Counterfactual Explanations for Cautious Random Forests

HAIFEI ZHANG, UMR CNRS 5516 Laboratoire Hubert Curien, Université Jean Monnet, Saint-Étienne, France
BENJAMIN QUOST, UMR CNRS 7253 HEUDIASYC, Université de Technologie de Compiègne, Compiègne, France
MARIE-HÉLÈNE MASSON, UMR CNRS 7253 HEUDIASYC, Université de Technologie de Compiègne, Compiègne, France, and IUT de l'Oise, Université de Picardie Jules Verne, Beauvais, France

Traditional machine learning models provide a single-class prediction for a given input instance. This may be inadequate in some scenarios, especially when the cost of erroneous predictions is high. Cautious random forests are cautious classification models that may output sets of possible classes as predictions when uncertainty is high, thus reducing the risk of making incorrect decisions. However, making such indeterminate predictions carries a cost, as resolving indeterminacy typically necessitates further analysis and manual intervention. This work focuses on explaining why an indeterminate prediction has been made and how indeterminacy can be resolved. To this end, we use counterfactual examples associated with determinate predictions. We propose a branch-and-bound algorithm that can efficiently generate proximal, plausible, and actionable counterfactual examples. Several experimental results are presented to demonstrate the advantages of our proposed method.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Ensemble methods**; **Knowledge representation and reasoning**;

Additional Key Words and Phrases: Cautious Classifiers, Explainable AI, Counterfactual Explanation, Feature Importance

Associate Editor: Jian Pei

ACM Reference format:
Haifei Zhang, Benjamin Quost, and Marie-Hélène Masson. 2026. Counterfactual Explanations for Cautious Random Forests. *ACM Trans. Knowl. Discov. Data.* 20, 3, Article 50 (March 2026), 35 pages.
https://doi.org/10.1145/3794856

This work was partly done while H. Zhang was a research assistant at the Université de Technologie de Compiègne; we thank the institution for their support.
This work received financial support from Université de Technologie de Compiègne and its subsidiary UTeam.
Authors' Contact Information: Haifei Zhang (corresponding author), **UMR CNRS 5516 Laboratoire Hubert Curien, Université Jean Monnet, Saint-Étienne, France**; e-mail: haifei.zhang@univ-st-etienne.fr; Benjamin Quost, **UMR CNRS 7253 HEUDIASYC, Université de Technologie de Compiègne, Compiègne, France**; e-mail: benjamin.quost@hds.utc.fr; Marie-Hélène Masson, **UMR CNRS 7253 HEUDIASYC, Université de Technologie de Compiègne, Compiègne, France, and IUT de l'Oise, Université de Picardie Jules Verne, Beauvais, France**; e-mail: mylene.masson@hds.utc.fr.

ACM 1556-472X/2026/3-ART50
https://doi.org/10.1145/3794856

1 Introduction

With the increasing availability of data and the considerable advances in computing power, machine learning models have become more sophisticated and can now handle complex tasks with high accuracy in various fields, including medical diagnosis, image recognition, and recommendation systems [48]. Among machine learning models, classifiers aim to predict the class label of a given input instance. Traditionally, the output of these models consists of a single class, corresponding to a precise guess for the actual label of the instance being processed. However, in some situations, the user may desire to control the risk of making errors by producing a set of possible classes for a given input instance, rather than a precise assessment. Models providing such imprecise (set-valued) outputs are called cautious classifiers [43].

Allowing for such indeterminate predictions, consisting of more than one class label, seems especially valuable in cases where the cost of erroneous predictions may be high: among the high-stakes applications, one may cite the fields of medical diagnosis or fraud detection [1, 2, 57]. A cautious classifier can arguably help users to minimize the occurrence of errors (false positives and false negatives in binary classification), for instance, when training data are scarce or pervaded with noise.

This article focuses on the binary classification case, using a set-valued classification model known as the **Cautious Random Forest (CRF)** [58, 60, 61]. A CRF generalizes the classical random forest classifier by leveraging the imprecise Dirichlet model [6, 55] in the theoretical framework of belief functions [18, 49]. Its main advantage is its ability to make indeterminate predictions in the presence of either epistemic uncertainty (when the tree outputs are based on scarce information) or aleatoric uncertainty (when the classes present are mixed and therefore the tree outputs are conflicting, which typically happens near the decision boundaries).

Obviously, making imprecise predictions carries a cost, since resolving the indeterminacy typically requires further analysis, typically via the intervention of a human expert [59]. Therefore, it appears highly desirable to be able to provide such an expert with explanations about why a prediction is indeterminate and what could be done to turn it into a determinate one. Such questions fall under the scope of explainable machine learning, which aims at a better understanding of machine learning models and their outputs, so as to improve their transparency and thereby the trust of the users [3, 5, 40].

In this article, we propose to use the concept of counterfactual examples to "explain" indeterminate predictions made by a CRF model. Such examples are a kind of contrastive explanation, which allows one to identify the minimal modifications of some feature values to resolve the indeterminacy and thus obtain a desired determinate prediction. We focus here on synthetic counterfactual examples, i.e., which are generated according to the data distribution so as to convey information on the prediction made for an instance of interest. We stress that generating such counterfactual examples amounts to solving a difficult optimization problem: counterfactual examples should indeed be close to the instance of interest, but at the same time "in-distribution" (i.e., they should not be outliers), and the associated model output should be determinate.

Our main contribution is an efficient algorithm to generate counterfactual examples, specifically designed for CRFs. This algorithm relies on a branch-and-bound procedure initially introduced in [8], to which several significant improvements are brought. Notably,

- —we propose a simple decomposition of the feature space into small decision regions;
- —we introduce a filtering step to narrow the search region of counterfactual examples;
- —we show how constraints on features can be taken into account so as to ensure the actionability of the generated counterfactual examples;

— we integrate a plausibility metric in the selection step, so as to make the generated counterfactuals as realistic as possible;
— we investigate how global and local feature importance can be leveraged to accelerate the counterfactual example generation process, and thereby lead to a more efficient and transparent resolution of indeterminacy.

The article is organized as follows. Section 2 recalls basic material on CRFs (Section 2.1) and gives a brief overview of explainable machine learning (Section 2.2). Section 3 focuses on counterfactual examples: after recalling the notion and mentioning desirable properties (Sections 3.1 and 3.2), it discusses how it can be used in the context of indeterminate predictions (Section 3.3). We present our first contribution, an efficient counterfactual example generation method in Section 4. The use of feature importance metrics to accelerate the counterfactual generation process, which constitutes our second contribution, is investigated in Section 5. Section 6 presents and discusses the results of our experimental study, which compares our approach to four other state-of-the-art counterfactual generation strategies on several binary datasets from the literature, demonstrating its efficiency and effectiveness. Finally, a conclusion is drawn in Section 7.

2 Background Knowledge

2.1 CRFs

We focus on binary classification problems: we denote by $\mathcal{Y} = \{c_1, c_2\}$ the set of classes. Let $h = \{h_t,\ t = 1, \ldots, T\}$ be a random forest composed of T decision trees, obtained from a training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$. Each observation $\boldsymbol{x} \in \mathcal{X}$ is assumed to be the realization of a random vector $\mathbf{X} = (\mathrm{X}_1, \ldots, \mathrm{X}_M)$ composed of M random variables. From now on, we will refer to the decision made by tree h_t as $h_t(\boldsymbol{x}) \in \mathcal{Y}$, and to the associated posterior probability distribution estimate as $\boldsymbol{\rho}_t(\boldsymbol{x}) \in \Delta^2$ (with Δ^2 the two-dimensional simplex): $\boldsymbol{\rho}_t(\boldsymbol{x}) = (\rho_{t1}(\boldsymbol{x}), \rho_{t2}(\boldsymbol{x}))$, with $\rho_{tj}(\boldsymbol{x}) := \widehat{\Pr}(c_j|\boldsymbol{x})$, for $j = 1, 2$.

For any input instance $\boldsymbol{x} \in \mathcal{X}$, let us denote by $n_{tlj}(\boldsymbol{x})$ the number of training samples of class c_j falling into the same leaf η_{tl} as $\boldsymbol{x}$ for tree h_t. According to the imprecise Dirichlet model [6, 55], the interval estimates for the class posterior probabilities $\rho_{tlj}(\boldsymbol{x})$ are defined by:

$$[\rho]_{tlj}(\boldsymbol{x}) = \left[\underline{\rho}_{tlj}(\boldsymbol{x}), \overline{\rho}_{tlj}(\boldsymbol{x})\right] = \left[\frac{n_{tlj}(\boldsymbol{x})}{N_{tl}(\boldsymbol{x}) + s}, \frac{n_{tlj}(\boldsymbol{x}) + s}{N_{tl}(\boldsymbol{x}) + s}\right], \tag{1}$$

where $N_{tl}(\boldsymbol{x}) = n_{tl1}(\boldsymbol{x}) + n_{tl2}(\boldsymbol{x})$ is the total number of instances in that leaf, and parameter s controls the size of the intervals and thus the imprecision of the model. Note that s can be interpreted as the number of virtual samples falling in the leaf and whose actual class is unknown. The choice of an appropriate value for the parameter s remains an open issue; yet it is often recommended to pick a value $s = 1$ or $s = 2$ [55].

In [58, 60], a strategy rooted in the theory of belief functions is proposed to aggregate the imprecise outputs (probability intervals) provided by these trees. The theory of belief functions, also referred to as the theory of evidence or the Dempster–Shafer theory [18, 49], provides a general framework for modeling and reasoning with uncertainty. When processing a given instance $\boldsymbol{x}$, each interval $[\rho]_{tj}$[1] provided by the tree h_t can be regarded as a focal element associated with a mass m_t on the interval $[0, 1]$. Then, the belief $bel_1(\boldsymbol{x})$ and plausibility $pl_1(\boldsymbol{x})$ of the event $\Pr(c_1 \mid \boldsymbol{x}) \in [0.5, 1]$ are calculated, which quantifies the available evidence regarding the proposition that instance $\boldsymbol{x}$

[1]Note that we drop the leaf index, as it does not matter in the decision-making process.

belongs to c_1—note that, by duality, we have $bel_2(\boldsymbol{x}) = 1 - pl_1(\boldsymbol{x})$ and $pl_2(\boldsymbol{x}) = 1 - bel_1(\boldsymbol{x})$. We have:

$$bel_1(\boldsymbol{x}) := Bel(\Pr(c_1 \mid \boldsymbol{x}) \in [0.5, 1]) = \sum_{t:[\rho]_{t1} \subseteq [0.5,1]} m_t = \sum_{t=1}^{T} m_t \mathbb{1}(\underline{\rho}_{t1}(\boldsymbol{x}) \geq 0.5), \tag{2a}$$

$$pl_1(\boldsymbol{x}) := Pl(\Pr(c_1 \mid \boldsymbol{x}) \in]0.5, 1]) = \sum_{t:[\rho]_{t1} \cap]0.5,1] \neq \emptyset} m_t = \sum_{t=1}^{T} m_t \mathbb{1}(\overline{\rho}_{t1}(\boldsymbol{x}) > 0.5), \tag{2b}$$

where $\mathbb{1}(\cdot)$ is the indicator function. The mass degree m_t is the degree of support to class c_1 derived from the set of intervals $[\boldsymbol{\rho}]_t$; it can also be seen as the weight of tree h_t in the ensemble—a typical weight value being $m_t = 1/T$. Alternatively, tree weights may be determined so as to optimize a criterion: a cost function that takes both determinacy and accuracy into account is proposed to determine tree weights in [60]. Due to these weights, the imprecise tree aggregation strategy can be seen as a generalized voting approach.

Given an interval $[bel_1(\boldsymbol{x}), pl_1(\boldsymbol{x})]$,[2] a decision, hereafter written $h(\boldsymbol{x})$, can be made by applying the interval dominance principle:

$$h(\boldsymbol{x}) = \begin{cases} \{c_1\} & \text{if } bel_1(\boldsymbol{x}) > 0.5, \\ \{c_2\} & \text{if } pl_1(\boldsymbol{x}) < 0.5; \\ \{c_1, c_2\} & \text{otherwise.} \end{cases} \tag{3}$$

We can see that the decision is indeterminate whenever $0.5 \in [bel_1(\boldsymbol{x}), pl_1(\boldsymbol{x})]$. This robust decision-making scheme based on random forests amounts to reaching a compromise between accuracy and cautiousness (avoidance of errors); notably when data are pervaded with noise [60].

2.2 Explanations in Artificial Intelligence

Artificial intelligence models, especially those based on statistical learning algorithms, are nowadays widely deployed in many application fields. Models exhibit significant performance due to their complex algorithmic design and their use of massive training data.

However, assessing the quality of predictive models solely on their accuracy may lead to neglecting important aspects tied to their use. Notably, decision-making processes often need to be explained [19]. This is particularly important in fields such as criminal justice, where providing evidence supporting the guilt or innocence of an individual is arguably essential to affect the court judgment; or in medical diagnosis, where providing a basis for the diagnosis is a key factor in choosing a treatment [39].

Therefore, the field known as **eXplainable Artificial Intelligence (XAI)** has emerged, which aims at making artificial intelligence models, and especially those inferred via machine learning, more transparent, interpretable, and understandable to human users [36].

2.2.1 Taxonomy of Methods. The categorization of XAI methods is subjective and largely depends on the criteria used, as outlined in [40]. We can nevertheless point out three main kinds of taxonomy.

Intrinsic vs. Post-Hoc Approaches. Intrinsic explainable models are designed to be inherently interpretable, which means that their internal mechanisms can be easily understood by a human. These models, such as linear regression, logistic regression, and (shallow) decision trees, can be analyzed to provide transparent explanations of how a specific decision or prediction was reached. *Post-hoc* explanation methods, on the other hand, are used to provide explanations for machine

[2] By duality, the interval $[bel_2(\boldsymbol{x}), pl_2(\boldsymbol{x})]$ can be deduced from $[bel_1(\boldsymbol{x}), pl_1(\boldsymbol{x})]$.

learning models referred to as black-box models. These latter studies analyze the input and output behavior of the model after training and somehow try to infer the decision-making process.

Model-Agnostic vs. Model-Specific Approaches. Explanation methods designed to provide insights into a particular model are referred to as model-specific; intrinsic explanation methods, as discussed above, are examples of model-specific methods, since they are tailored to the unique structure and properties of the model under consideration. In contrast, model-agnostic explanation methods, which generally examine the association between input features and output predictions but do not have access to the internal workings of the model, can in principle be applied to any model.

Global vs. Local Approaches. Global model explainability refers to the ability to explain the overall behavior of a model. This typically involves understanding the key factors or features based on which decisions are made. Global explanations provide a high-level understanding of the model decision-making process, but they do not necessarily give insights into the specific reasons why a particular prediction was made for a given input. For example, the so-called important features in a model (see below) are a kind of global explanation. Local model explainability, on the other hand, focuses on explaining model outputs for specific instances. This type of explanation is advantageous when the prediction of the model differs from what a human user would expect—such as, e.g., identifying the features that led to making a specific decision rather than a given alternative one. Local explanations can help to identify the specific factors that are driving the prediction for a particular input, which can also be helpful in identifying errors or biases in the model.

2.2.2 Types of Explanations. Apart from the above general characteristics, the nature of explanations can vary from one method to another. In this article, we focus on two particular kinds of explanations. For other types of explanations, such as model internals, proxy models, saliency maps, and concept-based explanations, we refer the reader to the well-known book of Molnar [40] and the survey papers such as [5, 44, 52].

Feature importance. One of the most commonly used explanation strategies, feature importance, amounts to assigning each feature of the input space a measure of its contribution to the model predictions. For inherently explainable models, such as linear regression and logistic regression, the learned weights can be seen as direct measures of feature importance. For tree-based models, a feature importance metric can be computed as the normalized total reduction of an impurity measure (such as the Gini index) [9]. For *post-hoc* explanation methods, the **Permutation Feature Importance (PFI)**, which can be seen as obtained via sensitivity analysis, evaluates the importance of a feature via the difference between the baseline performance metric (often the accuracy) and the one obtained from the same dataset with one column permuted [9, 21].

At the local explanation level, **Local Interpretable Model-agnostic Explanations (LIME)** [46] and **SHapley Additive exPlanations (SHAP)** [38] are two other popular feature importance evaluation methods for single query instances. LIME learns a sparse linear model based on instances sampled around the query instance to approximate the local classification boundary and uses the learned weights as feature importance. SHAP is based on the coalition game theory and relates the feature importance to the feature contribution to the predictions. Beyond assessing the importance of each feature specifically, taking into account interactions between features may be critical, as the influence of features on model outputs is generally not independent due to these interactions.

Example-Based Explanations. Following the philosophy of "similar questions, similar answers," prototypes can be used as a global model-agnostic explanation method: using this strategy, the entire dataset is summarized by a small number of representative samples selected, or computed, from the training data [16, 30, 51]. For a given query sample, the nearest prototype is used as the explanation. Akin to prototypes, influential instances, whose deletion would have a

significant impact on the model outputs, can be identified by deletion diagnosis [14] and influence functions [31].

A prominent example of local example-based explanation is that of counterfactual example. A counterfactual example makes it possible to identify the minimum changes that should be applied to the feature values of a given input instance to alter the associated model output into a desired one [54]. A large number of counterfactual example generation methods can be found in the survey of Guidotti [24], which can either be model-specific or model-agnostic. We note here that counterfactual examples can be related to adversarial instances, which are generated so as to trick the model into making erroneous decisions [7]. Adversarial instances can therefore be leveraged so as to make the model more robust.

3 Counterfactual Explanations for Indeterminate Predictions

Indeterminate predictions provided by a cautious classifier, such as the CRF considered here, allow users to avoid making erroneous decisions. However, indeterminacy must be resolved so that a precise (determinate) and right decision is made. In real-world applications, leaving the entire responsibility of resolving this indeterminacy to experts of the field would result in a very demanding decision process. Therefore, for each instance with an indeterminate prediction, we propose to provide counterfactual examples associated with each class, so that users can know how indeterminacy can be resolved by minimally modifying the feature values.

3.1 Counterfactual Explanations for Classical Predictions

We recall that a counterfactual explanation for a prediction characterizes the smallest changes (modifications) to the feature values which turn the prediction into a desired one—in our case, into a predefined class [40]. The examples, thus, obtained by modifying the original instance are called counterfactual examples. Formally, for a given classifier h, an instance $\boldsymbol{x}$ and a desired prediction $y' \neq h(\boldsymbol{x})$, the counterfactual example $\boldsymbol{x}' \in \mathcal{X}$ is defined as:

$$\boldsymbol{x}' = \arg\min_{z \in \mathcal{X}} d(\boldsymbol{x}, \boldsymbol{z}), \text{ s.t. } h(\boldsymbol{z}) = y', \tag{4}$$

where $d(\cdot)$ is a dissimilarity measure. It is recommended in [54] to use the Manhattan distance weighted with the inverse **Median Absolute Deviation (MAD)**:

$$d(\boldsymbol{x}, \boldsymbol{z}) = L_1^{mad}(\boldsymbol{x}, \boldsymbol{z}) = \sum_{j=1}^{M} \frac{|\boldsymbol{x}_j - \boldsymbol{z}_j|}{MAD_j}, \tag{5}$$

where x_j stands for the value of $\boldsymbol{x}$ as per feature X_j, and where the MAD is calculated on the training set as:

$$MAD_j = \underset{i \in \{1,\ldots,N\}}{\text{Median}} \left(\left| x_{ij} - \underset{l \in \{1,\ldots,N\}}{\text{Median}} (x_{lj}) \right| \right). \tag{6}$$

This normalized distance captures scale differences among features and is robust to outliers. Due to the properties of the L_1 norm, the method produces sparse counterfactual examples, i.e., with a small number of modified features with respect to the queried instance. Alternatively, the squared Euclidean distance weighted by the inverse **Standard Deviation (STD)** can be used:

$$d(\boldsymbol{x}, \boldsymbol{z}) = L_2^{std}(\boldsymbol{x}, \boldsymbol{z}) = \sum_{j=1}^{M} \frac{(x_j - z_j)^2}{STD_j}. \tag{7}$$

Note that the squared Euclidean distance can also be weighted by the inverse variance, which downscales the influence of features with high variability in a more significant way. In this article, we follow the definition in [54], using the inverse STD as the normalization factor.

3.2 Desirable Properties of Counterfactual Examples

Local explanations are by nature context-dependent and subjective; this makes it difficult to characterize what determines a "good" explanation. We nevertheless mention several desirable properties in the specific case of counterfactual explanations, the importance of which depends on the context in which such explanations are provided.

- *Validity*. A counterfactual example is said to be valid if its associated model output meets the requirements of the user, i.e., it corresponds to the desired class that is different from that of the query instance.
- *Proximity*. A counterfactual example $\boldsymbol{x}'$ for a query instance $\boldsymbol{x}$ should be as close as possible to the latter, according to the chosen dissimilarity measure. Note that proximity may be antagonistic to the plausibility criterion mentioned below.
- *Sparsity*. Counterfactual examples obtained by modifying only a small number of feature values can arguably be translated more easily into understandable explanations. A natural and efficient way of measuring sparsity consists in counting the number of differences across features between $\boldsymbol{x}$ and $\boldsymbol{x}'$.
- *Plausibility*. A counterfactual example is said to be plausible if its feature values are consistent with those observed in the training set. In other words, the counterfactual example should be in-distribution, i.e., situated in a high-density region of the input space.
- *Actionability*. Depending on the application, some features may be considered as immutable, i.e., impossible to modify for example, the case of race, gender, or age. Others can be considered as mutable under some constraints, for instance, in one direction only (monotonicity). To be actionable, a counterfactual example should result from the modification of mutable features under the possible existing constraints.
- *Diversity*. Producing several diverse counterfactual examples rather than a single one is likely to lead to more meaningful explanations, suitable for a higher number of users, or likely to be accepted by a given user.

3.3 Counterfactual Examples for Indeterminate Binary Predictions

We consider a binary cautious classification setting with $\mathcal{Y} = \{c_1, c_2\}$. We propose a natural way to generalize the notion of counterfactual example: we define, for a given instance $\boldsymbol{x}$ such that $h(\boldsymbol{x}) = \{c_1, c_2\}$ (indeterminate), two counterfactual examples $\boldsymbol{x}^{c_1}$ and $\boldsymbol{x}^{c_2}$ such that:

$$\boldsymbol{x}^{c_1} = \arg\min_{\boldsymbol{z}\in\mathcal{X}} d(\boldsymbol{x}, \boldsymbol{z}) \text{ s.t. } h(\boldsymbol{z}) = \{c_1\}, \tag{8a}$$

$$\boldsymbol{x}^{c_2} = \arg\min_{\boldsymbol{z}\in\mathcal{X}} d(\boldsymbol{x}, \boldsymbol{z}) \text{ s.t. } h(\boldsymbol{z}) = \{c_2\}, \tag{8b}$$

where the dissimilarity measure $d(\cdot)$ can be defined by Equation (5) or Equation (7). These counterfactual examples provide meaningful information to a user, as they can help determining the minimal modifications needed to obtain a desired precise (determinate) prediction; they make it possible to identify the closest class to an indeterminate instance, and they facilitate understanding the differences between the two classes.

Example 3.1 (Counterfactual Examples for Indeterminate Predictions). We provide here an example to illustrate the concept of counterfactual examples for indeterminate predictions. This example is

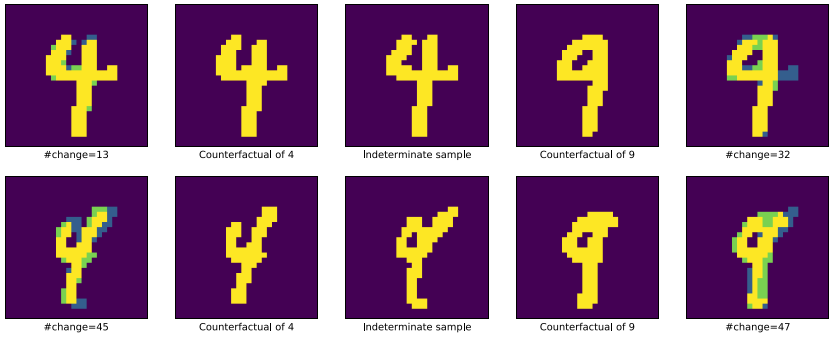


Fig. 1. Examples of indeterminate numbers (center) and corresponding counterfactual examples of class 4 (left) and 9 (right). Left- and right-most images display pixels to be added (green) and to be deleted (blue) in order to obtain the determinate predictions. Figures come from [59].

based on the MNIST dataset, a database of images of handwritten numbers containing about 60,000 training cases and 10,000 test cases.

Samples corresponding to classes "4" and "9" were selected so as to construct a binary classification problem. In the resulting dataset, some of the pictures are ambiguous: it is difficult to safely establish whether the number is a "4" or a "9." Generating counterfactual examples for such an ambiguous query instance is used to identify which parts of the image are responsible for the indeterminacy of the decision. Figure 1 illustrates this point for two indeterminate instances (in the center), displaying how they can be modified to reach both possible determinate decisions (i.e., either a "4" or a "9").

The number of pixels to be modified to transform the query instance into a counterfactual example can be seen as a similarity measure to this latter—and hence to its class: for example, in Figure 1, the first instance is clearly closer to a "4" than to a "9." Additionally, comparing the generated counterfactual instances provides some insights regarding the differences between the classes—in the present case, the presence of a circular shape hints at the number being a "9," and the presence of a horizontal bar protruding to the right suggests a "4." ■

We close this section with a short discussion on the multi-class setting, i.e., $|\mathcal{Y}| > 2$. Given a query instance with an indeterminate prediction, our approach can be extended in several ways. If the goal is to generate counterfactual examples for which the model issues precise predictions, then extending our method is straightforward: we may search for candidates in decision regions that turn the decision into a precise one (i.e., a single class), regardless of the number of classes of the problem. However, if we allow for generating counterfactuals associated with indeterminate predictions—even if more precise than that for the query instance, the problem is arguably much more complex. Let $h(\boldsymbol{x}) = A \subsetneq \mathcal{Y}$ be an indeterminate output for an instance $\boldsymbol{x}$, and $A' \subsetneq \mathcal{Y}$ be the desired (possibly indeterminate) prediction with $A' \neq A$; then, we can generalize the problem as follows:

$$\boldsymbol{x} = \arg\min_{\boldsymbol{z} \in \mathcal{X}} d(\boldsymbol{x}, \boldsymbol{z}) \text{ s.t. } h(\boldsymbol{z}) = A', \tag{9}$$

where the dissimilarity measure is again defined as per Equation (5) or Equation (7). According to possible additional constraints on A' (e.g., $|A'| = 1$, $A' \subset A$, $|A'| \leq |A|$), depending on the application, different kinds of explanations may be found. The interest of such counterfactual examples with given set-valued predictions, and how they can be generated, appear to be an interesting direction of research, which remains in the initial stages [22]. In this article, we focus on generating counterfactual examples with determinate predictions, leaving set-valued counterfactual explanations in a multi-class setting for further investigation.

4 Counterfactual Generation for CRFs

Solving the counterfactual example generation problem defined in Equation (8) is complex. A common approach is to minimize an objective function via a gradient-based optimization method. Wachter et al. [54] proposed the following optimization problem:

$$\boldsymbol{x}' = \arg\min_{\boldsymbol{z}\in\mathcal{X}} \ \alpha\mathcal{L}(h(\boldsymbol{z}), y') + d(\boldsymbol{x}, \boldsymbol{z}), \tag{10}$$

where $\mathcal{L}(\cdot)$ is a loss function, and α determines the compromise between generating a valid counterfactual example (ensured by minimizing the first term) close to the query instance (second term).

As we already stressed, besides validity and proximity which appear in Equation (10), counterfactual examples are often required to satisfy additional properties—sparsity, plausibility, actionability, and diversity [13, 24, 53]. To this end, Dandl et al. [17] proposed to generate counterfactual examples by solving a multi-objective optimization problem. Laugel et al. [32] and Mothilal et al. [41] enforced diversity by adding a specific term to Equation (10). However, for models such as those based on trees, generating counterfactual explanations based on minimizing a loss function via continuous optimization is difficult, since no gradient information can be leveraged.

In this case, model-agnostic counterfactual explanation methods can be used. A line of work attempts to identify suitable counterfactuals directly from the observed data. This so-called **Minimum Observable (MO)** principle has been extended in several directions. For example, **Discovering Counterfactual Explanations using Relevance Features from Neighborhoods (DisCERN)** [56] reduces the number of modified attributes by exploiting estimates of feature relevance. Another representative method is LORE [25], which constructs a local neighborhood of synthetic instances using an evolutionary algorithm and then fits a decision tree to approximate the model's local behavior. MACE [28] translates the forest into logical constraints and applies satisfiability modulo theories to search for counterfactuals. Other model-agnostic frameworks explicitly search the input space for counterfactuals: Growing Spheres [33] generates candidates by expanding hyperspheres around the factual instance until an instance with the desired prediction is identified; the performance of this approach suffers from the curse of dimensionality. While the original DiCE paper [41] formulates the counterfactual generation problem under the assumption that the predictive model is differentiable and static, subsequent implementations also provide non-gradient-based variants (e.g., random or genetic search) that can be applied to tree models. However, these alternatives tend to be computationally expensive, especially for high-dimensional data or large ensembles.

A number of methods have been specifically designed for random forests; they can be categorized as follows: (i) optimization formulations, (ii) simplification of the tree ensemble, and (iii) feature-space partitioning. Optimization-based approaches such as OAE [15], DACE [27], and OCEAN [42] encode counterfactual generation as integer or mixed-integer programs. FOCUS [37] instead provides a smooth approximation of trees with differentiable operators, thereby enabling gradient-driven search. Model simplification techniques replace the forest with a single surrogate tree. For instance, FBT [47] constructs global proxy trees that approximate the random forest on the whole input space—which may introduce a non-negligible mismatch with the original ensemble. OCSE [20] instead performs a local conversion around the instance of interest, yielding a tree that is locally exact and from which optimal counterfactual sets are extracted. While this substantially reduces approximation error, the local fusion process can still be computationally demanding, especially for large forests or points far from the decision boundaries. OFCC [58] restricts modifications to the query instance to a single feature change, which limits its ability to capture counterfactuals requiring coordinated multi-feature modifications. **Live Region (LIRE)** [12] generates counterfactual examples in "live regions" that are intersections of leaves where

Table 1. Taxonomy of Counterfactual Explanation Methods for Random Forests

Method	Validity	Proximity	Plausibility	Actionability	Diversity	Efficiency
Model-agnostic approaches						
MO	●	❍	✓	✓	~	●
DisCERN [56]	●	◗	✓	✗	~	●
LORE [25]	❍	◗	✗	✗	✗	❍
MACE [28]	●	●	✓	✓	✓	❍
DiCE [41]	●	●	✗	✓	✓	❍
Growing Spheres [33]	●	●	~	✗	~	◗
Tree-ensemble-specific approaches						
OAE [15]	●	●	✗	✓	✓	❍
DACE [27]	●	●	✓	✓	✓	❍
OCEAN [42]	●	●	✓	✓	~	❍
FOCUS [37]	❍	◗	✗	✗	✗	❍
FBT [47]	❍	◗	✗	✗	✗	◗
OCSE [20]	●	●	✗	✗	✓	◗
OFCC [59]	●	◗	✗	✗	~	●
LIRE [12]	●	◗	✗	✗	~	●
Ours	●	●	✓	✓	~	●

●, high; ◗, medium; ❍, Low; ✓, considered; ~, partially considered or easy to solve; ✗, not considered.

training samples fall. Our framework also belongs to this partition-based family, but searches for counterfactual examples by performing a branch-and-bound search over decision regions, which are defined as intersections of split intervals across all cautious trees.

As summarized in Table 1, most existing methods guarantee validity and often aim for proximity, but they typically do not address plausibility, actionability, or efficiency in a satisfactory manner. Our branch-and-bound framework stands out by jointly ensuring validity and proximity, while explicitly incorporating plausibility and actionability constraints, and maintaining competitive efficiency in the context of CRFs, as illustrated in Figure 2. This highlights the novelty and practical relevance of our contribution compared to prior works.

In a nutshell, our approach consists of dividing the input space into what we will call "decision regions," in which all data points have the same model output; then, we use a branch-and-bound strategy with a carefully designed search tree to find the closest counterfactual example satisfying the desired properties for a given query instance. Although diversity is not the primary focus of our framework, it can be naturally addressed within our branch-and-bound procedure. Indeed, once the search space has been explored to identify one counterfactual solution, additional diverse counterfactuals can be obtained at negligible extra cost by continuing the exploration and collecting alternative feasible regions that satisfy the desired constraints. In this sense, our approach can readily produce a set of diverse counterfactuals; however, optimizing the computation of this set while satisfying the diversity requirement may require adapting the search, for instance by modifying the order of features in the search tree once a new counterfactual is added to the set. We leave this problem for further investigation, focusing in the current article on validity, proximity, plausibility, and actionability.

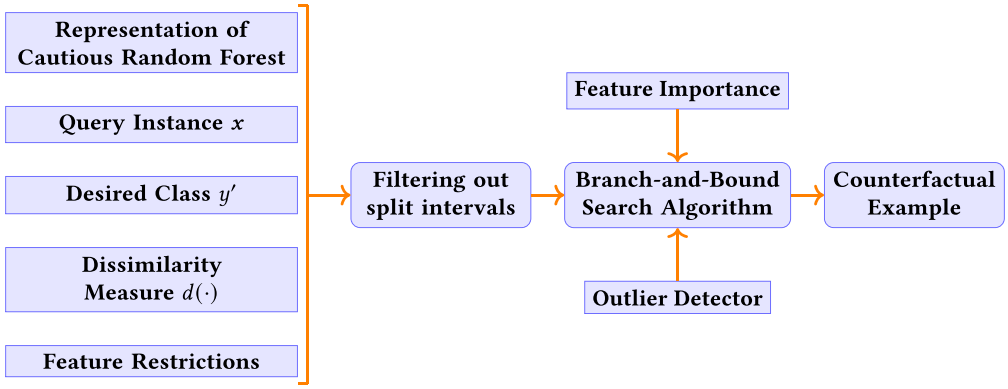


Fig. 2. Overview of our counterfactual example generation framework for CRFs.

4.1 Representation of Cautious Random Forests

The first step of our branch-and-bound strategy requires an efficient representation of a CRF. Let $h = \{h_t,\ t = 1, \ldots, T\}$ be a CRF consisting of T (imprecise) binary decision trees; for the sake of simplicity, and without loss of generality, we assume features to be numerical, i.e., $\boldsymbol{x} \in \mathbb{R}^M$.

4.1.1 Compact Representation of Trees. First, let us recall that each *decision region* associated with each leaf node in each tree can be seen as a multi-dimensional box, in which all instances share the same estimated posterior probability distribution.

Each leaf corresponds indeed to a partial region of the input space, determined by a series of split tests from the root to the leaf. Multiple splits based on the same feature in a root-leaf decision path can be summarized into a single interval. For instance, a leaf determined by $(\mathrm{X}_1 \leq 4 \wedge \mathrm{X}_2 > 3 \wedge \mathrm{X}_1 > 2)$ can be represented as a region defined by the multi-dimensional box $(\mathrm{X}_1, \mathrm{X}_2) \in]2, 4] \times]3, +\infty)$. If a feature X_j is never used, it can trivially be associated with the whole variable domain $\mathcal{X}_j$. Thus, the decision region $\mathcal{R}_{tl} = \mathcal{R}_{tl1} \times \cdots \times \mathcal{R}_{tlM}$ associated with a leaf η_{tl} of tree h_t can formally be defined as a box, i.e., a Cartesian product of lower-upper bounds $\mathcal{R}_{tlj}$ (one for each input variable X_j, and some of which may be trivial, i.e., may correspond to the variable domain bounds).

Each leaf η_{tl} is associated with an estimated probability distribution $\boldsymbol{\rho}_{tl}$—or in our case, a set of probability intervals $[\boldsymbol{\rho}]_{tl}$ such as defined in Equation (1). Therefore, $\eta_t = \{(\mathcal{R}_{tl}, [\boldsymbol{\rho}]_{tl}), l = 1, \ldots, \lambda_t\}$ is a compact representation of the tree h_t (with λ_t the number of leaves in the tree).

4.1.2 Decision Regions of a Random Forest. Computing efficiently the decision regions associated with the forest, i.e., the largest regions associated with the same decision, is far from trivial. We rather propose to compute an *inner partition* of the input space into such decision regions.

For a given feature X_j, let us consider the set of its associated distinct, nontrivial split values, observed across all trees in the forest:

$$\boldsymbol{v}_j = \{v_{ji},\ i = 1, \ldots, n_{vj}\}, \tag{11}$$

where n_{vj} is the number of split values for feature X_j. Without loss of generality, we assume the split values in $\boldsymbol{v}_j$ to be sorted in ascending order. Especially, we define v_{j0} and $v_{j(n_{vj}+1)}$ as the minimum and maximum of feature X_j.

These split values decompose each whole variable domain $\mathcal{X}_j$ into a set $\boldsymbol{I}_j$ of split intervals:

$$\boldsymbol{I}_j = \{I_{ji},\ i = 0, \ldots, n_{vj}\}, \quad \text{with } I_{ji} =]v_{ji}, v_{j(i+1)}]. \tag{12}$$

Hereafter, for the sake of simplification, we will use the generic notation $I_{ji} =]v_{ji}, w_{ji}]$ as notation for a split interval, i.e., its upper split value is written w_{ji}—assuming split values to be ordered, we have $w_{ji} = v_{j(i+1)}$. The total number of intervals for feature X_j is obviously $n_{Ij} = |\boldsymbol{I}_j| = n_{vj} + 1$.

Example 4.1 (Split Intervals of Random Forests). As an example, (a) and (b) display the decision regions obtained by a forest of two trees: decision boundaries are displayed in plain lines (variable splits being indicated in plain and dashed lines).

Since $\mathcal{X}_1 =]0, 5]$ and $\mathcal{X}_2 =]0, 4]$, and the nontrivial split values are $\boldsymbol{v}_1 = \{2, 4\}$ for X_1 and $\boldsymbol{v}_2 = \{1, 3\}$ for X_2, the sets of split intervals are $\boldsymbol{I}_1 = \{]0, 2],]2, 4],]4, 5]\}$ for X_1 and $\boldsymbol{I}_2 = \{]0, 1],]1, 3],]3, 4]\}$ for X_2.

Figure 3(c) displays the split intervals associated with the forest, obtained by computing the intersection of the intervals in $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$. ■

Given a set of split intervals $\{\boldsymbol{I}_j, j = 1, \ldots, M\}$, it should be clear that all instances $\boldsymbol{z} \in \boldsymbol{I}_1 \times \cdots \times \boldsymbol{I}_M$ share the same forest output $h(\boldsymbol{z})$: to that extent, this Cartesian product is thus a part of the decision region associated with that decision. Note, however, that the set of all Cartesian products of the split intervals is an *inner decomposition* of the input space into decision regions, as the same decision can be associated with several distinct Cartesian products. Nevertheless, aiming at computational efficiency rather than at compacity, we consider each Cartesian product as a decision region $\mathcal{R}_l \subset \mathcal{X}$ of the forest, with $l = 1, \ldots, \Lambda$ and Λ the total number of such decision regions (i.e., of Cartesian products).

4.1.3 Decision-Making. Let $\boldsymbol{x}$ be an instance to be processed by the forest. To efficiently process $\boldsymbol{x}$ and (as we will see later) generate counterfactual instances for $\boldsymbol{x}$, our procedure identifies the forest leaf in which $\boldsymbol{x}$ falls while keeping track of all associated tree leaves.

For a given variable X_j, we will refer by $I_j(\boldsymbol{x})$ to the split interval that contains the value x_j (the realization of X_j for $\boldsymbol{x}$). This split interval can easily and efficiently be identified using a simple search—with linear complexity in the number of intervals n_{Ij}, since these latter are ordered.

Identifying the set of leaves $\boldsymbol{\eta}(I_j(\boldsymbol{x}))$ associated with a given interval $I_j(\boldsymbol{x})$, i.e., whose associated decision regions have a nonempty intersection with it, can be achieved with linear complexity as well—this time in the total number of leaves in all trees; it is formally defined as:

$$\boldsymbol{\eta}(I_j(\boldsymbol{x})) = \left\{\eta_{tl} : \mathcal{R}_{tlj} \cap I_j(\boldsymbol{x}) \neq \emptyset, \quad t = 1, \ldots, T, \quad l = 1, \ldots, \lambda_t\right\}. \tag{13}$$

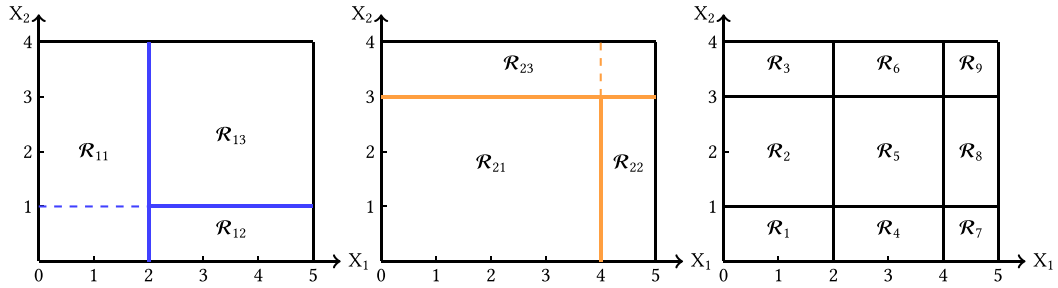
Consequently, the set of leaves which have a nonempty intersection with the forest leaf $\boldsymbol{\mathcal{R}}(\boldsymbol{x})$ in which $\boldsymbol{x}$ falls—i.e., the Cartesian product $I_1(\boldsymbol{x}) \times \cdots \times I_M(\boldsymbol{x})$—can be computed as:

$$\boldsymbol{\eta}(\boldsymbol{x}) := \boldsymbol{\eta}(\times_{j=1}^{M} I_j(\boldsymbol{x})) = \bigcap_{j=1}^{M} \boldsymbol{\eta}(I_j(\boldsymbol{x})). \tag{14}$$

In summary, for a given instance $\boldsymbol{x}$, a decision can be made by applying the following steps:

(1) compute the split intervals $I_{ji}(\boldsymbol{x})$ associated with $\boldsymbol{x}$;
(2) retrieve the sets of associated leaves $\boldsymbol{\eta}(I_j(\boldsymbol{x}))$ as defined by Equation (13);
(3) compute the set of leaves $\boldsymbol{\eta}(\boldsymbol{x})$ associated with the (forest) decision region $\boldsymbol{\mathcal{R}}(\boldsymbol{x})$ as per Equation (14);
(4) return the prediction made according to a predefined decision-making strategy, of the (imprecise-) probabilistic output computed according to a predefined aggregation operator, based on the set of leaves $\boldsymbol{\eta}(I_j(\boldsymbol{x}))$.

Note that once the forest output (decision and/or probabilistic output) has been computed for a given Cartesian product of split intervals, it can be stored in an adequate structure for future use,



(a) Split intervals for tree h_1 (b) Split intervals for tree h_2 (c) Split intervals of the forest

Fig. 3. Example of a forest of two decision trees: decision regions for each of the trees and of the forest.

Table 2. Example of Leaves in a Cautious Random Forest

η	Interval of values for X_1	Interval of values for X_2	$[\rho]_{tl1}$	$[\rho]_{tl2}$
η_{11}	$]0, 2]$	$]0, 4]$	$[0.8, 0.9]$	$[0.1, 0.2]$
η_{12}	$]2, 5]$	$]0, 1]$	$[0.4, 0.6]$	$[0.4, 0.6]$
η_{13}	$]2, 5]$	$]1, 4]$	$[0.1, 0.2]$	$[0.8, 0.9]$
η_{21}	$]0, 4]$	$]0, 3]$	$[0.7, 0.8]$	$[0.2, 0.3]$
η_{22}	$]\ 4, 5]$	$]0, 3]$	$[0.3, 0.4]$	$[0.6, 0.7]$
η_{23}	$]0, 5]$	$]3, 4]$	$[0, 0.2]$	$[0.8, 1]$

i.e., so as to avoid recomputing the same forest output for future instances located in the same region.

Example 4.2 (Representation of CRFs). We continue with the example displayed in Figure 3. Table 2 provides, for each leaf η_{tl} ($l = 1, 2$) in each tree h_t ($t = 1, 2$), the associated intervals of values for each of the two variables X_j ($j = 1, 2$) and probability intervals $[\rho]_{tlj}$. We note that regions $\mathcal{R}_1$ and $\mathcal{R}_2$ are actually associated with the same forest outputs, as well as regions $\mathcal{R}_6$ and $\mathcal{R}_9$. The leaves associated with each split interval are given in Table 3.

Assume a test instance $\boldsymbol{x} = (1, 2)$, is located in the split intervals $I_1(\boldsymbol{x}) = I_{11} =]0, 2]$ and $I_2(\boldsymbol{x}) = I_{22} =]1, 3]$. We can immediately see that it is located into region $\mathcal{R}(\boldsymbol{x}) = \mathcal{R}_2 = I_1(\boldsymbol{x}) \times I_2(\boldsymbol{x})$. The associated leaves are:

$$\boldsymbol{\eta}(\boldsymbol{x}) = \boldsymbol{\eta}(I_{11} \times I_{22}) = \{\eta_{11},\ \eta_{21},\ \eta_{23}\} \cap \{\eta_{11},\ \eta_{13},\ \eta_{21},\ \eta_{22}\} = \{\eta_{11},\ \eta_{21}\}.$$

If a decision is made based on the interval dominance principle (see Equations (2) and (3)), we obtain $h(\boldsymbol{x}) = \{c_1\}$. ■

4.2 Generation of Determinate Counterfactual Examples

Given a query instance $\boldsymbol{x}$ and a target (desired) class y', our approach to solving Equation (10) consists in generating a counterfactual example $\boldsymbol{x}'$ in the closest decision region to $\boldsymbol{x}$ in which the decision is y'. For this purpose, we need first to define a suitable dissimilarity measure between query instances and decision regions, and then to propose an efficient procedure for identifying the closest region to the query instance $\boldsymbol{x}$.

Table 3. For Each Feature, Set of Leaves Associated with Each Possible Split Interval

I_{1i}	$\boldsymbol{\eta}(I_{1i})$	I_{2i}	$\boldsymbol{\eta}(I_{2i})$
$]0,\ 2]$	$\{\eta_{11},\ \eta_{21},\ \eta_{23}\}$	$]0,\ 1]$	$\{\eta_{11},\ \eta_{12},\ \eta_{21},\ \eta_{22}\}$
$]2,\ 4]$	$\{\eta_{12},\ \eta_{13},\ \eta_{21},\ \eta_{23}\}$	$]1,\ 3]$	$\{\eta_{11},\ \eta_{13},\ \eta_{21},\ \eta_{22}\}$
$]4,\ 5]$	$\{\eta_{12},\ \eta_{13}, \eta_{22},\ \eta_{23}\}$	$]3,\ 4]$	$\{\eta_{11},\ \eta_{13},\ \eta_{23}\}$

4.2.1 Dissimilarity between Instances and Regions. Let $\mathcal{R} = \{]v_1,\ w_1]\times]v_M,\ w_M]\}$ be a (forest) multivariate decision region. The dissimilarity $d(\boldsymbol{x}, \mathcal{R})$ between $\boldsymbol{x}$ and $\mathcal{R}$ can be computed in various ways, for example, by degenerating the MAD or the weighted squared Euclidean distance to corresponding dissimilarities. Directly using Equation (5) or Equation (7) yields:

$$L_1^{mad}(\boldsymbol{x}, \mathcal{R}) = \sum_{j=1}^{M} \frac{d(x_j,]v_j,\ w_j])}{MAD^j}, \quad L_2^{std}(\boldsymbol{x}, \mathcal{R}) = \sum_{j=1}^{M} \frac{d(x_j,]v_j,\ w_j])^2}{STD^j}. \tag{15}$$

For the sake of simplicity, we will refer in the following to the adjusted dissimilarity along a single feature by $ad(\cdot)$:

$$ad(x_j,]v_j,\ w_j]) = \frac{d(x_j,]v_j, w_j])}{MAD^j}, \quad \text{or} \quad ad(x_j,]v_j,\ w_j]) = \frac{d(x_j,]v_j, w_j])^2}{STD^j}. \tag{16}$$

Computing either of the dissimilarities above requires to define the dissimilarity $d(x_j,]v_j,\ w_j])$ of a feature value to an interval, we propose:

$$d(x_j,]v_j,\ w_j]) = \begin{cases} 0 & \text{if } v_j < x_j \leq w_j, \\ v_j - x_j + \epsilon & \text{if } x_j \leq v_j, \\ x_j - w_j & \text{if } x_j > w_j. \end{cases} \tag{17}$$

The parameter ϵ makes sure that for a given feature X_j, whenever an instance falls on the lower interval bound v_j (which is assumed to be excluded from the interval), the dissimilarity will not be zero: it must be chosen small enough (and in particular such that $\epsilon < w_j - v_j$), yet greater than the machine epsilon. Therefore, we can define the dissimilarity of region $\mathcal{R}$ to instance $\boldsymbol{x}$ as the dissimilarity between $\boldsymbol{x}$ and the counterfactual instance $\boldsymbol{x}'$ for the query instance $\boldsymbol{x}$ in the decision region $\mathcal{R}$ by proceeding variable-wise, and pick, for $j = 1, \ldots, M$:

$$x'^j = \begin{cases} x_j & \text{if } v_j < x_j \leq w_j, \\ v_j + \epsilon & \text{if } x_j \leq v_j, \\ w_j & \text{if } x_j > w_j. \end{cases} \tag{18}$$

Note that if feature X_j is an integer, we can then define:

$$d(x_j,]v_j, w_j]) = \begin{cases} 0 & \text{if } v_j < x_j \leq w_j, \\ \lceil v_j + \epsilon \rceil - x_j & \text{if } x_j \leq v_j, \\ \lceil x_j - w_j \rceil & \text{if } x_j > w_j; \end{cases} \quad \text{and} \quad x'^j = \begin{cases} x_j & \text{if } v_j < x_j \leq w_j, \\ \lceil v_j + \epsilon \rceil & \text{if } x_j \leq v_j, \\ \lfloor w_j \rfloor & \text{if } x_j > w_j. \end{cases} \tag{19}$$

Figure 4 provides an example of the closest point generated in each region with respect to a query instance using Equations (18) and (19).

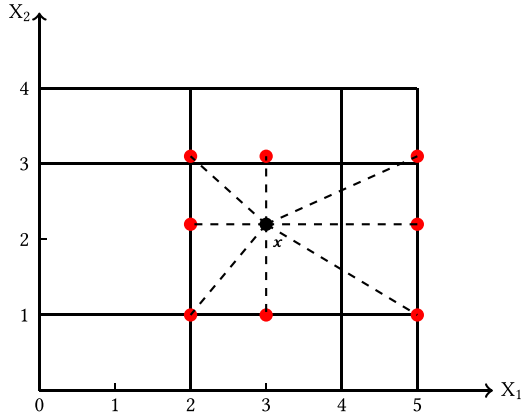


Fig. 4. The closest point in each region to query instance $\boldsymbol{x}$ assuming feature X_1 is integer and X_2 is continuous.

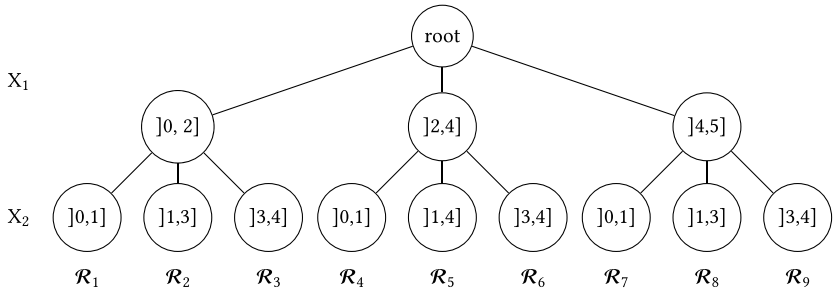


Fig. 5. Example of search tree for $\boldsymbol{x} = (1, 0.5)$ according to the forest illustrated in Figure 3(c).

4.2.2 Closest Decision Region to a Query Instance. We now turn to identifying the decision region closest to a query instance $\boldsymbol{x}$ while associated with a decision $y' \neq h(\boldsymbol{x})$, according to one of the dissimilarity measures L_1^{mad} or L_2^{std} proposed above. A straightforward way would be to compute the dissimilarity from $\boldsymbol{x}$ to all regions, the complexity of which is intractable due to the potentially high number of decision regions. To overcome this issue, our strategy starts from the region $\mathcal{R}(\boldsymbol{x})$ and expands the search to further regions $\mathcal{R}'$, exploring them by increasing dissimilarity $d(\boldsymbol{x}, \mathcal{R}')$, and stopping when a suitable region is identified.

To this end, we build a search tree, where each level corresponds to a feature and each node on a given level to a split interval for the corresponding feature. The path from the root to a leaf corresponds to a sequence of (at most) M split intervals defining a decision region. Assume that for each feature X_j, the split intervals are *sorted in ascending order according to their dissimilarity to* x_j. Then, the leftmost node on the level associated with X_j is the split interval $I_j(\boldsymbol{x})$ where $\boldsymbol{x}$ is located.

Figure 5 displays such a search tree, constructed from the forest in Figure 3(c). As mentioned before, exploring all decision regions in a forest is intractable due to its size being exponential. In the following, we propose several methods to reduce the complexity of the search procedure by filtering out regions that are known not to satisfy some constraints or to be suboptimal.

4.3 Speeding up the Search

The first way to accelerate the search is to consider feature restrictions. In some applications, features are immutable (they cannot be modified) or can only be increased or decreased. These constraints allow us to filter out some of the split intervals in the search tree.

In addition, we remark that a counterfactual example that satisfies all feature restrictions can be found or generated using simple heuristic methods. One popular such heuristic method is the so-called MO approach, which searches in the training set for the closest counterfactual example that satisfies feature restrictions. Alternatively, the **One-Feature-Changed Counterfactual (OFCC)** approach tries to vary the value of only one feature and keeps the remaining features unchanged to generate counterfactual examples. Experimental results [59] showed that OFCC can generate closer counterfactual examples than MO, thus achieving a smaller initial dissimilarity d_{sup}. We stress that using either of these methods provides us with an initial guess for the solution to Equation (10), associated with an upper bound d_{sup} on the dissimilarity between the query instance and potential counterfactual examples.

Our procedure aims at improving this initial guess, using d_{sup} to narrow down the search for the corresponding region. Together with the feature restrictions due to actionability, split intervals of features can be filtered as follows:

(1) If a feature X_j has no restrictions, split intervals whose dissimilarity to x_j are larger than d_{sup} should be filtered out, giving a set of remaining split intervals $\boldsymbol{I}_j^{\text{rem}} = \{I_{ji} : ad(x_j, I_{ji}) \leq d_{sup},\ i = 1, \ldots, n_{Ij}\}$.
(2) If a feature X_j is immutable, the value x'^j of any counterfactual instance should equal that of the query instance $\boldsymbol{x}$ (x_j): thus, all split intervals in $\boldsymbol{I}_j$ that do not contain x_j should be filtered out, leaving us with the remaining split intervals $\boldsymbol{I}_j^{\text{rem}} = \{I_{ji} : x_j \in I_{ji},\ i = 1, \ldots, n_{Ij}\}$—note that this step results in $\boldsymbol{I}_j^{\text{rem}}$ containing a single split interval, which amounts to removing a dimension in the search tree.
(3) If a feature X_j can be only increased, all split intervals whose upper bounds are smaller than x_j should be filtered out: the remaining split intervals are $\boldsymbol{I}_j^{\text{rem}} = \{I_{ji} : w_{ji} \geq x_j,\ ad(x_j, I_{ji}) \leq d_{sup},\ i = 1, \ldots, n_{Ij}\}$.
(4) If on the contrary a feature X_j can be only decreased, all split intervals whose lower bounds are larger than x_j should be filtered out: the remaining split intervals are $\boldsymbol{I}_j^{\text{rem}} = \{I_{ji} : v_{ji} \leq x_j, ad(x_j, I_{ji}) \leq d_{sup},\ i = 1, \ldots, n_{Ij}\}$.

In this filtering process, if no actionability constraints are considered, the algorithm will never eliminate the region that contains the closest counterfactual example. Conversely, if actionability constraints are taken into account, the counterfactual found may not be optimal in terms of proximity (since the closest counterfactual may have become non-admissible as per the additional constraints). Example 4.3 illustrates the filtering step on a simple example with a CRF trained on 2D binary classification data.

Example 4.3 (Filtering Procedure). Figure 6 illustrates the filtering procedure described above on an example where a CRF was trained on 2D binary classification data. The determinate decision regions (i.e., where the decision is a single class) are depicted in blue and orange, while the gray regions correspond to indeterminate decisions. The problem is to find a counterfactual of the orange class for a query instance $\boldsymbol{x}$ with an indeterminate prediction represented by the black point. The closest of these counterfactual examples is represented by the red point.

An initial counterfactual example (green point) obtained through the MO method gives an upper bound dissimilarity d_{sup} for the search procedure. Subsequently, if we use the Euclidean dissimilarity, the optimal counterfactual example must lie within the circle (or a hyper-sphere, in the case of high-dimensional data) centered on $\boldsymbol{x}$ with a radius of d_{sup}.

If there are no constraints on the features, then the search can be restricted to the textured area in the graph, consisting of 25 decision regions, rather than the entire feature space of 64 bins.

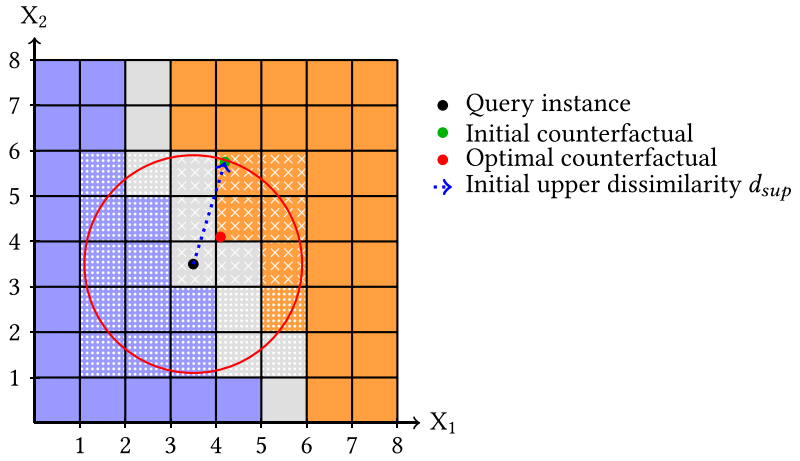


Fig. 6. An example of a CRF based on 2D data.

Additionally, if we assume that the values of the two features at hand cannot be decreased, the search range can be further narrowed down to the crosshatched area, consisting of only 9 bins. ■

Algorithm 1 compiles the filtering steps presented above and returns the subset of regions consisting of the remaining split intervals, where the closest counterfactual examples are to be generated. Via this filtering, the number of remaining split intervals of each feature will be significantly smaller than the original number.

4.4 Branch-and-Bound Search Procedure

Assume that, for a given query instance, the search procedure reached a node in the search tree. As the dissimilarity to a region of the space can be computed dimension-wise, it is easy to obtain a lower bound on the dissimilarity between the query and all the regions below this node. Given an upper bound d_{sup} on the normalized dissimilarity between the query and the counterfactual example, we can design a branch-and-bound algorithm able to stop the exploration of the nodes in the search tree whenever the dissimilarity of the remaining nodes is known to exceed the said upper bound. This allows to considerably decrease the time required to explore the search tree.

The branch-and-bound search strategy requires to keep track of the cumulative dissimilarity from the root to a current node in the search tree, so as to manage the forward and backward movements of the search. If the sub-tree of the current node is entirely explored, the algorithm should step back to the previous level and select a new sub-tree to explore. Conversely, if the sub-tree of the current node is incompletely explored and the cumulative dissimilarity from the root node to the current node does not exceed the upper bound distance d_{sup}, then the unexplored sub-tree should be prioritized for forward movement. Should instead the cumulative dissimilarity from the root node to the current node exceed d_{sup}, the exploration of all remaining sub-trees of its parent node should be terminated, given that those sub-trees to its right are guaranteed to be further away, and the cumulative dissimilarity of any node in these remaining sub-trees is consequently guaranteed to exceed d_{sup}.

Upon arriving at a leaf node of the search tree with a cumulative dissimilarity smaller than the current upper bound dissimilarity, the algorithm checks the prediction of the corresponding decision region. If the prediction matches the given desired class, a counterfactual example is

Algorithm 1: Filtering

Input: Query instance $\boldsymbol{x}$; initial upper bound dissimilarity d_{sup}; leaves of cautious random forest $\boldsymbol{\eta}$; feature restrictions; adjusted dissimilarity measure $ad(\cdot)$.

Output: Remaining split intervals $\boldsymbol{I}^{\text{rem}}$.

$\boldsymbol{I}^{\text{rem}} = \{\}$
for $j = 1, \ldots, M$ **do**
  $\boldsymbol{I}_j^{\text{rem}} = \{\}$
  if X_j *is immutable* **then**
    for $i = 1, \ldots, n_{Ij}$ **do**
      if $v_{ji} < x_j \leq w_{ji}$ **then**
        $\boldsymbol{I}_j^{\text{rem}} = \boldsymbol{I}_j^{\text{rem}} \cup I_{ji}$
  else if X_j *is only increasing* **then**
    for $i = 1, \ldots, n_{Ij}$ **do**
      if $w_{ji} \geq x_j$ **and** $ad(x_j, I_{ji}) \leq d_{sup}$ **then**
        $\boldsymbol{I}_j^{\text{rem}} = \boldsymbol{I}_j^{\text{rem}} \cup I_{ji}$
  else if X_j *is only decreasing* **then**
    for $i = 1, \ldots, n_{Ij}$ **do**
      if $v_{ji} \leq x_j$ **and** $ad(x_j, I_{ji}) \leq d_{sup}$ **then**
        $\boldsymbol{I}_j^{\text{rem}} = \boldsymbol{I}_j^{\text{rem}} \cup I_{ji}$
  else
    for $i = 1, \ldots, n_{Ij}$ **do**
      if $ad(x_j, I_{ji}) \leq d_{sup}$ **then**
        $\boldsymbol{I}_j^{\text{rem}} = \boldsymbol{I}_j^{\text{rem}} \cup I_{ji}$
  $\boldsymbol{I}^{\text{rem}} = \boldsymbol{I}^{\text{rem}} \cup \boldsymbol{I}_j^{\text{rem}}$
return $\boldsymbol{I}^{\text{rem}}$

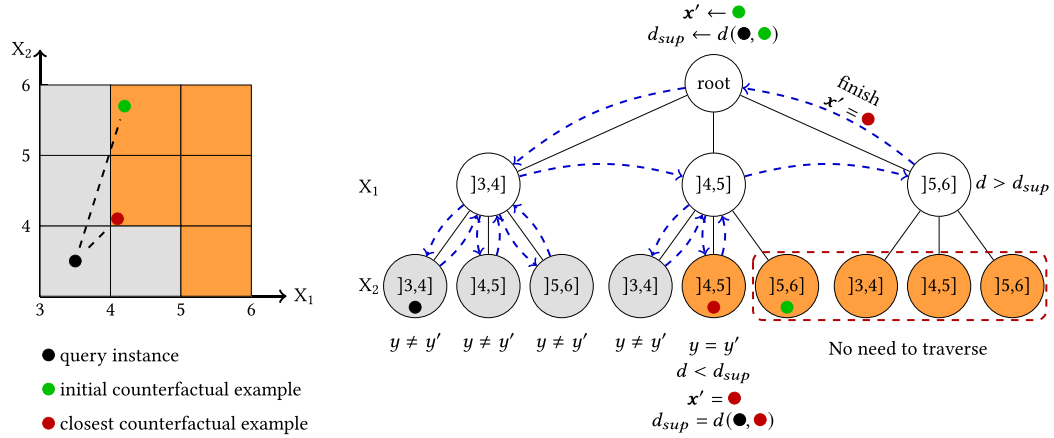
generated within this region according to Equations (18) and (19), and the upper bound dissimilarity d_{sup} is updated to the (smaller) current cumulative dissimilarity before resuming the search.

Example 4.4 illustrates this branch-and-bound search procedure as a continuation of Example 4.3.

Example 4.4 (Branch-and-Bound Search for the Closest Counterfactual Example). Figure 7(a) displays the decision regions in Example 4.3, narrowed down by using the upper bound dissimilarity d_{sup} and the actionability constraints that attributes can only be increased. The corresponding search tree is represented in Figure 7(b).

The search starts at the root and explores the regions such that $3 < \mathrm{X}_1 \leq 4$, corresponding to the leftmost node at the first level. The first child of this node corresponds to the region $(\mathrm{X}_1, \mathrm{X}_2) \in]3, 4] \times]3, 4]$, which is within the upper-bound dissimilarity but does not correspond to the desired prediction—the associated decision is indeterminate (the node contains the query instance), as are its two sibling nodes.

The search therefore goes back to the previous level in the tree and proceeds with exploring the regions such that $4 < \mathrm{X}_1 \leq 5$, corresponding to the children of the central node. The first child does not give the desired prediction. However, the second one, defined by $(\mathrm{X}_1, \mathrm{X}_2) \in]4, 5] \times]4, 5]$,



(a) Search regions after filtering

(b) Search tree and branch-and-bound search process

Fig. 7. Example of a search region (after application of the filtering procedure), and associated search tree with branch-and-bound search process for the closest counterfactual example.

produces the desired prediction and is within the upper-bound dissimilarity: a counterfactual example is therefore generated in this region, and the upper-bound dissimilarity is updated. We note that the third child is not visited, since the dissimilarity to $\boldsymbol{x}$ now exceeds the updated upper-bound dissimilarity.

The search goes back again to the first level to explore the rightmost node, i.e., the regions such that $5 < \mathrm{X}_1 \leq 6$. Since the dissimilarity of this node to the root exceeds the updated upper-bound dissimilarity as well, exploring the sub-trees rooted in this node is pointless: the search is therefore terminated by going back to the root node. ■

4.5 Plausible Counterfactual Generation

In order to guarantee the plausibility of the counterfactual instances produced by the algorithm, outlier detection techniques such as the **Local Outlier Factor (LOF)** [4, 11] and the *connectedness* of counterfactual examples to training data [34] can be employed in the process prior to updating $\boldsymbol{x}'$ and d_{sup}. In this article, we adopt the LOF as the indicator of plausibility since it is more sensitive than connectedness.

The LOF detects outliers by comparing the local density of a point to the densities of its neighbors, which assumes an outlier will have a significantly lower density than the points around it. To calculate the LOF score, for a given point $z \in \mathcal{X}$, the algorithm first finds its k-nearest neighbors, noted as $\mathcal{N}_k(z)$. Let $d_{(k)}(z)$ stand for the distance between an instance z and its kth nearest neighbor. LOF computes the "reachability distances" associated with each neighbor z' of z in $\mathcal{N}_k(z)$:

$$r_k(z, z') = \max(d_{(k)}(z'), d(z, z')); \tag{20a}$$

Note that $r_k(z, z')$ is formally not a distance since it does not satisfy symmetry. Then, LOF calculates the local reachability density of z as the inverse of the average reachability distance to z from its neighbors:

$$q_k(z) = \left(\frac{1}{|\mathcal{N}_k(z)|} \sum_{z' \in \mathcal{N}_k(z)} r_k(z, z') \right)^{-1}; \tag{20b}$$

Algorithm 2: Branch-and-bound Search for Counterfactual Examples

Input: Query instance $\boldsymbol{x}$; desired class y'; intial counterfactual example $\boldsymbol{x}'_{init}$; initial upper bound dissimilarity d_{sup}; sorted split intervals $\boldsymbol{I}$, leaves of CRF $\boldsymbol{\eta}$; adjusted dissimilarity measure along a single feature $ad(\cdot)$; local outlier factor $LOF(\cdot)$.

Output: Counterfactual example $\boldsymbol{x}'$.

$\boldsymbol{x}' = \boldsymbol{x}'_{init}$
$n_{I=}[n_{I1}, \ldots, n_{IM}]$
$n\_checked\_interval = [0, \ldots, 0]$ `// initial lists are all of` M `elements`
$dim\_dis = [\emptyset, \ldots, \emptyset]$
$dim\_leaves = [\emptyset, \ldots, \emptyset]$
$region = [\emptyset, \ldots, \emptyset]$
$cumu\_dis = 0$ `// cumulative dissimilarity from root to a search tree node`
$j = 1$
while *True* **do**
  if $j = 0$ **then**
    break
  else if $n\_checked\_intervals[j] = n_{Ij}$ **then**
    $n\_checked\_intervals[j] = 0$
    $dim\_dis[j] = 0$
    $j = j - 1$
  else
    $n\_checked\_intervals[j] = n\_checked\_intervals[j] + 1$
    $i = n\_checked\_intervals[j]$
    $interval = I_{ji}$
    $region[j] = interval$
    $dim\_dis[j] = ad(x_j, interval)$
    $cumu\_dis = sum(dim\_dist[1], \ldots, dim\_dist[j])$
    if $cumu\_dis > d_{sup}$ **then**
      $n\_checked\_intervals[j] = 0$
      $dim\_dis[j] = 0$
      $j = j - 1$
    else
      $dim\_leaves[j] = R(interval, j, \boldsymbol{\eta})$
      if $j{=}M$ **then**
        $leaves = dim\_leaves[1] \cap \cdots \cap dim\_leaves[M]$
        if $predict(leaves) = y'$ **then**
          $cf\_candidate = generate\_cf\_in\_region(\boldsymbol{x}, region)$ via Eq. (18) or (19)
          if $LOF(cf\_candidate)$ *is plausible* **then**
            $x' = cf\_candidate$
            $d_{sup} = cumu\_dist$
      else
        j = j + 1
return $\boldsymbol{x}'$

in a nutshell, $q_k(z)$ measures to which extent z can easily be reached from its neighbors, on average. Finally, the LOF score $s_k(z)$ is computed as the average relative reachability density of the neighbors of z from z:

$$s_k(z) = \frac{1}{|\mathcal{N}_k(z)|} \sum_{z' \in \mathcal{N}_k(z)} \frac{q_k(z')}{q_k(z)}. \tag{20c}$$

In our implementation, we followed the setting for LOF which is widely adopted in the anomaly detection literature. Specifically, we set the neighborhood size to $k = 20$, which is also the default value in the `scikit-learn` implementation and lies within the range of 10–30 recommended in [11, 23]. For the decision threshold, we considered a candidate counterfactual z implausible whenever its LOF score exceeded $s_k(z) = 1.5$, a value commonly used in practice and suggested in the original LOF paper [11]. This choice reflects the fact that points with $s_k(z) \approx 1$ have a density comparable to their neighbors, while significantly larger values (e.g., > 1.5) indicate outlier-like behavior.

Algorithm 2 formalizes the branch-and-bound search presented above, where we generate a single counterfactual candidate near the boundary of the decision region according to Equations (18) and (19). One might argue that such boundary candidates could occasionally be flagged as outliers by the plausibility filter, while interior points in the same region might not. Our method does not further search interior points for three reasons. First, due to the fine partitioning behavior of random forests near decision boundaries and our indeterminate predictions often occur near decision boundaries, interior points are typically very close to the boundary candidates and hence yield similar plausibility outcomes. Second, interior points necessarily have larger dissimilarity and are less aligned with our optimization goal. Third, explicitly searching interior points would incur substantial computational overhead with limited expected gain. This reflects an intended tradeoff between efficiency and completeness. Therefore, this design choice is an intended tradeoff between completeness and efficiency of counterfactual generation.

Algorithm 3, which calls Algorithm 2, presents the complete process for generating counterfactual examples. It features a reconstruction step, required to further ensure that the generated counterfactual examples are effective. The justification for this is twofold. First, the order of features in the search tree may be different from the original (arbitrary) one—for example, if they are ordered according to an importance score so as to increase the efficiency of the search procedure (see Section 5): then, the original feature order needs to be restored. Second, subsets of features might be subject to specific constraints: e.g., for a set of features resulting from one-hot encoding, there should be a single feature equal to one, the others being necessarily set to zero.

5 Increasing Counterfactual Generation Efficiency Using Feature Importance

5.1 Motivations

An important factor that significantly influences the efficiency of counterfactual example generation is the order of features within the search tree. Intuitively, modifying a given feature to generate a valid counterfactual example might require visiting the right-most child at the corresponding level in the search tree. Therefore, placing *decisive features* (i.e., which probably require to be modified to reach a counterfactual example) at the bottom of the search tree, and unimportant features near its root, increases the chance of identifying valid counterfactuals earlier, which allows for reducing the search space more quickly. It should be noted that the reordering of features will never eliminate the decision region with the minimal dissimilarity to the query instance, thus preserving the completeness of the search.

Algorithm 3: Counterfactual Generation

Input: Query instance $\boldsymbol{x}$; classes $\mathcal{Y} = \{c_1, c_2\}$; leaves of cautious random forest $\boldsymbol{\eta}$; dissimilarity measure $d(\cdot)$; feature restrictions FR; feature ordering FO; local outlier factor $LOF(\cdot)$.

Output: Counterfactuals $cfs\{\boldsymbol{x}^{c_1}, \boldsymbol{x}^{c_2}\}$.

$cfs = \{\}$
for $c_k \in \mathcal{Y}$ **do**
  Initialize $\boldsymbol{x}_{init}^{c_k}$ via MO or OFCC method for class c_k
  $d_{sup} = d(\boldsymbol{x}, \boldsymbol{x}^{c_k})$
  $\boldsymbol{I}$ = Filtering($\boldsymbol{x}$, d_{sup}, $\boldsymbol{\eta}$, FR, $d(\cdot)$) // Algorithm 1
  Sort $\boldsymbol{I}_j$ according to $d(\boldsymbol{x}, \boldsymbol{I}_{ji})$, $j = 1, \ldots, M$, $i = 1, \ldots, n_{Ij}$
  Reorder features in $\boldsymbol{x}$, SI, and $\boldsymbol{\eta}$ according to feature ordering FO
  $\boldsymbol{x}^{c_k}$ = Branch-and-Bound Search($\boldsymbol{x}$, c_k, $\boldsymbol{x}_{init}^{c_k}$, d_{sup}, $\boldsymbol{I}$, $\boldsymbol{\eta}$, $d(\cdot)$, $LOF(\cdot)$) // Algorithm 2
  $\boldsymbol{x}^{c_k}$ = Reconstruction($\boldsymbol{x}^{c_k}$, FO, FR)
  $cfs = cfs \cup \boldsymbol{x}^{c_k}$
return cfs

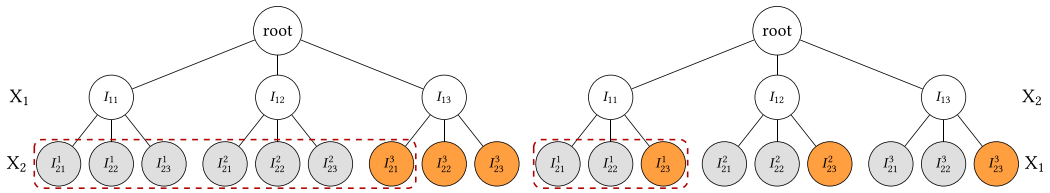


Fig. 8. Two search trees with different orderings of features: (X_1, X_2) (left) and (X_2, X_1) (right). The procedure aims at generating a counterfactual in an orange region. Feature X_1 is decisive. The right-most tree incorporates this latter at the bottom, which makes it possible to reach an appropriate generation region (and thus to decrease the upper-bound dissimilarity d_{sup}) earlier during the search, thereby increasing the chance to terminate the search more quickly.

As an illustration, assume feature X_1 is decisive. In tree t_1, the feature order is (X_1, X_2), and the procedure must visit seven left-most leaves before generating the first valid counterfactual example. In contrast, in tree t_2, the feature order is (X_2, X_1), and only three left-most leaves need to be explored to obtain the first valid counterfactual, thereby reducing the upper bound dissimilarity more quickly and increasing the chance of skipping subsequent leaves (see Figure 8).

This observation motivates the use of feature importance to guide the ordering of features in the search, which is crucial for improving the practical efficiency of the proposed algorithm.

5.2 Feature Importance-Based Strategy

In previous works, the concept of feature importance has been widely used for assessing which features may need to be modified to generate counterfactual examples. For example, Keane et al. proposed in [29] to identify the features to be modified based on the counterfactual examples of data points close to the query instance—as per the principle that similar instances should require similar modifications to obtain counterfactual examples. In [45, 56], the authors proposed to modify features that contribute against the desired predictions, i.e., features with negative SHAP values associated with the query instance and the desired prediction.

We propose to establish the order of features in the search tree by evaluating their importance with respect to the resolution of prediction indeterminacy, expecting features with higher importance to be more likely to be modified during the generation of counterfactual examples: such features should therefore be positioned closer to the bottom of the search tree. Given a list of feature importance values $\mathbf{\Phi} = \{\phi^1, \ldots, \phi^M\}$, we define the feature ordering $\preceq_{\mathbf{\Phi}}$ as:

$$\mathrm{X}_j \preceq_{\mathbf{\Phi}} \mathrm{X}_{j'} \text{ if } \phi^j \leq \phi^{j'},\ \forall j, j' \in \{1, \ldots, M\},\ j \neq j'. \tag{21}$$

We finally note that, in addition to helping determine the order of features in the search tree, the feature importance values assessed constitute an explanation in itself, by pointing out the features that are expected to be important in the counterfactual generation process.

5.3 Existing Feature Importance Measures

Many strategies have been proposed based on which we can estimate the feature importance values in $\mathbf{\Phi}$. We detail some of the most prominent ones below, and we point out how they can be adapted to rank the features in the search tree. We made a distinction between local measures, and notably the LIME [46] and SHAP [38] methods, which focus on the contribution of features for a particular instance and prediction; and global measures, which take all instances into account. These latter may notably be helpful for the purpose of understanding the model.

5.3.1 Local Feature Importance. For a given query instance $\boldsymbol{x}$ and model h to be explained, the LIME approach aims at locally approximating h by an interpretable surrogate model ξ chosen in a family $\mathcal{H}$ of models (e.g., linear regression models) [46]. For this purpose, LIME creates a new dataset by generating samples around $\boldsymbol{x}$ and collects the corresponding predictions provided by h. A new interpretable model, e.g., linear regression, is trained using the new dataset in which each sample is then weighted according to its proximity to $\boldsymbol{x}$. The best surrogate model can formally be defined as follows:

$$\xi(\boldsymbol{x}) = \arg\min_{h' \in \mathcal{H}} \mathcal{L}(h, h', \boldsymbol{\varepsilon}(\boldsymbol{x})) + O(h'), \tag{22}$$

where $\mathcal{L}(\cdot)$ is a loss function measuring the fidelity of h' to h (e.g., mean squared error), $O(\cdot)$ measures the model complexity, and $\boldsymbol{\varepsilon}(\boldsymbol{x})$ defines a neighborhood of $\boldsymbol{x}$ in which the approximation is sought. In practice, LIME only optimizes the loss part while the complexity of the model (number of features) is fixed by the user. In our case, the outputs of model h and h' are replaced by a measure of prediction indeterminacy—see Equation (25a) or Equation (25b) below.

Unlike LIME, which learns an approximate model, the SHAP approach makes use of the Shapley values [50], a concept from cooperative game theory that measures the contribution of each player to the total payoff of a game. The basic idea behind SHAP is to assign a score $\phi(j \mid h, \boldsymbol{x})$ to each feature X_j that measures its contribution to the model prediction for a given instance $\boldsymbol{x}$ and classifier h. This score is based on the difference between the model prediction for the instance with and without the feature X_j. More precisely, the SHAP score for feature X_j and instance $\boldsymbol{x}$ is defined as:

$$\phi(j \mid \boldsymbol{x}, h) = \sum_{S \subseteq \{1,2,\ldots,M\} \setminus \{j\}} \frac{|S|!(M - |S| - 1)!}{M!} (f_h(\boldsymbol{x}^{S \cup \{j\}}) - f_h(\boldsymbol{x}^S)), \tag{23}$$

where M is the total number of features, S is a subset of the features excluding feature X_j, $f_h(\boldsymbol{x}^S)$ is a payoff associated with model h for instance $\boldsymbol{x}$ with only features in S, and $f_h(x^{S \cup \{j\}})$ is the one with feature X_j added to S. Again, in our case, the payoff function $f_h(\cdot)$ is a measure of prediction indeterminacy, defined either by Equation (25a) or by Equation (25b) below.

Intuitively, the SHAP score measures the average marginal contribution of feature X_j across all possible subsets of features that do not contain X_j. It can be interpreted as the contribution of

feature X_j to the model output for $\boldsymbol{x}$. In order to estimate the SHAP score, Lundberg et al. proposed Kernel SHAP [38], which is a model-agnostic approach. Kernel SHAP connects LIME and Shapley values, i.e., it is a kind of LIME where the proximity measurement $\boldsymbol{\varepsilon(x)}$ in LIME is no longer based on distance but replaced by the SHAP kernel.

5.3.2 Global Feature Importance. When growing a decision tree, upon adding a new node, the best split is determined so as to maximize the decrease in impurity. For a given feature, the **Mean Decrease in Impurity (MDI)** is the average decrease in the impurity of the nodes where the feature is selected as split feature, weighted by the proportion of samples in the nodes [10, 35]. Intuitively, features achieving a great decrease of impurity over a tree are more discriminative and are therefore important for the separation of the feature space. In a random forest, the MDI can be defined as the average MDI across all trees. MDI is a model-specific method, only suitable for tree-based models. Features with a large MDI generally appear in the shallow layers of the trees, while features with a small MDI often appear at the bottom, corresponding to regions of the input space where the different classes are more likely to be mixed—which corresponds to the main cause of indeterminacy for a CRF. Therefore, features with a higher MDI are more likely to be helpful in resolving indeterminacy.

Since Shapley values are considered to be consistent, this local feature importance measure can be used to construct a global measure. The SHAP feature importance measure (SHAP-FI) is calculated by averaging the absolute Shapley values per feature across all instances with indeterminate predictions:

$$\Phi(j \mid h) = \frac{1}{n_{\text{indet}}} \sum_{\boldsymbol{x} \mid |h(\boldsymbol{x})|>1} |\phi(j \mid h, \boldsymbol{x})|, \tag{24}$$

where n_{indet} is the number of instances with indeterminate predictions in the dataset and $\phi(j \mid h, \boldsymbol{x})$ is defined by Equation (23). Besides, SHAP-FI can also provide a summary plot for all features to illustrate the relation between SHAP values and feature effects, which enables us to better understand the model dependency on features.

In traditional classification problems, the PFI approach measures the decrease in the prediction accuracy of the model after the feature values have been swapped. It is also referred to as the mean decrease in accuracy [40]. Since machine learning models learn the association between input features and outputs, if there is a significant decrease in the accuracy of the prediction when a feature in the dataset is perturbed, it indicates that the prediction of the model is highly dependent on this feature [10]. PFI is a model-agnostic method, and the performance measurement can also be different from the accuracy.

5.4 Proposed Feature Importance Assessment Strategy for Indeterminacy Resolution

5.4.1 Measuring Prediction Indeterminacy. Feature importance must be associated with a prediction evaluation metric, such as the accuracy in precise classification problems. In our case, we focus on the indeterminacy of predictions. Thus, we propose two measures of indeterminacy. The first one, written Imp_1, indicates whether the prediction is indeterminate or not. The second one, denoted by Imp_2, is a measure of uncertainty based on the prediction intervals. If the output $h(\boldsymbol{x})$ for an instance $\boldsymbol{x}$ consists of a set of classes, we propose to define Imp_1 as follows:

$$\text{Imp}_1(h, \boldsymbol{x}) = \begin{cases} 0 & \text{if } |h(\boldsymbol{x})| = 1, \\ 1 & \text{otherwise.} \end{cases} \tag{25a}$$

If we rather consider interval-valued outputs ($I_1 = [bel_1(\boldsymbol{x}),\ pl_1(\boldsymbol{x})]$ and $I_2 = [bel_2(\boldsymbol{x}),\ pl_2(\boldsymbol{x})]$), following the definition in [26], we may define the indeterminacy measure Imp_2 as:

$$\mathrm{Imp}_2(h, \boldsymbol{x}) = \min(pl_1(\boldsymbol{x}), pl_2(\boldsymbol{x})). \tag{25b}$$

By duality, the uncertainty quantification can be calculated either using I_1 or I_2 due to the relationship $pl_1(\boldsymbol{x}) = 1 - bel_2(\boldsymbol{x})$ and $pl_2(\boldsymbol{x}) = 1 - bel_1(\boldsymbol{x})$, i.e., $\mathrm{Imp}_2(h, \boldsymbol{x}) = \min(pl_1(\boldsymbol{x}), 1 - bel_1(\boldsymbol{x}))$ or $\mathrm{Imp}_2(h, \boldsymbol{x}) = \min(1 - bel_2(\boldsymbol{x}), pl_2(\boldsymbol{x}))$.

Remark 1. Since $pl_1(\boldsymbol{x}) = pl_1(\boldsymbol{x}) + bel_1(\boldsymbol{x}) - bel_1(\boldsymbol{x})$ and $1 - bel_1(\boldsymbol{x}) = 1 - bel_1(\boldsymbol{x}) + pl_1(\boldsymbol{x}) - pl_1(\boldsymbol{x})$, we have:

$$\begin{aligned}
\mathrm{Imp}_2(h, \boldsymbol{x}) &= \min(pl_1(\boldsymbol{x}), 1 - bel_1(\boldsymbol{x})) \\
&= \min(bel_1(\boldsymbol{x}) + pl_1(\boldsymbol{x}) - bel_1(\boldsymbol{x}), 1 - pl_1(\boldsymbol{x}) + pl_1(\boldsymbol{x}) - bel_1(\boldsymbol{x})) \\
&= \min(bel_1(\boldsymbol{x}), 1 - pl_1(\boldsymbol{x})) + pl_1(\boldsymbol{x}) - bel_1(\boldsymbol{x}) \\
&= \min(bel_1(\boldsymbol{x}), bel_2(\boldsymbol{x})) + (pl_1(\boldsymbol{x}) - bel_1(\boldsymbol{x})).
\end{aligned}$$

The first part in this decomposition has been linked to aleatoric uncertainty, and the second part, representing the length of the probability intervals I_1 and I_2, to the epistemic uncertainty in the prediction.

Remark 2. According to the decision strategy in Equation (3), all determinate predictions ($bel_1(\boldsymbol{x}) > 0.5$ or $bel_2(\boldsymbol{x}) > 0.5$) yield a value $\mathrm{Imp}_2 < 0.5$, and indeterminate predictions ($bel_1(\boldsymbol{x}) \leq 0.5$ and $bel_2(\boldsymbol{x}) \leq 0.5$) yield a value $\mathrm{Imp}_2 \geq 0.5$. In addition, for any $k, l \in \{1, 2\}$ with $l \neq k$, we have that $\mathrm{Imp}_2 \to 0$ when $bel_k(\boldsymbol{x}) \to 1$ or $pl_k(\boldsymbol{x}) \to 0$, and in contrast $\mathrm{Imp}_2 \to 1$ when both $bel_k(\boldsymbol{x}) \to 0$ and $pl_k(\boldsymbol{x}) \to 1$: in these cases, Imp_2 coincides with Imp_1.

5.4.2 Local Permutation Feature Importance. Let $\boldsymbol{x}$ be a query instance associated with an indeterminate model output; assume that the joint distribution $f_{\mathbf{X}}$ of the underlying random vector $\mathbf{X}$ is known (or has been estimated).

Let $\boldsymbol{x}^{\sim j} \in \mathcal{X}$ be a vector obtained from $\boldsymbol{x}$ by modifying feature X_j while keeping all others unchanged, i.e., the jth element x_j of $\boldsymbol{x}$ was replaced by a new value z_j: this modification can be seen as a sampling according to the conditional distribution:

$$f_{\mathbf{X}^{\sim j}}(z_j) := f_{\mathrm{X}_j \mid \boldsymbol{x}^{-j}}(z_j) = \mathbb{P}(\mathrm{X}_j = z_j \mid \mathrm{X}_{j'} = x_{j'}, \forall j' \neq j), \tag{26a}$$

where $\boldsymbol{x}^{-j}$ is the sub-vector consisting of all elements of $\boldsymbol{x}$ except the jth one. In general, the conditional distribution $f_{\mathbf{X}^{\sim j}}$ is difficult to calculate analytically; a notable exception is the Gaussian case, where any conditioning of a Gaussian random vector by a sub-vector is Gaussian, with parameters that can be computed analytically.

Note that if the modification of $\boldsymbol{x}$ is to be performed "toward a given class k," i.e., the instance should be modified so as to transform an indeterminate prediction into a desired one, the class-conditional distributions for the feature to be modified, i.e., we should replace (26a) with:

$$f_{\mathbf{X}^{\sim j} \mid k}(z_j) := f_{\mathrm{X}_j \mid \boldsymbol{x}^{-j}, Y=k}(z_j) = \mathbb{P}(\mathrm{X}_j = z_j \mid Y = k, \mathrm{X}_{j'} = x_{j'}, \forall j' \neq j), \tag{26b}$$

where the considered class $Y = k$ matches the desired prediction $h(\boldsymbol{x})$.

Given a test instance $\boldsymbol{x}$, we can define the *local PFI* of X_j as the expected reduction of indeterminacy induced by modifying feature value x_j only:

$$\phi(j \mid h, \boldsymbol{x}) = \mathbb{E}_{\mathbf{X}^{\sim j}}\left[\mathrm{Imp}(h(\boldsymbol{x})) - \mathrm{Imp}(h(\mathbf{X}^{\sim j}))\right] = \mathrm{Imp}(h(\boldsymbol{x})) - \mathbb{E}_{\mathbf{X}^{\sim j}}\left[\mathrm{Imp}(h(\mathbf{X}^{\sim j}))\right]. \tag{27}$$

Again, this expectation can be computed according to the class-conditional distribution (26b) instead of (26a), should a modification be made so as to obtain a specific decision, i.e., a given class k.

Note that even if the conditional distribution (26a) or (26b) can be computed or approximated efficiently, the expectation (27) is generally difficult to compute. Therefore, we propose in practice to estimate it by averaging the reduction of indeterminacy over the instances similar to $\boldsymbol{x}$ except for value x_j, i.e., instances obtained by replacing the jth value of $\boldsymbol{x}$ by values sampled according to the conditional distribution of the random variable $\mathbf{X}^{\sim j}$. Alternatively, provided a sufficiently large validation set is available, we can also estimate the local feature importance based on the samples similar to $\boldsymbol{x}$ except for value x_j.

5.4.3 Global Permutation Feature Importance. The PFI measure defined above can be used to assess feature importance at the population level, so as to globally explain the relationship between the features and the indeterminacy of the CRF model. Since our purpose is to resolve the indeterminacy of set-valued predictions, we focus on observations $\boldsymbol{x}$ with indeterminate predictions, i.e., $\operatorname{Card} h(x) > 1$.

We define the global feature importance of X_j as the expected gain of the reduction of indeterminacy associated with classifier outputs based on the conditional distribution $f_{\mathbf{X} \mid \operatorname{Card} h(\mathbf{X})>1}$:

$$\Phi(j \mid h) = \mathbb{E}_{\mathbf{X} \mid \operatorname{Card} h(\mathbf{X})>1} [\phi(j \mid h, \boldsymbol{x})], \tag{28}$$

where $\phi(j \mid h, \boldsymbol{x})$ is the measure of local feature importance of X_j associated with a specific observation $\boldsymbol{x}$, as defined in Equation (27), in which an appropriate class-conditional distribution in (27) can be used so as to compute an expected gain specific to a desired prediction $h(\boldsymbol{x}) = k$ as mentioned above.

The theoretical calculation of the global feature importance is generally difficult. However, Equation (28) can be practically estimated by replacing the expectation with an empirical average calculated on the instances of a dataset (e.g., test or validation set) for which the model outputs are indeterminate.

5.5 Complexity Analysis of Our Proposed Counterfactual Example Generation Framework

We now analyze the computational complexity of the proposed branch-and-bound approach. Suppose each of the M features is partitioned into N intervals by the random forest. Then the worst-case complexity of the search is exponential, i.e., $O(N^M)$. Even if filtering irrelevant intervals reduces the effective number of partitions per feature to $N_* \ll N$ (see Section 4.3), the worst-case complexity remains $O(N_*^M)$. This exponential bound is inherent to exact search methods.

In practice, not all features need to be modified to reach a valid counterfactual. Let $J < M$ denote the number of decisive features that must be modified. If these decisive features are located deeper in the search tree, the effective complexity reduces to $O(N_*^J)$. This exponential dependence on J rather than M represents a significant gain, particularly when $J \ll M$.

From the practical point of view, we tested our framework with CRFs consisting of 100 trees, each trained to maximum depth. This results in very fine partitions of the feature space, already representing a large-scale scenario. The reported runtimes in Section 6 demonstrate that the algorithm remains tractable under such conditions. In addition, as shown in Table 7, the generated counterfactuals typically involve modifications to only 1–3 features. This observation indicates that J is quite small in practice, which greatly reduces the effective complexity relative to the theoretical worst case.

Table 4. Datsets Used in Experiments

Data name	Abbreviation	n-feature	n-sample	Majority class (%)
Adult	ADLT	11	45,222	75.22
Biodeg	BIOD	41	1,053	66.38
Compas	COMP	6	2,652	53.13
German	GERM	24	1,000	70.00
Heloc	HELO	23	10,459	52.19
Liver	LIVR	6	345	57.97
Mammographic	MAMO	5	830	51.45
Pima	PIMA	8	768	65.10
Spam	SPAM	57	4,594	60.69
Wine	WINE	11	1,599	53.47

While the scalability of a branch-and-bound procedure in very high-dimensional domains is acknowledged as a challenge, the combination of (i) filtering by preprocessing, (ii) ordering decisive features based on expected feature importance, and (iii) the empirical sparsity of counterfactual changes ensures that the search remains feasible in practice for the datasets considered.

We stress here that additional advanced implementation strategies may be considered to further reduce the computational complexity of our approach. For instance, spatial indexing structures such as KD-trees are often used to accelerate nearest-neighbor queries in continuous low-dimensional spaces. While promising in principle, their applicability may be questioned in our setting for two reasons: (i) tabular data typically involve mixed feature types (continuous, ordinal, and categorical), where KD-trees offer little benefit; and (ii) our branch-and-bound procedure already exploits the axis-aligned partitions induced by the forest, which plays a role similar to space partitioning. Therefore, we chose not to investigate the integration of spatial indexing into our implementation. Nevertheless, in applications with continuous features and low dimensionality, specific structures such as KD-trees could be explored as a further optimization of our counterfactual generation approach.

6 Experiments

We conducted two experiments to establish the interest of our approach. First, we compared our counterfactual example generation method with other methods by evaluating the effectiveness of the generated counterfactual examples and their efficiency in handling large-scale problems. Second, we demonstrated the efficiency gain for the branch-and-bound search by ordering the features according to their importance with respect to the reduction of indeterminacy.

All of the aforementioned experiments were performed on 10 distinct binary classification datasets sourced from the UCI repository. Table 4 recalls details on these datasets, including the number of features and samples.

In our experiments, we used CRFs with 100 decision trees, all trained to the maximum purity (with a purity of 1 at each leaf). The parameter of the imprecise Dirichlet model was set to $s = 2$. The random state was fixed to 42 for all experiments. Below, we report average metric values computed over four-fold cross-validations and their fold-wise STD. By this setting, we had 25% test data in each fold to have enough indeterminate predictions. All of our experiments were conducted on a Windows 10 system, powered by an Intel Ultra-7-165H CPU and 32 GB of RAM.

Table 5. Comparison of Different Counterfactual Generation Approaches in Terms of L_2^{std} Dissimilarity

Data	MO	DisCERN	LIRE	OFCC	Ours
ADLT	2.9544 ± 1.1706	2.3369 ± 1.2030	2.2740 ± 0.9960	1.7801 ± 0.8391	**1.7123 ± 0.8548**
BIOD	3.1315 ± 0.5267	0.9909 ± 0.1652	2.3432 ± 0.3674	0.6444 ± 0.1667	**0.6177 ± 0.1525**
COMP	1.1348 ± 0.1590	1.0120 ± 0.1293	1.0647 ± 0.1798	1.0296 ± 0.1149	**0.6487 ± 0.0973**
GERM	4.3264 ± 0.1291	3.0958 ± 0.1671	2.9975 ± 0.0980	2.1815 ± 0.1919	**1.8523 ± 0.1322**
HELO	7.6164 ± 0.2360	4.7660 ± 0.3593	6.9940 ± 0.1660	1.6802 ± 0.2295	**1.5423 ± 0.2014**
LIVR	4.1412 ± 0.4126	3.1374 ± 0.4070	3.7520 ± 0.3892	1.5609 ± 0.6758	**1.1803 ± 0.5746**
MAMO	1.5090 ± 0.8999	1.4339 ± 0.9204	1.1032 ± 0.8564	1.2162 ± 0.5430	**0.8376 ± 0.4950**
PIMA	4.6446 ± 0.1405	3.0986 ± 0.2662	4.2328 ± 0.1640	1.4915 ± 0.2630	**1.3176 ± 0.2450**
SPAM	3.7464 ± 0.4694	1.2830 ± 0.0701	2.6797 ± 0.3356	0.1316 ± 0.0735	**0.1302 ± 0.0745**
WINE	1.5406 ± 0.0715	0.8147 ± 0.0427	1.2937 ± 0.0572	0.4081 ± 0.1405	**0.2585 ± 0.0596**

Table 6. Comparison of Different Counterfactual Generation Approaches in Terms of L_1^{mad} Dissimilarity

Data	MO	DisCERN	LIRE	OFCC	Ours
ADLT	2.1763 ± 0.2049	1.4225 ± 0.1187	1.7010 ± 0.1371	0.4385 ± 0.0285	**0.4267 ± 0.0316**
BIOD	19.6806 ± 8.3335	3.8764 ± 0.8769	13.6125 ± 5.0628	1.8555 ± 0.6457	**1.6416 ± 0.5035**
COMP	2.1317 ± 0.7937	1.4352 ± 0.5168	2.1857 ± 1.0048	1.2389 ± 0.5602	**0.8790 ± 0.4585**
GERM	12.2793 ± 1.1329	5.6872 ± 0.4590	7.9907 ± 0.5206	2.8918 ± 0.2080	**2.6376 ± 0.1954**
HELO	9.1232 ± 0.5293	3.2291 ± 0.3028	8.1836 ± 0.2956	0.9090 ± 0.1847	**0.8603 ± 0.1878**
LIVR	3.3464 ± 0.3249	1.8550 ± 0.3322	2.9197 ± 0.3128	0.6230 ± 0.2747	**0.5652 ± 0.2664**
MAMO	2.2311 ± 1.4992	2.1199 ± 1.5405	1.6697 ± 1.4313	1.3309 ± 0.8715	**1.1444 ± 0.8559**
PIMA	3.8787 ± 0.1511	1.9301 ± 0.2127	3.4428 ± 0.1412	0.5817 ± 0.0805	**0.5603 ± 0.0768**
SPAM	29.0034 ± 6.1094	5.3566 ± 0.9527	18.9356 ± 3.6862	0.4882 ± 0.3137	**0.4342 ± 0.3203**
WINE	5.3058 ± 0.1610	2.2654 ± 0.2382	4.5340 ± 0.1783	0.6686 ± 0.1909	**0.5087 ± 0.0892**

6.1 Comparison of Different Counterfactual Generation Methods

Here, we compare different counterfactual example generation methods in terms of the properties of counterfactual examples, including dissimilarity (L_2^{std} and L_1^{mad}), sparsity (L_0), plausibility (whether the generated counterfactual is an outlier or not), and efficiency (time cost); all these quality measures are reported in Tables 5–9, where best results are highlighted in bold. Note that since all methods implemented here generate valid counterfactuals by design, the validity is not reported. The compared methods in this experiment are:

(1) MO, which searches the closest counterfactual example in the training set;
(2) DisCERN [56], which replaces values of features ordered by feature importance with the corresponding feature values from MO until valid counterfactuals are returned;
(3) OFCC [59], which varies the value of only one feature and keeps the remaining features unchanged to generate counterfactual examples;
(4) LIRE [12], a method tailored to random forests, which generates counterfactual examples by considering only the leaf intersections that contain at least one training sample.

We note that when choosing baselines for empirical comparison, we deliberately restricted our attention to methods that guarantee validity while maintaining practical efficiency. Although several other counterfactual generators for tree ensembles have been proposed, many of them

Table 7. Number of Modified Features (L_0-*norm* or Sparsity) When Optimizing L_2^{std} and L_1^{mad}, Respectively

Data	MO	DisCERN	LIRE	OFCC	Ours
When optimizing L_2^{std}					
ADLT	3.2146 ± 0.1730	1.8479 ± 0.0804	3.3000 ± 0.1621	**1.0000 ± 0.0000**	1.1750 ± 0.0613
BIOD	18.5184 ± 0.5158	6.4423 ± 0.1601	16.0689 ± 0.4861	**2.6383 ± 0.3827**	2.6480 ± 0.2746
COMP	1.9542 ± 0.0570	1.7292 ± 0.0488	2.5146 ± 0.1283	**1.0417 ± 0.0300**	1.6271 ± 0.0997
GERM	8.4250 ± 0.3124	4.7938 ± 0.2870	9.0625 ± 0.2036	**2.1500 ± 0.1595**	2.4333 ± 0.0854
HELO	16.1625 ± 0.1799	6.5437 ± 0.6348	16.9042 ± 0.1415	**1.7375 ± 0.3119**	2.0063 ± 0.3534
LIVR	5.2355 ± 0.1017	2.8107 ± 0.3008	5.1993 ± 0.1473	**1.0435 ± 0.0435**	1.9370 ± 0.1021
MAMO	1.9036 ± 0.0664	1.6631 ± 0.1384	1.9878 ± 0.0477	**1.2212 ± 0.0573**	1.4314 ± 0.0636
PIMA	6.5495 ± 0.1310	2.9739 ± 0.1389	6.4343 ± 0.1487	**1.0068 ± 0.0117**	1.4854 ± 0.0501
SPAM	14.7424 ± 0.8757	2.8445 ± 0.2664	14.0828 ± 0.8056	**1.0677 ± 0.1173**	1.0911 ± 0.1045
WINE	10.1063 ± 0.0474	3.6063 ± 0.0997	9.7524 ± 0.0693	**1.1302 ± 0.0917**	1.7678 ± 0.0571
when optimizing L_1^{mad}					
ADLT	2.4250 ± 0.1320	1.5438 ± 0.0988	2.6312 ± 0.1669	**1.0000 ± 0.0000**	1.0750 ± 0.0228
BIOD	17.7212 ± 0.7181	6.5568 ± 0.4956	15.5073 ± 0.6615	2.5552 ± 0.5095	**2.5457 ± 0.4871**
COMP	1.7354 ± 0.0360	1.5229 ± 0.0501	2.2750 ± 0.0981	**1.0354 ± 0.0384**	1.3125 ± 0.0411
GERM	6.7229 ± 0.2682	3.7208 ± 0.1842	6.9667 ± 0.2488	**1.9146 ± 0.0564**	1.9896 ± 0.0330
HELO	13.7084 ± 0.1939	5.3562 ± 0.2299	14.8208 ± 0.1580	**1.6438 ± 0.2507**	1.7625 ± 0.2177
LIVR	4.9384 ± 0.1148	2.6527 ± 0.2885	4.9133 ± 0.1968	**1.0380 ± 0.0388**	1.3800 ± 0.1480
MAMO	1.5898 ± 0.0686	1.4737 ± 0.1109	1.6954 ± 0.0983	**1.1719 ± 0.0252**	1.2689 ± 0.0418
PIMA	6.4567 ± 0.1443	2.9405 ± 0.2049	6.3358 ± 0.1219	**1.0068 ± 0.0117**	1.2616 ± 0.0436
SPAM	12.2568 ± 0.7802	2.8739 ± 0.2263	11.9241 ± 0.8344	**1.0364 ± 0.0631**	1.2631 ± 0.0644
WINE	9.9485 ± 0.0105	3.8430 ± 0.2807	9.4544 ± 0.0364	**1.1420 ± 0.0871**	1.7527 ± 0.0657

Table 8. Proportion of Generated Counterfactual Instances Considered as Plausible (Based on Their LOF Score)

Data	MO	DisCERN	LIRE	OFCC	Ours
ADLT	**0.9667 ± 0.0161**	0.9542 ± 0.0208	0.9125 ± 0.0336	0.9156 ± 0.0184	0.9188 ± 0.0149
BIOD	**0.9628 ± 0.0207**	0.9005 ± 0.0149	0.9348 ± 0.0384	0.8976 ± 0.0090	0.8976 ± 0.0090
COMP	0.9375 ± 0.0247	0.9312 ± 0.0143	**0.9531 ± 0.0239**	0.8635 ± 0.0186	0.9094 ± 0.0245
GERM	**0.9927 ± 0.0065**	0.9885 ± 0.0058	0.9927 ± 0.0050	0.9760 ± 0.0135	0.9771 ± 0.0137
HELO	0.9958 ± 0.0041	0.9896 ± 0.0143	**0.9979 ± 0.0036**	0.9802 ± 0.0200	0.9812 ± 0.0207
LIVR	**0.9547 ± 0.0196**	0.9254 ± 0.0216	0.9359 ± 0.0309	0.8880 ± 0.0382	0.8959 ± 0.0365
MAMO	0.9843 ± 0.0179	**0.9919 ± 0.0085**	0.9808 ± 0.0168	0.9847 ± 0.0160	0.9847 ± 0.0160
PIMA	**0.9790 ± 0.0243**	0.9550 ± 0.0498	0.9682 ± 0.0337	0.9488 ± 0.0374	0.9488 ± 0.0374
SPAM	0.9323 ± 0.0312	0.9192 ± 0.0314	**0.9403 ± 0.0328**	0.8876 ± 0.0437	0.8938 ± 0.0421
WINE	**0.9864 ± 0.0144**	0.9619 ± 0.0080	0.9750 ± 0.0177	0.9619 ± 0.0067	0.9675 ± 0.0066

require dozens to hundreds of seconds to produce a single counterfactual, which makes them impractical for real-world usage scenarios. Some others, however efficient, cannot guarantee that the generated counterfactual examples are valid. In contrast, MO, DisCERN, OFCC, and LIRE satisfy the two criteria that we regard as indispensable for real-world counterfactual explanation: (i) they always return valid counterfactuals whenever possible, and (ii) they achieve runtimes that scale

Table 9. Time to Generate One Counterfactual Sample (Seconds)

Data	MO	DisCERN	LIRE	OFCC	Ours
ADLT	**0.0019 ± 0.0006**	0.0161 ± 0.0008	0.0069 ± 0.0008	0.4948 ± 0.0084	1.4706 ± 0.2123
BIOD	**0.0001 ± 0.0002**	0.0305 ± 0.0017	0.0011 ± 0.0005	1.2313 ± 0.0293	5.6882 ± 0.0683
COMP	0.0002 ± 0.0001	0.0071 ± 0.0013	**0.0001 ± 0.0001**	0.1609 ± 0.0058	0.6160 ± 0.1681
GERM	**0.0001 ± 0.0002**	0.0299 ± 0.0051	0.0009 ± 0.0003	0.2329 ± 0.0159	4.3557 ± 0.4441
HELO	**0.0011 ± 0.0002**	0.0311 ± 0.0044	0.0066 ± 0.0006	1.1488 ± 0.0174	5.4889 ± 0.1025
LIVR	**0.0001 ± 0.0002**	0.0095 ± 0.0019	0.0001 ± 0.0002	0.1381 ± 0.0080	2.5674 ± 0.5040
MAMO	**0.0001 ± 0.0001**	0.0072 ± 0.0006	0.0001 ± 0.0001	0.0454 ± 0.0010	0.5619 ± 0.1826
PIMA	**0.0001 ± 0.0001**	0.0108 ± 0.0008	0.0002 ± 0.0003	0.5127 ± 0.0128	4.8612 ± 0.2521
SPAM	**0.0024 ± 0.0007**	0.0235 ± 0.0031	0.0052 ± 0.0010	3.0554 ± 0.0656	5.3627 ± 0.1608
WINE	**0.0001 ± 0.0001**	0.0132 ± 0.0008	0.0006 ± 0.0002	0.8301 ± 0.0429	5.5450 ± 0.1709

reasonably on the datasets considered. We therefore chose these four methods to compare to, ensuring that the comparison remains both fair and practically meaningful.

From Tables 5 and 6, our method consistently yields the smallest dissimilarity across all datasets, for both L_2^{std} and L_1^{mad}. OFCC is usually the second-best method in terms of proximity, while LIRE generally improves over MO but remains clearly behind OFCC and our approach, especially under L_1^{mad}. This confirms that, starting from MO, DisCERN, and OFCC as initializations, the additional search over decision regions allows our method to find counterfactuals that are closer to the query instance, so that users need to change feature values less to obtain a determinate prediction.

The sparsity of counterfactual examples is reported in Table 7. Generating counterfactual examples via MO and LIRE requires modifying a high number of features, in particular for datasets with a high number of continuous features, such as the Wine dataset. DisCERN significantly reduces the number of modified features by utilizing feature importance, but it is still relatively high. OFCC is designed to modify only one feature, but for some samples, this is insufficient to change the prediction into a desired one. Note that to ensure the validity of counterfactuals in the implementation, if OFCC fails to generate valid counterfactuals, it returns the counterfactuals generated by DisCERN: this is why the number of features changed by OCCF on some datasets in the table may slightly exceed one. Our approach is reasonable in terms of the number of modified features, since it generally results in fewer than three features being modified. This level of sparsity is arguably consistent with the cognitive constraints pertaining to contrastive explanations. This sparsity also implies that the branch-and-bound search procedure discards the majority of branches at an early stage, effectively achieving a high pruning rate without the need to explore the full search space. Additionally, owing to the characteristics of the L1-norm, it can be observed that the counterfactual instances generated using L_1^{mad} are sparser compared to those generated using L_2^{std}.

Table 8 reports the proportion of generated counterfactual instances that are considered as plausible—i.e., with LOF score less than 1.5. As can be seen, MO and LIRE attain the highest plausibility scores, which is expected since they return training samples or points in densely populated leaf intersections. DisCERN also achieves very high plausibility by reusing feature values from MO. Our method remains close to these baselines: the proportion of plausible counterfactuals is almost always above 90%, while still improving substantially over MO, DisCERN, and LIRE in terms of dissimilarity and sparsity. OFCC tends to generate slightly less plausible counterfactuals, reflecting the fact that single-feature changes tend to produce counterfactuals outside of the data manifold.

Table 10. Feature Importance for the Acceleration of Brand-and-Bound Counterfactual Search, Reported with the Percentage of Improvement (%) of Average Time Compared to Table 9

Data	MDI (Global)	PFI (Global)	SHAP-FI (Global)	PFI (Local)	SHAP (Local)	LIME (Local)
ADLT	35.77	**39.77**	16.89	29.16	7.95	−3.61
BIOD	80.63	**90.15**	87.15	71.97	30.47	55.59
COMP	22.08	22.26	12.95	7.18	10.57	−65.50
GERM	**77.11**	71.71	71.70	51.71	71.19	70.37
HELO	46.60	49.95	**52.93**	35.32	24.42	8.97
LIVR	**15.96**	13.55	8.62	6.75	7.75	7.49
MAMO	63.04	**82.24**	56.75	52.61	56.29	48.05
PIMA	8.46	9.45	7.75	**11.08**	−1.57	8.82
SPAM	5.48	5.58	4.04	3.88	**18.00**	8.46
WINE	2.50	**7.81**	3.02	2.36	−3.25	2.46

Table 9 reports the average time cost of generating counterfactual examples. It confirms that MO and LIRE are by far the fastest methods, requiring only a few milliseconds per counterfactual, since they essentially operate by querying existing training points or a small proportion of all leaf intersections. DisCERN remains efficient despite repeated evaluations with feature replacements. OFCC and our method are more costly because they perform structured searches in the space induced by the forest. OFCC remains reasonably fast, whereas our method is the slowest overall; however, it still generates counterfactuals within a few seconds per instance (well below 10 seconds on all datasets) for forests with 100 fully-grown trees. Given the clear gains in proximity and sparsity, these runtimes appear acceptable.

6.2 Counterfactual Example Generation Acceleration via Feature Importance

As presented in Section 5, global or local feature importance measures can be used to establish a feature ordering so as to optimize the construction of the search tree. Here, we evaluate the effectiveness of three global feature importance assessment methods (MDI, PFI, and SHAP-FI) and three local methods (PFI-Local, SHAP, and LIME) in accelerating the branch-and-bound search for proximal valid counterfactual examples. Table 10 reports the improvement in terms of mean elapsed time achieved by incorporating feature order based on feature importance, as a percentage of the elapsed time when using the default feature order in datasets. Best results are highlighted in bold. This comparison can be considered as an ablation study, confirming that feature-importance-guided ordering accelerates the search without degrading the other performance metrics considered.

Overall, global feature importance has the most positive and stable effect on efficiency. For all datasets, MDI, PFI, and SHAP-FI yield a reduction in runtime, often by a large margin. PFI achieves the largest improvement on most datasets, while MDI and SHAP-FI are competitive and can be best on specific cases (for instance GERM and LIVR for MDI, and HELO for SHAP-FI). This behavior is consistent with the fact that global scores are computed from the trees themselves and capture how often and how effectively features contribute to splits, which aligns well with our goal of focusing the search on features that are most likely to resolve indeterminate predictions.

In contrast, local feature importance leads to smaller and more variable gains, and can even slow down the search on some datasets (e.g., LIME on ADLT and COMP, or local SHAP on PIMA and WINE). Local scores are estimated around individual instances and rely on sampling, which

makes them more sensitive to noise, correlations between features, and the limited number of indeterminate predictions. When the local estimates are not reliable enough to reflect which features actually drive the CRF decisions, the induced ordering may push uninformative features toward the top of the search tree, increasing the number of visited nodes instead of reducing it. This explains why global feature importance tends to provide more robust acceleration across datasets.

7 Conclusion

In this article, we have proposed a framework where counterfactual examples are used to explain indeterminate predictions made by a binary CRF. The generated counterfactual explanations aim to address the questions of why a given input instance is classified indeterminately, and how some feature values can be modified so as to achieve a determinate prediction. We have proposed a branch-and-bound approach to search for the closest counterfactual examples and integrated plausibility and actionability considerations into the process. To accelerate the generation of counterfactual examples, we have proposed to use local and global feature importance measures to determine important features and locate them at the bottom of the search tree.

By comparing our proposed method with several contenders, we have demonstrated its advantages in terms of dissimilarity, sparsity, and plausibility of the generated counterfactual examples, at a slightly higher—yet arguably perfectly reasonable—computational cost. These results are obtained on tabular data using CRFs with axis-aligned trees, which induce box-shaped decision regions and typically involve only a few decisive features, which makes the exact search tractable. The LOF-based plausibility constraint further assumes that the training data provide a reasonable local approximation of the underlying distribution, a condition that may be more challenging to satisfy in very high-dimensional or sparse feature spaces.

In future work, we will study how to rely on explanations to make a diagnosis of the model and possibly reduce the indeterminacy and increase the accuracy of CRFs. Another challenging open problem is the extension of the current work to generating counterfactual examples associated with indeterminate predictions in multi-class problems. Additionally, further optimizations of the approach could be explored, for instance through advanced data structures or approximate search strategies when the number of decisive features is large, as well as the efficient computation of sets of diverse counterfactual examples, which will likely require adapting the branch-and-bound procedure.

Data Availability

Data and code are available on https://github.com/Haifei-ZHANG/Explainable-Cautious-Random-Forest.

References

[1] Joaquin Abellan and Andrés R. Masegosa. 2012. Imprecise classification with credal decision trees. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 20, 5 (2012), 763–787. DOI: https://doi.org/10.1142/S0218488512500353

[2] Joaquín Abellán and Serafín Moral. 2003. Building classification trees using the total uncertainty criterion. *International Journal of Intelligent Systems* 18, 12 (2003), 1215–1225. DOI: https://doi.org/10.1002/int.10143

[3] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160. DOI: https://doi.org/10.1109/ACCESS.2018.2870052

[4] Omar Alghushairy, Raed Alsini, Terence Soule, and Xiaogang Ma. 2020. A review of local outlier factor algorithms for outlier detection in big data streams. *Big Data and Cognitive Computing* 5, 1 (2020), 1. DOI: https://doi.org/10.3390/bdcc5010001

[5] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. 2020. Explainable artificial intelligence

(XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82–115. DOI: https://doi.org/10.1016/j.inffus.2019.12.012

[6] Jean-Marc Bernard. 2005. An introduction to the imprecise Dirichlet model for multinomial data. *International Journal of Approximate Reasoning* 39, 2–3 (2005), 123–150. DOI: https://doi.org/10.1016/j.ijar.2004.10.002

[7] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2154–2156. DOI: https://doi.org/10.1145/3243734.3264418

[8] Pierre Blanchart. 2021. An exact counterfactual-example-based approach to tree-ensemble models interpretability. arXiv:2105.14820. Retrieved from https://arxiv.org/abs/2105.14820

[9] Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1 (2001), 5–32. DOI: https://doi.org/10.1023/A:1010933404324

[10] Leo Breiman. 2017. *Classification and Regression Trees*. Routledge. DOI: https://doi.org/10.1201/9781315139470

[11] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, Vol. 29. ACM, 93–104. DOI: https://doi.org/10.1145/335191.335388

[12] Miguel A. Carreira-Perpinán and Suryabhan Singh Hada. 2023. Very fast, approximate counterfactual explanations for decision forests. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37, 6935–6943. DOI: https://doi.org/10.1609/aaai.v37i6.25848

[13] Yu-Liang Chou, Catarina Moreira, Peter Bruza, Chun Ouyang, and Joaquim Jorge. 2022. Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications. *Information Fusion* 81 (2022), 59–83. DOI: https://doi.org/10.1016/j.inffus.2021.11.003

[14] R. Dennis Cook. 2000. Detection of influential observation in linear regression. *Technometrics* 42, 1 (2000), 65–68. DOI: https://doi.org/10.1080/00401706.1977.10489493

[15] Zhicheng Cui, Wenlin Chen, Yujie He, and Yixin Chen. 2015. Optimal action extraction for random forests and boosted trees. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 179–188. DOI: https://doi.org/10.1145/2783258.2783281

[16] Enyan Dai and Suhang Wang. 2025. Towards prototype-based self-explainable graph neural network. *ACM Transactions on Knowledge Discovery from Data* 19, 2 (2025), 1–20. DOI: https://doi.org/10.1145/3689647

[17] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. 2020. Multi-objective counterfactual explanations. In *Proceedings of the 16th International Conference on Parallel Problem Solving from Nature—PPSN XVI (PPSN '20)*. Springer, 448–469. DOI: https://doi.org/10.1007/978-3-030-58112-1_31

[18] Arthur P. Dempster. 2008. Upper and lower probabilities induced by a multivalued mapping. In *Classic Works of the Dempster-Shafer Theory of Belief Functions*. Roland R. Yager and Liping Liu (Eds.), Springer, 57–72. DOI: https://doi.org/10.1007/978-3-540-44792-4_3

[19] Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. arXiv:1702.08608. Retrieved from https://arxiv.org/abs/1702.08608

[20] Rubén R. Fernández, Isaac Martín De Diego, Víctor Aceña, Alberto Fernández-Isabel, and Javier M. Moguerza. 2020. Random Forest explainability using counterfactual sets. *Information Fusion* 63 (2020), 196–207. DOI: https://doi.org/10.1016/j.inffus.2020.07.001

[21] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. 2019. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research* 20, 177 (2019), 1–81. Retrieved from https://www.jmlr.org/papers/v20/18-760.html

[22] Gabriele Gianini, Jianyi Lin, Corrado Mio, and Ernesto Damiani. 2022. Set-based counterfactuals in partial classification. In *Proceedings of the 19th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, 560–571. DOI: https://doi.org/10.1007/978-3-031-08974-9_45

[23] Markus Goldstein and Seiichi Uchida. 2016. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS One* 11 (2016), e0152173. DOI: https://doi.org/10.1371/journal.pone.0152173

[24] Riccardo Guidotti. 2022. Counterfactual explanations and how to find them: Literature review and benchmarking. *Data Mining and Knowledge Discovery* 38, 5 (2022), 1–55. DOI: https://doi.org/10.1007/s10618-022-00831-6

[25] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. 2018. Local rule-based explanations of black box decision systems. arXiv:1805.10820. Retrieved from https://arxiv.org/abs/1805.10820

[26] Eyke Hüllermeier, Sébastien Destercke, and Mohammad Hossein Shaker. 2022. Quantification of credal uncertainty in machine learning: A critical analysis and empirical comparison. In *Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence*. PMLR, 548–557. Retrieved from https://proceedings.mlr.press/v180/hullermeier22a.html.

[27] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Hiroki Arimura. 2020. DACE: Distribution-aware counterfactual explanation by Mixed-Integer linear optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2855–2862. DOI: https://doi.org/10.24963/ijcai.2020/395

[28] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. 2020. Model-agnostic counterfactual explanations for consequential decisions. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*. PMLR, 895–905. Retrieved from https://proceedings.mlr.press/v108/karimi20a

[29] Mark T. Keane and Barry Smyth. 2020. Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable AI (XAI). In *Proceedings of the 28th International Conference on Case-Based Reasoning Research and Development (ICCBR '20)*. Springer, 163–178. DOI: https://doi.org/10.1007/978-3-030-58342-2_11

[30] Been Kim, Rajiv Khanna, and Oluwasanmi O. Koyejo. 2016. Examples are not enough, learn to criticize! Criticism for interpretability. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2016/hash/5680522b8e2bb01943234bce7bf84534-Abstract.html

[31] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1885–1894. Retrieved from https://proceedings.mlr.press/v70/koh17a

[32] Thibault Laugel, Adulam Jeyasothy, Marie-Jeanne Lesot, Christophe Marsala, and Marcin Detyniecki. 2023. Achieving diversity in counterfactual explanations: A review and discussion. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, 1859–1869. DOI: https://doi.org/10.1145/3593013.3594122

[33] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. 2018. Comparison-based inverse classification for interpretability in machine learning. In *Proceedings of the 17th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Theory and Foundations (IPMU '18)*. Springer, 100–111. DOI: https://doi.org/10.1007/978-3-319-91473-2_9

[34] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. 2019. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. arXiv:1907.09294. Retrieved from https://arxiv.org/abs/1907.09294

[35] Xiao Li, Yu Wang, Sumanta Basu, Karl Kumbier, and Bin Yu. 2019. A debiased MDI feature importance measure for random forests. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Vol. 32. Curran Associates Inc. Retrieved from https://proceedings.neurips.cc/paper/2019/hash/702cafa3bb4c9c86e4a3b6834b45aedd-Abstract.html

[36] Zachary C. Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue* 16, 3 (2018), 31–57. DOI: https://doi.org/10.1145/3236386.3241340

[37] Ana Lucic, Harrie Oosterhuis, Hinda Haned, and Maarten de Rijke. 2022. FOCUS: Flexible optimizable counterfactual explanations for tree ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36, 5313–5322. DOI: https://doi.org/10.1609/aaai.v36i5.20468

[38] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Vol. 30. Curran Associates Inc., 4768–4777. Retrieved from https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html

[39] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267 (2019), 1–38. DOI: https://doi.org/10.1016/j.artint.2018.07.007

[40] Christoph Molnar. 2020. Interpretable Machine Learning. *Lulu.com*. Retrieved from https://christophm.github.io/interpretable-ml-book/

[41] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 607–617. DOI: https://doi.org/10.1145/3351095.3372850

[42] Axel Parmentier and Thibaut Vidal. 2021. Optimal counterfactual explanations in tree ensembles. In *Proceedings of the International Conference on Machine Learning*. PMLR, 8422–8431. Retrieved from https://proceedings.mlr.press/v139/parmentier21a.html

[43] Foster Provost and Tom Fawcett. 2001. Robust classification for imprecise environments. *Machine Learning* 42, 3 (2001), 203–231. DOI: https://doi.org/10.1023/A:1007601015854

[44] Gabrielle Ras, Ning Xie, Marcel van Gerven, and Derek Doran. 2022. Explainable deep learning: A field guide for the uninitiated. *Journal of Artificial Intelligence Research* 73 (2022), 329–396. DOI: https://doi.org/10.1613/jair.1.13200

[45] Shubham Rathi. 2019. Generating counterfactual and contrastive explanations using SHAP. arXiv:1906.09293. Retrieved from: https://arxiv.org/abs/1906.09293

[46] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144. DOI: https://doi.org/10.1145/2939672.2939778

[47] Omer Sagi and Lior Rokach. 2020. Explainable decision Forest: Transforming a decision forest into an interpretable tree. *Information Fusion* 61 (2020), 124–138. DOI: https://doi.org/10.1016/j.inffus.2020.03.013

[48] Iqbal H. Sarker. 2021. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science* 2, 3 (2021), 1–21. DOI: https://doi.org/10.1007/s42979-021-00592-x

[49] Glenn Shafer. 1976. *A Mathematical Theory of Evidence*. Princeton University Press. DOI: https://doi.org/10.2307/j.ctv10vm1qb

[50] Lloyd S. Shapley. 1953. *A Value for n-Person Games*. Princeton University Press. DOI: https://doi.org/10.1515/9781400829156-012

[51] Sarah Tan, Matvey Soloviev, Giles Hooker, and Martin T. Wells. 2020. Tree space prototypes: Another look at making tree ensembles interpretable. In *Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference*, 23–34. DOI: https://doi.org/10.1145/3412815.3416893

[52] Naeem Ullah, Javed Ali Khan, Ivanoe De Falco, and Giovanna Sannino. 2024. Explainable artificial intelligence: Importance, use domains, stages, output shapes, and challenges. *ACM Computing Surveys* 57, 4 (2024), 1–36. DOI: https://doi.org/10.1145/3705724

[53] Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan Hines, John Dickerson, and Chirag Shah. 2024. Counterfactual explanations and algorithmic recourses for machine learning: A review. *ACM Computing Surveys* 56, 12 (2024), 1–42. DOI: https://doi.org/10.1145/3677119

[54] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *SSRN Electronic Journal* 31 (2017), 841. Retrieved from https://heinonline.org/HOL/LandingPage?handle=hein.journals/hjlt31&div=29

[55] Peter Walley. 1996. Inferences from multinomial data: Learning about a bag of marbles. *Journal of the Royal Statistical Society: Series B (Methodological)* 58, 1 (1996), 3–34. Retrieved from https://www.jstor.org/stable/2346164

[56] Nirmalie Wiratunga, Anjana Wijekoon, Ikechukwu Nkisi-Orji, Kyle Martin, Chamath Palihawadana, and David Corsar. 2021. Discern: Discovering counterfactual explanations using relevance features from neighbourhoods. In *Proceedings of the 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 1466–1473. DOI: https://doi.org/10.1109/ICTAI52525.2021.00233

[57] Marco Zaffalon, Giorgio Corani, and Denis Mauá. 2012. Evaluating credal classifiers by utility-discounted predictive accuracy. *International Journal of Approximate Reasoning*, 53 (2012), 1282–1301. DOI: https://doi.org/10.1016/j.ijar.2012.06.022

[58] Haifei Zhang, Benjamin Quost, and Marie-Helène Masson. 2021. Cautious random forests: A new decision strategy and some experiments. In *Proceedings of the International Symposium on Imprecise Probability: Theories and Applications*. PMLR, 369–372. Retrieved from https://proceedings.mlr.press/v147/zhang21a.html

[59] Haifei Zhang, Benjamin Quost, and Marie-Hélène Masson. 2022. Explaining cautious random forests via counterfactuals. In *Proceedings of the Building Bridges between Soft and Statistical Methodologies for Data Science*. Springer, 390–397. DOI: https://doi.org/10.1007/978-3-031-15509-3_51

[60] Haifei Zhang, Benjamin Quost, and Marie-Hélène Masson. 2023. Cautious weighted random forests. *Expert Systems with Applications* 213 (2023), 118883. DOI: https://doi.org/10.1016/j.eswa.2022.118883

[61] Haifei Zhang, Benjamin Quost, and Marie-Hélène Masson. 2025. Cautious classifier ensembles for set-valued decision-making. *International Journal of Approximate Reasoning* 177 (2025), 109328. DOI: https://doi.org/10.1016/j.ijar.2024.109328

Received 1 July 2025; revised 12 December 2025; accepted 17 January 2026