Counterfactual Metarules for Local and Global Recourse

Tom Bewley [1] **Salim I. Amoukou** [1] **Saumitra Mishra** [1] **Daniele Magazzeni** [1] **Manuela Veloso** [1]

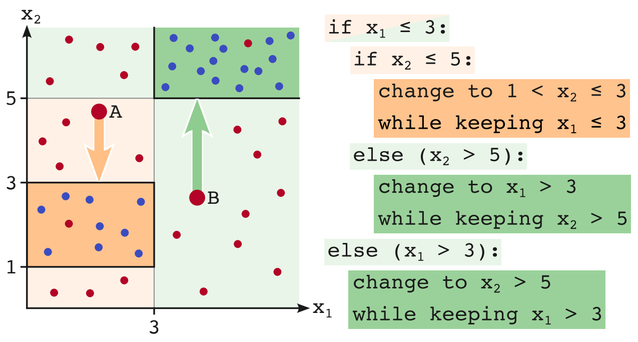
Abstract

We introduce **T-CREx**, a novel model-agnostic method for local and global counterfactual explanation (CE), which summarises recourse options for both individuals and groups in the form of human-readable rules. It leverages tree-based surrogate models to learn the counterfactual rules, alongside *metarules* denoting their regions of optimality, providing both a global analysis of model behaviour and diverse recourse options for users. Experiments indicate that **T-CREx** achieves superior aggregate performance over existing rule-based baselines on a range of CE desiderata, while being orders of magnitude faster to run.

1. Introduction

Counterfactual explanation (CE), which describes how input features could be changed to alter a model's output, is a ubiquitous technique in eXplainable AI (XAI). As AI models make increasingly many decisions that impact human users, CEs provide a foundation for recourse, whereby users act to change an adverse output (e.g. loan rejection) to a desirable one (e.g. acceptance) (Wachter et al., 2017).

While the most basic objective is to find one CE example per instance, several works have generalised this to find either a single group-level CE for a *set* of instances (Carrizosa et al., 2024), or a diverse *set* of CEs for a single instance (Mothilal et al., 2020), optionally summarising each kind of set using human-interpretable rules (Kanamori et al., 2022; Rawal & Lakkaraju, 2020). Both generalisations bring benefits: group-level CEs provide a route to globally analysing a model's behaviour and subgroup fairness properties, while diverse CEs present users with a range of recourse options, which increases the chance of one being practically actionable. In addition, summarising diverse CEs in a compact rule-based form mitigates the information overload that may result from diversity, and provides robustness to the problem of recourse noise (Pawelczyk et al., 2022) by specifying an extended range of values that each feature can take.



```
if x1 ≤ 3:
  if x2 ≤ 5:
    change to 1 < x2 ≤ 3
    while keeping x1 ≤ 3
  else (x2 > 5):
    change to x1 > 3
    while keeping x2 > 5
else (x1 > 3):
  change to x2 > 5
  while keeping x1 > 3
```

Figure 1. Application of **T-CREx** to a binary classifier, producing CEs for the red class ●. Shown are two rules (dark green/orange) for which $\geq 90\%$ of contained points have the alternative blue class ●. The rules are paired with metarules denoting regions of the input space where each rule is an optimal CE (light green/orange), which can be interpreted as follows. For inputs with $x_1 \leq 3$ and $x_2 \leq 5$, the orange rule is optimal because it requires changing only one feature (sparsity). Elsewhere, the green rule is preferred because it contains a greater number of points (feasibility). The two rules and three metarules can be combined to create a global textual summary of all recourse options (shown on the right), which in turn enables the near-instantaneous generation of a CE for any single instance (e.g. A or B) via a simple lookup.

We present **T-CREx**, a method combining both forms of generalisation. It uses tree-based surrogate models to learn counterfactual rules denoting regions of the input space where a model changes its output with high probability, alongside *metarules* denoting regions for which each rule is an optimal CE. The optimal rule for each input (and thus for each metarule) is a joint function of the number of features that must change to satisfy the rule (sparsity) and the rule's population under a given data distribution (feasibility). The method is model-agnostic, requiring only black box access to the target model, and handles both numerical and categorical features and both classification and regression problems. Figure 1 shows the rules and metarules produced by **T-CREx** for a simple binary classification example.

We are aware of one prior method that uses rules to provide diverse CEs for groups of instances comparably to our rule/metarule formulation (Rawal & Lakkaraju, 2020), and several others that find rule-based CEs for single instances (Guidotti et al., 2019; Fernández et al., 2020). Through direct comparison, we show that **T-CREx** reliably matches or outperforms these baselines on various CE desiderata, and

[1]J.P. Morgan AI Research. Correspondence to: Tom Bewley <tom.bewley@jpmorgan.com>.

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

is orders of magnitude faster to run. We also present a selection of qualitative analyses, aided by our ability to represent rules and metarules in a human-readable tree structure.

Our contributions can be summarised as follows:

- A general formulation of the rule/metarule approach to CE, which could in principle be applied to any black box predictive model, and a concrete instantiation for real vector input spaces using hyperrectangles.
- The **T-CREx** algorithm, which uses trees to learn hyperrectangular counterfactual rules and metarules for a specific sparsity- and feasibility-based cost function.
- Experiments demonstrating **T-CREx**'s strong quantitative performance relative to baseline methods, most notably in terms of computational efficiency.
- An exploration of the qualitative structure of learnt counterfactual rules and metarules, demonstrating how they provide both local and global insights.

2. Counterfactual Rules and Metarules

Let $\mathcal{X}$ and $\mathcal{Y}$ be input and output spaces and $f : \mathcal{X} \to \mathcal{Y}$ be a model. Given $x^0 \in \mathcal{X}$ with model output $f(x^0)$ and a set of target outputs $Y^* \subseteq \mathcal{Y} \setminus \{f(x^0)\}$, counterfactual point explanation (CPE) seeks the lowest-cost transformation of x^0 that yields an output in Y^*. Formally, the aim is to find

$$\text{CPE}(x^0 \mid Y^*, P_\mathcal{X}) = \underset{x^i \in \mathcal{X} : f(x^i) \in Y^*}{\text{argmin}} \ \text{cost}(x^0, x^i \mid P_\mathcal{X}), \quad (1)$$

where the cost may depend only on the relationship between x^0 and x^i (e.g. their separation under some distance metric), or also on a given distribution $P_\mathcal{X}$ of *realistic* inputs (e.g. assigning low cost to points in high-density regions, which are seen as feasible recourse targets (Poyiadzi et al., 2020)).

In this work, we seek counterfactuals expressed not as single points, but as *rules* covering regions of the input space where the model f gives an output in Y^* with high probability, thereby summarising a range of possible recourse options. Formally, we define rules $\mathcal{R} \subseteq 2^\mathcal{X}$ as a class of subsets of $\mathcal{X}$. To ensure that the resultant explanations are understandable to humans, we require these rules to satisfy some definition of interpretability; one such definition is adopted below. We say that a rule $R^i \in \mathcal{R}$ is *valid* for a given realistic distribution $P_\mathcal{X}$ and output set Y^* if it contains at least a fraction $\rho \in (0, 1]$ of realistic inputs, and f gives an output in Y^* for at least a fraction $\tau \in (0, 1]$ of those inputs:[1]

$$\text{val}(R^i) = \mathbb{1}[\text{feasibility}(R^i) \geq \rho \land \text{accuracy}(R^i) \geq \tau], \quad (2)$$

where

$$\text{feasibility}(R^i) = \mathbb{P}_{x \sim P_\mathcal{X}}\{x \in R^i\}; \quad (3)$$

$$\text{accuracy}(R^i) = \mathbb{P}_{x \sim P_\mathcal{X} : x \in R^i}\{f(x) \in Y^*\}. \quad (4)$$

[1] In practice, these values are estimated over a finite sample.

Let the set of *maximal*-valid rules be those that are not proper subsets of any other valid rule:

$$\mathcal{R}^{\max}_{Y^*} = \{R^i \in \mathcal{R} \mid \text{val}(R^i) \land (\nexists R^j \in \mathcal{R} : R^i \subset R^j \land \text{val}(R^j))\}. \quad (5)$$

In other words, a maximal-valid rule is one that cannot be made any larger without violating the validity conditions.

Given a set of candidate maximal-valid rules, the aim of counterfactual rule explanation (CRE) is to find the one that minimises some cost function,

$$\text{CRE}(x^0 \mid Y^*, P_\mathcal{X}) = \underset{R^i \in \mathcal{R}^{\max}_{Y^*}}{\text{argmin}} \ \text{cost}(x^0, R^i \mid P_\mathcal{X}). \quad (6)$$

As shorthand, let $R^* = \text{CRE}(x^0 \mid Y^*, P_\mathcal{X})$. Depending on the choice of cost function, there may be many other inputs in $\mathcal{X}$ for which R^* is the optimal (lowest-cost) counterfactual rule among the maximal-valid set $\mathcal{R}^{\max}_{Y^*}$. That is, there exists a *group* of inputs which receive the *same* CE as x^0. We propose to describe this group of commonly-explained inputs using *metarules*, drawn from the same class of interpretable rules $\mathcal{R}$. Retaining the terminology used above, we say that a metarule $M^i \in \mathcal{R}$ is valid if it only contains inputs for which R^* is the optimal rule given Y^* and $P_\mathcal{X}$,

$$\text{val}_{\text{meta}}(M^i) = \mathbb{1}[\forall x \in M^i, \text{CRE}(x \mid Y^*, P_\mathcal{X}) = R^*], \quad (7)$$

and maximal if it is not a subset of another valid metarule,

$$\mathcal{M}^{\max}_{R^*} = \{M^i \in \mathcal{R} \mid \text{val}_{\text{meta}}(M^i) \land (\nexists M^j \in \mathcal{R} : M^i \subset M^j \land \text{val}_{\text{meta}}(M^j))\}. \quad (8)$$

The addition of metarules enhances the explanatory power of counterfactual rules in at least two ways. When presenting R^* as the local explanation of x^0 to a user, it could be beneficial to accompany it with a member of $\mathcal{M}^{\max}_{R^*}$ containing x^0 itself, as this provides background context on *where else* R^* is the optimal rule, and thus the robustness of the explanation to input perturbations (Mishra et al., 2021). Furthermore, the full set of counterfactual rules and metarules for an entire dataset provides a summary of the model's global recourse properties, which may assistive for model development, debugging and auditing (Ley et al., 2023). We explore this opportunity for global analysis in Section 6.

The rule/metarule formulation is extremely general and could be applied to any kind of input space, including images and text. Henceforth, however, we assume real vector input spaces $\mathcal{X} = \mathbb{R}^D$, spanned by features corresponding to intrinsically interpretable quantities, and consider rules and metarules with axis-aligned hyperrectangular geometry. That is, each $R^i \in \mathcal{R}$ is defined by (potentially infinite) lower and upper bounds $l^i_d, u^i_d \in \mathbb{R} \cup \{-\infty, \infty\}$, $u^i_d \geq l^i_d$, along each feature $d \in \{1, \ldots, D\}$. Rules of this form can be seen as preserving feature-level interpretability as they can be expressed as a conjunction of single-feature terms, e.g. "age > 25 and \$$30k <$ income $\leq$ \$$50k$". We note that one-hot encoded categorical features can be handled with a few additional constraints, discussed in Appendix A.1.

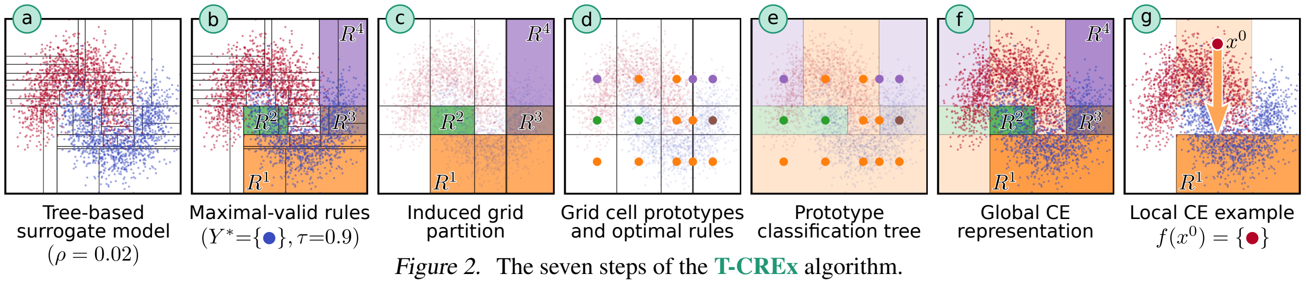

a
Tree-based surrogate model ($\rho=0.02$)
b
R^4
R^2
R^3
R^1
Maximal-valid rules ($Y^*=\{●\}, \tau=0.9$)
c
Induced grid partition
d
Grid cell prototypes and optimal rules
e
Prototype classification tree
f
Global CE representation
g
x^0
Local CE example $f(x^0)=\{●\}$

Figure 2. The seven steps of the **T-CREx** algorithm.

The cost function in Equation 6 can also be defined in many reasonable ways depending on context. Here, we consider

$$\text{cost}(x^0, R^i \mid P_{\mathcal{X}}) = \text{changes}(x^0, R^i) - \text{feasibility}(R^i), \tag{9}$$

where

$$\text{changes}(x^0, R^i) = \textstyle\sum_{d=1}^{D} \mathbb{1}[(x_d^0 \leq l_d^i) \vee (u_d^i < x_d^0)]. \tag{10}$$

This cost function prefers *sparse* rules that require a small number of feature changes, thereby making them simpler to describe and use for recourse. Sparsity objectives have significant precedence in the CE literature (Guidotti, 2022). It also prefers rules with higher feasibility under $P_{\mathcal{X}}$, which also follows prior work in hypothesising that counterfactuals in highly-populated regions can be more realistically used for recourse (Poyiadzi et al., 2020). Importantly, since $\text{changes}(\cdot,\cdot)$ is an integer and $\text{feasibility}(\cdot) \leq 1$, the first term always takes priority in cost calculations.

Returning to the example in Figure 1, the green and orange rules are maximal-valid for $Y^* = \{●\}$ and any $\tau \leq 0.9$ and $\rho \leq 0.2$ (i.e. 10/50 points). The inputs for which the orange rule is an optimal CE under Equation 9 are contained in a single maximal-valid metarule ($x_1 \leq 3$ and $x_2 \leq 5$) while the green rule has two such metarules ($x_1 \leq 3$ and $x_2 > 5$, or $x_1 > 3$). An important property of textual representations of counterfactual rules (shown on the right) is that they differentiate between features that need to change ("*change to...*") and those that need to remain within a specified range ("*while keeping...*"), both of which are required for the CE to inform reliable recourse for an end user. Notice how this leads the same rule to be expressed differently in different cases: satisfying the green rule requires changing feature 1 in its first metarule, and feature 2 in its second metarule.

3. T-CREx (<u>T</u>rees for <u>C</u>ounterfactual <u>R</u>ule <u>Ex</u>planation)

We now describe **T-CREx**, an efficient, model-agnostic algorithm for generating valid (and approximately maximal) hyperrectangular rules and metarules for the cost function in Equation 9. It consists of seven steps, which are visualised for a toy example in Figure 2. For simplicity, we assume here that all features are numerical, and present refinements for handling one-hot encoded categoricals in Appendix A.2.

(a) Given a dataset of realistically-distributed inputs and associated model outputs, $\mathcal{D} = \{(x \sim P_{\mathcal{X}}, y = f(x))\}_{n=1}^{N}$, we first grow a tree-based *surrogate* model, which can be either a random forest or a single decision tree.[2] Each of the L leaves and $L-1$ internal nodes of each tree equates to a hyperrectangle in $\mathcal{X} = \mathbb{R}^D$, whose bounds are determined by splits made at that node's ancestors. The tree growth algorithm optimises for a measure of *purity* in the model outputs of data at every node, which closely aligns with our aim of finding accurate rules. We also use a stopping criterion to ensure that each leaf contains a minimum fraction $\rho \in (0,1]$ of the data in $\mathcal{D}$, thereby enforcing the feasibility constraint in Equation 2. We then take all hyperrectangles corresponding to the nodes of the surrogate tree(s) as a candidate rule set $\mathcal{R}^{\text{surr}}$, from which counterfactuals will be constructed. In Figure 2, the surrogate is single tree which (for $\rho = 0.02$) has $L = 37$ leaves, yielding $|\mathcal{R}^{\text{surr}}| = 2L - 1 = 73$ rules.

(b) From this step onwards, we assume that a set of target outputs $Y^* \subset Y$ and an accuracy threshold $\tau \in (0,1]$ have been specified. We use these parameters (together with the guarantee that all rules in $\mathcal{R}^{\text{surr}}$ satisfy the feasibility constraint) to identify the maximal-valid rules,[3]

$$\mathcal{R}_{Y^*}^{\max} = \{R^i \in \mathcal{R}^{\text{surr}} \mid \text{accuracy}(R^i) \geq \tau \;\wedge$$
$$(\nexists R^j \in \mathcal{R}^{\text{surr}} : R^i \subset R^j \wedge \text{accuracy}(R^j) \geq \tau)\}, \tag{11}$$

where the subset relation for hyperrectangles is defined as

$$R^i \subset R^j \equiv \textstyle\prod_{d=1}^{D} \mathbb{1}[(l_d^j \leq l_d^i) \wedge (u_d^i \leq u_d^j)] \wedge R^i \neq R^j \tag{12}$$

and $\text{accuracy}(\cdot)$ is computed via Equation 4, using $\mathcal{D}$ as a finite-sample approximation of $P_{\mathcal{X}}$. In the Figure 2 example, we obtain $\mathcal{R}_{Y^*}^{\max} = \{R^1, R^2, R^3, R^4\}$ for $Y^*=\{●\}$, τ=0.9.

(c) Next, we use all bounds occurring in the maximal-valid rules to partition the input space into a grid of hyperrectangular cells. That is, we find the set of unique bounds along each feature $d \in \{1, \dots, D\}$ (always including $\pm\infty$),

$$\text{bounds}(\mathcal{R}_{Y^*}^{\max}, d) = \textstyle\bigcup_{R^i \in \mathcal{R}_{Y^*}^{\max}} \{l_d^i, u_d^i\} \cup \{-\infty, \infty\}, \tag{13}$$

then construct cells using all possible combinations of consecutive pairs of bounds along all features. The worst-case size of the resultant grid is $(2|\mathcal{R}_{Y^*}^{\max}| + 1)^D$ cells, which in the Figure 2 example is 27, but the actual number is only 15 here due to bound values being duplicated across rules.

[2] We use a classification forest/tree if f is a classifier, and a regression forest/tree if f is a regressor.

[3] For some (ρ, τ) pairs, it is possible that $\mathcal{R}_{Y^*}^{\max} = \emptyset$. In such cases, the algorithm should be run with different hyperparameters.

To understand why this grid partition is meaningful, consider the following reasoning:

- For any two points in the same grid cell C, the set of features that must change to move from those points to each $R^i \in \mathcal{R}^{\max}_{Y^*}$ must be the same, by virtue of the cell's construction from consecutive bounds in $\mathcal{R}^{\max}_{Y^*}$.
- This means that the cost of each $R^i \in \mathcal{R}^{\max}_{Y^*}$ under Equation 9 must be constant throughout C.
- As a result, there exists an $R^{*C} \in \mathcal{R}^{\max}_{Y^*}$ that has the lowest cost throughout C.
- Therefore, every grid cell is a valid metarule.

d With the above in mind, the next step is to find the optimal rule for each grid cell C. An efficient way to do this is to pick an arbitrary point $x^C \in C$, which we call a *prototype*, and use Equation 6 (with Equation 9 as the cost function) to find $R^{*C} = \text{CRE}(x^C \mid Y^*, P_{\mathcal{X}})$,[4] where $\mathcal{D}$ again serves as a finite-sample approximation of $P_{\mathcal{X}}$. By definition, R^{*C} will also be optimal throughout the rest of the cell. In Figure 2, we show a prototype for each of the 15 cells and use colours to denote their optimal rules.

e At this point, we have identified the maximal-valid rules $\mathcal{R}^{\max}_{Y^*} \subseteq \mathcal{R}^{\text{surr}}$ and valid metarules (i.e. grid cells) denoting their regions of optimality. However, the cells are unlikely to be maximal; Figure 2 includes several instances of adjacent cells that could be merged to give a larger metarule with the same optimal rule. While one could develop a bottom-up algorithm for iteratively merging cells into larger metarules as the previous sentence implies, we instead pursue an efficient top-down approach. That is, we consider the set of cell prototypes, together with their optimal rule assignments, as a kind of labelled dataset, and grow *another tree model* (specifically a CART classifier (Breiman, 2017)) to classify prototypes based on their labels. Crucially, we constrain this tree's growth algorithm to only consider split thresholds in bounds$(\mathcal{R}^{\max}_{Y^*}, d)$ for each $d \in \{1, \ldots, D\}$, and grow the tree to purity. The result is that every leaf of the tree is a union of cells sharing a common optimal rule, and hence is an (approximately maximal) valid metarule. In the Figure 2 example, the 15 cells are aggregated into 7 metarules: three for R^1, two for R^4, and one each for R^2 and R^3.

f Together, the rules and metarules globally characterise the counterfactual structure of f for the given (ρ, Y^*, τ). The visual representation shown in Figure 2 is only possible when $D = 2$, but the textual form exemplified in Figure 1 is more scalable. As we show in Section 6, the model is also amenable to the kind of regional and feature-level interpretability analysis that is normally possible with trees.

g In turn, the global rule/metarule structure trivially enables the explanation of a single instance $x^0 : f(x^0) \notin Y^*$ by identifying its containing metarule (in Figure 2, this leads to R^1 being returned as the CE). Since metarules are arranged in a conventional classification tree structure, highly optimised implementations can be leveraged to rapidly generate CEs for many instances in parallel.

Steps b – e can be repeated for any new (Y^*, τ) combination as required, to extract paired sets of rules and metarules from the same underlying surrogate. Once this has been done once, the rule structures can be reused indefinitely for local explanation. This front-loading of computation makes **T-CREx** highly scalable to large datasets. In Appendix B, we provide a complexity analysis of each stage of the algorithm. The step that is most computationally expensive in practice, d, has complexity $O(\frac{DT^{D+1}}{\rho^{D+1}})$, which increases polynomially with higher numbers of trees in the surrogate model (denoted by T) and lower values of the hyperparameter ρ, and increases exponentially with the input space dimensionality D. Despite this exponential scaling, we find in Section 5.2 that **T-CREx** runtimes are orders-of-magnitude faster than baseline methods.

The lack of restriction on Y^* (it can be any subset of $\mathcal{Y}$) makes **T-CREx** very flexible. If f is a K-class classifier (i.e. $|\mathcal{Y}| = K$), we can handle both *targeted* CE, in which $Y^* = \{y\}$ for some $y \in \mathcal{Y}$, and *untargeted* CE, in which $Y^* = \mathcal{Y} \setminus \{y\}$. For regression models, the CE problem can be formalised in an even greater diversity of ways (Spooner et al., 2021). We consider a simple treatment in Section 5.4.

4. Related Work

As noted in the introduction, our work connects two strands of prior research. The first is concerned with finding group-level CE representations for sets of inputs, which may consist of single counterfactual points, (Warren et al., 2023; Carrizosa et al., 2024), single vectors by which all inputs should be translated (Kanamori et al., 2022), or single translation directions while allowing magnitudes to vary (Ley et al., 2023). Such aggregation provides a basis for high-level auditing of a model's counterfactual fairness properties and may enhance the trust and understanding of non-expert users (Warren et al., 2023). The second strand aims to find diverse sets of CEs for single inputs, typically via gradient-based methods (Mothilal et al., 2020; Rodríguez et al., 2021) or genetic algorithms (Dandl et al., 2020). By providing fuller insight into a model's counterfactual options, it has been argued that diverse CEs improve actionability in the recourse setting (Wachter et al., 2017), although user studies suggest that presenting too many options can create *"increased cognitive load that* [*hinders*] *understanding"* (Tompkins et al., 2022). This drawback motivates providing compressed summaries of diverse CEs in the form of human-readable rules.

Outside of the counterfactual context, rules have been used as factual explanations, denoting the sufficient conditions

[4]We break ties in the cost of two or more rules by taking the first in a fixed (but otherwise arbitrary) ordering of $\mathcal{R}$.

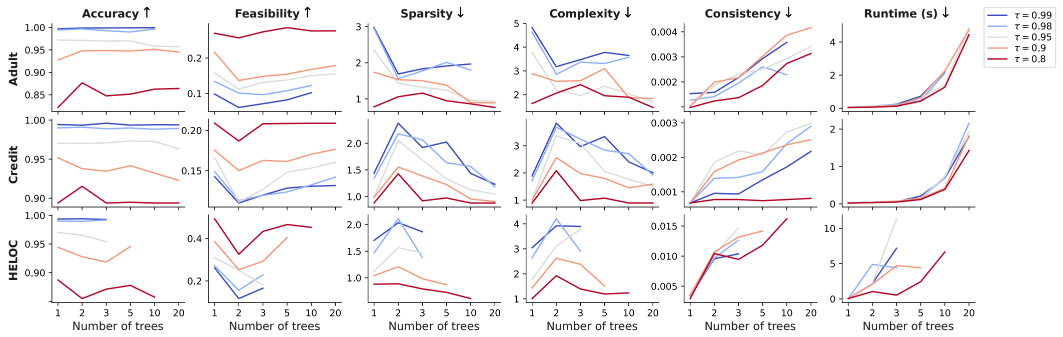


Figure 3. Performance of **T-CREx** as a function of the number of trees and τ (arrows indicate 'better' direction for each desideratum).

for a model to produce a given output (Ribeiro et al., 2018; I. Amoukou & Brunel, 2022). To our knowledge, **LORE** (Guidotti et al., 2019) is the first method to combine such factual rules with counterfactual ones, describing ranges of feature value changes that (starting from a given reference input) would change the model output with high probability. Fernández et al. (2020) propose **RF-OCSE**, which finds similar counterfactual rules for the specific class of random forest classifiers, and demonstrate greatly improved computational speed over **LORE** (although the resultant rules have much lower feasibility). Most related to our proposal is **AReS** (Rawal & Lakkaraju, 2020), which learns counterfactual rules for sets of inputs described by *"subgroup descriptors"* and *"inner rules"* (collectively analogous to our metarules). Rules and subgroups are jointly optimised to yield CEs that are accurate, incur low feature-changing costs, and cover as many instances as possible. One drawback of **AReS**, alongside its extremely long runtimes (Ley et al., 2023), is its use of arbitrary binning for numerical features. By extracting split thresholds from a tree-based surrogate model, our **T-CREx** method learns more adaptive rules while avoiding the need for binning. In addition, **AReS**, **LORE** and **RF-OCSE** alike are only designed to work with binary classification models. **T-CREx** natively handles both multi-class classifiers and regressors.

Several works cited above leverage tree models, including Kanamori et al. (2022) and both **LORE** and **RF-OCSE**. Our use is rather different, and in particular, the growth of a secondary tree for metarule aggregation is entirely novel.

5. Quantitive Experiments

To evaluate **T-CREx**, we use it to generate counterfactual rules for unseen test data $\mathcal{D}_{\text{test}}=\{(x \sim P_{\mathcal{X}}, y=f(x))\}_{m=1}^{M}$ and score the results according to six desiderata. The first of these are **feasibility** and **accuracy** (Equations 3 and 4), both approximated by using $\mathcal{D}_{\text{test}}$ itself. In rare cases that feasibility is 0, we fall back to a default accuracy based on the marginal probability of Y^* across $\mathcal{D}_{\text{test}}$. We also report the **sparsity** of each rule (Equation 10), as well as its **complexity**, which we define as the number of finite terms in its hyperrectangle bounds. We propose this desideratum because it equates to the number of expressions needed to describe a rule in text (fewer is better). Our fifth desideratum is **consistency**, which is the number of unique rules returned across all inputs in $\mathcal{D}_{\text{test}}$ (divided by $|\mathcal{D}_{\text{test}}|$). We suggest that having few unique rules is preferable because this implies robustness of the explanations to input perturbations. It also enables a compact representation of all counterfactuals for $\mathcal{D}_{\text{test}}$, which is a key motivation for group-level CE methods. Finally, we report the **runtime** (on an `r6i.large` AWS instance) of all algorithmic variants and baselines. This is a crucial consideration for the deployment of explanation methods in practice. Throughout this section, f is an XGBoost model (Chen & Guestrin, 2016), trained on $\mathcal{D}$ with `n_estimators=50` and `max_leaves=8`. However, **T-CREx** is model-agnostic, and we report similar results for a neural network model in Appendix F.

5.1. Hyperparameter Study

We begin by characterising the performance of **T-CREx** as a function of key hyperparameters, specifically the number of trees in the surrogate model ($\in \{1, 2, 3, 5, 10, 20\}$) and the accuracy threshold τ ($\in \{0.8, 0.9, 0.95, 0.98, 0.99\}$) while holding the feasibility threshold constant at $\rho = 0.02$. We run this experiment on nine binary classification datasets (details in Appendix C), using 10-fold cross-validation (CV) to split the datasets into train and test components $(\mathcal{D}, \mathcal{D}_{\text{test}})$, and aggregate results across all folds. We focus on the three largest datasets in Figure 3, and report results for all nine datasets (and alternative ρ values) in Appendix D.

The first high-level trend to note is a clear and intuitive trade-off between the accuracy of returned rules on the one hand, and their feasibility, sparsity and complexity on the other. This trade-off is mediated by the accuracy threshold τ. When higher thresholds are specified, this necessitates more specific rules with narrower bounds, which in turn sacrifices performance on the other three desiderata.

The trends with tree count align less with our a priori expectations. One might assume that most desiderata would robustly improve as more trees are used, as this creates a

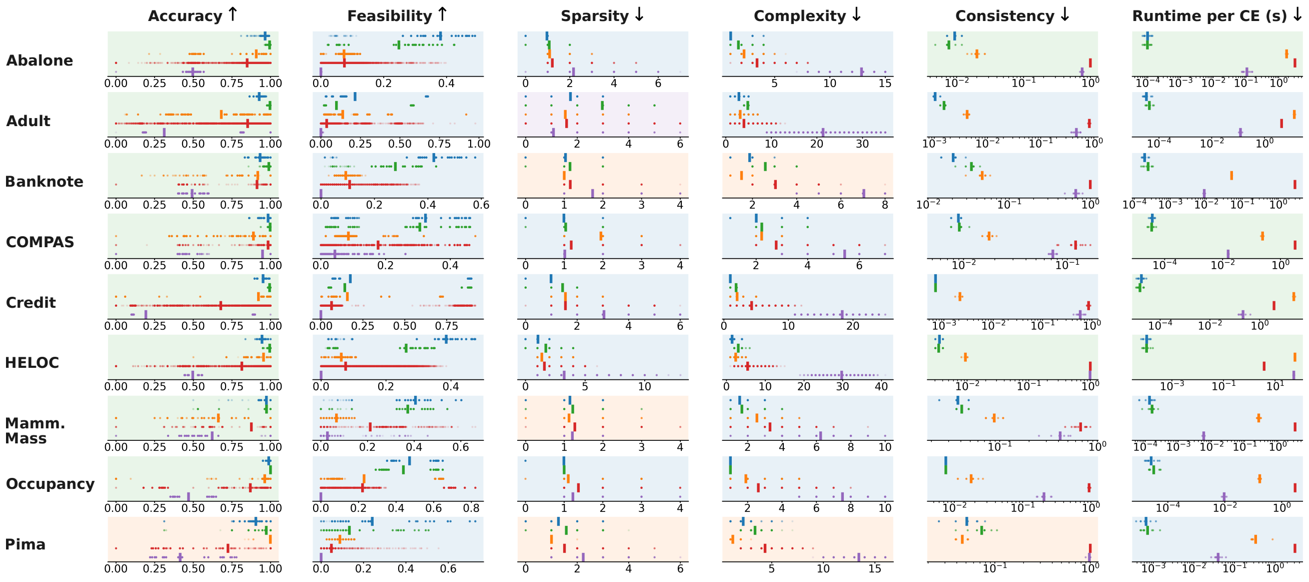


Figure 4. Comparative evaluation of **T-CREx$_{0.9}$**, **T-CREx$_{0.99}$**, **AReS**, **LORE** and **RF-OCSE** on nine binary classification datasets.

larger pool of candidate rules to optimise over. For feasibility, sparsity and complexity, this usually appears to be true for 2-20 trees,[5] but in many cases (most visibly for Credit) using a single tree markedly outperforms using two. We believe the outsized performance of single trees is due to their use of the entire dataset $\mathcal{D}$ for growth rather than bootstrap samples, as in typical random forest implementations.

There is no consistent trend towards more accurate rules as tree count increases, and consistency reliably worsens, as having more valid rules to select from leads to more fragmented results. The runtime required to learn all rules and metarules also increases superlinearly. Together with the ambiguous and inconsistent trends in the other desiderata, this provides a strong practical reason for initially running the **T-CREx** algorithm with a single surrogate tree, and only deviating from this if unsatisfactory results are obtained. For this reason, we use a single tree in all remaining experiments. Specifically, we consider two variants with $\tau = 0.9$ (**T-CREx$_{0.9}$**) and $\tau = 0.99$ (**T-CREx$_{0.99}$**).

5.2. Baseline Comparison

We now present our main results, which compare **T-CREx$_{0.9}$** and **T-CREx$_{0.99}$** to the three relevant baselines from prior work: **AReS**, **LORE** and **RF-OCSE** (see Appendix E for baseline details). We use the same nine datasets and CV setup as above, and evaluate the rules returned by each method using our six key desiderata. Figure 4 shows the distribution of results for each desideratum, dataset and method. Thick vertical lines denote mean values, and subplot background colours indicate the best-performing method.

[5]Some results for higher tree counts are absent here due to a manually-imposed cell limit being reached; see Appendix D.

Moving left-to-right through the columns, we immediately encounter a positive result. **T-CREx$_{0.99}$** (the variant with the more stringent accuracy threshold τ) returns the most accurate rules on eight out of nine datasets, only being narrowly beaten by **AReS** on Pima. As expected, **T-CREx$_{0.9}$** returns somewhat less accurate rules, but this is counterbalanced by their superior feasibility: this variant of our method returns the most feasible rules on *every* dataset. **T-CREx$_{0.99}$** still performs well on feasibility, ranking second on seven datasets, often with a sizeable gap to the third-ranked baseline. This indicates that our method achieves strong compromises on the accuracy-feasibility trade-off. **RF-OCSE**'s poor feasibility results are notable; it frequently returns rules so specific that they contain zero instances from the test set.

The methods perform more similarly in terms of sparsity. In most cases, they return rules requiring just one or two features to be changed. Overall, we feel comfortable in declaring **T-CREx$_{0.9}$** a narrow winner on this desideratum, as it ranks highest on five of nine datasets. This is true to a greater extent for complexity, where **T-CREx$_{0.9}$** ranks best on seven datasets, and never outside the top two. The only consistently strong baseline on both sparsity and complexity is **AReS**. **T-CREx$_{0.99}$** is fairly competitive, outperforming at least two baselines in most cases, but is always beaten by **T-CREx$_{0.9}$**. This reinforces our earlier observation that more accurate rules come at the cost of other desiderata.

On consistency, which measures the fraction of unique rules as an indicator of robustness and explanatory simplicity, **T-CREx** outperforms all baselines for eight of nine datasets. Consistencies of 10^{-1} to 10^{-3} indicate the same rule is returned for tens or hundreds of test instances. Once again, **AReS** is the only close competitor, which is intuitive as it also learns aggregated group-level explanations. **LORE** and

RF-OCSE perform no such aggregation, so any repetition of rules (consistency $< 10^0 = 1$) is coincidental.

The disparity of runtimes is the most stark, varying by 3-4 orders of magnitude between both T-CREx variants (which are always fastest) and the slowest baseline. To illustrate the strength of this result: the total runtime for *all* test inputs in *all* CV folds of *all* datasets is $4.16s$ and $4.52s$ for T-CREx$_{0.9}$ and T-CREx$_{0.99}$ respectively, which is $13\times$ less than the mean runtime *per input* of AReS on HELOC. T-CREx is at least an order of magnitude faster than RF-OCSE on every dataset, which is notable because this baseline's speed is hailed as a key advantage by its authors.

5.3. Counterfactual Distance Evaluation

An apparent drawback of T-CREx is that it does not optimise for rules that require small magnitudes of feature changes to the original input, as measured by metrics such as the Manhattan or Euclidean distance to the closest point satisfying the rule. This desideratum, which is distinct from sparsity, is very common in the literature (Karimi et al., 2022), and is included to some extent in all three baselines. We can justify our omission: since most distance metrics vary continuously, their inclusion would prevent us from finding non-infinitesimal hyperrectangular metarules for which a single counterfactual rule is guaranteed to be optimal. However, given its ubiquity, it is important to know how our method compares to baselines on this desideratum.

The results, shown in Figure 5, are encouraging. For all nine datasets, we report the distribution of distances between each test input and the closest point in the rule returned by each method, where distance is measured as the *total percentile shift* (Pawelczyk et al., 2020). Despite T-CREx making no explicit attempt to minimise counterfactual distances, it performs comparably to the baselines, with both T-CREx$_{0.9}$ and T-CREx$_{0.99}$ outperforming AReS and LORE on seven of the nine datasets. In most cases, RF-OCSE is strongest on this metric, but T-CREx$_{0.9}$ actually does best on three datasets. The fact that T-CREx$_{0.9}$ always yields lower distances than T-CREx$_{0.99}$ suggests a reason for these positive results: by optimising for feasibility (which T-CREx$_{0.9}$ does to a greater extent than T-CREx$_{0.99}$), we indirectly incentivise rules that occupy as much volume in the input space as possible, which in turn reduces the expected distance to any other point in the space.

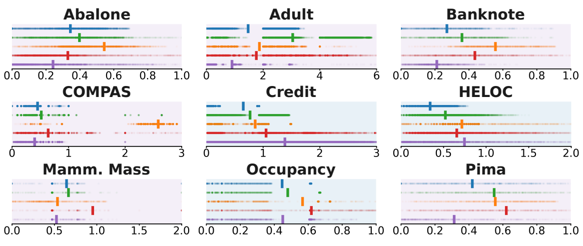


Figure 5. Distribution of counterfactual distances for all methods.

5.4. Regression Example

All three baselines are designed exclusively for binary classification, but T-CREx can operate in a much wider range of contexts, including regression. To demonstrate this, we evaluate the method on an XGBoost regression model for the Wine Quality dataset (Cortez et al., 2009). As discussed above, the CE problem for regression has many valid formulations, but we use a simple approach that partitions the output space into 'low' and 'high' halves using the mean model output μ across $\mathcal{D}_{\text{test}}$. Concretely, we set $Y^* = (\mu, \infty)$ if $f(x^0) \leq \mu$ and $Y^* = (-\infty, \mu]$ otherwise. As in binary classification, this requires steps (b) – (e) of the T-CREx algorithm to be completed twice: once for low-to-high counterfactuals, and once for high-to-low.

The performance of T-CREx$_{0.9}$ and T-CREx$_{0.99}$ on our six key desiderata are shown in Figure 6. The high-level outcomes are consistent with the binary classification setting, insofar as T-CREx$_{0.99}$ reliably returns more accurate rules, at the expense of somewhat worse feasibility, sparsity and complexity. Both variants produce CEs in $\approx 0.1ms$ per instance, similar to comparably-sized datasets in Figure 4 (COMPAS, HELOC). This experiment thus provides good evidence that T-CREx generalises well to regression.

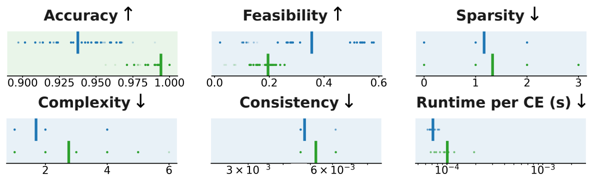


Figure 6. Evaluating T-CREx$_{0.9}$ and T-CREx$_{0.99}$ for regression.

6. Qualitative Analysis

Having provided evidence of the promising quantitative performance of T-CREx, we now briefly explore the qualitative structure of the rule-based counterfactuals it provides.

6.1. Global CE Summary

We begin by examining the rules and metarules learnt by T-CREx$_{0.9}$ for $Y^* = \{high\ income\}$ on the Adult dataset (CV fold 7). Figure 7 depicts this information as a tree diagram, as an alternative to the purely textual form exemplified in Figure 1. In this case, our method learns a total of five metarules (leaves of this tree) for two distinct counterfactual rules (green/orange colouring), whose accuracy, feasibility and complexity are shown in italics. We also show the sparsity, which can differ between metarules for the same rule.[6] This diagram provides a complete global summary of the counterfactual rules learnt for this dataset,

[6]Strictly speaking, this is the *worst-case* sparsity for each (rule, metarule) pair. It is possible to satisfy a metarule and need to change fewer than the stated number of features (for example, if a person is already married).

which describe options for changing the model output from $low\ income$ to $high\ income$ in terms of a range of alternative values for an individual's age, education level, marital status and working hours per week. We can also see that the rule returned for any given individual depends on their current age and working hours.

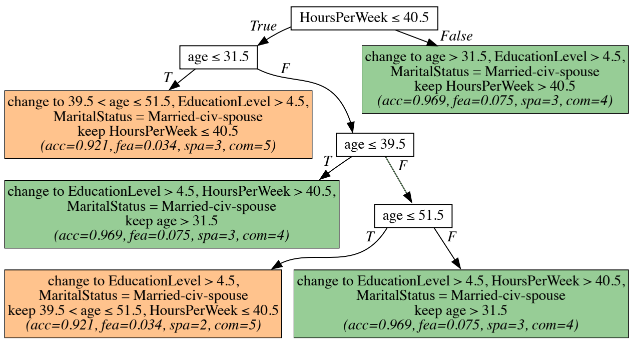

HoursPerWeek ≤ 40.5
True
False
age ≤ 31.5
T
F
change to age > 31.5, EducationLevel > 4.5, MaritalStatus = Married-civ-spouse keep HoursPerWeek > 40.5 (acc=0.969, fea=0.075, spa=3, com=4)
change to 39.5 < age ≤ 51.5, EducationLevel > 4.5, MaritalStatus = Married-civ-spouse keep HoursPerWeek ≤ 40.5 (acc=0.921, fea=0.034, spa=3, com=5)
age ≤ 39.5
change to EducationLevel > 4.5, HoursPerWeek > 40.5, MaritalStatus = Married-civ-spouse keep age > 31.5 (acc=0.969, fea=0.075, spa=3, com=4)
age ≤ 51.5
change to EducationLevel > 4.5, MaritalStatus = Married-civ-spouse keep 39.5 < age ≤ 51.5, HoursPerWeek ≤ 40.5 (acc=0.921, fea=0.034, spa=2, com=5)
change to EducationLevel > 4.5, HoursPerWeek > 40.5, MaritalStatus = Married-civ-spouse keep age > 31.5 (acc=0.969, fea=0.075, spa=3, com=4)

Figure 7. Rules and metarules for the Adult dataset.

6.2. Local CE Example and Metacounterfactual

Remaining with the above example, consider an unmarried individual aged 45, who has no degree and works 35 hours per week, and for whom the model output is $low\ income$. They may wish to know why the output is not $high\ income$ instead. Running their features through the metarule tree leads to the bottom-left of Figure 7, and hence to the yellow counterfactual rule. That is, the model would output $high\ income$ with high probability if this individual were married and had an education level of 5 (Bachelor's degree) or higher, provided their age remained in the 40-51 range and their working hours remained at 40 or fewer.

Now consider what would happen if this individual were 55 years old instead of 45. They would be led to the bottom-right of Figure 7, and hence to the green metarule. Now, in order to be predicted as $high\ income$ with high probability, they would need to make the same feature changes as before (married, education level of 5 or higher) *and* work 41 hours per week or more. The reason for this change, which likely results from trends in the training data reflected in the model, can be pinpointed to their age now being 52 or higher. Such *metacounterfactual* reasoning (i.e. concerning the input changes required to yield an alternative CE) offers an interesting perspective on the issue of explanation robustness (Mishra et al., 2021), and provides a basis for understanding changes in recourse recommended for individuals as a result of natural changes in features such as their age. We believe this basic idea warrants deeper investigation in future.

6.3. Dataset-level Analysis

Figure 8 (a) shows another set of rules and metarules learnt by **T-CREx$_{0.9}$**, in this case for $Y^* = \{no\ diabetes\}$ on the Pima diabetes dataset (CV fold 8). Here, we indicate where 23 unseen test inputs (for which the model output is $has\ diabetes$) fall in the metarule tree. We take the opportunity to collapse any parts of the tree where no data reside, which allows us to focus on the information that is relevant to this particular dataset. The test inputs are distributed between seven metarules for three distinct counterfactual rules. The most common metarule denotes individuals aged 31 or older, with a diabetes pedigree function of 0.636 or below and a plasma glucose concentration exceeding 154.5. For these individuals, it is predicted that the model would output $no\ diabetes$ with high probability if their glucose levels were reduced to 108.5 or below (and their age and diabetes pedigree function value remained in the same range).

High-level counterfactual summary statistics can be derived for this test set, such as Figure 8 (b), which shows the number of inputs for which each feature is included in the returned rule (*"change"* and *"keep"* conditions are plotted separately). This highlights the importance of glucose concentration for the diabetes diagnosis problem: the glucose feature must either be changed or kept within a specified range for all 23 test inputs (change: 16, keep: 7).

12 of the test inputs fall in metarules for which the green counterfactual rule (glucose $\leq$ 139.5 and age $\leq$ 30.5) is optimal. This rule only includes two features, so we can represent it via a 2D plot similar to those in Figures 1 and 2. This is shown in Figure 8 (c), which also includes the values of these features for the training data $\mathcal{D}$ (small points) and the 12 test inputs themselves (large points). The five individ-

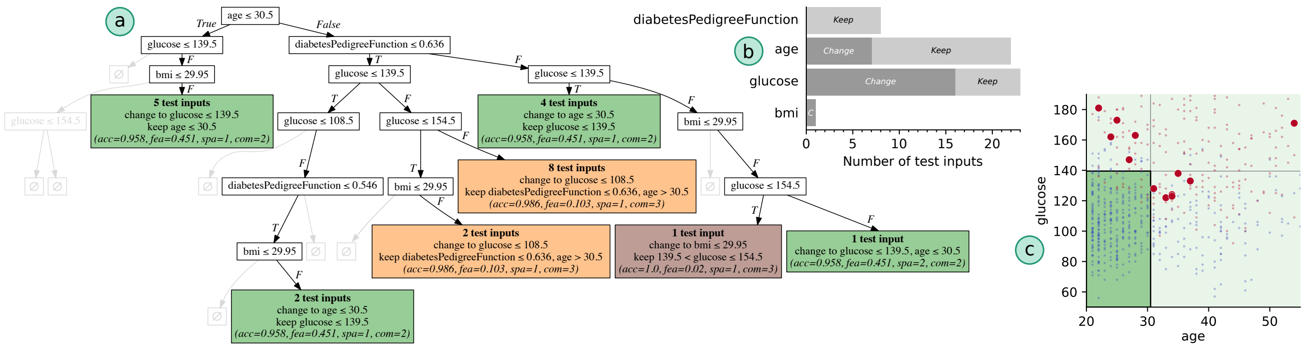

a
age ≤ 30.5
True
False
glucose ≤ 139.5
F
∅
bmi ≤ 29.95
F
5 test inputs
change to glucose ≤ 139.5
keep age ≤ 30.5
(acc=0.958, fea=0.451, spa=1, com=2)
glucose ≤ 154.5
∅
∅
diabetesPedigreeFunction ≤ 0.636
T
glucose ≤ 139.5
T
F
glucose ≤ 108.5
glucose ≤ 154.5
F
∅
diabetesPedigreeFunction ≤ 0.546
T
bmi ≤ 29.95
F
2 test inputs
change to age ≤ 30.5
keep glucose ≤ 139.5
(acc=0.958, fea=0.451, spa=1, com=2)
∅
∅
∅
T
bmi ≤ 29.95
F
2 test inputs
change to glucose ≤ 108.5
keep diabetesPedigreeFunction ≤ 0.636, age > 30.5
(acc=0.986, fea=0.103, spa=1, com=3)
8 test inputs
change to glucose ≤ 108.5
keep diabetesPedigreeFunction ≤ 0.636, age > 30.5
(acc=0.986, fea=0.103, spa=1, com=3)
glucose ≤ 139.5
4 test inputs
change to age ≤ 30.5
keep glucose ≤ 139.5
(acc=0.958, fea=0.451, spa=1, com=2)
bmi ≤ 29.95
∅
glucose ≤ 154.5
1 test input
change to bmi ≤ 29.95
keep 139.5 < glucose ≤ 154.5
(acc=1.0, fea=0.02, spa=1, com=3)
1 test input
change to glucose ≤ 139.5, age ≤ 30.5
(acc=0.958, fea=0.451, spa=2, com=2)
b
diabetesPedigreeFunction
age
glucose
bmi
Keep
Change
C
0
5
10
15
20
Number of test inputs
c
glucose
180
160
140
120
100
80
60
age
20
30
40
50

Figure 8. Rules and metarules for the Pima diabetes dataset, with analysis for a particular sample of 23 test inputs.

uals aged 30 or below require different minimum glucose reductions to satisfy the rule, ranging from 7.5 in the lowest case to 41.5 in the highest. Excluding one outlier, those older than 30 already have glucose levels below the required threshold, and thus would be likely to receive a $no\ diabetes$ prediction if only they were a few years younger. A plot of this type may help to illustrate the differing implications of counterfactual rules for different individuals.

7. Conclusion

We have introduced a fast, model-agnostic method for learning accurate and feasible counterfactual rules, alongside metarules for choosing between them, thereby enabling both local (individual) and global (group-level) CE. We have demonstrated the method's efficacy on benchmark classification and regression datasets with a mix of numerical and categorical features, and shown strong performance relative to baselines on a range of CE desiderata.

A limitation of our current implementation is that is tied to a particular cost function and lacks the facility for domain-specific actionability constraints. This can lead it to return non-actionable counterfactuals (e.g. requiring an individual to lower their age). Although such constraints may not be relevant for understanding a model's bias and fairness, they are crucial for recourse (Karimi et al., 2022). We believe that the method could be generalised to handle actionability and alternative costs without major changes. Also important is the question of tree count and other hyperparameters for surrogate tree growth. While we found single-tree surrogates to be notably effective in our experiments, this issue warrants deeper theoretical and empirical investigation.

Impact Statement

This paper presents a method for explaining the outputs of black box AI models and summarising recourse options for both individuals and groups using human-interpretable rules. As with many proposals made within the wider XAI field, the responsible deployment of more mature versions of such a technology could have a positive societal impact, enabling more understandable, trustworthy and user-friendly AI systems in deployment.

Disclaimer

References

Becker, B. and Kohavi, R. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

Breiman, L. *Classification and regression trees*. Routledge, 2017.

Candanedo, L. Occupancy Detection. UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C5X01N.

Carrizosa, E., Ramírez-Ayerbe, J., and Morales, D. R. Mathematical optimization modelling for group counterfactual explanations. *European Journal of Operational Research*, 2024.

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

Cortez, P., Cerdeira, A., Almeida, F., Matos, T., and Reis, J. Wine Quality. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C56S3T.

Dandl, S., Molnar, C., Binder, M., and Bischl, B. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*, pp. 448–469. Springer, 2020.

Elter, M. Mammographic Mass. UCI Machine Learning Repository, 2007. DOI: https://doi.org/10.24432/C53K6Z.

Fernández, R. R., De Diego, I. M., Aceña, V., Fernández-Isabel, A., and Moguerza, J. M. Random forest explainability using counterfactual sets. *Information Fusion*, 63: 196–207, 2020.

FICO. Explainable Machine Learning Challenge, 2018. https://community.fico.com/s/explainable-machinelearning-challenge.

Guidotti, R. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pp. 1–55, 2022.

Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S., and Turini, F. Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems*, 34(6):14–23, 2019.

I. Amoukou, S. and Brunel, N. Consistent sufficient explanations and minimal local rules for explaining the decision of any classifier or regressor. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 8027–8040. Curran Associates, Inc., 2022.

Kanamori, K., Takagi, T., Kobayashi, K., and Ike, Y. Counterfactual explanation trees: Transparent and consistent actionable recourse with decision trees. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I. (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 1846–1870. PMLR, 28–30 Mar 2022.

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.

Ley, D., Mishra, S., and Magazzeni, D. Globe-ce: A translation based approach for global counterfactual explanations. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Lohweg, V. Banknote Authentication. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C55P57.

Mishra, S., Dutta, S., Long, J., and Magazzeni, D. A survey on the robustness of feature importance and counterfactual explanations. *arXiv preprint arXiv:2111.00358*, 2021.

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

Nash, W., Sellers, T., Talbot, S., Cawthorn, A., and Ford, W. Abalone. UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C55C7W.

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.

Pawelczyk, M., Datta, T., van-den Heuvel, J., Kasneci, G., and Lakkaraju, H. Algorithmic recourse in the face of noisy human responses. *arXiv preprint arXiv:2203.06768*, 2022.

Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., De Bie, T., and Flach, P. Face: feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 344–350, 2020.

ProPublica. compas-analysis: Data and analysis for 'Machine Bias', 2016. https://github.com/propublica/compas-analysis.

Rawal, K. and Lakkaraju, H. Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses. *Advances in Neural Information Processing Systems*, 33:12187–12198, 2020.

Ribeiro, M. T., Singh, S., and Guestrin, C. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Rodríguez, P., Caccia, M., Lacoste, A., Zamparo, L., Laradji, I., Charlin, L., and Vazquez, D. Beyond trivial counterfactual explanations with diverse valuable explanations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1056–1065, 2021.

Sani, H. M., Lei, C., and Neagu, D. Computational complexity analysis of decision tree algorithms. In *Artificial Intelligence XXXV: 38th SGAI International Conference on Artificial Intelligence, AI 2018, Cambridge, UK, December 11–13, 2018, Proceedings 38*, pp. 191–197. Springer, 2018.

Smith, J. W., Everhart, J. E., Dickson, W., Knowler, W. C., and Johannes, R. S. Using the adap learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium on computer application in medical care*, pp. 261. American Medical Informatics Association, 1988.

Spooner, T., Dervovic, D., Long, J., Shepard, J., Chen, J., and Magazzeni, D. Counterfactual explanations for arbitrary regression models. *arXiv preprint arXiv:2106.15212*, 2021.

Tompkins, R., Singh, R., and Miller, T. The effect of diversity in counterfactual machine learning explanations. In *Proceedings of the IJCAI 2022 workshop on Explainable Artificial Intelligence (XAI)*, 2022.

Wachter, S., Mittelstadt, B., and Russell, C. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

Warren, G., Keane, M. T., Gueret, C., and Delaney, E. Explaining groups of instances counterfactually for xai: A use case, algorithm and user study for group-counterfactuals. *arXiv preprint arXiv:2303.09297*, 2023.

Yeh, I.-C. Default of Credit Card Clients. UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C55S3H.

A. Handling of One-hot Encoded Categorical Features

A.1. Well-formedness Constraints on Rules

In one-hot encoding, a categorical feature c taking D_c possible values is mapped to a length-D_c vector in which one 'hot' element is equal to 1 and the remaining 'cold' elements are equal to 0. We can therefore represent an arbitrary mix of numerical and categorical features as a single real vector space $\mathbb{R}^D$, of which a subspace $S_c \subseteq \{1, \dots, D\} : |S_c| = D_c$ represents each categorical c and thus only contains 0 and 1 values.

As discussed in the main paper, a hyperrectangular rule R^i is defined by lower and upper bounds $l_d^i, u_d^i \in \mathbb{R} \cup \{-\infty, \infty\}$ along each feature $d \in \{1, \dots, D\}$. For numerical features, these bounds are unconstrained (except that $u_d^i \geq l_d^i$), but for each categorical feature c, the following conditions must hold for the rule to be well-formed:

- R^i must either specify a single hot category $\text{hot}_c^i \in S_c$ or up to $D_c - 1$ cold categories $\text{cold}_c^i \subset S_c$,[7] but never multiple hot categories (which would be impossible to satisfy) and never a mixture of hot and cold (which would be over-specified; a single hot category fully determines the feature's value).
- In the one-hot case, $l_{\text{hot}_c^i}^i$ must be set to a value in $[0, 1)$. In practice, we use 0.5.
- In the multi-cold case, for each $d \in \text{cold}_c^i$, u_d^i must be set to a value in $[0, 1)$. As above, we use 0.5.
- All other lower and upper bounds for each $d \in S_c$ must be set to $-\infty$ and ∞ respectively.

As a result, for any given point $x^0 \in \mathbb{R}^D$, each categorical feature c can contribute a value of either 0 or 1 to the $\text{changes}(x^0, R^i)$ calculation in Equation 10:

- In the one-hot case where $x_{\text{hot}_c^i}^0 = 1$, the contribution is 0 because $\mathbb{1}[(1 \leq 0.5) \vee (\infty < 1)] = 0$.
- In the one-hot case where $x_{\text{hot}_c^i}^0 = 0$, the contribution is 1 because $\mathbb{1}[(0 \leq 0.5) \vee (\infty < 0)] = 1$.
- In the multi-cold case where $\nexists d \in \text{cold}_c^i : x_d^0 = 1$, the contribution is 0 because $\mathbb{1}[(0 \leq -\infty) \vee (0.5 < 0)] = 0$.
- In the multi-cold case where $\exists d \in \text{cold}_c^i : x_d^0 = 1$, the contribution is 1 because $\mathbb{1}[(1 \leq -\infty) \vee (0.5 < 1)] = 1$.

A.2. Algorithmic Refinements

The **T-CREx** algorithm includes several minor adjustments for the correct handling of categorical features.

A.2.1. Rule Simplification on Extraction from the Surrogate

During the greedy growth process of a standard classification or regression tree, it is possible to create branches with cold split outcomes for one or more categories of a categorical feature, followed by (lower down in the branch) a hot split outcome for another category. As discussed above, the presence of a hot category makes the specification of cold categories redundant, so when extracting rules from the surrogate in step a, we apply a post hoc correction to all extracted rules to set the corresponding bounds to ∞. Secondly, while a rule specifying $D_c - 1$ cold categories for a categorical feature c is technically well-formed, it is equivalent to the simpler and more direct specification of the remaining hot category. We replace any such 'all-but-one-cold' rules with their equivalent one-hot representations.

A.2.2. Modified Subset Relation

A more subtle correction is needed for the hyperrectangle subset relation in Equation 12, which is used to identify maximal-valid rules in step b. From a semantic perspective, a rule R^j that is multi-cold for some categorical c is less specific than another rule R^i that is one-hot with $\text{hot}_c^i \notin \text{cold}_c^j$ (assuming equal bounds on all other features), since the latter is a special case of the former. R^i should therefore be considered a subset of R^j, but the way the bounds of one-hot rules are specified above (i.e. $\pm\infty$ for all $d \in S_c \setminus \{\text{hot}_c^i\}$) leads this not to be the case. To fix this, we temporarily modify all one-hot rules to have explicit upper bounds of 0.5 for all cold categories, which equates to applying the following function:

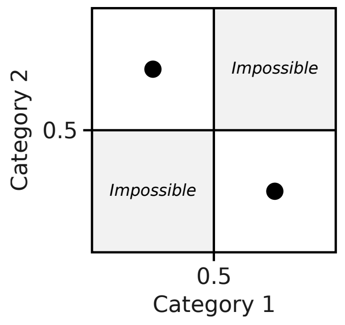
$$\hat{u}_d^i = \begin{cases} 0.5 & \text{if } \exists\, c : (d \in S_c) \wedge (\exists\, d' \in S_c \setminus \{d\} : l_{d'}^i = 0.5), \\ u_d^i & \text{otherwise.} \end{cases} \tag{14}$$

The strict subset relation $R^i \subset R^j$ is then defined exactly as in Equation 12, except using $\hat{u}_d^i$ and $\hat{u}_d^j$ instead of u_d^i and u_d^j.

[7] cold_c^i can be the empty set, in which case the rule effectively ignores feature c.

A.2.3. Skipping of Impossible Grid Cells

In step (c), the bounds of the maximal-valid rules are used to partition the input space into a grid of hyperrectangular cells. In the presence of categoricals, a large proportion of these cells can never be occupied by any real input point, because they specify multiple (or zero) hot categories for the same feature. Consider the following minimal example of an input space consisting of a single binary feature, which is partitioned into a 2×2 grid:



The two cells on the off-diagonal can be occupied by real data as they specify one of the two categories being hot, but the other two cells are impossible as they specify either both or neither. Since we are only interested in finding CEs for inputs that are possible in practice, we skip all such impossible cells when generating prototypes. This delivers a significant efficiency saving: the maximum factor by which a categorical feature c can increase the size of the grid is reduced from 2^{D_c} ($= 4$ in the above example) to D_c ($= 2$).

B. Computational Complexity Analysis

Let D be the dimensionality of the input space, N be the size of the dataset used to grow the surrogate model and T be the number of trees in the surrogate model. The following is a computational complexity analysis of each step of the **T-CREx** algorithm (excluding step (f), which describes no specific computation):

(a) This step consists of growing standard classification or regression trees with an early-stopping criterion enforcing that each leaf has a minimum fraction $\rho \in (0, 1]$ of the data. This creates a maximum of $L_{\max} = \text{ceil}(1/\rho)$ leaves per tree via $L_{\max} - 1$ splitting operations. Since each such operation involves searching over $O(DN)$ possible splits of the data, the complexity of tree growth is $O(DN(L_{\max} - 1)) = O(\frac{DN}{\rho})$. This implies a complexity of $O(\frac{TDN}{\rho})$ when growing T independent trees to form a random forest, which will be somewhat reduced in typical implementations where each tree only uses a subset of features.

(b) The greatest number of valid rules extracted from a single tree, denoted by V, will tend to increase as the hyperparameter $\tau \in (0, 1]$ is decreased, but is upper-bounded by the maximum total number of rules: $V \leq 2L_{\max} - 1 = 2\text{ceil}(1/\rho) - 1$. Evaluating the pairwise subset relation to find the maximal-valid rules involves comparing each pair of rules on each feature, so for T trees is $O(D(VT)^2) = O(D((2\text{ceil}(1/\rho) - 1)T)^2) = O(\frac{DT^2}{\rho^2})$.

(c) Let $M \leq V$ be the greatest number of maximal-value rules extracted from a single tree. Because these rules have been filtered by the subset relation, M is upper-bounded by the maximum number of leaves: $M \leq L_{\max} = \text{ceil}(1/\rho)$. In the worst-case grid size scenario described in the paragraph for this step in Section 3, we must create $O((MT)^D) = O(\frac{T^D}{\rho^D})$ grid cell prototypes, which involves computing $O(\frac{DT^D}{\rho^D})$ individual feature values.

(d) Using Equations 6 and 9 to identify the optimal rule is $O(DMT) = O(\frac{DT}{\rho})$ for each prototype, and hence is $O(\frac{T^D}{\rho^D} \times \frac{DT}{\rho}) = O(\frac{DT^{D+1}}{\rho^{D+1}})$ for all prototypes.

(e) This step involves growing another standard classification tree, but this time without an early-stopping criterion and with the $O(\frac{T^D}{\rho^D})$ prototypes as the points to be classified rather than the original dataset. The complexity of this process depends on how balanced the classification problem is, but has been generically estimated as $O(D[\text{num points}] \log[\text{num points}])$ per tree (Sani et al., 2018). This evaluates to $O(D\frac{T^D}{\rho^D} \log(\frac{T^D}{\rho^D})) = O(D^2 \frac{T^D}{\rho^D}(\log T - \log \rho))$.

g Finding the local explanation for a single instance involves propagating that instance through the metarule tree structure. In the worst case, this involves a number of feature comparisons equal to the depth of this tree, which is $O(\log([\text{num points}])) = O(\log(\frac{T^D}{\rho^D})) = O(D(\log T - \log \rho))$ if the prototype classification problem in step e is relatively balanced.

In practice, we find that d is most computationally expensive step of the algorithm in all experimental settings studied in this paper. It is $O(\frac{DT^{D+1}}{\rho^{D+1}})$, which increases polynomially with higher tree counts T and lower values of the hyperparameter ρ, and increases exponentially with the input dimensionality D.

Steps a, e and g can all leverage highly-optimised decision tree implementations so have a small impact on the total runtime. Although we made an effort to implement the other steps efficiently using parallelised array operations, it is likely that further optimisation is possible.

C. Dataset Details

Our experimental setup is somewhat inspired by that used in the **RF-OCSE** paper (Fernández et al., 2020), and we retain eight out of the 10 public-access datasets studied there. We remove Post-Operative Patient and Seismic Bumps due to their small size (86 instances) and extreme class imbalance (0.07) respectively. In their place, we add Home Equity Line of Credit (HELOC) and Wine Quality, the latter of which is a regression dataset to serve as a demonstration of our method's suitability for that context (see Section 5.4). Otherwise, our data preprocessing pipeline is very similar to that of Fernández et al. (2020). Details of the 10 datasets used in our experiments are as follows (# Inst. = number of instances, # Feat. = number of features, # Cat. = number of categorical features, Class Balance = proportion of instances with the positive class):

Short Name	Full Name	Citation	# Inst.	# Feat.	# Cat.	Class Balance
Abalone	Abalone	(Nash et al., 1995)	4177	8	1	0.50*
Adult	Adult / Census Income	(Becker & Kohavi, 1996)	30,718	11	5	0.25
Banknote	Banknote Authentication	(Lohweg, 2013)	1372	4	0	0.44
COMPAS	COMPAS Recidivism Racial Bias	(ProPublica, 2016)	5278	5	3	0.53
Credit	Default of Credit Card Clients	(Yeh, 2016)	29,623	14	3	0.78
HELOC	Home Equity Line of Credit	(FICO, 2018)	9871	23	0**	0.52
Mamm. Mass	Mammographic Mass	(Elter, 2007)	830	5	2	0.51
Occupancy	Occupancy Detection	(Candanedo, 2016)	2665***	5	0	0.64
Pima	Pima Indians Diabetes	(Smith et al., 1988)	768	8	0	0.65
Wine Quality	Wine Quality	(Cortez et al., 2009)	6497****	12****	1****	N/A

* Abalone is natively a regression dataset, but Fernández et al. (2020) transform it into a classification one by splitting the target variable at the median (hence the equal class balance).

** Strictly speaking, the `MaxDelq2PublicRecLast12M` and `MaxDelqEver` features are categoricals, mapping onto strings such as "120+ Days Delinquent" and "Never Delinquent" (see https://docs.interpretable.ai/stable/examples/fico/). However, the 0-9 numerical encoding used in the dataset file imposes a semantically meaningful ordering on the categories, allowing these features to be treated as numerical in practice.

*** This actually appears to be only the test split of the original dataset, but is the one used in Fernández et al. (2020)'s GitHub repository at https://github.com/rrunix/libfastcrf, so we retain it here.

**** We concatenate the red and white wine subsets of the original dataset and add a binary categorical feature to indicate the colour of each instance. The 11 original features are all numerical.

D. Extended Hyperparameter Study

Figure 8 (overleaf) shows full results for the surrogate tree count and accuracy threshold (τ) hyperparameter study across all nine binary classification datasets.

Before turning to the desiderata, the leftmost columns consider two reasons why our **T-CREx** implementation may fail to return rules. Firstly, no valid rules may be found. In general, this becomes more likely for higher values of τ, as it becomes harder to find rules that satisfy both the feasibility and accuracy criteria. Its likelihood decreases as more surrogate trees are used (more trees means more chances of finding a valid rule), with the notable exception of a single tree. As discussed in the main paper, we believe the strong performance of the single-tree case is due to its use of the entire dataset rather than bootstrap samples as in a random forest. We note that this failure mode never occurs for six out of nine datasets, indicating that it is not always an issue, even for very high τ values.

Alternatively, the algorithm might violate a self-imposed limit on the number of grid cells (we use $1e^5$), which we include to pre-empt and prevent long runtimes. Again, this failure mode does not occur for all datasets, but when it does, it becomes markedly more likely as the number of surrogate trees increases. This is because more trees tend to yield more maximal-valid rules, and thus more unique boundaries from which to construct the cell grid. It also tends to increase with τ, as higher values of this accuracy threshold yield a larger number of more specific maximal-valid rules.

The remaining columns focus on cases for which the algorithm does not fail by either of the two modes described above. The key observations made in Section 5.1 continue to apply across this larger suite of datasets, including:

- The τ-mediated trade-off between accuracy on the one hand, and feasibility, sparsity, complexity and (to a lesser extent) consistency is clear in the vast majority of cases.
- The trends with tree count are less straightforward. Feasibility, sparsity and complexity usually improve as the number of trees is increased from 2-20, although in many cases using a single tree markedly outperforms using two. Consistency, on the other hand, worsens with tree count, as more unique maximal-valid rules exist.
- The trend of accuracy with tree count is either flat or, in some cases, actually decreasing. We believe that this is due to our use of accuracy as a minimum threshold for validity, rather than an objective to be optimised. Having more rules to choose from increases the optimisation space for the other desiderata of feasibility and sparsity, which (due to the aforementioned trade-off) indirectly leads to somewhat less accurate rules being returned.
- The runtime required to learn all rules and metarules increases superlinearly with tree count, which provides a strong motivation for keeping the number low.

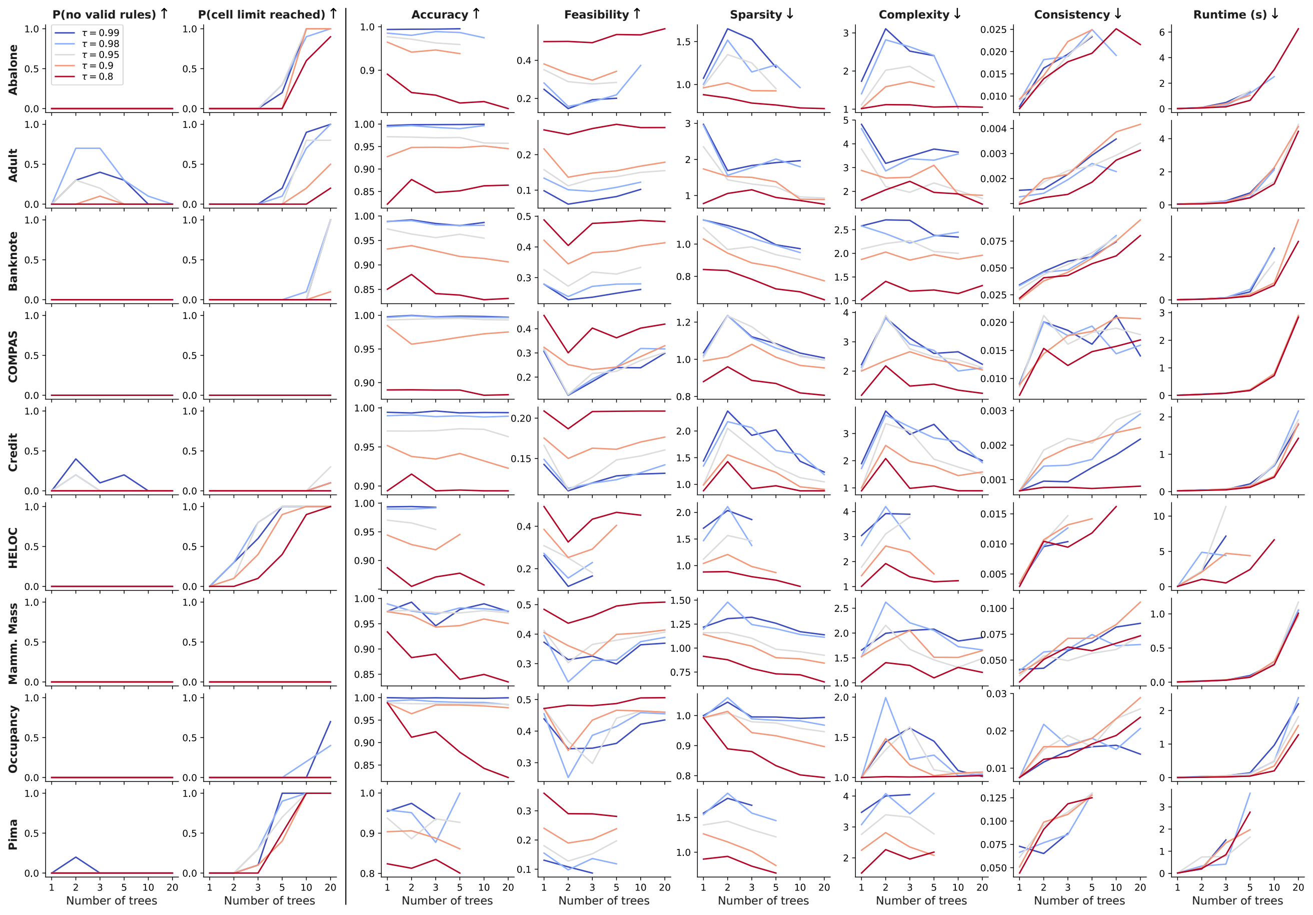


Figure 8. Hyperparameter study for all nine binary classification datasets.

For the Adult dataset, we expand the hyperparameter study to consider variations in the feasibility threshold ρ ($\in \{0.01, 0.02, 0.03, 0.04, 0.05\}$), and plot the results as a grid of heatmaps in Figure 9. The probability of the two failure modes (no valid rules and cell limit reached) increase and decrease with ρ respectively, both of which make sense because higher ρ values yield fewer, larger rules. Overall, the fewest failures occur with $\rho = 0.02$. Together with our informal finding that this value yields good results in practice, this explains our decision to use $\rho = 0.02$ in all other experiments.

The consistency metric, as well as the algorithm's runtime, tend to decrease (i.e. improve) as ρ is increased. The variations in other metrics exhibit less significant trends.

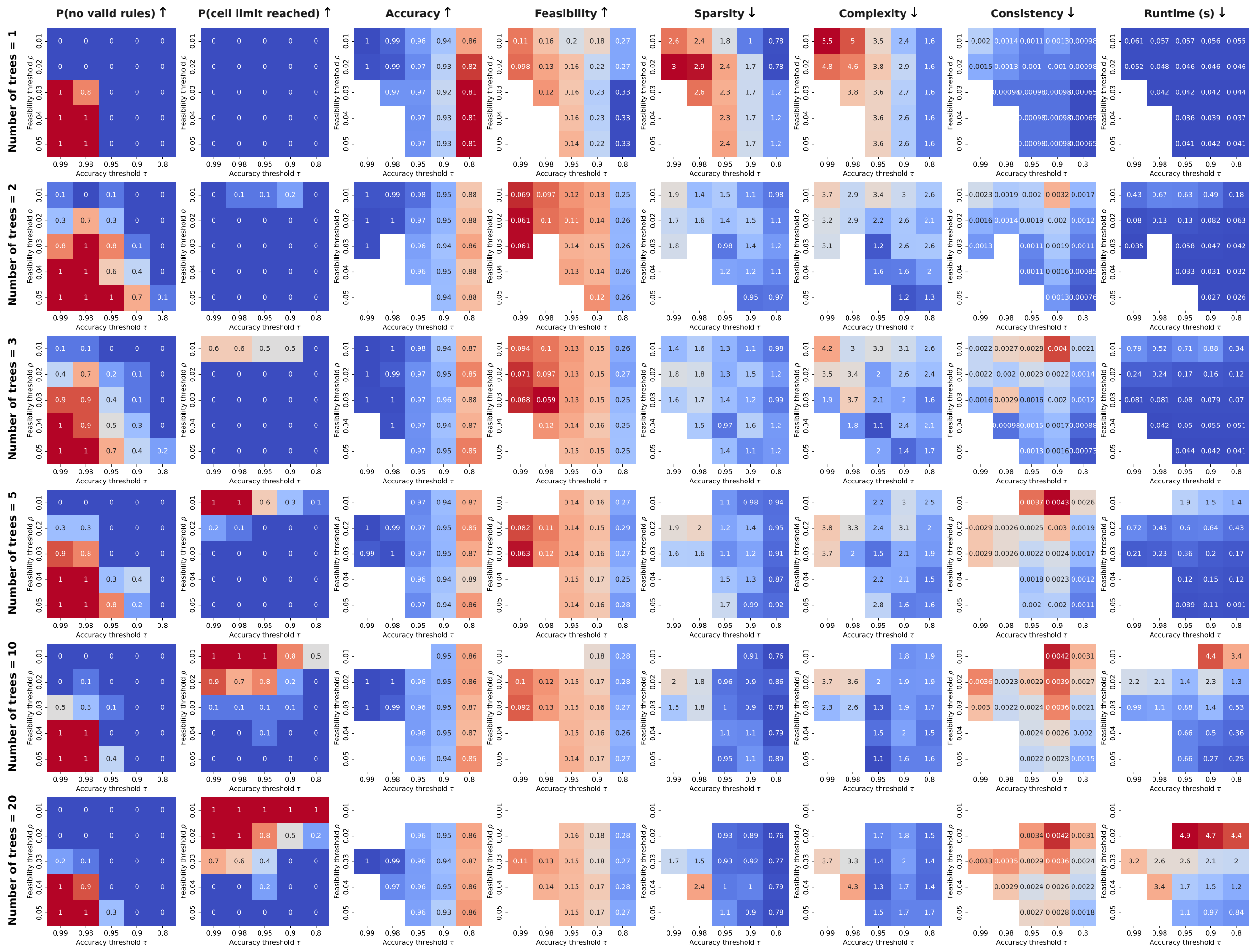


Figure 9. Extended hyperparameter study for the Adult data, also considering variable ρ.

E. Baseline Details

For all three baselines, we use existing public Python implementations and retain default hyperparameter values (or values used in provided exemplar scripts or notebooks).

E.1. AReS

We use a third-party implementation by Kanamori et al. (2022) at `https://github.com/kelicht/cet`. We retain most of the default hyperparameter values (`max_rule=8, max_rule_length=8, discretization_bins=10, max_candidates=50`), but change `minimum_support` from `0.6` to `0.05`, following Kanamori et al. (2022) themselves (see `userstudy.py` on the GitHub repository). The implementation includes a `tuning` method to automatically set the objective weighting hyperparameters λ. We use all default hyperparameter values for this method (`max_change_num=4, cost_type=TLPS, gamma=1.0, lambda_acc=[1.0], lambda_cov=[1.0], lambda_cst=[1.0], objective=origin`).

A slight complication is that the implementation can only natively produce counterfactuals for cases where $f(x^0) = 1$. To produce counterfactuals for $f(x^0) = 0$, we artificially invert the model outputs and run the algorithm a second time.

E.2. LORE

We use the original authors' implementation at `https://github.com/kdd-lab/XAI-Lib`, which they recommend over an older version at `https://github.com/riccotti/LORE`. We use the same hyperparameter values for both the `fit` method (`neigh_type=rndgen, size=1000, ocr=0.1, ngen=10`) and the `explain` method (`samples=1000, use_weights=True, metric=neuclidean`) as in `tabular_explanations_example.ipynb` on the `XAI-Lib` GitHub repository.

In general, **LORE** returns a list of counterfactual rules. To enable direct comparison with the other methods, we retain only the single rule with the fewest finite boundaries (lowest complexity) with ties broken by taking the earliest in the returned list. In rare cases, **LORE** can also fail to return any rules. We exclude such cases from our evaluation.

The implementation handles categorical features using one-hot encoding but proceeds to treat each category feature as if it were numerical, allowing arbitrary rule boundaries that may fall outside the meaningful $[0, 1)$ range. It is unclear whether the authors intended this, but it is visible in `tabular_explanations_example.ipynb` on the `XAI-Lib` repository. We apply a post hoc correction to category feature boundaries, snapping those inside the $[0, 1)$ range to 0.5, and ignoring those outside the range. The authors' naïve treatment of categoricals can also yield impossible 'multi-hot' rules where more than one category feature is positive. In such cases, we apply another correction to choose one of these categories at random.

E.3. RF-OCSE

We use the original author's implementation at `https://github.com/rrunix/libfastcrf`, using default hyperparameter values for the `batch_extraction` function (`max_distance=-1, search_closest=True, max_iterations=-1, epsilon=0.0005`).

The authors present **RF-OCSE** as a method for generating counterfactual rules for random forests given direct access to their internal rule structure, but in this work, we are interested in explaining arbitrary black box models. To adapt **RF-OCSE** for this context, we first fit a random forest surrogate to a training set labelled with model outputs, then apply the algorithm to the surrogate (i.e. mirroring our own approach). We use a `scikit-learn RandomForestClassifier` with `n_estimators=10` and otherwise default hyperparameters, which matches the models studied in the **RF-OCSE** paper (Fernández et al., 2020).

In that paper, it is stated that *"categorical and binary variables are represented using numerical encoding"*, and rules are constructed as if those features were numerical. Reviewing the GitHub repository (specifically `research_paper/dataset_reader.py`), it appears that the numerical encoding is based on an alphabetical ordering of category names, and the rule examples given in Table 6 of (Fernández et al., 2020) are consistent with this. This is clearly an arbitrary and restricting choice, but we retain it to remain faithful to the original implementation.

F. Experiment with Neural Network Model

For the Adult dataset, we repeat the comparative evaluation in Figure 4 for a neural network target model (specifically an `sklearn.neural_network.MLPClassifier` with two hidden layers of 100 units each and ReLU activations). The results are shown in Figure 10, with the equivalent XGBoost results (from the main paper) copied below for reference. Although there are some changes in the rankings of methods on some of the desiderata, overall magnitudes are broadly similar in all cases. This indicates that neither our method, nor any of the baselines, exhibits any special sensitivity to the class of the target model.

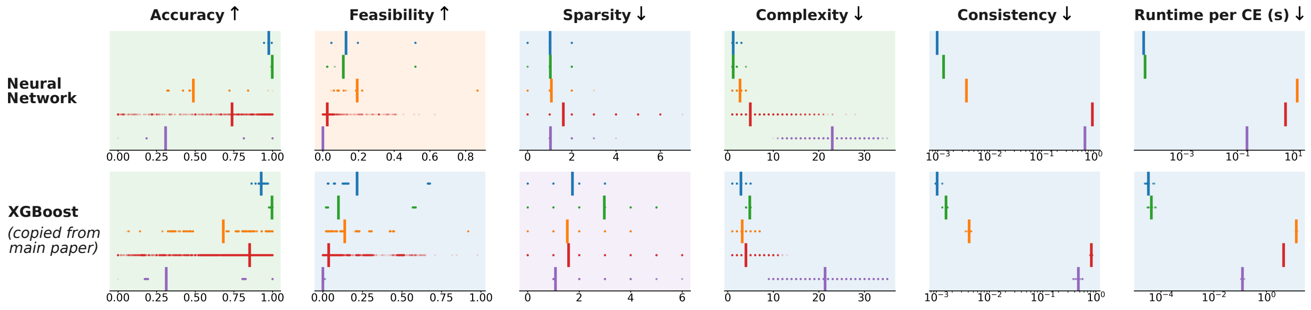


Figure 10. Performance of **T-CREx$_{0.9}$**, **T-CREx$_{0.99}$**, **AReS**, **LORE** and **RF-OCSE** for neural network and XGBoost models (Adult).