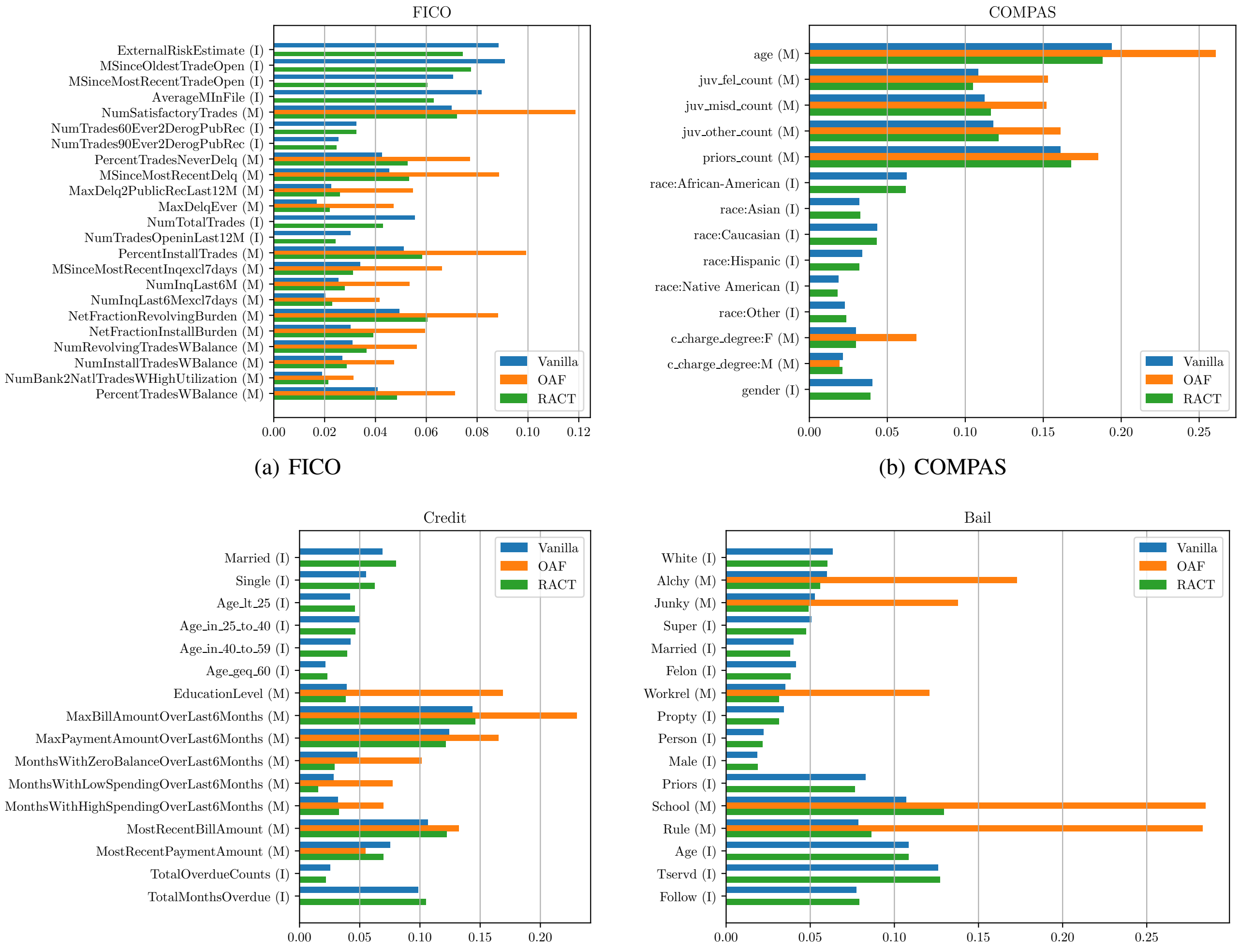
Feature	**Type**	**Min**	**Max**	**Immutable**	**Constraint**
Married	Binary	0.0	1.0	Yes	Fix
Single	Binary	0.0	1.0	Yes	Fix
Age_lt_25	Binary	0.0	1.0	Yes	Fix
Age_in_25_to_40	Binary	0.0	1.0	Yes	Fix
Age_in_40_to_59	Binary	0.0	1.0	Yes	Fix
Age_geq_60	Binary	0.0	1.0	Yes	Fix
EducationLevel	Integer	0.0	3.0	No	Nothing
MaxBillAmountOverLast6Months	Integer	0.0	50810.0	No	Nothing
MaxPaymentAmountOverLast6Months	Integer	0.0	51430.0	No	Nothing
MonthsWithZeroBalanceOverLast6Months	Integer	0.0	6.0	No	Nothing
MonthsWithLowSpendingOverLast6Months	Integer	0.0	6.0	No	Nothing
MonthsWithHighSpendingOverLast6Months	Integer	0.0	6.0	No	Nothing
MostRecentBillAmount	Integer	0.0	29450.0	No	Nothing
MostRecentPaymentAmount	Integer	0.0	26670.0	No	Nothing
TotalOverdueCounts	Integer	0.0	3.0	Yes	Fix
TotalMonthsOverdue	Integer	0.0	36.0	Yes	Fix

Table 9. Details of each feature of the Bail dataset (Schmidt & Witte, 1988).

Feature	**Type**	**Min**	**Max**	**Immutable**	**Constraint**
White	Binary	0.0	1.0	Yes	Fix
Alchy	Binary	0.0	1.0	No	Nothing
Junky	Binary	0.0	1.0	No	Nothing
Super	Binary	0.0	1.0	Yes	Fix
Married	Binary	0.0	1.0	Yes	Fix
Felon	Binary	0.0	1.0	Yes	Fix
Workrel	Binary	0.0	1.0	No	Nothing
Propty	Binary	0.0	1.0	Yes	Fix
Person	Binary	0.0	1.0	Yes	Fix
Male	Binary	0.0	1.0	Yes	Fix
Priors	Integer	0.0	40.0	Yes	Fix
School	Integer	1.0	19.0	No	Nothing
Rule	Integer	0.0	39.0	No	Nothing
Age	Integer	15.0	72.0	Yes	Fix
Tservd	Integer	1.0	287.0	Yes	Fix
Follow	Integer	46.0	57.0	Yes	Fix

Table 10. Experimental results on the average running time [s] of our performance comparison. There is no significant difference between the baselines and RACT on almost all the datasets.

	Classification Tree			**Random Forest**		
Dataset	**Vanilla**	**OAF**	**RACT**	**Vanilla**	**OAF**	**RACT**
FICO	0.483 ± 0.04	0.324 ± 0.02	0.429 ± 0.06	42.09 ± 1.8	35.25 ± 1.52	48.07 ± 2.33
COMPAS	0.098 ± 0.02	0.043 ± 0.01	0.036 ± 0.0	2.31 ± 0.32	1.76 ± 0.14	2.07 ± 0.13
Credit	1.827 ± 0.07	1.37 ± 0.01	1.587 ± 0.1	292.53 ± 21.6	268.41 ± 18.9	277.40 ± 25.8
Bail	0.337 ± 0.02	0.054 ± 0.01	0.2 ± 0.05	24.84 ± 1.93	3.26 ± 0.35	23.04 ± 1.64



(a) FICO

(b) COMPAS

(c) Credit

(d) Bail

Figure 5. Average feature importance of random forests leaned by each method in the performance comparison. We measured the importance score of each feature by averaging the number of times the feature is used among trees, and normalized the scores so that the sum equals to 1. For each feature, "I" (resp. "M") stands for an immutable (resp. mutable) feature.

Table 11. Average cost of extracted actions (lower is better). We used the MPS (Ustun et al., 2019) as a cost function c. Our RACT attained lower costs than the baselines regardless of the datasets.

	Classification Tree			**Random Forest**		
Dataset	**Vanilla**	**OAF**	**RACT**	**Vanilla**	**OAF**	**RACT**
FICO	0.324 ± 0.11	0.174 ± 0.02	$\mathbf{0.11 \pm 0.02}$	0.447 ± 0.05	0.407 ± 0.03	$\mathbf{0.283 \pm 0.01}$
COMPAS	0.26 ± 0.02	0.257 ± 0.02	$\mathbf{0.184 \pm 0.05}$	0.298 ± 0.02	0.28 ± 0.01	$\mathbf{0.232 \pm 0.02}$
Credit	0.265 ± 0.03	N/A	$\mathbf{0.217 \pm 0.05}$	0.293 ± 0.02	N/A	$\mathbf{0.166 \pm 0.04}$
Bail	0.6 ± 0.09	0.47 ± 0.03	$\mathbf{0.221 \pm 0.01}$	0.763 ± 0.03	0.525 ± 0.04	$\mathbf{0.419 \pm 0.05}$

Table 12. Average plausibility of extracted actions (lower is better). Following (Parmentier & Vidal, 2021), we measured the ourlier score estimated by isolation forests. There is no significant difference on the plausibility between the baselines and our RACT.

	Classification Tree			**Random Forest**		
Dataset	**Vanilla**	**OAF**	**RACT**	**Vanilla**	**OAF**	**RACT**
FICO	0.481 ± 0.01	$\mathbf{0.46 \pm 0.0}$	0.476 ± 0.0	0.456 ± 0.0	0.446 ± 0.0	$\mathbf{0.437 \pm 0.0}$
COMPAS	$\mathbf{0.46 \pm 0.01}$	$\mathbf{0.46 \pm 0.01}$	0.477 ± 0.03	$\mathbf{0.44 \pm 0.01}$	0.447 ± 0.01	0.453 ± 0.01
Credit	0.525 ± 0.01	N/A	$\mathbf{0.506 \pm 0.01}$	0.526 ± 0.01	N/A	$\mathbf{0.523 \pm 0.01}$
Bail	0.511 ± 0.01	$\mathbf{0.506 \pm 0.01}$	0.517 ± 0.01	$\mathbf{0.504 \pm 0.01}$	0.507 ± 0.0	0.512 ± 0.01

Table 13. Average validity of extracted actions under the causal recourse constraint proposed by (Karimi et al., 2021) (higher is better). Our RACT attained higher validity than the baselines even in the causal recourse setting.

	Classification Tree			**Random Forest**		
Dataset	**Vanilla**	**OAF**	**RACT**	**Vanilla**	**OAF**	**RACT**
FICO	0.554 ± 0.15	0.799 ± 0.05	$\mathbf{0.944 \pm 0.04}$	0.316 ± 0.04	0.502 ± 0.03	$\mathbf{0.648 \pm 0.03}$
COMPAS	0.602 ± 0.04	0.611 ± 0.04	$\mathbf{0.838 \pm 0.14}$	0.599 ± 0.02	0.642 ± 0.04	$\mathbf{0.721 \pm 0.02}$
Credit	0.531 ± 0.05	N/A	$\mathbf{0.672 \pm 0.1}$	0.669 ± 0.05	N/A	$\mathbf{0.964 \pm 0.03}$
Bail	0.17 ± 0.1	0.37 ± 0.07	$\mathbf{0.726 \pm 0.02}$	0.174 ± 0.04	0.201 ± 0.04	$\mathbf{0.539 \pm 0.03}$

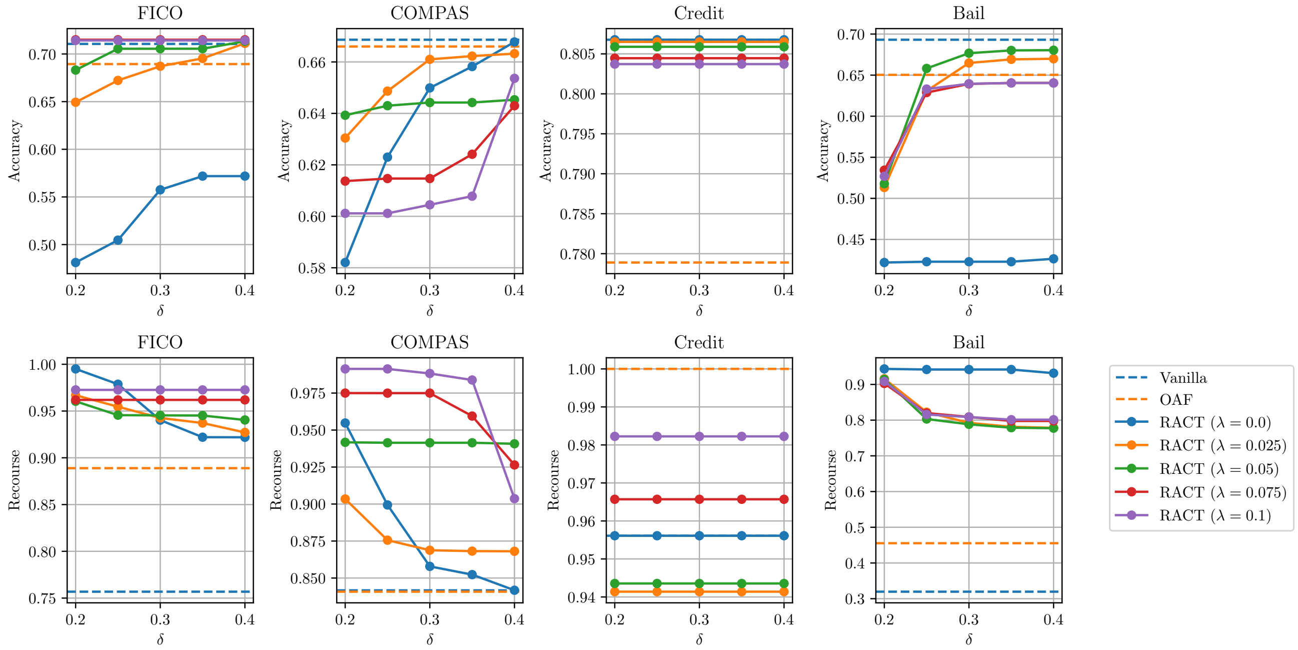
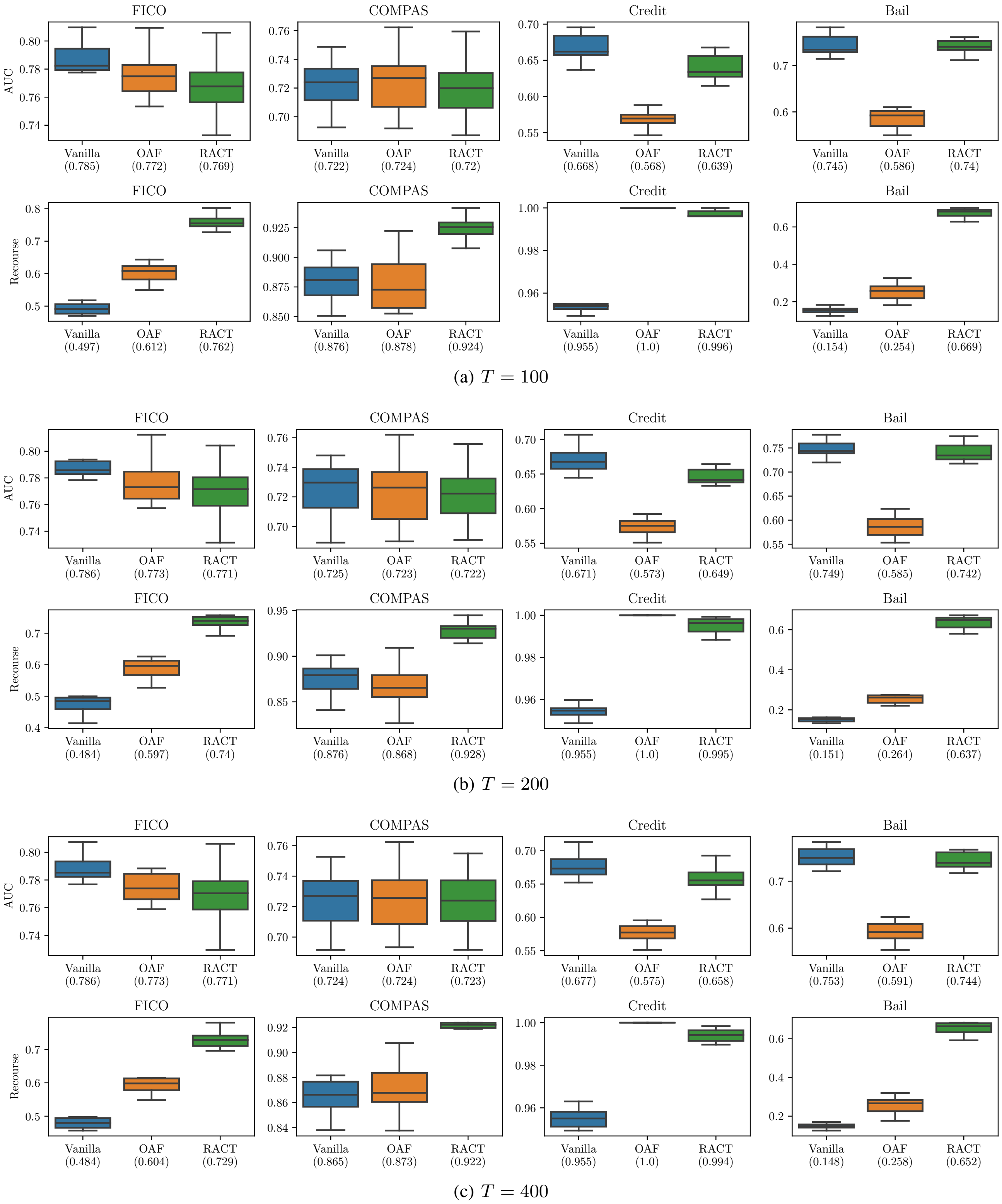


Figure 6. Sensitivity analyses of the parameter δ with respect to the average accuracy and recourse ratio of classification trees. We can see the trend that the recourse ratio (resp. AUC) was improved by increasing (resp. decreasing) δ.

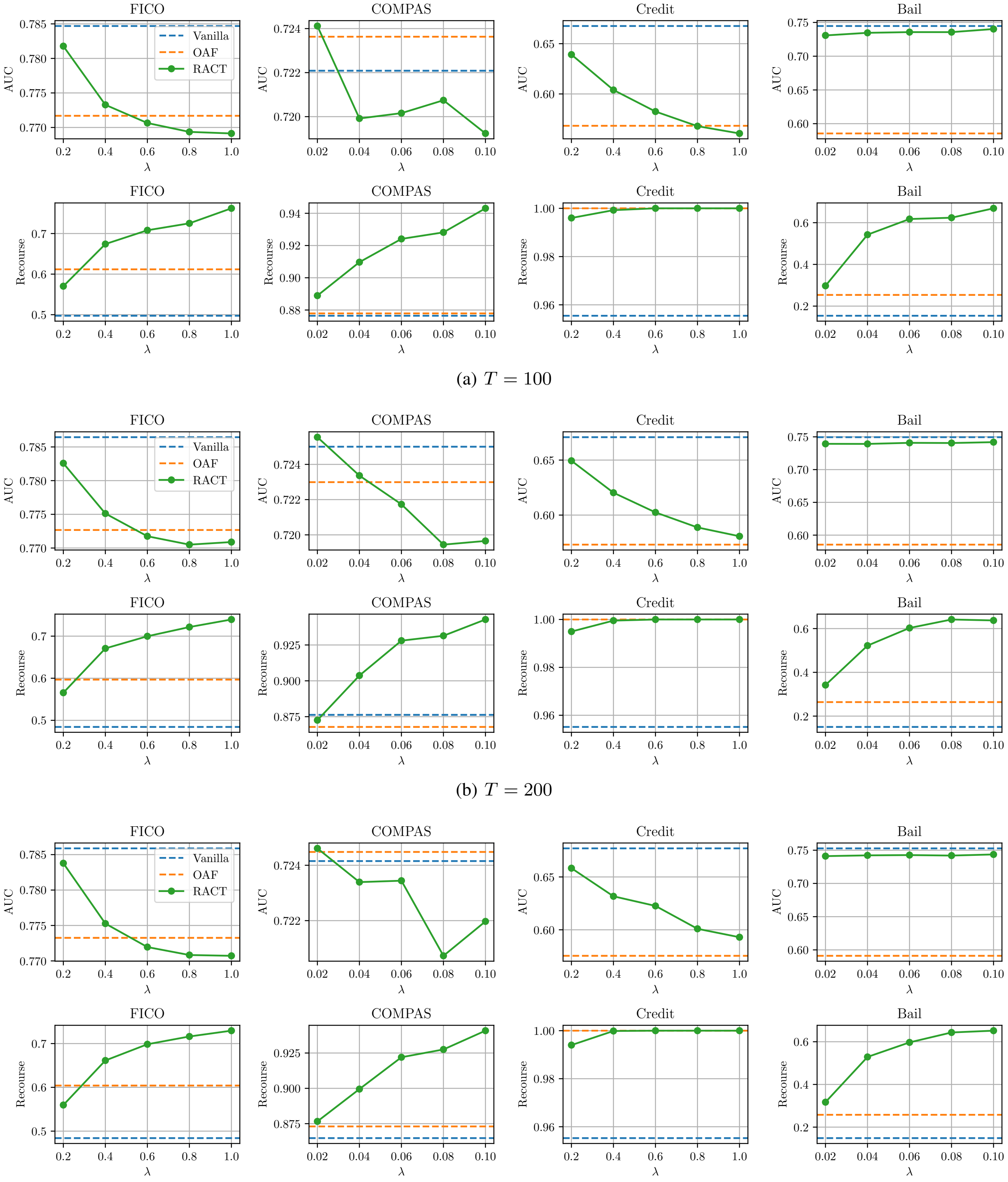


(a) $T = 100$

(b) $T = 200$

(c) $T = 400$

Figure 7. Experimental results of the performance comparison for random forests with different numbers of trees $T \in \{100, 200, 400\}$.

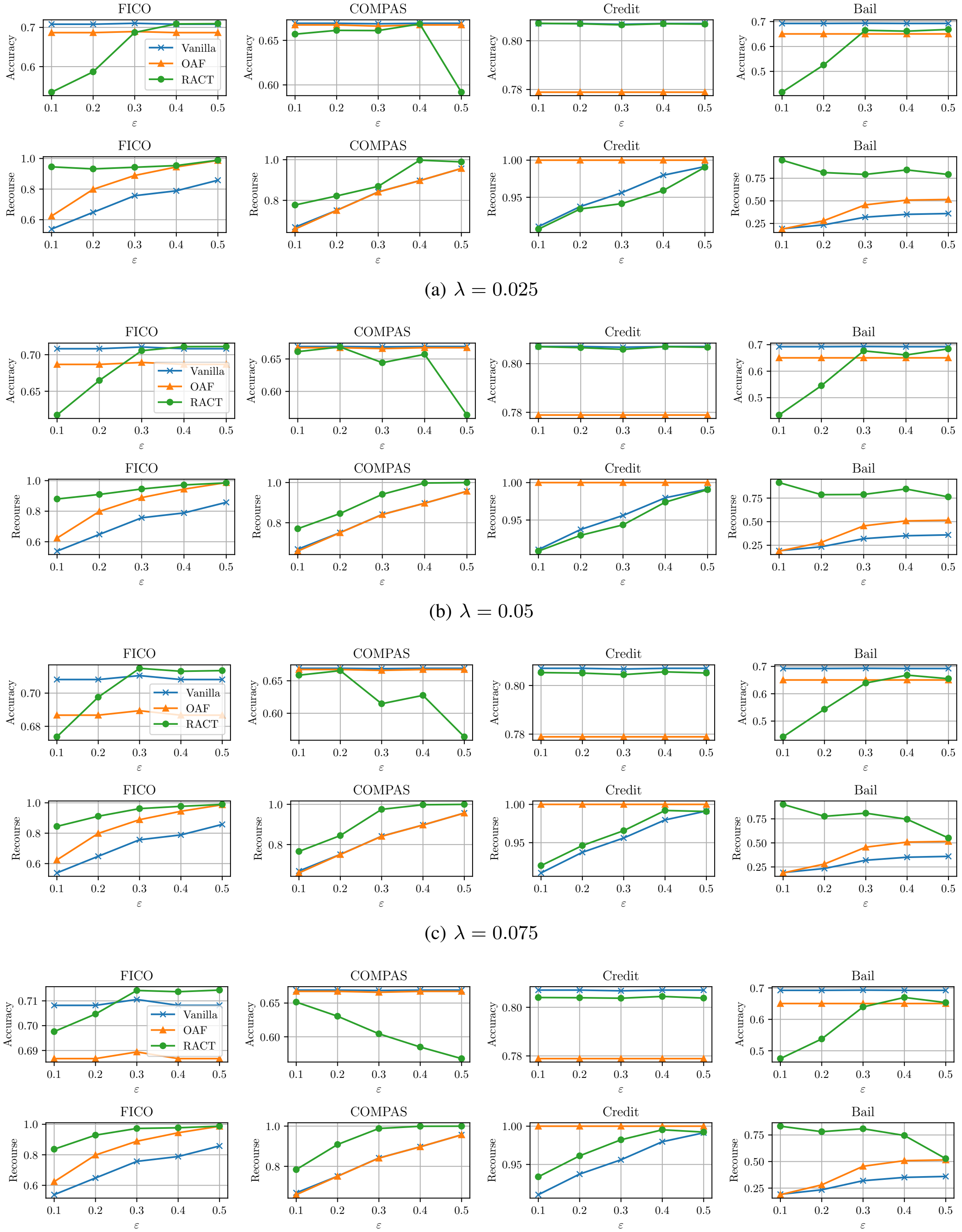


(a) $T = 100$

(b) $T = 200$

(c) $T = 400$

Figure 8. Sensitivity analyses of the trade-off parameter λ with respect to the average accuracy and recourse ratio of random forests for each number of trees $T \in \{100, 200, 400\}$.



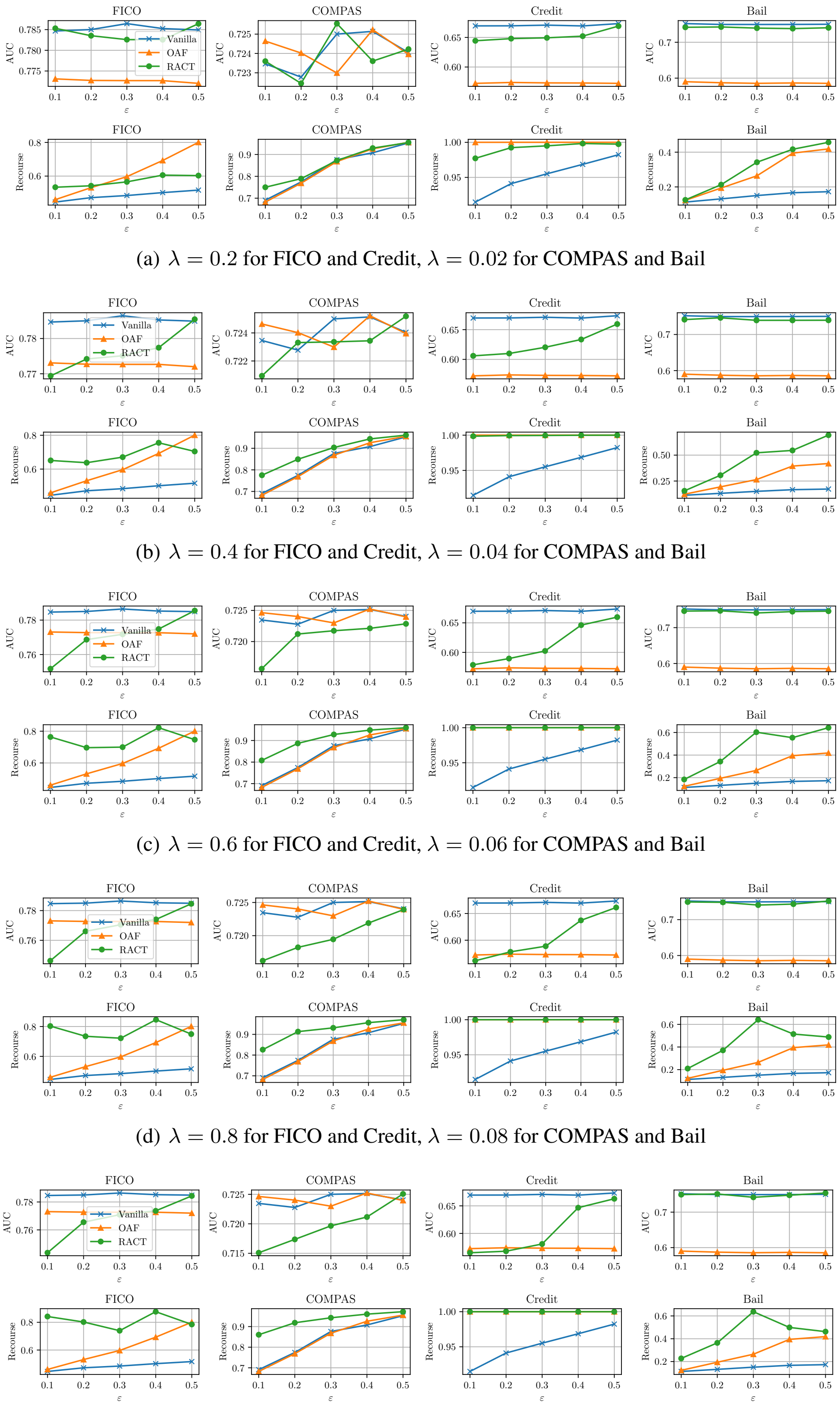
(a) $\lambda = 0.025$

(b) $\lambda = 0.05$

(c) $\lambda = 0.075$

(d) $\lambda = 0.1$

Figure 9. Sensitivity analyses of the budget parameter ε with respect to the average accuracy and recourse ratio of classification trees.



(a) $\lambda = 0.2$ for FICO and Credit, $\lambda = 0.02$ for COMPAS and Bail

(b) $\lambda = 0.4$ for FICO and Credit, $\lambda = 0.04$ for COMPAS and Bail

(c) $\lambda = 0.6$ for FICO and Credit, $\lambda = 0.06$ for COMPAS and Bail

(d) $\lambda = 0.8$ for FICO and Credit, $\lambda = 0.08$ for COMPAS and Bail

(e) $\lambda = 1.0$ for FICO and Credit, $\lambda = 0.1$ for COMPAS and Bail

Figure 10. Sensitivity analyses of the budget parameter ε with respect to the average accuracy and recourse ratio of random forests.

Table 14. Average validity of actions extracted by the MILO-based AR method (higher is better). Our RACT attained higher validity than the baselines regardless of the datasets.

Dataset	Vanilla	OAF	RACT
FICO	0.234 ± 0.08	0.132 ± 0.05	$\mathbf{0.896 \pm 0.05}$
COMPAS	0.748 ± 0.08	0.79 ± 0.06	$\mathbf{0.938 \pm 0.07}$
Credit	0.704 ± 0.08	N/A	$\mathbf{0.976 \pm 0.02}$
Bail	0.076 ± 0.04	0.25 ± 0.08	$\mathbf{0.724 \pm 0.07}$

Table 15. Average cost of actions extracted by the MILO-based AR method (lower is better). We used the MPS (Ustun et al., 2019) as a cost function c. Our RACT attained lower costs than the baselines regardless of the datasets.

Dataset	Vanilla	OAF	RACT
FICO	0.429 ± 0.08	0.553 ± 0.02	$\mathbf{0.217 \pm 0.04}$
COMPAS	0.226 ± 0.03	0.207 ± 0.02	$\mathbf{0.124 \pm 0.03}$
Credit	0.219 ± 0.03	N/A	$\mathbf{0.119 \pm 0.01}$
Bail	0.73 ± 0.08	0.493 ± 0.06	$\mathbf{0.236 \pm 0.03}$