Selected as Spotlight Paper

41st International Conference on Machine Learning

ICML 2024

Learning Decision Trees and Forests with Algorithmic Recourse

Kentaro Kanamori (Artificial Intelligence Laboratory, Fujitsu Limited)

Takuya Takagi (Artificial Intelligence Laboratory, Fujitsu Limited)

Ken Kobayashi (School of Engineering, Tokyo Institute of Technology)

Yuichi Ike (Institute of Mathematics for Industry, Kyushu University)

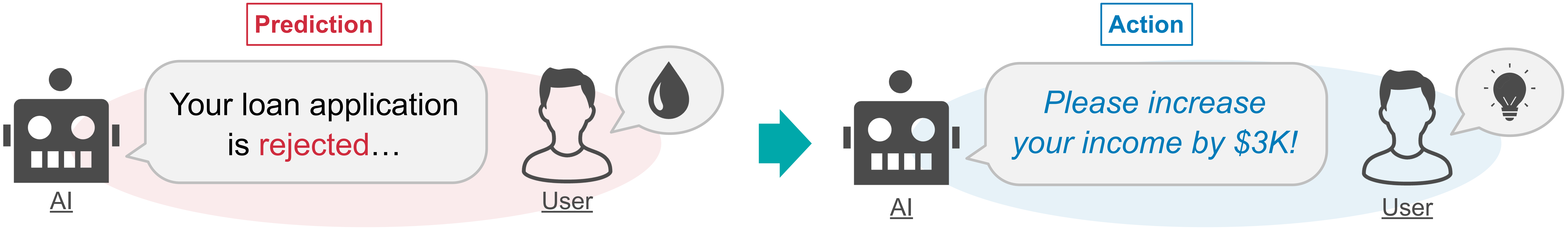
Table of Contents

Background | Algorithmic Recourse (1/2)

Explain a "recourse action" for obtaining the desired prediction result from a model

- *Algorithmic decision-making with machine learning models* has been applied to various tasks in the real world (e.g., loan approvals, judicial decisions, …)
  - Because the model's predictions have a *significant impact on individual human users* [Rudin, 19], decision-makers *need to explain how individuals should act to alter the undesired decisions* [Miller 19]

- **Algorithmic Recourse** [Ustun+ 19]:
  Explaining a *"recourse action"* for *obtaining the desired prediction outcome* from a model

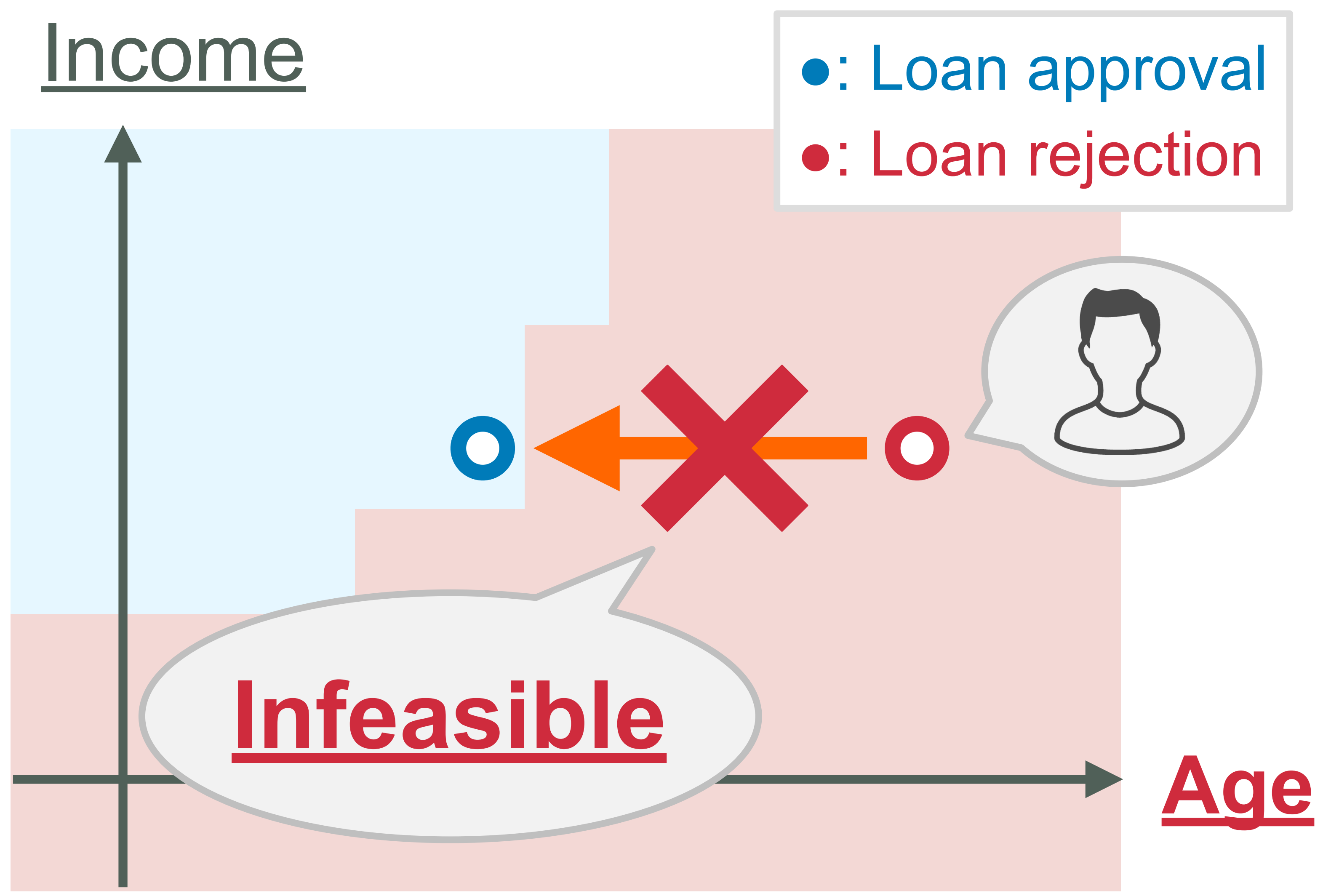


Background | Algorithmic Recourse (2/2)

There is no guarantee that executable actions for users exist for a learned model

- Most of existing studies focus on *how to extract actions from a given learned model*
  - Among actions altering the prediction result into the desired one (*validity*), existing methods often try to find an optimal action that is reasonable for users (*feasibility*) and minimizes the required effort (*cost*)
- In general, however, *such executable actions do not always exist for the given learned model*
  - This is mainly because models are often optimized only for their predictive performance (*without considering recourse actions!*)

▸ We need to ensure the existence of executable actions *at the stage of learning models* [Ross+ 21]



Our Contributions

Learning decision trees that can provide accurate predictions and executable actions

1. Propose a top-down greedy algorithm for learning a *decision tree* by taking into account the *recourse risk* (ratio of instances having no valid and executable action)
   - Its time complexity is equivalent to that of the standard algorithm like *CART*
   - Can be easily applied to the framework of *random forest*
2. Introduce a post-processing task of modifying a learned tree under the constraint on our recourse risk
   - Can be reduced to a variant of the *minimum set-cover problem*
   - Provide a theoretical guarantee by a *PAC-style analysis*
3. Demonstrate the efficacy of our method by experiments
   - Our method could provide executable actions for more instances without degrading accuracy and computational efficiency

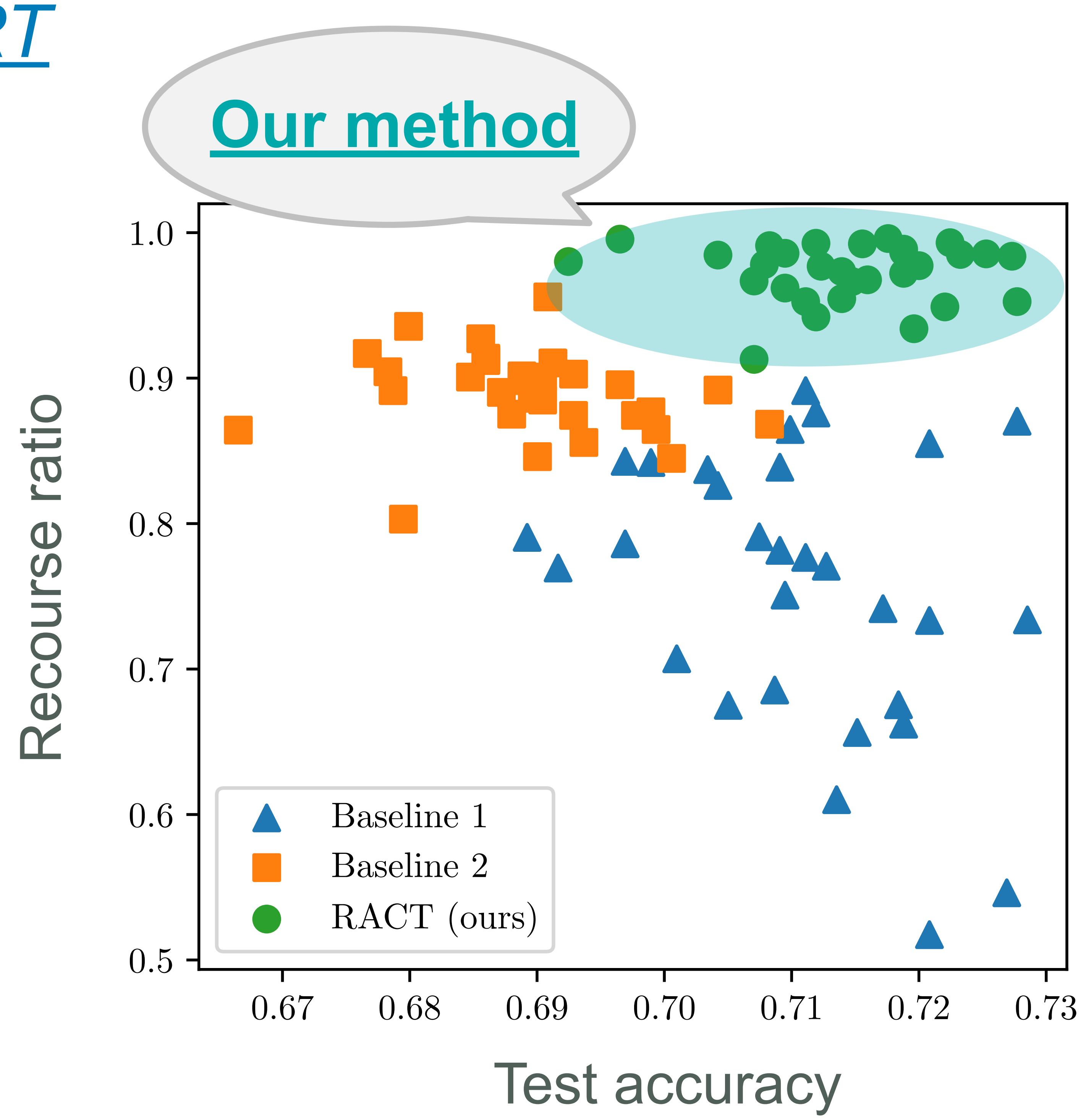
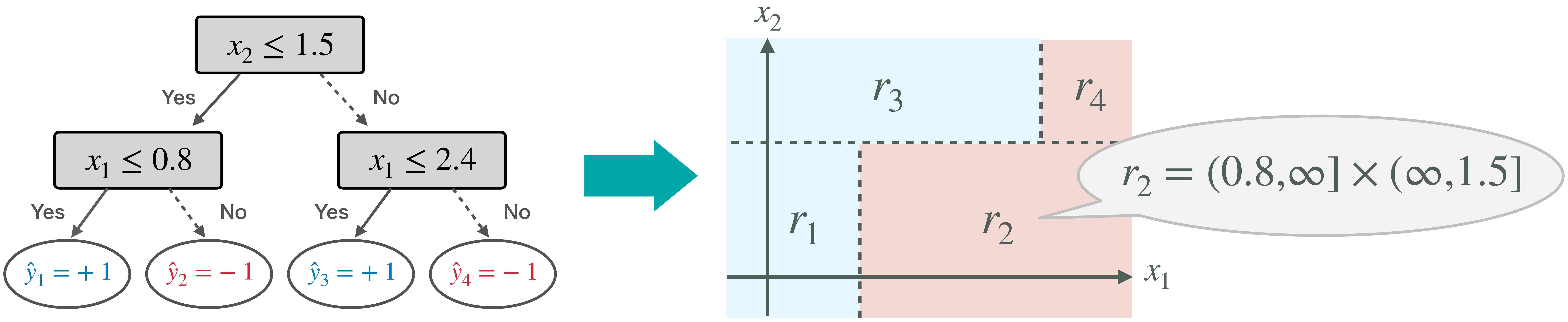


Table of Contents

Preliminaries | Decision Tree

A popular model performing well for tabular datasets as a base learner of ensemble

- A *decision tree* is a model consisting of “if-then-else” rules expressed as a binary tree
- It makes a prediction according to the *predictive label* $\hat{y}$ of the leaf that an input x reaches by traversing the tree depending on the *split conditions* $x_d \leq b$ of each internal node
    - A subspace r_i corresponds to each leaf $i \in [I]$ and $\{r_1, \ldots r_I\}$ gives a partition of the input space $\mathscr{X} \subseteq \mathbb{R}^D$



Preliminaries | Algorithmic Recourse (AR)

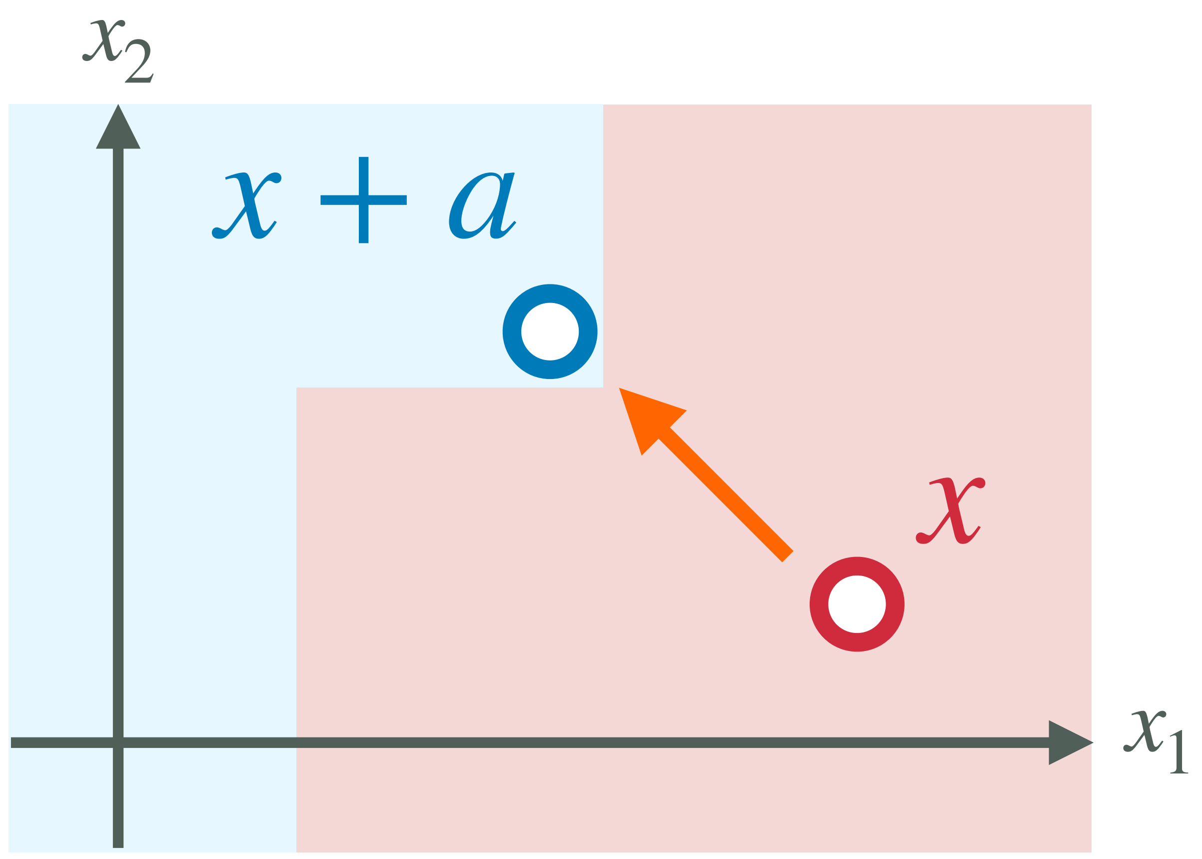
Explain a "recourse action" for obtaining the desired prediction result from a model

Algorithmic Recourse (AR) [Ustun+ 19]

Given an input $x = (x_1, \ldots, x_D) \in \mathcal{X}$ and a classifier $h\colon \mathcal{X} \to \{\pm 1\}$, find an *action* a^* that is an optimal solution for the following problem:

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

where c is a *cost function* that measures the required effort of a and $\mathcal{A}(x) = [l_1, u_1] \times \ldots \times [l_D, u_D]$ is a pre-defined *feasible action set*.



✓ This paper assumes the *ℓ_∞-type cost function* $c(a \mid x) = \max_{d \in [D]} c_d(a_d \mid x_d)$

- Ex 1) Weighted ℓ_∞-norm [Ross+ 21]: $c(a \mid x) = \max_{d \in [D]} w_d \cdot |a_d|$
- Ex 2) *Max percentile shift* [Ustun+ 19]: $c(a \mid x) = \max_{d \in [D]} |Q_d(x_d + a_d) + Q_d(x_d)|$ (Q_d: CDF for x_d)

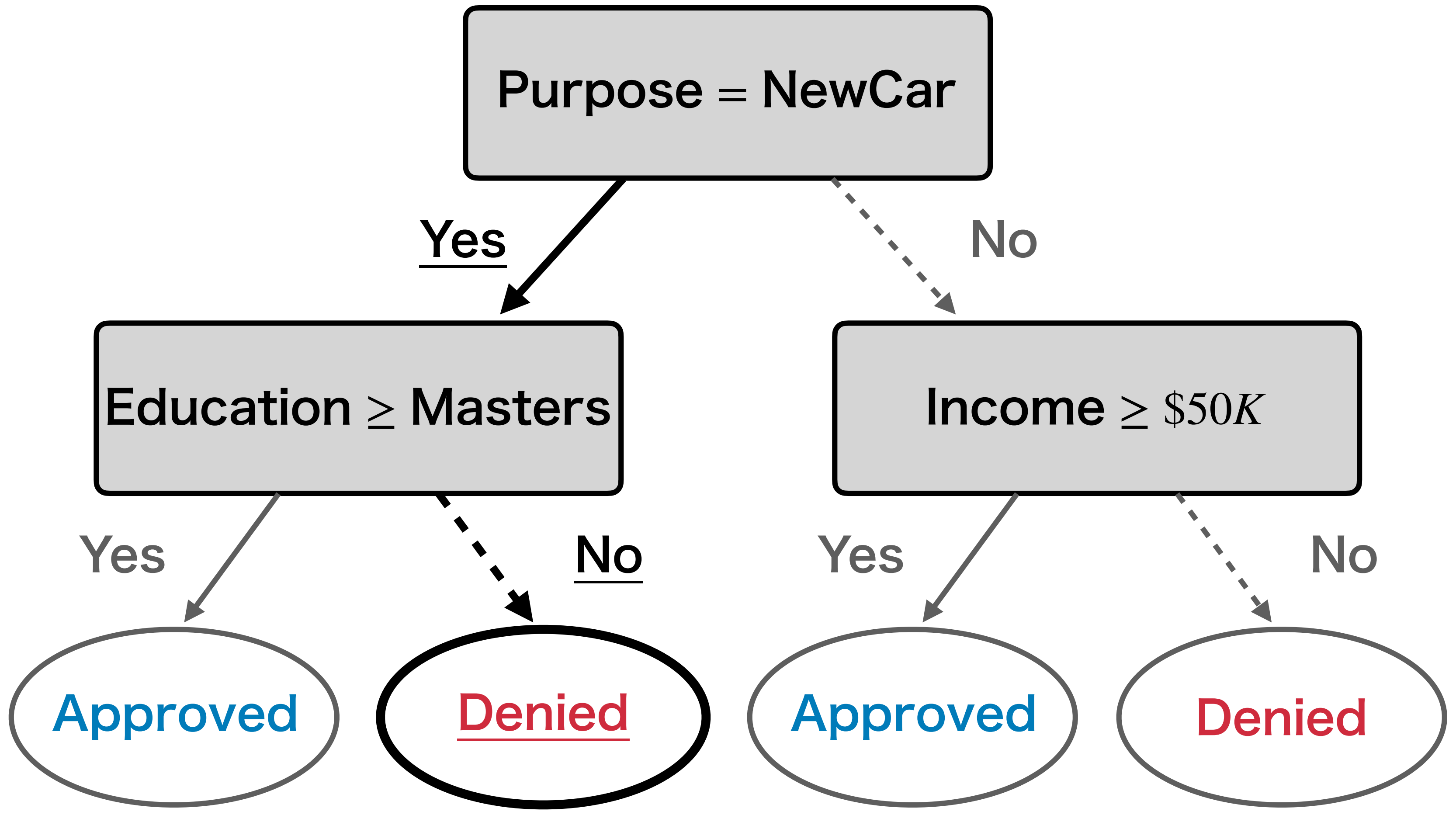
Our Goal | Learning Models with AR

Learn accurate models while ensuring valid and executable actions for instances

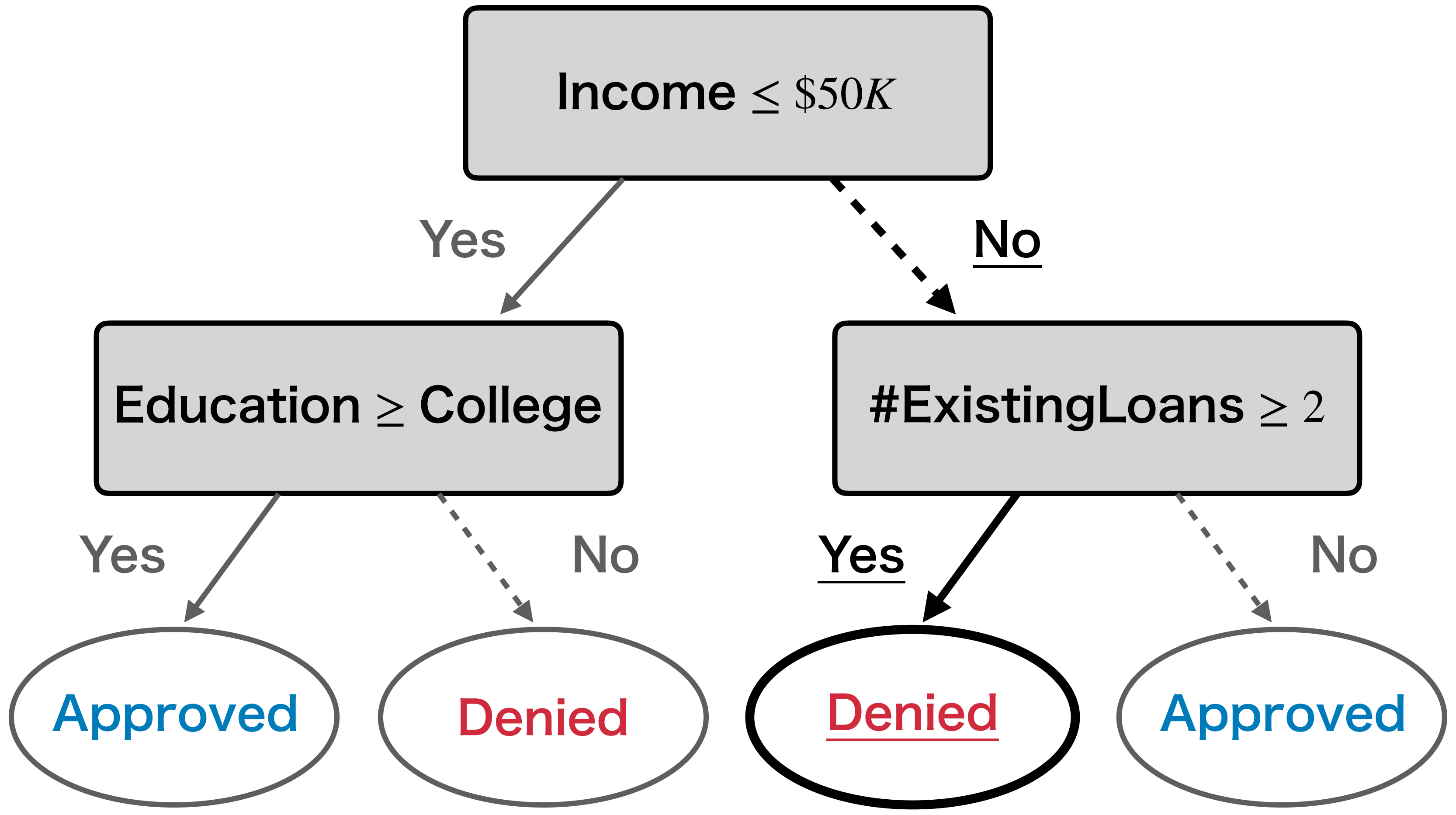
User x

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

To get the loan approved, the user x should …



Improve "Education" or change "Purpose" (*difficult to execute…*)



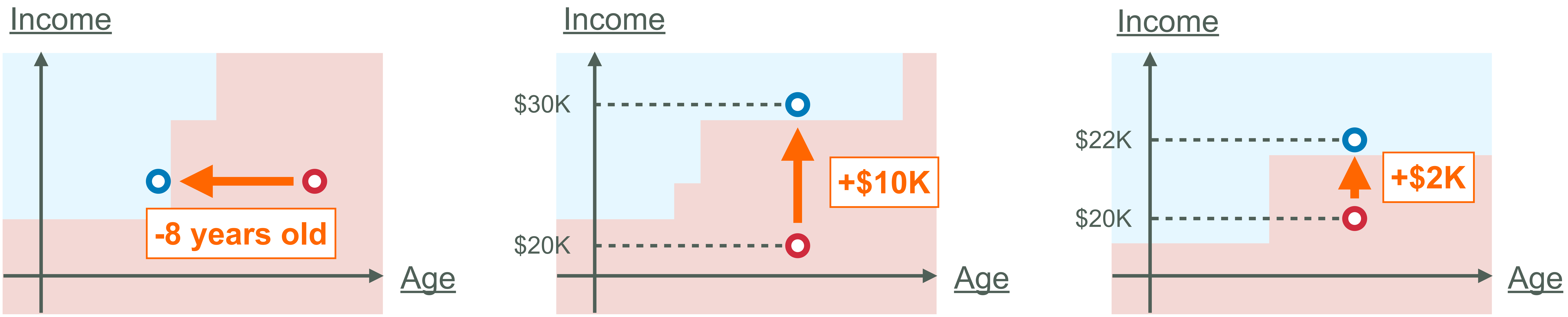
Just reduce "#ExistingLoans" (*relatively easy to execute!*)

Our Goal

Learn an accurate decision tree *while ensuring executable actions for as many instances as possible*

Formulation | Recourse Loss (1/2)

"Ensure executable actions" = There exists at least one feasible and low-cost action



Decrease age by 8 years old (*not feasible…*)	Increase income by $10K (*significant cost…*)	Increase income by $2K (*reasonable cost!*)

- We need to take into account *not only the feasibility but also the cost of provided actions*

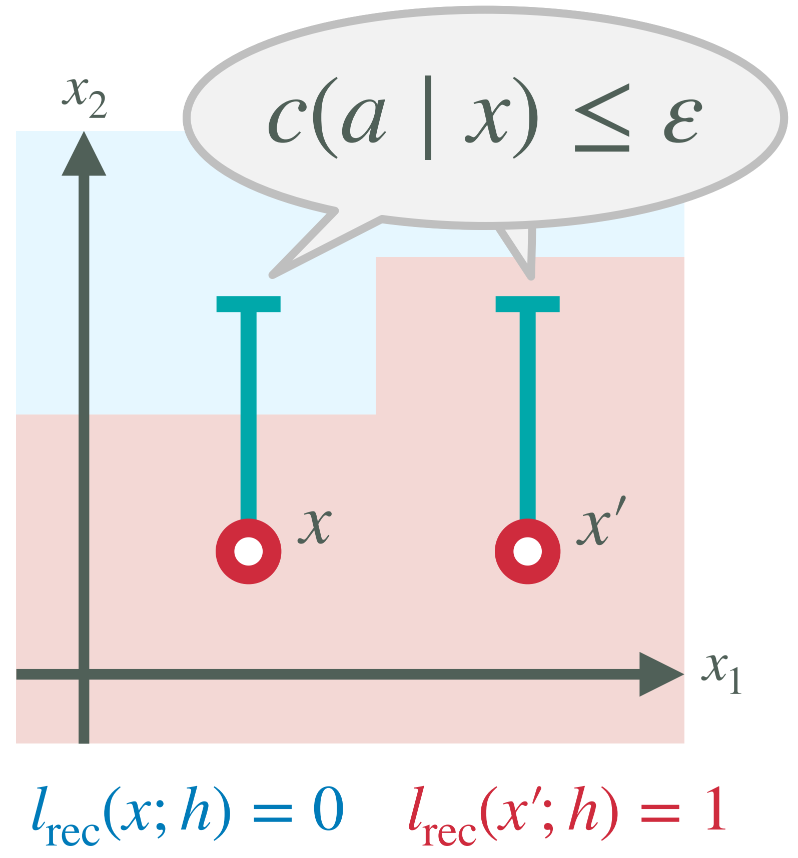
Formulation | Recourse Loss (2/2)

Introduce the recourse risk for evaluating the ratio of instances having actions

Definition (empirical recourse risk)

For a given *cost budget* $\varepsilon > 0$, we denote by $\mathcal{A}_\varepsilon(x) = \{a \in \mathcal{A}(x) \mid c(a \mid x) \leq \varepsilon\}$, and define the *recourse loss* by $l_{\mathrm{rec}}(x; h) := \min_{a \in \mathcal{A}_\varepsilon(x)} l_{01}(+1, h(x+a))$ (l_{01} : 0-1 loss).
Then, for a sample $S = \{(x_n, y_n)\}_{n=1}^N$, we define the *empirical recourse risk* as

$$\hat{\Omega}_\varepsilon(h) := \frac{1}{N} \sum_{n=1}^N l_{\mathrm{rec}}(x_n; h)$$



- Our empirical recourse risk is equivalent to the ratio of input instances x in the sample S that do not have any *action $a \in \mathcal{A}(x)$ such that $h(x+a) = +1$ and $c(a \mid x) \leq \varepsilon$.*
  - $a \in \mathcal{A}(x)$: feasible
  - $h(x+a) = +1$: valid
  - $c(a \mid x) \leq \varepsilon$: low-cost

Formulation | Learning with Recourse Loss

Learn an accurate decision tree under the constraint on the empirical recourse risk

Problem (Recourse-Aware Classification Tree; RACT)

Given a sample $S = \{(x_n, y_n)\}_{n=1}^{N} \subseteq \mathcal{X} \times \{\pm 1\}$ and parameters $\delta, \varepsilon > 0$,
find a decision tree $h^* \colon \mathcal{X} \to \{\pm 1\}$ that is a solution for the following problem:

$$\min_{h \in \mathcal{H}} \hat{R}(h) \quad \text{s.t.} \quad \hat{\Omega}_{\varepsilon}(h) \leq \delta,$$

where $\mathcal{H}$ is a set of decision trees and $\hat{R}(h) = \frac{1}{N} \sum_{n=1}^{N} l_{01}(y_n, h(x_n))$ is the empirical risk.

- Aim to learn a decision tree that minimizes the empirical risk on a training sample S *while ensuring valid and executable actions for at least* $100 \cdot (1 - \delta)$ *% instances in* S

Table of Contents

Algorithm | Outline

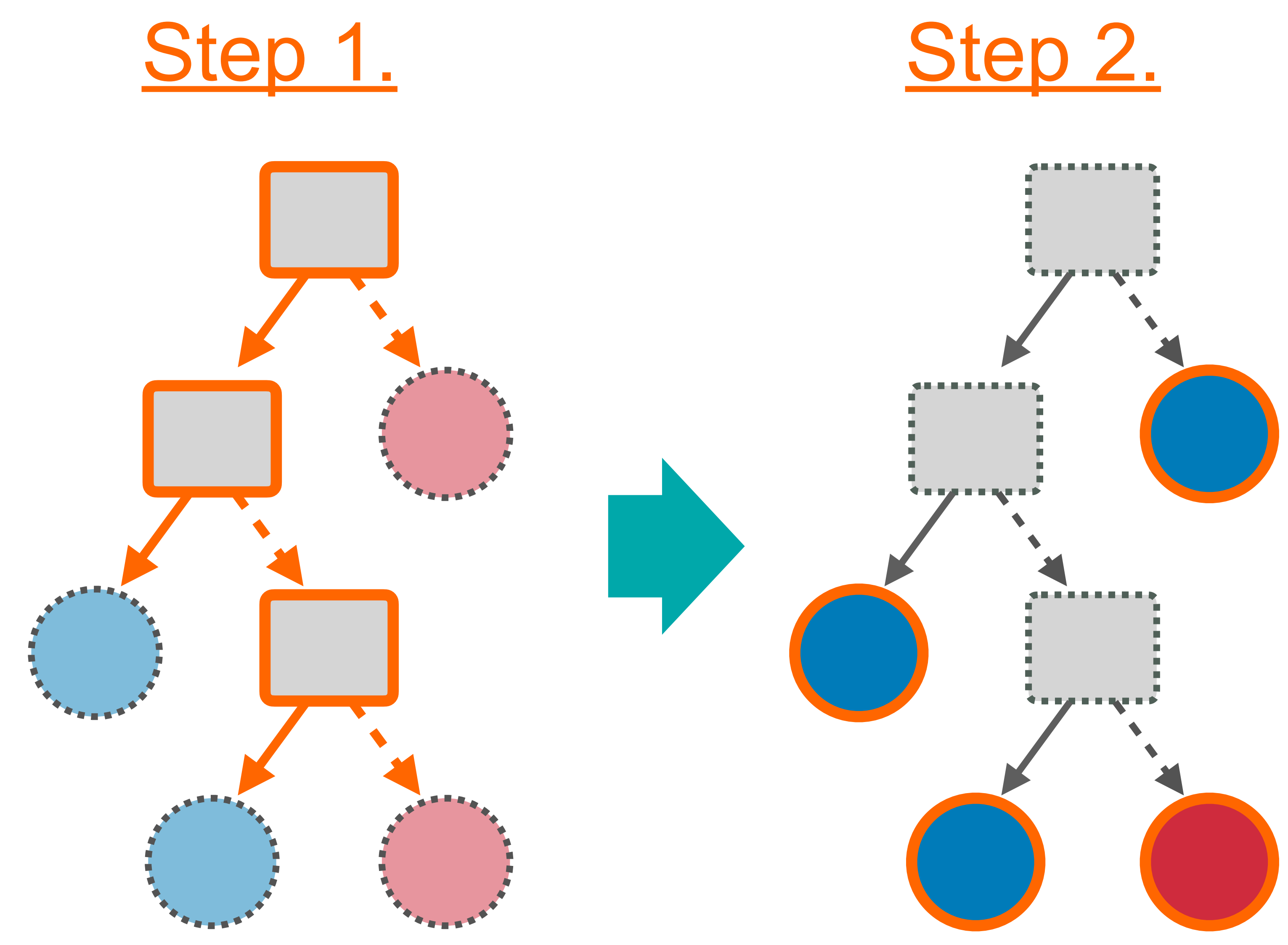
Our algorithm consists of "Top-Down Greedy Splitting" and "Set-Cover Relabeling"

- Even for the case without the constraint on the empirical recourse risk $\hat{\Omega}_\varepsilon(h) \leq \delta$, exactly *learning optimal decision tree is known to be a computationally challenging task*

Our Idea

Extend the standard *top-down greedy approach* like Classification And Regression Tree (CART) [Breiman+ 84]

1. *Learn the split condition $x_d \leq b$ of each internal node* based on both the empirical risk $\hat{R}$ and our empirical recourse risk $\hat{\Omega}_\varepsilon$
2. *Modify the predictive labels $\hat{y}_i$ of selected leaves $\mathcal{I} \subseteq [I]$* in the learned tree h so as to satisfy the constraint $\hat{\Omega}_\varepsilon(h) \leq \delta$



Algorithm | Top-Down Greedy Splitting (1/3)

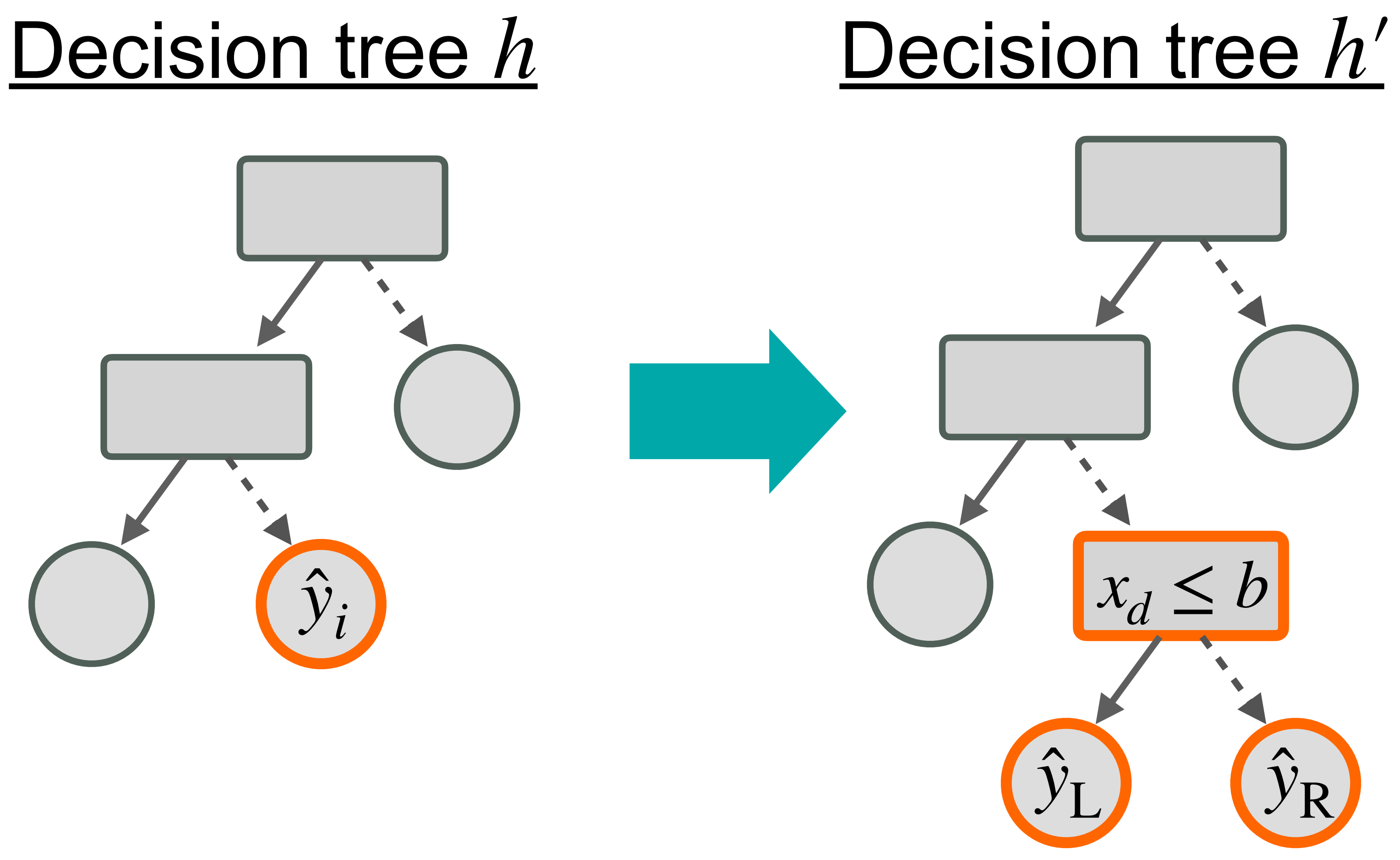
Formulate the task of determining the best split condition with our recourse risk

- For a leaf i of the current tree h, we consider to add a new a split condition $x_d \leq b$
  - We need to determine the best split condition (d, b) for the node and the predictive labels $\hat{y}_\mathrm{L}, \hat{y}_\mathrm{R} \in \{\pm 1\}$ for its left and right child leaves (denote such a new tree by h')
- *Determine the best split condition (d, b) and predictive labels $\hat{y}_\mathrm{L}, \hat{y}_\mathrm{R}$* by solving the following task:

Greedy Splitting Problem (GSP)

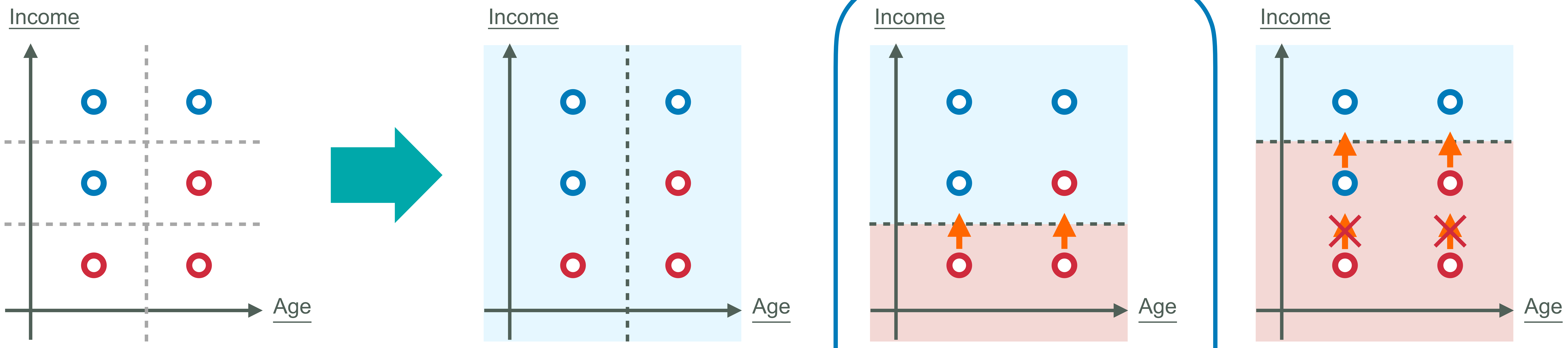
$$\min_{d,b} \min_{\hat{y}_\mathrm{L}, \hat{y}_\mathrm{R}} \Phi_\lambda(d, b, \hat{y}_\mathrm{L}, \hat{y}_\mathrm{R}) := \hat{R}(h') + \lambda \cdot \hat{\Omega}_\varepsilon(h')$$

Relaxation of $\hat{\Omega}_\varepsilon(h') \leq \delta$



Algorithm | Top-Down Greedy Splitting (2/3)

Recursively optimize split conditions with both the empirical risk and recourse risk



$l_{\mathrm{rec}}(x; h) = 0$ if $h(x) = +1$

$\min_{\hat{y}_L, \hat{y}_R} \Phi_1(d_1, b_1, \hat{y}_L, \hat{y}_R) = 3/6 + 0/6 = \underline{3/6}$

$\min_{\hat{y}_L, \hat{y}_R} \Phi_1(d_2, b_2, \hat{y}_L, \hat{y}_R) = 1/6 + 0/6 = \underline{1/6}$

$\min_{\hat{y}_L, \hat{y}_R} \Phi_1(d_3, b_3, \hat{y}_L, \hat{y}_R) = 1/6 + 2/6 = \underline{3/6}$

- Learn a tree structure by *recursively determining the split conditions and predictive labels*

Algorithm | Top-Down Greedy Splitting (3/3)

Time complexity of our algorithm is equivalent to the standard algorithm like CART

- The standard algorithm (e.g., CART) can be regarded as solving the GSP with $\lambda = 0$
  - GSP with $\lambda = 0$ *can be solved in* $\mathcal{O}(D \cdot N)$

Greedy Splitting Problem (GSP)

$$\min_{d,b} \min_{\hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}}} \Phi_\lambda(d, b, \hat{y}_{\mathrm{L}}, \hat{y}_{\mathrm{R}}) := \hat{R}(h') + \lambda \cdot \hat{\Omega}_\varepsilon(h')$$

- In contrast, *how to efficiently solve the GSP with $\lambda > 0$ is not trivial…*
  - We show that *we can compute our empirical recourse risk $\hat{\Omega}_\varepsilon$ in amortized constant time*, as well as $\hat{R}$

Proposition 1.

Our proposed algorithm solves the greedy splitting problem with $\lambda > 0$ in $\mathcal{O}(D \cdot N)$.

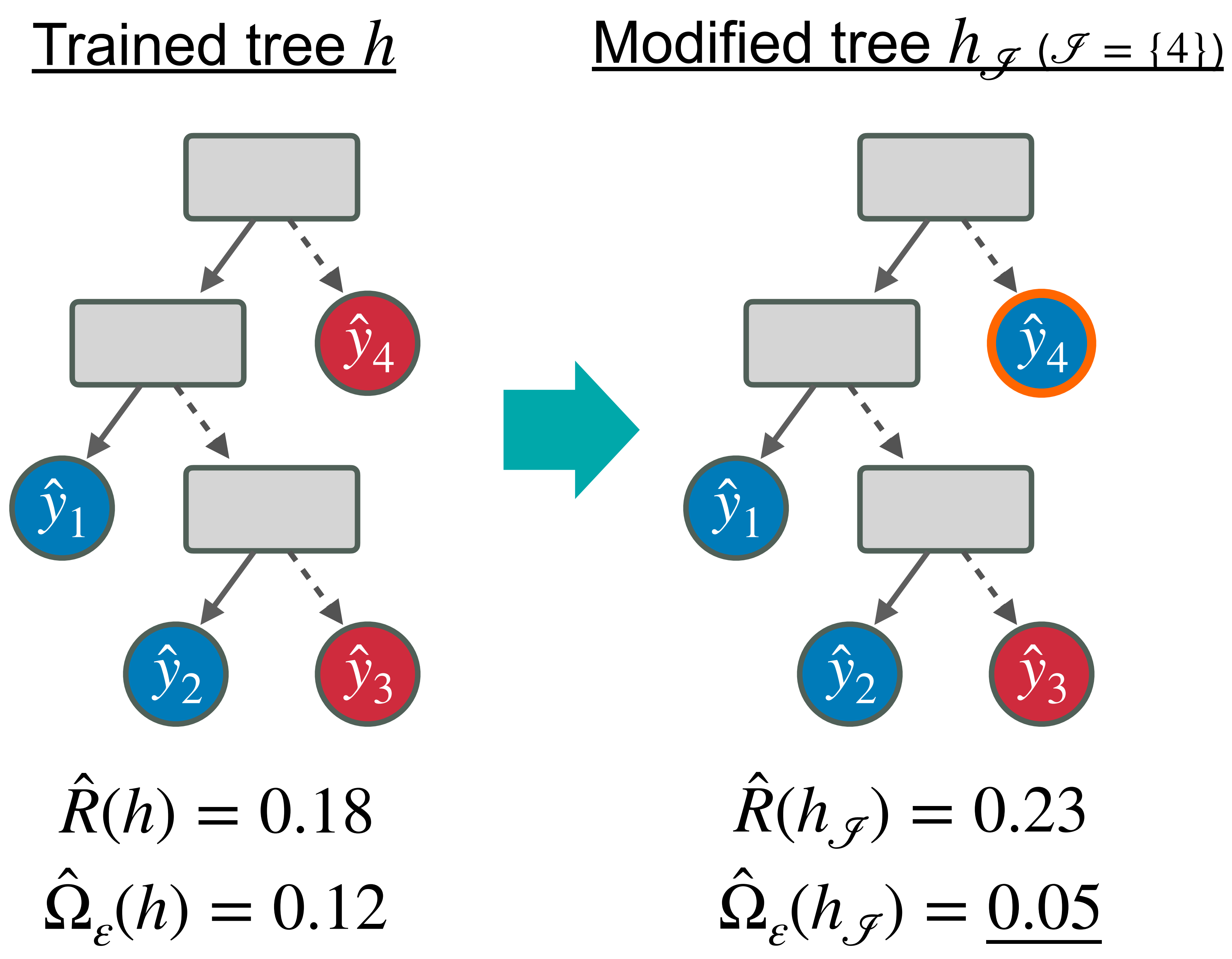
Algorithm | Set-Cover Relabeling (1/2)

Modify the predictive labels so as to satisfy the constraint on the recourse risk

- A decision tree h trained by our algorithm does *not necessarily satisfy the constraint* $\hat{\Omega}_\varepsilon(h) \leq \delta$ since the constraint is relaxed…
- Fix the split condition of each internal node in h and *flip the predictive labels of selected leaves* $\mathcal{I} \subseteq [I]$ by solving the following problem:

Relabeling Problem

$$\min_{\mathcal{I} \subseteq [I]} \hat{R}(h_{\mathcal{I}}) \quad \text{s.t.} \quad \hat{\Omega}_\varepsilon(h_{\mathcal{I}}) \leq \delta$$



  - Select leaves $\mathcal{I}$ so as to *satisfy* $\hat{\Omega}_\varepsilon(h) \leq \delta$ *without increasing the empirical risk* $\hat{R}(h_{\mathcal{I}})$ *as much as possible*

Algorithm | Set-Cover Relabeling (2/2)

Reduced to the set-cover problem and can be efficiently solved approximately

- The empirical recourse risk of a decision tree can be expressed as a *coverage function*
  - "x_n can reach a leaf i" $\iff \exists a \in \mathcal{A}_\varepsilon(x_n) : x_n + a \in r_i$
  - Let the instances that can reach a leaf i be $\mathcal{N}_i$, then we have $\hat{\Omega}_\varepsilon(h) = 1 - \frac{1}{N} \left| \bigcup_{i \in [I]: \hat{y}_i = +1} \mathcal{N}_i \right|$

coverage function

Relabeling Problem

$$\min_{\mathcal{J} \subseteq [I]} \hat{R}(h_\mathcal{J}) \quad \text{s.t.} \quad \hat{\Omega}_\varepsilon(h_\mathcal{J}) \leq \delta$$

Proposition 2.

The relabeling problem is reduced to the *weighted partial cover problem*.

- There exist *polynomial-time algorithms with an approximation guarantee* [Kearns 90]

Appendix | PAC-Analysis of Recourse Loss

Provide a probabilistic guarantee of the recourse loss for unseen test instances

Proposition 3.

Let $\Omega_\varepsilon(h) := \mathbb{P}_x[\exists a \in \mathscr{A}_\varepsilon(x) : h(x+a) = +1]$ be the expected recourse loss. For any $\alpha > 0$, classifier $h \in \mathscr{H}$, and sample S, the following inequality holds with probability at least $1 - \alpha$:

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

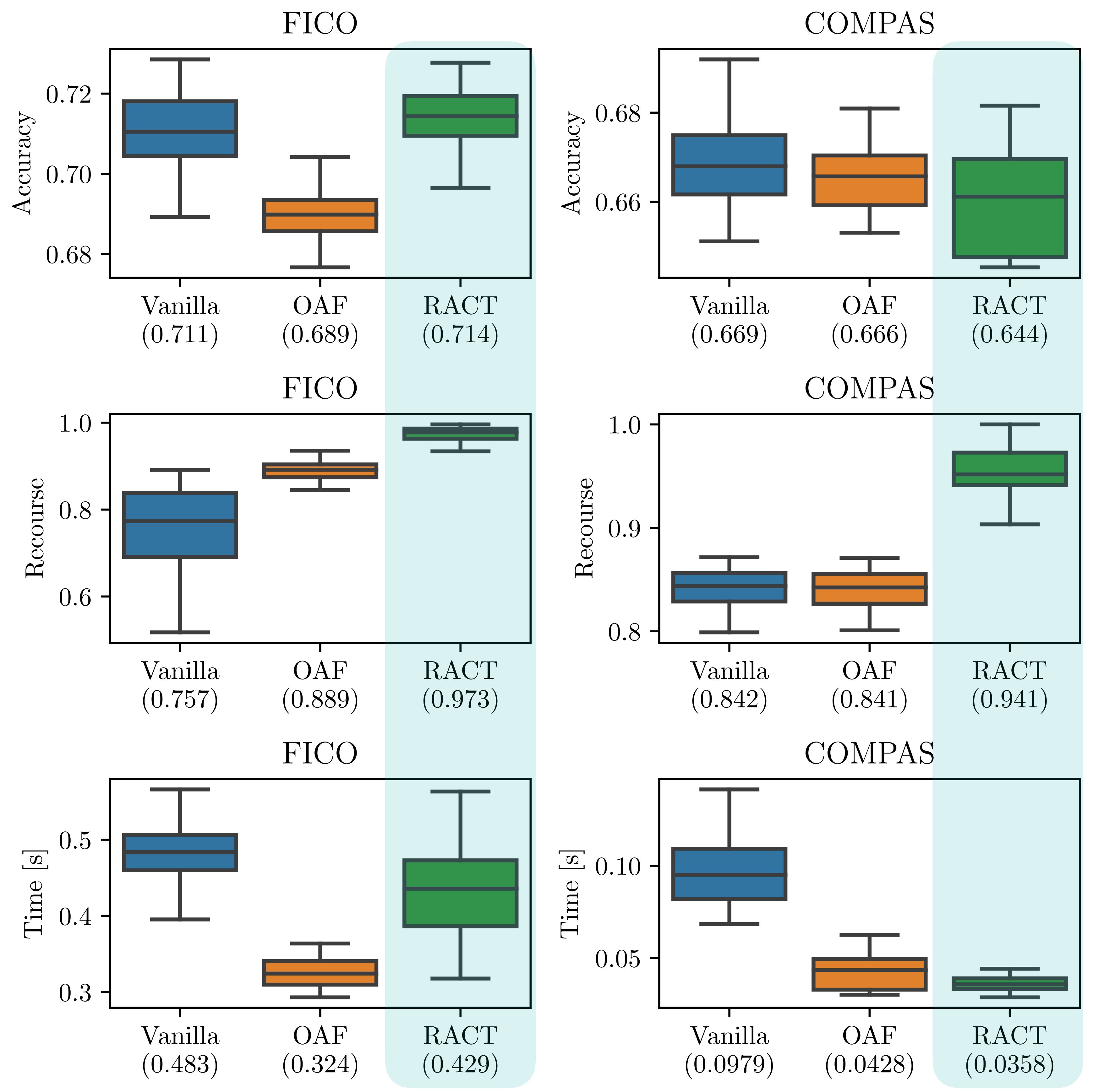
- By replacing δ with $\delta - \sqrt{(I \cdot \ln 2 - \ln \alpha)/(2 \cdot N)}$ in our set-cover relabeling problem, we *can obtain a decision tree h that satisfies $\hat{\Omega}_\varepsilon(h) \leq \delta$* at high probability

Table of Contents

Experiments | Baseline Comparison

Attained higher recourse ratio while maintaining comparable accuracy and efficiency

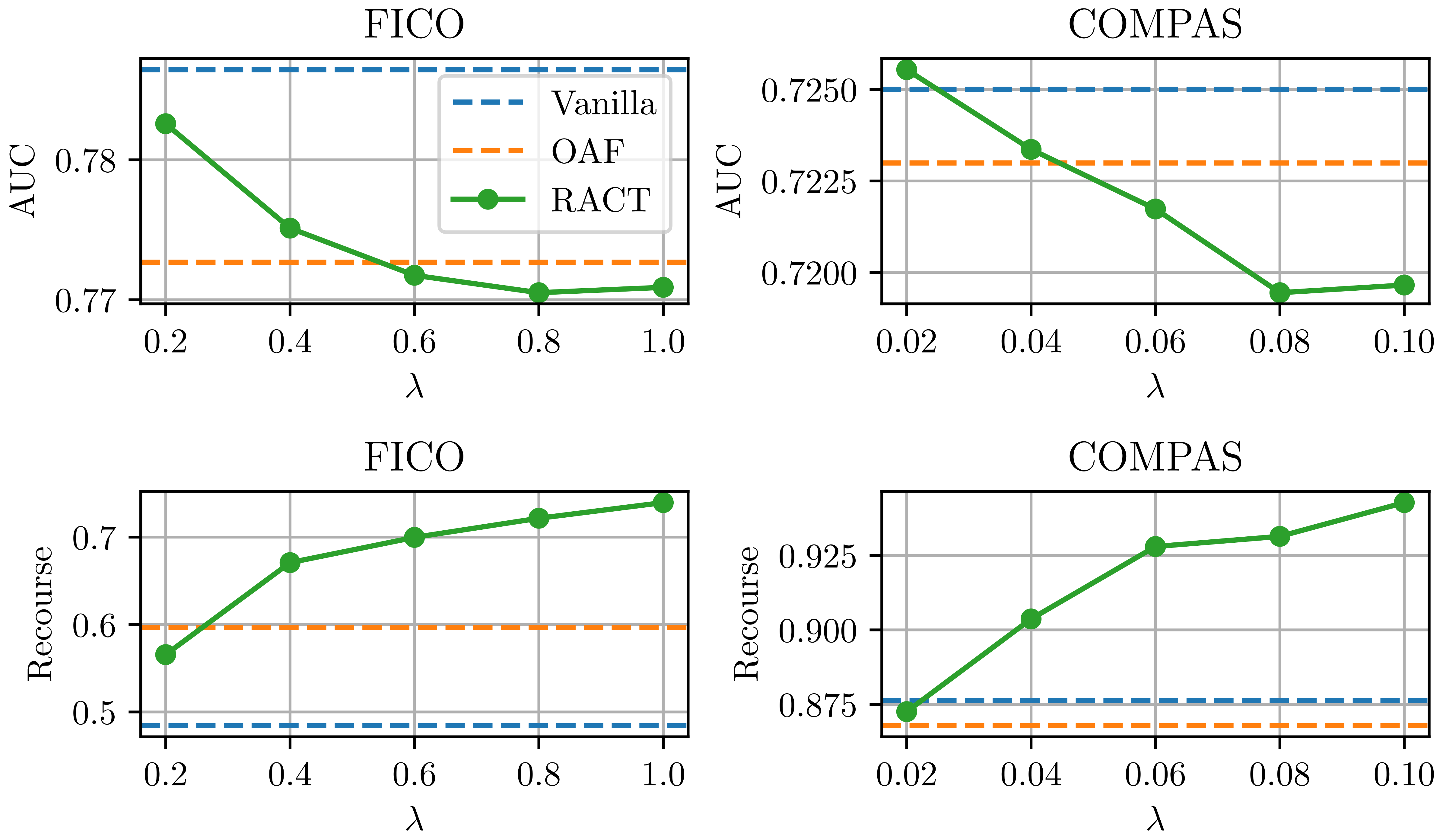
- **Setting** (10-fold CV)
  - *Baselines*:
    - Standard learning algorithm with no constraint on recourse (**Vanilla**)
    - Standard learning algorithm using only actionable features (**OAF**)
  - *Evaluation*:
    - Top: ***Accuracy*** (test accuracy)
    - Middle: ***Recourse*** (ratio of instances having at least one executable valid action)
    - Bottom: ***Time [s]*** (running time)
- **Result**
  - Our proposed method (**RACT**) *achieved higher recourse ratio*
  - There is *no significant difference in accuracy and running time* between the baselines and our method



Experiments | Sensitivity Analysis

Achieved to balance the trade-off between accuracy and recourse guarantee

- **Setting** (10-fold CV)
  - Train **random forest classifiers** by the baselines and our proposed method
  - Evaluate the accuracy (***AUC***) and recourse ratio (***Recourse***) by varying the trade-off parameter λ



- **Result**
  - By increasing the value of λ, the recourse ratio of our method was improved
  - By decreasing the value of λ, the AUC score of our method was improved
  - ‣ There is a chance to *attain better trade-off between the accuracy and recourse ratio by tuning λ*

Conclusion

Learning decision trees that can provide accurate predictions and executable actions

1. Propose a top-down greedy algorithm for learning a *decision tree* by taking into account the *recourse risk* (ratio of instances having no valid and executable action)
   - Its time complexity is equivalent to that of the standard algorithm like *CART*
   - Can be easily applied to the framework of *random forest*
2. Introduce a post-processing task of modifying a learned tree under the constraint on our recourse risk
   - Can be reduced to a variant of the *minimum set-cover problem*
   - Provide a theoretical guarantee by a *PAC-style analysis*
3. Demonstrate the efficacy of our method by experiments
   - Our method could provide executable actions for more instances without degrading accuracy and computational efficiency