Efficient and Effective Counterfactual Explanations for Random Forests

Haifei Zhang[a,*], Jinfeng Zhong[b]

[a]Hubert Curien Lab - UMR CNRS 5516, Jean Monnet University, Saint-Etienne 42000, France
[b]CEREMADE - UMR CNRS 7534, Paris-Dauphine University, PSL Research University, Paris 75775, France

ARTICLE INFO

Keywords:
Explainable AI
Counterfactual explanations
Ensemble learning
Random forests

ABSTRACT

Random forests are widely used in machine learning due to their excellent predictive performance and computational efficiency. However, their inherent complexity often hinders interpretability, making it challenging for users to understand the decision-making process. Explainable Artificial Intelligence (XAI) techniques aim to mitigate this issue by improving model transparency and providing explanations for predictions. Among these techniques, counterfactual explanations provide intuitive insights by describing the minimal modifications needed to achieve a desired outcome. However, generating counterfactual explanations of good quality is still challenging for random forests. In this paper, we propose a novel explanation approach called Efficient and Effective Counterfactual Explanation (EECE) for random forests, which generates counterfactual explanations by leveraging the structure of decision tree leaves. EECE not only ensures efficient explanation generation but also satisfies essential properties for high-quality counterfactual explanations, such as validity, proximity, sparsity, diversity, plausibility, and actionability. We compare EECE with existing methods across 15 datasets using multiple evaluation metrics, demonstrating its effectiveness in generating high-quality counterfactual explanations.

1. Introduction

Since the introduction of the Defense Advanced Research Projects Agency (DARPA) program in 2017 (Gunning et al., 2021) and the enforcement of the General Data Protection Regulation (GDPR) in 2018 (Voigt & Von dem Bussche, 2017), there has been an increasing demand, from both regulatory and practical standpoints, to provide effective explanations for widely adopted AI systems. It is also demonstrated that ensuring transparency in automated decision-making processes is crucial for fostering user trust and acceptance (Lipton, 2018). At the same time, a well-recognized challenge in AI is the trade-off between model performance and interpretability (Rudin et al., 2022). Highly predictive models, such as deep learning and ensemble learning models, often lack interpretability (Doshi-Velez & Kim, 2017; Saeed & Omlin, 2023), whereas more transparent models, such as linear regression and decision trees, tend to have lower predictive power. To address this issue, eXplainable Artificial Intelligence (XAI) has emerged as an attractive research area, which aims to bridge this gap by either developing intrinsically interpretable models with competitive performance or providing post-hoc explanations that can clarify the decisions made by complex ("black-box") models (Adadi & Berrada, 2018; Arrieta et al., 2020).

Explanations in XAI can be categorized in several ways. Based on scope, explanations are classified as global or local. Global explanations provide insights into the overall decision-making behavior of a model, while local explanations focus on a specific input instance and its corresponding output (Molnar, 2020). Another categorization is based on applicability, distinguishing between model-agnostic and model-specific explainers. Model-agnostic techniques, such as SHAP (Lundberg & Lee, 2017) and LIME (Ribeiro et al., 2016), can be applied to any machine learning model by analyzing only its inputs and outputs, without requiring access to its internal structure. Model-specific explainers, on the other hand, leverage the internal mechanics of a particular model architecture to generate explanations. Finally, explanations can take various forms, including feature importance or attribution scores (Apley & Zhu, 2020; Breiman, 2001; Friedman, 2001), rule-based representations (Deng, 2019; Kozielski et al., 2025; Li & Zaiane, 2017; Motallebi et al., 2023), sample-based explanations such as counterfactual examples (Wachter et al., 2017) and prototypes (Kim et al., 2016), saliency maps (Selvaraju et al., 2020; Simonyan et al., 2013), etc.

Among these explanations, counterfactual explanations are quite special since they do not directly explain why a decision is made but rather suggest actionable changes to achieve a desired outcome (Wachter et al., 2017), i.e., to answer the "why not" or "how" question. In essence, they describe the minimal modifications needed to alter an AI system's decision for a given instance. For example, considering a loan application scenario: if a bank rejects a loan application based on a client's financial profile, the client may not only want to understand the reasons for the rejection but also how to qualify for approval. A counterfactual explanation might state: "If you increase your monthly income by $500 and reduce your number of credit cards by one, your loan application will be approved." Such explanations implicitly provide insight into why the decision has been made, as users can infer that a low income and multiple credit cards were key factors in the rejection.

*Corresponding author
haifei.zhang@univ-st-etienne.fr (H. Zhang); jinfeng.zhong@dauphine.psl.eu (J. Zhong)
ORCID(s): 0000-0003-4488-1631 (H. Zhang); 0000-0001-9267-590X (J. Zhong)

Counterfactual explanations are quite useful, but generating them for random forests, high-performing ensemble learning models, presents significant challenges due to their exponential complexity of classification boundaries and lack of differentiability (Haddouchi & Berrado, 2019). This challenge intensifies when accounting for key properties like validity, proximity, plausibility, actionability, diversity, efficiency, and stability of counterfactual explanations (Verma et al., 2024). Existing state-of-the-art solutions to this problem can be categorized into four main approaches: (1) searching for counterfactual examples within an existing dataset, (2) formulating the problem as a (mixed-integer) optimization task, (3) conducting a search within the solution space constructed with the random forest structure, and (4) approximating a random forest with a simplified surrogate model (e.g., a decision tree). Each approach has its advantages and limitations. Searching within a dataset is computationally efficient but cannot guarantee the proximity to the original instance. Optimization-based methods and search within the exhaustive solution space are able to produce counterfactual examples that satisfy multiple desirable properties, while often suffer from exponential complexity, limiting their practical applicability to large random forests. Lastly, surrogate model approaches may generate counterfactual examples that do not align with predictions of original models, leading to invalid explanations. Existing methods struggle to balance these properties, particularly in the context of random forests, where decision paths are fragmented across multiple trees.

Considering these issues, in this paper, we propose a novel method to generate counterfactual explanations for random forests, called Efficient and Effective Counterfactual Explanation (EECE) generator for random forests. EECE leverages the structure of decision trees to efficiently search for high-quality counterfactual examples. Our contributions can be summarized as follows.

1. Inspired by the Feature-Tweaking method (Tolomei et al., 2017), EECE identifies candidate counterfactual examples by searching within individual leaves of decision trees and selecting the most suitable one, but EECE enhances both efficiency and effectiveness by introducing an improved tree representation and a more reasonable counterfactual generation strategy.
2. By incorporating active regions proposed in LIRE (Carreira-Perpinán & Hada, 2023) as additional candidate regions, we guarantee the availability of counterfactual examples for all input instances, which is a key limitation of the Feature-Tweaking method.
3. Beyond ensuring validity, proximity, and efficiency, EECE also comprehensively accounts for other critical desiderata of counterfactual explanations, such as plausibility, actionability, and diversity. We provide detailed discussion for each of them.
4. Experimental results demonstrate that EECE can efficiently generate valid, more proximal, more sparse, and more plausible counterfactual examples, i.e., more effective.

This paper is structured as follows. Section 2 provides the definition and desired properties of counterfactual explanations, then reviews the state-of-the-art counterfactual explanation generation methods for random forests. In Section 3, we detail our novel proposed EECE method and discuss how it satisfies the desiderata of counterfactual explanations. Section 4 presents experiments conducted on 15 datasets and analyses the obtained results. Finally, a conclusion is drawn in Section 5.

2. Related work

In this section, we first give the definition and the desired properties of counterfactual explanations. Then, we will review commonly used counterfactual generation methods for random forests, including model-agnostic and model-specific ones. We will also provide a comparison analysis among them.

2.1. Counterfactual explanations

In explainable machine learning, a counterfactual explanation of a prediction for a given query instance can be described as the smallest change to its feature values that will alter the prediction to a predefined one (Wachter et al., 2017). Similar to the paper of Guidotti (2024), we formalize the definition of counterfactual explanation and counterfactual explainer as follows.

Definition 1. (Counterfactual explanation) Given a trained classifier f, a query instance x with prediction $\hat{y} = f(x)$, a counterfactual example is an instance x' that requires minimal perturbation to x and obtains a desired prediction $y' = f(x') \neq \hat{y}$. The difference between x' and x, noted as $\Delta(x', x)$, is defined as the counterfactual explanation.

Here, the counterfactual explanation (perturbation) can be interpreted as the cost required to obtain the desired prediction. The measure of minimum perturbation depends on different settings, which could be different distance metrics. In the following, we generally use counterfactual examples x' as the output of an explainer. The counterfactual explanation can simply be obtained by $\Delta(x', x)$. Counterfactual examples can be either sought from existing datasets or generated virtually. Both approaches can be seen as producing a counterfactual example through a function that serves as a counterfactual explainer, defined as follows.

Definition 2. (Counterfactual explainer) A counterfactual explainer is a function $g(x, f, y', d, \mathcal{X}, k)$ that returns k valid counterfactual examples for the query instance x, where f is a trained underlying classifier, y' is the predefined desired prediction, d is a measure of the difference between two instances, and $\mathcal{X}$ denotes the feature space of the classification problem.

In this definition, if the classifier f is only applied to predict and has no need to know its inner functions, then $g(\cdot)$ is called a model-agnostic counterfactual explainer, otherwise, it is a model-specific one. If we consider generating only the

closest counterfactual example ($k = 1$), it can be modeled as an optimization problem :

$$x' = \arg\min_{z \in \mathcal{X}} d(x, z) \text{ s.t. } f(z) = y'. \quad (1)$$

Unlike other forms of explanation, the quality of counterfactual examples can be more easily quantified, allowing for a more rigorous comparison of counterfactual explainer. Below, we summarize some key properties of counterfactual examples and their associated explainers, drawing insights from recent studies (Guidotti, 2024; Verma et al., 2024).

1. **Validity**. A counterfactual example x' is valid if and only if it achieves the predefined desired prediction, i.e., $y' = f(x') \neq f(x)$.
2. **Proximity**. As the perturbation on x is a cost, the counterfactual example x' should be as close as possible to x in terms of a specific distance measure $d(x, x')$. It should be noted that if $d(x, x')$ is L_0-Norm distance, proximity is referred to as **sparsity**, i.e., changing as few features as possible. A counterfactual example x' is said to be minimal if and only if there is no valid counterfactual example x'' such that $d(x, x'') < d(x, x')$.
3. **Plausibility**. A plausible counterfactual example should be coherent with the given data distribution of the feature space where the classifier is trained. In other words, counterfactual examples should be realistic but not outliers, such as regions with very low data density.
4. **Actionability**. A counterfactual example is a solution offered to the user to change the status quo, and if this solution is not actionable, then there is no real significance. A counterfactual example and its corresponding counterfactual explanation are actionable when they satisfy the following restrictions: (1) the data type of each feature of the generated counterfactual example must be consistent with the training data of the classifier; (2) it is guaranteed that the protected (immutable) features are not modified; (3) monotonic variation of specific features is guaranteed, e.g., age can only grow; (4) there is no contradiction between the variation of features, such as one-hot encoding and highly correlated features; (5) user-specific limitations are considered.
5. **Diversity**. The explainer should be capable of providing a set of different counterfactual explanations instead of a single one. The advantage of diversity is that it gives users more possible choices and thus increases the possibility of their acceptance of explanations.

In addition to the above properties of counterfactual examples, counterfactual explainers also need to hold the following necessary properties.

1. **Efficiency**. In practice, an explainer has to generate explanations as fast as possible. This is a very important metric, especially for classifiers of huge scales, such as deep learning models and tree ensembles.
2. **Stability**. The explanation of the identical pair of instance and classifier should always be the same and two similar instances should have similar explanations. This requires that an explainer minimizes or avoids the introduction of random uncertainty during the process of explanation generation.

2.2. Explaining random forests via counterfactuals

In this section, various counterfactual explainers for random forests will be reviewed, which are categorized into two groups: model-agnostic and model-specific methods.

2.2.1. Model-agnostic approaches

A fundamental approach to counterfactual explanations is to search for appropriate counterfactual examples directly from the accessible data, referred to as the Minimum Observable (MO) approach. Building on MO, the DisCERN method (Wijekoon & Wiratunga, 2023; Wiratunga et al., 2021) selects fewer features to modify based on estimated feature importance. Additionally, LORE (LOcal Rule-based Explainer) (Guidotti et al., 2018) generates synthetic samples around the query instance using a genetic algorithm and approximates the local behavior of the original classifier with a decision tree trained on the generated samples labeled by the original model. Since random forests are non-differentiable, many model-agnostic methods relying on gradient-based optimization are not applicable. MACE (Model-Agnostic Counterfactual Explanations) (Karimi et al., 2020) addresses this limitation by converting the underlying random forests into logical formulae and employing satisfiability modulo theories to generate counterfactual explanations. For more model-agnostic counterfactual example generation methods, we recommend that readers refer to the survey of Guidotti (2024).

2.2.2. Model-specific approaches

Methods designed specifically for random forests can be grouped into three categories: optimization-based methods, model simplification approaches, and feature-space partitioning techniques.

OAE (Optimal Action Extraction) (Cui et al., 2015), DACE[1] (Distribution-Aware Counterfactual Explanation) (Kanamori et al., 2020), and OCEAN (Optimal Counterfactual ExplAiNer for Tree Ensembles) (Parmentier & Vidal, 2021) leverage integer or mixed-integer programming to generate counterfactual explanations. FOCUS (Flexible Optimizable Counterfactual Explanations for Tree Ensembles) (Lucic et al., 2022) introduces a differentiable approximation of tree ensembles using sigmoid and softmax functions, enabling gradient-based optimization to generate counterfactual examples for random forests.

Among model simplification approaches, FBT (Forest-Based Tree) (Sagi & Rokach, 2020) and OCSE (Random Forest Optimal Counterfactual Set Extractor) (Fernández et al., 2020) transform a random forest into a single decision

[1] DACE can be also applied to other additive models, such as linear models, but it is still not model-agnostic.

Table 1
Comparison of Counterfactual Explainers for Random Forests

Explainer	Category	Validity	Proximity	Plausibility	Actionability	Diversity	Efficiency
MO	Agnostic	High	Low	Explicit	Explicit	Yes	High
DisCERN	Agnostic	High	Medium	Explicit	None	Yes	High
LORE	Agnostic	Low	Medium	None	None	None	Low
MACE	Agnostic	High	High	None	Explicit	Yes	Low
DACE	Specific	High	High	Explicit	Explicit	Yes	Low
OAE	Specific	High	High	None	Explicit	Yes	Low
OCEAN	Specific	High	High	Explicit	Explicit	Yes	Medium
FOCUS	Specific	Low	Medium	None	None	None	Low
FBT	Specific	Low	Medium	None	None	None	Medium
OCSE	Specific	High	High	Implicit	Implicit	Yes	Medium
ExactCF	Specific	High	High	None	Explicit	Yes	Medium
FT	Specific	Low	Medium	None	None	None	Medium
LIRE	Specific	High	Medium	Explicit	None	Yes	High
EECE (ours)	Specific	High	High	Explicit	Explicit	Yes	High

tree to extract counterfactual examples, with FBT applying this globally and OCSE focusing on local conversion. FBT constructs high-coverage rules from trees using conjunctions, which may lead to non-valid counterfactual examples when the proxy tree's predictions deviate from the original random forest. OCSE mitigates this issue by conducting a partial conversion near the factual instance, reducing approximation errors. However, its efficiency remains a concern for large-scale forests and instances far from decision boundaries.

FT (Feature-Tweaking) (Tolomei et al., 2017), ExactCF (Exact Counterfactual-example-based Approach to Tree-Ensemble) (Blanchart, 2021), and LIRE (Lightweight Interpretation for Random Ensembles) (Carreira-Perpinán & Hada, 2023) explore feature-space partitioning. Given an instance x, FT bypasses trees already predicting the desired output and generates ϵ-satisfactory instances from decision paths in trees that do not predict the desired output. Therefore, FT only considers parts of the decision path. However, changes in one tree's prediction may not influence the overall forest's prediction. ExactCF addresses this by computing the smallest decision regions (intersections of decision paths) in a tree ensemble and applying a branch-and-bound search strategy to find the most proximal counterfactual examples. Despite its improvements, ExactCF remains computationally expensive, as the smallest decision region corresponds to the intersection of multiple decision paths (leaves). LIRE provides a trade-off between FT and ExactCF by computing intersections of leaves where each training sample falls, forming active regions that approximately represent the random forest.

2.2.3. Comparative analysis

A cross-comparison of reviewed counterfactual generation methods in random forests is presented in Table 1. Methods relying on approximate surrogate models, such as LORE, FOCUS, and FBT, exhibit low or medium validity. Optimization-based methods, including MACE, OAE, and DACE, require substantial computational time, rendering them inefficient. OCSE and ExactCF explore only the subspace around the query instance, improving validity (relative to surrogate models) and efficiency (compared to optimization-based approaches). However, for instances distant from decision boundaries, the required search space expands, leading to inefficiencies. OCEAN maintains a satisfactory level of efficiency for random forests consisted of small number of shallow trees but it explodes when the depth of trees are larger than five. Regarding FT, a primary drawback is the lack of a theoretical guarantee to find valid counterfactual examples for all query instances. In the context of large-scale random forests, the efficiency of the FT method is also low.

Table 1 also highlights that only a few methods explicitly consider the plausibility and actionability of generated counterfactual examples. For optimization-based methods, these aspects can be incorporated as constraints. Other methods must integrate these considerations into the selection criteria for determining high-quality counterfactual examples. Based on the above comparison analysis, we propose an approach similar to FT combined with LIRE to efficiently generate effective counterfactual examples for large-scale random forests, which will be detailed in section 3.

3. Proposed efficient and effective counterfactual explanation approach

In this section, we present our proposed Efficient and Effective Counterfactual Example (EECE) generation method for random forests. From an overview, we first construct candidate regions from a trained random forest model. For

each input instance, we then generate candidates of counterfactual examples in these regions and finally introduce a counterfactual example selection process considering the desiderata of high-quality counterfactual explanations that we have discussed in Section 2.1.

3.1. Candidate regions for counterfactual examples in random forests

In a decision tree, each internal node is a condition test to split current (sub) feature space into two parts and decides whether an instance falls into the left or the right sub-tree. These tests will stop till the instance reaches a leaf node. Each test consists of three elements: a split feature, a split value, and a comparison operator ($\leq$ or $>$). Therefore, each path from the root to a leaf node corresponds to a region defined by a conjunction of all condition tests along the path, which is also called a decision path and is defined as follows.

Definition 3. (Decision path) For a given feature space of D dimensions and a decision tree trained on it, a decision path dp is the conjunction of all the condition tests from the root to a leaf node l. The region corresponding to a decision path can be represented in the form of the multi-dimensional interval $r = \{I_d,\ d = 1, \dots, D\}$ where $I_d =]\underline{b}_d,\ \overline{b}_d]$.

In this definition, if a feature X_d is never used in any internal node as condition test on the decision path from the root to the corresponding leaf l, the value of $\underline{b}_d$ and $\overline{b}_d$ should be the minimum minus ϵ (or $-\infty$) and maximum (or $+\infty$) of feature X_d, respectively, where ϵ is a very small positive value to ensure the minimum value is included in the interval. Meanwhile, if a feature X_d is used multiple times on a decision path, the value of $\underline{b}_d$ and $\overline{b}_d$ should be calculated by the "AND" operation of these tests associated with feature X_d.

Example 1. (Decision path) For a given decision path in a decision tree trained on the Iris dataset, an example of the conjunction of all condition tests may look like:

$$\begin{aligned}&(\text{petal length} > 2.45) \wedge (\text{petal width} \leq 1.75) \wedge \\ &(\text{petal length} > 4.95) \wedge (\text{petal width} > 1.55) \wedge \\ &(\text{sepal length} \leq 6.95)\end{aligned}$$

If we fix the order of features as {petal length, petal width, sepal length, sepal width}, the decision path from the root to the leaf can be represented as

$$\begin{aligned}\{I_1, I_2, I_2, I_4\} = \{&]4.95, +\infty],\ ]1.55, 1.75], \\ &]0-\epsilon, +\infty],\ ]0-\epsilon, +\infty]\}.\end{aligned}$$

Here, we fix the minimum value of each feature to 0 and the maximum value to $+\infty$, respectively.

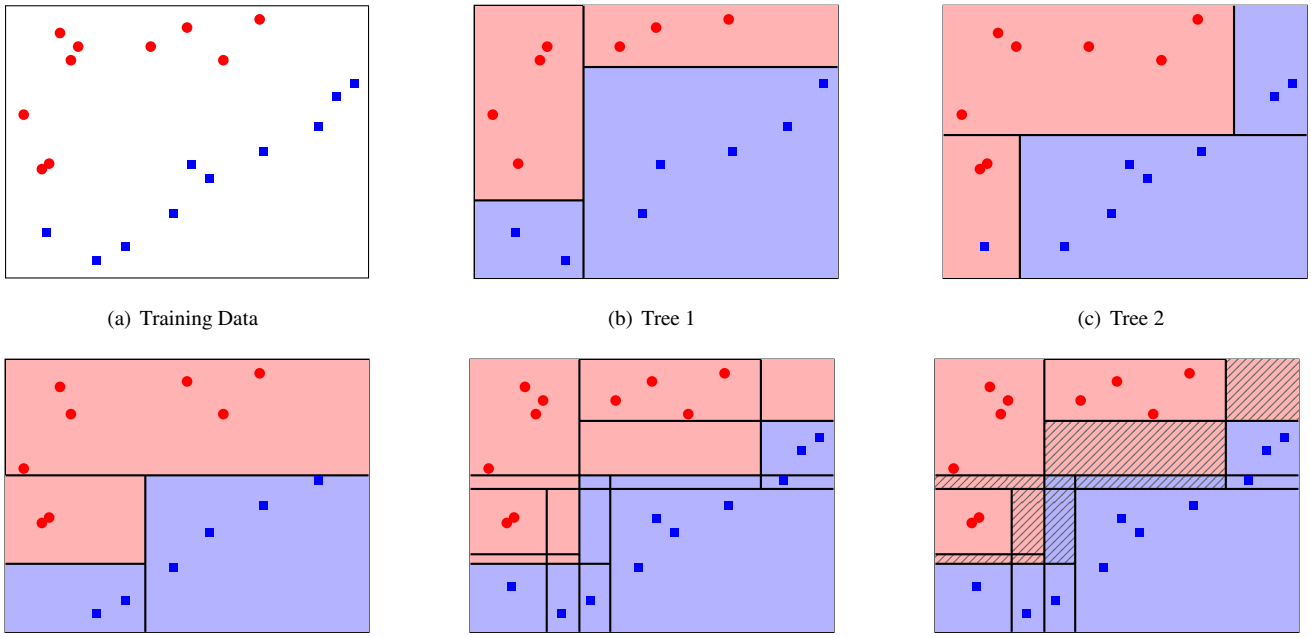
In a random forest, the decision path of each tree can be extracted. If we want to represent random forests equivalently via the smallest decision regions where all samples have the same prediction, we would have to calculate all possible intersections by merging decision paths from different trees. In this case, it is theoretically intractable for large-scale forests. If a forest consists of T trees and each contains M decision paths, the complexity will be $O(M^T)$.

For this issue, inspired by the Feature-Tweaking method, we concatenate regions of corresponding decision paths from different trees into a single set in the form of Definition 3, i.e., $R = \{\{]\underline{b}_{id},\ \overline{b}_{id}], d = 1, \dots, D\}, i = 1, \dots, N\}$, where N is the total number of decision paths in the forest and D is the number of features. Then, each decision path could be utilized to generate counterfactual example candidates for given query instances, which is called candidate region. It should be noted that the estimated class probability information is not necessary in this framework because the prediction of a tree might be inconsistent with the final prediction of the forest, which means counterfactual example candidates generated in candidate regions should be verified by the random forest prediction. Therefore, only the information associated with decision paths need to be stored. For simplifying the implementation, the set of decision paths can be divided into two $N \times D$ matrices that are called lower bound matrices L (where $L_{id} = \underline{b}_{id}$) and upper bound matrix U (where $U_{id} = \overline{b}_{id}$). The advantage of this data structure is that it allows for convenient broadcast manipulation and thus increases computational efficiency.

As discussed before, a major drawback of the Feature-Tweaking method is its inability to guarantee the generation of valid counterfactual examples for all input samples. To address this problem, we incorporate active regions proposed in the LIRE method as additional candidate regions in our framework. An active region is defined as the sub-feature space (non-empty intersection of decision paths from different trees) where at least one training sample falls into and all samples in this region have the same prediction. It is easy to get all active regions by investigating leaves where each training sample falls into and calculating their intersection. It should be noted that several training samples may reach the same active region and we can simply remove the duplication.

In real-world applications, another important issue that we should consider is categorical variables in datasets, which are typically transformed into numerical inputs through encoding schemes such as one-hot encoding, where only one feature within each group is allowed to be active (i.e., equal to 1), while others must remain 0. If we treat each feature independently without special treatment, it may lead to semantically invalid regions on one-hot encoded features.

To ensure semantic correctness of the regions derived from decision paths, we incorporate an additional step to handle one-hot encoded features after constructing the initial region. We note that for binary features, their split values are always 0.5. For each predefined one-hot group, if exactly one feature in the group is presented as $]0.5, 1]$, we enforce the one-hot semantics by constraining all other features in the group to $]0-\epsilon, 0.5]$. If no feature in the group is restricted to $]0.5, 1]$ (i.e., some features are fixed to $]0-\epsilon, 0.5]$ and others are still as $]0-\epsilon, 1]$), we split the current region into multiple sub-regions, each corresponding to one possible activation



(a) Training Data (b) Tree 1 (c) Tree 2

(d) Tree 3 (e) Random forest (f) Active regions

Figure 1: Illustration of decision trees, random forest, and active regions

of the one-hot group: in each sub-region, one feature is constrained to $]0.5, 1]$, and the rest are constrained to $]0 - \epsilon, 0.5]$. This transformation ensures that all resulting regions comply with the one-hot encoding semantics, and enables us to treat these features uniformly as independent binary variables in subsequent counterfactual generation tasks.

As an example in Fig. 1, the candidate regions of the random forest (see Fig. 1(e)) trained on the dataset of Fig.1(a) are all leaves (segmentation of regions) illustrated in trees from Fig. 1(b) to Fig. 1(d) along with segmented regions without shading in Fig. 1(f). Algorithm 1 and Algorithm 2 summarizes the process of constructing all candidate regions based on these two parts.

3.2. Generation of counterfactual examples

For a given query instance x whose prediction provided by the random forest is $\hat{y} = f(x)$ and a target class y', the Feature-Tweaking method ignores all trees that predict x as the target class y' and only considers generating counterfactual candidates from decision paths that predict x as the target class in trees that do not predict x as the target class. However, in a random forest, this consideration is questionable because changing the prediction of a particular tree does not guarantee a change in the final aggregated prediction of random forest. In the same way, even if the generated counterfactual candidate is not predicted as the desired class by a certain tree, it may still be predicted as the desired class by the random forest. Therefore, we propose to generate counterfactual candidates in all decision paths and then select valid counterfactual examples according to their predictions given by the random forest.

Another problem of the Feature-Tweaking approach is that it makes some unnecessary modifications to feature values. For example, for a given instance of Iris dataset with sepal length equal (associated with I_3) to 5 and we want to generate a candidate in the region $\{I_1, I_2, I_3, I_4\} = \{]4.95, 8.35],]1.55, 1.75],]0 - \epsilon, 6.95],]1, 4.05]\}$, Feature-Tweaking approach will generate example in this region with sepal length equals to $6.95 - \epsilon$. In fact, for this instance and this region, modifying the value of sepal length is unnecessary. In other words, the Feature-Tweaking approach modifies the values of all features that appear in the internal node tests. There are two shortcomings of this generation strategy. First, it leads the generated counterfactual candidates to the edges or to the corners of the regions, which increases the risk of outliers. Second, it may also generate counterfactual examples difficult to understand due to a large number of modified features. Therefore, we propose the following methods to overcome this problem: for a given instance x for which we want to generate a counterfactual candidate in candidate region r, then

$$z_d = \begin{cases} \underline{b}_d + \epsilon & \text{if } x_d \leq \underline{b}_d, \\ \overline{b}_d & \text{if } x_d > \overline{b}_d, \\ x_d & \text{if } \underline{b}_d < x_d \leq \overline{b}_d, \end{cases} \tag{2}$$

where x_d is the feature value of x on feature X_d, and $\underline{b}_d$ ($\overline{b}_d$) is the lower (upper) bound of candidate region r on feature

Algorithm 1: Construct candidate regions from one decision path

Input: Decision path dp, initial lower bounds $\underline{b}$ and upper bounds $\overline{b}$, one-hot groups $\mathcal{G}$
Output: Final set of lower bounds $\underline{B}$ and set of upper bounds $\overline{B}$

foreach *test in dp* **do**
  $c \leftarrow$ comparison operator in the test
  $d \leftarrow$ index of split feature in the test
  $v \leftarrow$ split value in the test
  if c *is* $\leq$ *and* $v < \overline{b}_d$ **then**
    $\overline{b}_d \leftarrow v$
  if c *is* $>$ *and* $v > \underline{b}_d$ **then**
    $\underline{b}_d \leftarrow v$
$\underline{B} \leftarrow \{\underline{b}\}$, $\overline{B} \leftarrow \{\overline{b}\}$ // Treat one-hot encoding groups
foreach *group* $g \in \mathcal{G}$ **do**
  $\underline{B}_{new} \leftarrow [\,]$, $\overline{B}_{new} \leftarrow [\,]$
  for $i = 1$ *to* $|\underline{B}|$ **do**
    $\underline{b} \leftarrow \underline{B}_i$, $\overline{b} \leftarrow \overline{B}_i$
    $A \leftarrow \{d \in g \mid \underline{b}_d = 0.5 \text{ and } \overline{b}_d = 1\}$
    if $|A| = 1$ **then**
      foreach $d \in g \setminus A$ **do**
        $\underline{b}_d \leftarrow 0 - \epsilon$, $\overline{b}_d \leftarrow 0.5$
      $\underline{B}_i \leftarrow \underline{b}$, $\overline{B}_i \leftarrow \overline{b}$
      $\underline{B}_{new} \leftarrow \underline{B}_{new} \cup \{\underline{b}\}$
      $\overline{B}_{new} \leftarrow \overline{B}_{new} \cup \{\overline{b}\}$
    else
      $A \leftarrow \{d \in g \mid \underline{b}_d = 0 - \epsilon \text{ and } \overline{b}_d = 1\}$
      foreach $d \in A$ **do**
        $\underline{b}' \leftarrow$ copy of $\underline{b}$, $\overline{b}' \leftarrow$ copy of $\overline{b}$
        $\underline{b}'_d \leftarrow 0.5$, $\overline{b}'_d \leftarrow 1$
        foreach $j \in g$ *and* $j \neq d$ **do**
          $\underline{b}'_j \leftarrow 0 - \epsilon$, $\overline{b}'_j \leftarrow 0.5$
        $\underline{B}_{new} \leftarrow \underline{B}_{new} \cup \underline{b}'$
        $\overline{B}_{new} \leftarrow \overline{B}_{new} \cup \overline{b}'$
  $\underline{B} \leftarrow \underline{B}_{new}$, $\overline{B} \leftarrow \overline{B}_{new}$
return Bounds sets: $\underline{B}$, $\overline{B}$

X_d, respectively. If feature X_d is of integer type, then

$$z_d = \begin{cases} \lceil \underline{b}_d + \epsilon \rceil & \text{if } x_d \leq \underline{b}_d, \\ \lfloor \overline{b}_d \rfloor & \text{if } x_d > \overline{b}_d, \\ x_d & \text{if } \underline{b}_d < x_d \leq \overline{b}_d. \end{cases} \tag{3}$$

Note that this paper only considers implementations of random forest models with numerical features. For categorical features, we should first convert them into numerical ones with a certain encoding method, such as one-hot encoding.

Algorithm 2: Construct all candidate region

Input: Random forest f, Training set X, one-hot groups $\mathcal{G}$
Output: Lower bounds L and upper bounds U

$L \leftarrow \{\}$
$U \leftarrow \{\}$
// Extract decision paths from the given random forest
foreach *tree t in f* **do**
  foreach *decision path dp in t* **do**
    initialize $\underline{b}$ with minimums minus ϵ of each feature
    initialize $\overline{b}$ with maximums of each feature
    $\underline{B}$, $\overline{B} \leftarrow$ Construct candidate regions from decision path $(dp, \underline{b}, \overline{b}, \mathcal{G})$ using Alg. 1
    $L \leftarrow L \cup \underline{B}$, $U \leftarrow U \cup \overline{B}$
// Extract active regions
duplication $\leftarrow \{\}$
foreach *instance x in X* **do**
  dps $\leftarrow$ decision paths for x from all trees
  if *dps* $\notin$ *duplication* **then**
    duplication $\leftarrow$ *duplication* $\cup$ *dps*
    initialize $\underline{b}$ with minimums of each feature
    initialize $\overline{b}$ with maximums of each feature
    foreach *decision path dp in dps* **do**
      $\underline{B}$, $\overline{B} \leftarrow$ Construct candidate region from decision path $(dp, \underline{b}, \overline{b})$ using Alg. 1
    $L \leftarrow L \cup \underline{B}$, $U \leftarrow U \cup \overline{B}$
return L, U

The proximity between x and generated instance z can be measured with different distance measures. As recommended by Wachter et al. (2017), it is a good choice to use the Manhattan distance weighted with the inverse median absolute deviation (MAD):

$$d(x, z) = L_1^{mad}(x, z) = \sum_{d=1}^{D} \frac{|x_d - z_d|}{MAD_d}, \tag{4}$$

where x_d stands for the value of x as per feature X_d, and where the MAD is calculated on the training set of N_t samples as

$$MAD_d = \underset{i \in \{1,\ldots,N_t\}}{\text{Median}} \left(\left| x_{id} - \underset{j \in \{1,\ldots,N_t\}}{\text{Median}} (x_{jd}) \right| \right). \tag{5}$$

This normalized distance captures scale differences among features and is robust to outliers. Due to the properties of the L_1 norm, this distance measure may lead to sparser counterfactual examples, i.e., with a small number of modified features with respect to the query instance. Alternatively, the squared Euclidean distance weighted by the inverse variance

Algorithm 3: Generate counterfactual examples

Input: Random forest f, query instance x, target class $y' \neq f(x)$, number of counterfactual examples k, lower bounds L, upper bounds U, ϵ, feature types, proximity measure $Prox$, local outlier factor LOF, actionable constraints AC

Output: Counterfactual cf

```
// initial counterfactual and distance
init_cf, init_dist ← applying LIRE method
candidates ← {init_cf}
N ← number of lines of L or U
for i = 1, ..., N do
    z ← x
    for d = 1, ..., D do
        if X_d is integer then
            z_d ← applying Eq. (3)
        else
            z_d ← applying Eq. (2)
    // using init_dist to filter far candidates
    if Prox(x, z) ≤ init_dist then
        if f(z) == c then
            // only keep candidates reach target class
            candidates ← candidates ∪ z
// to select k high-quality counterfactual examples
cfs ← CFSelection(candidates, k, Prox, LOF, AC)
return cfs
```

(V) can also be used, which is defined as

$$d(x,z) = L_2^v(x,z) = \sum_{d=1}^{D} \frac{(x_d - z_d)^2}{V_d}. \tag{6}$$

Finally, the closest instance among all generated counterfactual candidates that successfully achieves the desired prediction is selected as the final counterfactual example. Additionally, to enhance the quality of the explanations, we incorporate plausibility and actionability constraints to filter counterfactual candidates. These aspects will be discussed in detail in the following section.

3.3. Discussion on the desiderata of generated counterfactual examples

In this section, we will discuss the key qualities in our proposed approach that make generated counterfactual examples useful and meaningful, in terms of all desiderata introduced in Section 2.1.

3.3.1. Validity

In Algorithm 3, all generated counterfactual examples are aforehand verified by the prediction of random forest, i.e., only examples that match with the given target class are held. Therefore, the validity of generated counterfactual examples is surely guaranteed. Moreover, different from the Feature-Tweaking method, our method can generate counterfactual examples for all query instances with the help of active regions provided by the LIRE method.

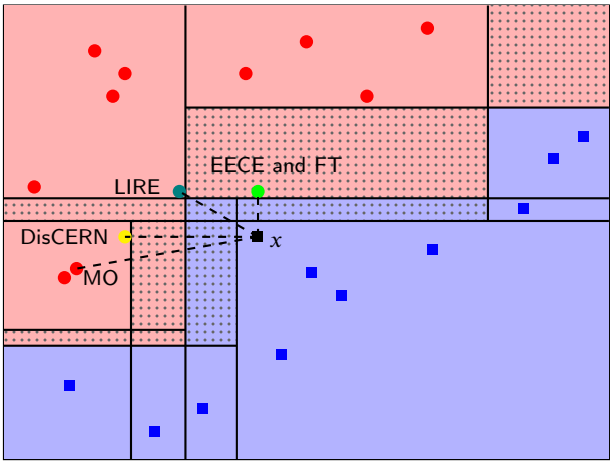


Figure 2: Counterfactual examples generated by different methods

3.3.2. Proximity

Fig. 2 illustrates counterfactual examples generated by different methods. For Mo and DisCERN methods, they highly depend on training instances, which is difficult to obtain proximal and sparse counterfactual explanations. With the same selection criteria, the counterfactual examples generated by our method will not be farther away from the query instance than those generated by LIRE and Feature-Tweaking methods, because our method has more candidates to consider on their respective bases. Although in the above example, the counterfactual example generated by our method is the same as that generated by Feature-Tweaking, on the whole, the counterfactual examples generated by our method will be more proximal than those generated by Feature-Tweaking.

It is difficult to find precisely the closest counterfactual example in a random forest for a query instance. Each instance is located in the intersection of decision paths of T decision trees. The number of such intersections is M^T, where T is the number of trees and M is the number of decision paths per tree (Sagi & Rokach, 2020), which is intractable for random forests of large scale. Our proposed method reduces the solution space, thus it has no guarantee to generate the closest counterfactual examples. However, in practice, we argue that when considering the plausibility and actionability of generated counterfactual examples, and efficiency of the generation process, the minimality of counterfactual example is a negotiable demand.

3.3.3. Plausibility

To enhance the plausibility of counterfactual examples generated by our approach, we integrate novelty and outlier detection techniques, such as Local Outlier Factor (LOF) (Breunig et al., 2000), Isolation Forest (Liu et al., 2008) and One-Class SVM (Alam et al., 2020), as a plausibility

indicator into the selection process for plausible counterfactual examples. LOF determines whether a sample deviates from the normal data distribution by calculating the density relationship between the sample and its neighboring points, thereby helping to identify counterfactual examples that conform to the global pattern but are unreasonable in local space. In contrast, Isolation Forest focuses mainly on global outliers and may not be able to identify minor local inconsistencies, while One-Class SVM is less adaptable to high-dimensional data and local anomalies. Besides, LOF also outperforms other methods by its speed[2]. Therefore, we selected LOF in our proposed method.

In detail, LOF identifies outliers under the assumption that an outlier will exhibit significantly lower density than its surroundings. Its computation for a given point x consists of the following steps:

1. Compute the reachability distance between x and each neighbor q of its k-nearest neighbors $N_k(x)$, defined as

$$rd_k(x, q) = \max(kd(q), d(x, q)), \tag{7}$$

where $kd(q)$ represents the distance from x to its k-th nearest neighbor. In our study, we set $k = 20$ in evaluation for all counterfactual example generation method and datasets, which is also the default parameter value in scikit-learn for the implementation of LOF. This choice is aligned with recommendations in literature, which achieves a good balance between capturing meaningful local density variations and providing stable LOF estimation, while also offering computational efficiency (Breunig et al., 2000; Goldstein & Uchida, 2016).

2. Calculate the local reachability density (LRD) of x, which is the inverse of the average reachability distance to its neighbors:

$$lrd_k(x) = \left(\frac{\sum_{q \in N_k(x)} rd_k(x, q)}{|N_k(x)|} \right)^{-1}. \tag{8}$$

3. Compute the LOF score as the ratio of the average LRD of x's neighbors to its own LRD:

$$lof_k(x) = \frac{\sum_{q \in N_k(x)} \frac{lrd_k(x)}{lrd_k(q)}}{|N_k(x)|}. \tag{9}$$

In practice, a point x with a LOF score around 1 indicates that its local density is comparable to its neighbors, i.e., the point lies in a dense region of the data. LOF scores significantly greater than 1 indicate increasingly lower local density, i.e., outlier-like behavior. We set the threshold $\delta = 1.5$ following common conventions in anomaly detection literature, where points with LOF ≤ 1.5 are considered to lie within relatively high-density regions and points with LOF > 1.5 are considered as mild to moderate outliers (Breunig et al., 2000). By selecting counterfactual candidates with LOF scores smaller than δ, we enhance the plausibility of provided counterfactual examples, ensuring that they remain within realistic data distributions. The plausibility of a counterfactual explainer can be evaluated as:

$$pl = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \mathbb{1}(lof_k(x_i') \leq \delta), \tag{10}$$

where x_i' represents the counterfactual example for the query instance x_i, $\mathbb{1}(\cdot)$ is the indicator function and x_i' is plausible if its LOF score is smaller than δ. It should be noted that the threshold δ can still be adjusted depending on the dataset characteristics or domain-specific requirements, allowing the plausibility criterion to remain flexible and context-aware.

3.3.4. Actionability

Actionability serves as an additional filtering criterion in our selection process of counterfactual candidates. Any counterfactual candidate that violates actionability constraints is excluded from consideration. For instance, a counterfactual requiring a decrease in age is unrealistic and therefore infeasible. To address different problems, specific actionability constraints must be incorporated into the selection process. However, it is important to note that imposing too many constraints may hinder the generation of desired counterfactual explanations.

As outlined in Algorithm 3, the selection process for the best counterfactual example considers three key criteria: proximity, plausibility, and actionability. In general, the prioritization follows a hierarchical order: actionability is assessed first, followed by plausibility, and finally, counterfactual examples with the best proximity to the query instance are selected. This structured approach ensures that the generated counterfactual examples remain high quality.

3.3.5. Diversity

Our proposed method naturally facilitates the generation of diverse counterfactual examples. Among all valid, plausible, and actionable counterfactual candidates, we can efficiently select multiple distinct ones that are as close as possible to the query instance while satisfying specific diversity criteria. For example, counterfactual examples can be chosen to modify different subsets of features, ensuring a broader range of actionable insights.

Given a set of generated counterfactual examples $cfs = \{cf_1, cf_2, \ldots, cf_k\}$ for a query instance $x \in \mathbb{R}^d$, we apply here two different diversity measures as follows.

- **Subset Diversity**. It is defined as the number of unique binary masks corresponding to feature-change patterns,

$$SD(x, cfs) = \left| \text{Unique} \left\{ \delta_1, \ldots, \delta_k \right\} \right|, \tag{11}$$

where $\delta_i \in \{0, 1\}^D$ and $\delta_{id} = \mathbb{1}[cf_{id} \neq x_d]$ indicating which features are changed in cf_i, and $|\cdot|$ is the cardinality of sets.

[2] See Scikit-learn Example for the comparison of different outlier detection methods.

Table 2
Example of counterfactual examples and corresponding modifications generated by different methods for Pima dataset.

	Pregnancies	Glucose	BP	ST	Insulin	BMI	DPF	Age
Patient-1	0	118	84	47	230	45.8	0.551	31
MO	**3**	**89**	**74**	**16**	**85**	**30.4**	0.551	**38**
	(↑ 3)	(↓ 29)	(↓ 10)	(↓ 31)	(↓ 145)	(↓ 15.4)		(↑ 7)
DisCERN	0	118	84	**16**	**85**	45.8	0.551	31
				(↓ 31)	(↓ 145)			
FT	0	118	84	47	230	45.8	**0.317**	31
							(↓ 0.234)	
LIRE	**3**	**89**	**74**	**17**	**85**	**30.445**	0.551	**38**
	(↑ 3)	(↓ 29)	(↓ 10)	(↓ 30)	(↓ 145)	(↓ 15.355)		(↑ 7)
EECE	0	118	**75**	47	230	45.8	0.551	31
			(↓ 9)					
Patient-2	5	155	84	44	545	38.7	0.619	34
MO	-	-	-	-	-	-	-	-
DisCERN	-	-	-	-	-	-	-	-
FT	5	155	84	**40.505**	545	38.7	**0.343**	**30.495**
				(↓ 3.495)			(↓ 0.276)	(↓ 3.505)
LIRE	**6**	**129**	**90**	**7**	**343**	**22.65**	0.619	**60**
	(↑ 1)	(↓ 26)	(↑ 6)	(↓ 37)	(↓ 202)	(↓ 16.05)		(↑ 26)
EECE	5	**127**	84	44	**352**	38.7	0.619	34
		(↓ 28)			(↓ 193)			

BP: Blood Pressure, ST: Skin Thickness, BMI: Body Mass Index, DPF: Diabetes Pedigree Function

- **Jaccard Diversity**. It is the average of Jaccard distance between all pairs of change masks:

$$JD(x, cfs) = \frac{1}{\binom{k}{2}} \sum_{1 \le i < j \le k} \left(1 - \frac{|\delta_i \cap \delta_j|}{|\delta_i \cup \delta_j|}\right). \quad (12)$$

For both of these two measures, a higher value indicates greater diversity in the set of k counterfactual examples generated.

3.3.6. *Efficiency*

In our proposed approach, complexity can be quantified based on the number of candidate regions, which comprise two components: (1) decision paths directly extracted from the trees in the given random forest and (2) active regions derived from training samples. Consequently, the complexity of our method is given by $O(TM + N_{ar})$, where T denotes the number of trees in the forest, M represents the average number of leaves per tree, and N_{ar} corresponds to the number of training samples.

In practice, we can further improve the efficiency of counterfactual example generation by filtering out far counterfactual candidates with the help of LIRE method, which will significantly reduce the call of random forest prediction function and the evaluation of plausibility. Moreover, since our candidate regions are all independent, it is natural and easy to parallelize the counterfactual example generation process.

3.3.7. *Stability*

In our approach, the stability of generated counterfactual explanations is guaranteed in the range of each candidate region. We take the range $I_d =]\underline{b}_d,\ \overline{b}_d]$ of a feature X_d of a candidate region $\boldsymbol{R}$ as an example. Consider a query instance x_1 and its neighbor x_2. If x_{1d} and x_{2d} are both included in I_d, their necessary modifications on feature X_d to the generated counterfactual example in the region $\boldsymbol{R}$ are both equal to zero. If $x_{1d} \le \underline{b}_d$ and $x_{2d} \le \underline{b}_d$, the values of feature X_d of their counterfactual examples in the region $\boldsymbol{R}$ will be equal, i.e., equal to $\underline{b}_{id} + \epsilon$. Respectively, if $x_{1d} > \underline{b}_d$ and $x_{2d} > \underline{b}_d$, the values of feature X_d of their counterfactual examples in the region $\boldsymbol{R}$ will also be equal, i.e., equal to $\overline{b}_{id}$. In these two cases, the difference between their modifications is the same as the difference between themselves.

3.3.8. *Example case*

Table 2 provides an example of case-study on the Pima Indians Diabetes Database, which is a medical dataset used for predicting whether a patient has diabetes based on diagnostic measurements. It contains 768 records of Pima Indian women (age ≥ 21) with 8 numerical features, including times of pregnancy, blood glucose, blood pressure, skin thickness, insulin levels, body mass index, value of diabetes pedigree function and age. We trained a random forest based on it and generate counterfactual examples with different methods for patients having diabetes.

For practical considerations, we have specified that the value of the diabetes pedigree function is immutable, as it is

calculated based on family genetic information. In addition, the times of pregnancy and age are specified to only increase. It should be noted that the above-mentioned actionability is considered in our all implementations of the MO, DisCERN, LIRE and EECE methods. However, the implementation of FT provided in paper (Tolomei et al., 2017) does not consider these restrictions.

As can be seen from the two examples of the studies in Table 2, MO either changes numerous features for Patient-1 or fails to provide a counterfactual example because no cases with diabetes pedigree function values consistent with Patient-2 can be found in the dataset. DisCERN can indeed significantly reduce the number of modified features, on the premise that MO can provide a counterfactual example. The FT method does not consider actionability, thus producing counterfactual examples that violate common sense, such as changing the diabetes pedigree function value of Patient-1 and decreases the age of Patient-2. In the case of Patient-1, the LIRE method is very similar to MO. In the case of Patient-2, LIRE also changes a considerable number of features, and some of the changes are also inconceivable, such as increasing the age from 34 to 60. For both patients, our proposed EECE method not only fully respects the constraints of actionability, but also changes factors such as blood glucose, blood pressure, and insulin levels that can effectively treat diabetes. Moreover, the cost of our method is minimal compared to the counterfactual examples generated by other methods.

The above counterfactual explanations generated by EECE can be textually described to improve understandability. For Patient-1, the counterfactual explanation suggests: *"You are currently predicted to have diabetes. However, if your blood pressure were reduced by 9 units (from 84 to 75), the prediction would change to non-diabetic."* For Patient-2, a more complex adjustment is required. The description is as follows: *"You are currently predicted to have diabetes. If your glucose level were reduced by 28 units (from 155 to 127), and your insulin level were reduced by 193 units (from 545 to 352), the prediction would change to non-diabetic."*

In addition, we believe that counterfactual explanations can be effectively combined with other explanation methods, such as feature importance scores or partial dependence plots (PDPs) to provide a more comprehensive understanding of model behavior and the rationale behind the suggested modifications.

From the above discussions and case-study analysis, it is evident that our proposed method comprehensively accounts for the key desiderata of counterfactual explanations. By systematically incorporating validity, proximity, plausibility, actionability, and diversity into the selection process, our approach ensures the efficient generation of high-quality counterfactual examples for random forest models. A more wide and comprehensive evaluation will be provided in the following experimental Section 4.

Table 3
Datasets used in experiments.

#	Data name	n-features	n-samples
1	Adult	11	45222
2	Banknote	4	1372
3	Biodeg	41	1053
4	Breast-cancer	30	568
5	Compas	6	2652
6	German	24	1000
7	Heart	13	303
8	Heloc	23	10459
9	Liver	6	345
10	Magic	57	2300
11	Mammographic	5	830
12	Phishing	30	11054
13	Pima	8	768
14	Spam	57	4594
15	Wine	11	1599

4. Experiments

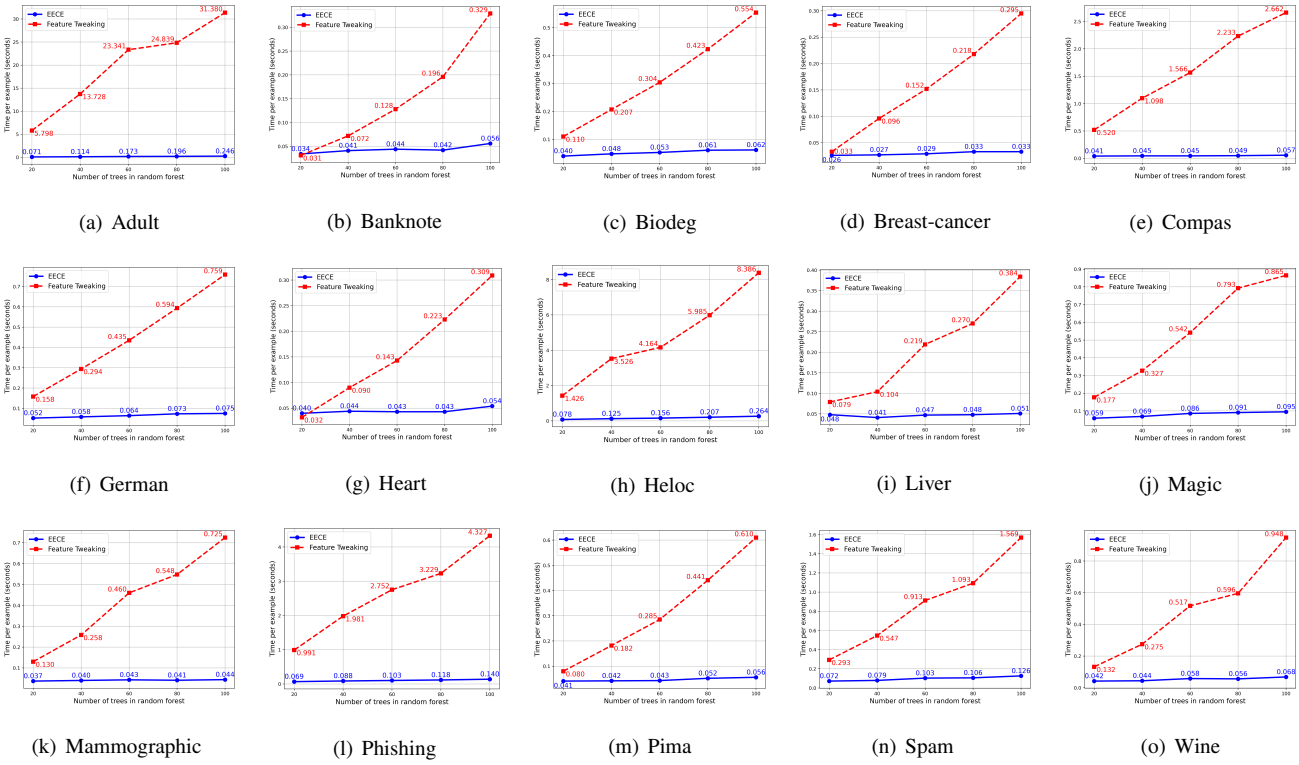
In practice, the validity and efficiency of counterfactual example generation methods are the most important requirements. For this reason, we selected the methods with both high validity and high efficiency mentioned in Section 2.2 as comparison methods in the experiments, including MO, DisCERN, and LIRE. In addition, we compared with Feature-Tweaking to demonstrate our improvements on it in terms of efficiency. The comparison among them was performed on 15 datasets from the UCI repository (Dua & Graff, 2017) shown in Table 3.

In this paper, we employed the scikit-learn implementation of Random Forests and the LOF algorithm (Pedregosa et al., 2011). For LOF, we used the default parameter settings: the number of neighbors was set to 20, and counterfactual examples were considered implausible if their LOF scores exceeded a threshold of 1.5. The model parameter settings for random forests will be explained in detail in the following sub-sections. Our experiments were conducted on a machine running Windows-10 system, equipped with an Intel(R) Core(TM) Ultra-7 165H processor. All of our code can be found on GitHub[3].

4.1. Efficiency of counterfactual example generation process

As mentioned in Table 1, the efficiency of the Feature-Tweaking method is not very high. Therefore, in this experiment, we first analyzed the advantages of our proposed method compared to the Feature-Tweaking method in terms of the average time required to generate a counterfactual example. Specifically, we examined the relationship between required time and two key factors: (1) the number of trees

[3]Code and datasets are available on https://github.com/Haifei-ZHANG/EECE



(a) Adult (b) Banknote (c) Biodeg (d) Breast-cancer (e) Compas

(f) German (g) Heart (h) Heloc (i) Liver (j) Magic

(k) Mammographic (l) Phishing (m) Pima (n) Spam (o) Wine

Figure 3: The efficiency of generating counterfactual examples with respect to the number of trees in the forest.

in the random forest and (2) the depth of the trees. In addition, we also compare with the MO, DisCERN and LIRE methods.

4.1.1. Experiment setting

To compare our approach with the Feature-Tweaking method, we conducted two sets of experiments. In the first set, considering the inefficiency of the Feature-Tweaking method, we fixed the maximum tree depth at 20 and varied the number of trees across $\{10, 40, 60, 80, 100\}$. In the second set, we fixed the number of trees at 100 and varied the tree depth across $\{5, 10, 15, 20\}$. For both cases, we reported the average computational time to generate a counterfactual example per test instance by optimizing the L_2^v distance measure as defined in Eq. (6). Moreover, we investigated the scalability of our EECE method across five large datasets (Adult, Compas, Heloc, Phishing and Spam) by varying the number of trees across $\{100, 200, 300, 400, 500\}$ and all trees were trained to the maximum depth. In this experiment, we used a standard data split, with 80% of the data allocated for training and the remaining 20% reserved for testing.

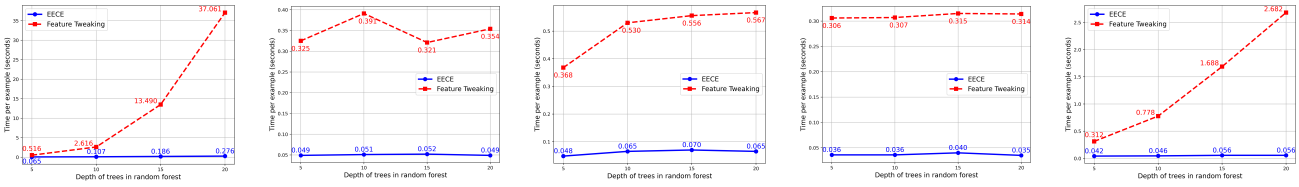
To compare our method with other efficient counterfactual generation methods, we standardized the experimental setup by fixing the number of trees to 100 and training all trees to their maximum depth across all datasets. The reported computational time is the average time required to generate one counterfactual example using each of the two distance metrics (L_1^{mad} and L_2^v) per test instance. That is, for each test sample, two counterfactual examples are generated, one optimized with L_1^{mad}, and the other with L_2^v, and the average time is computed over all such examples. The results were obtained by conducting a 10-fold cross-validation (Kohavi et al., 1995).

4.1.2. Result analysis

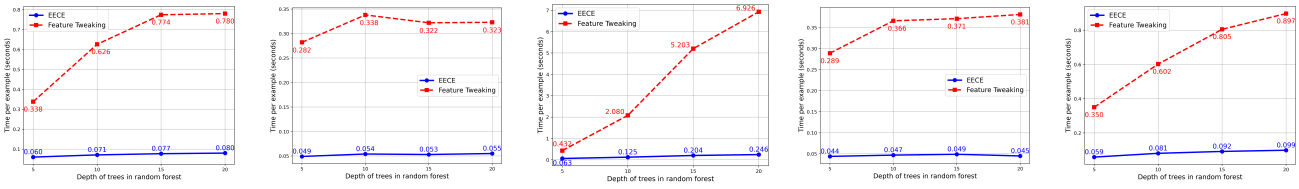
Figures 3 and 4 illustrate a comparison between our method and Feature-Tweaking across all datasets. As shown in Fig. 3, when tree depth is fixed, both methods exhibit an approximately linear relationship between required time and the number of trees. However, our method consistently requires significantly less time to generate counterfactual examples than Feature-Tweaking method. Fig. 4 further reveals that for certain datasets, such as Adult, the Feature-Tweaking method's required time grows exponentially with tree depth, reaching 30 seconds per counterfactual example when the forest has 100 trees with a maximum depth of 20, making it impractical for real-world applications. On other datasets, this exponential trend is less pronounced, because shallower trees are already sufficient for classification.

Fig. 5 shows that our proposed EECE method possesses a good scalability to large random forests and datasets. In forests consisting of 500 trees trained to maximum depth, EECE can always generate a counterfactual example in less than two seconds even within one second when dealing with datasets of more than 10k samples like Adult, Heloc and Phishing.

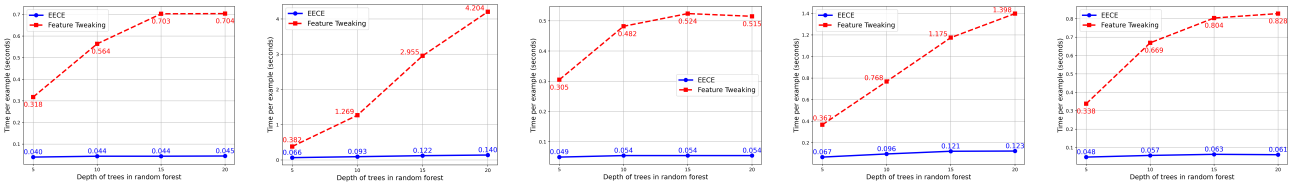
Table 4 compares the efficiency of our method with MO, DisCERN, and LIRE. LIRE is the fastest method across



(a) Adult (b) Banknote (c) Biodeg (d) Breast-cancer (e) Compas

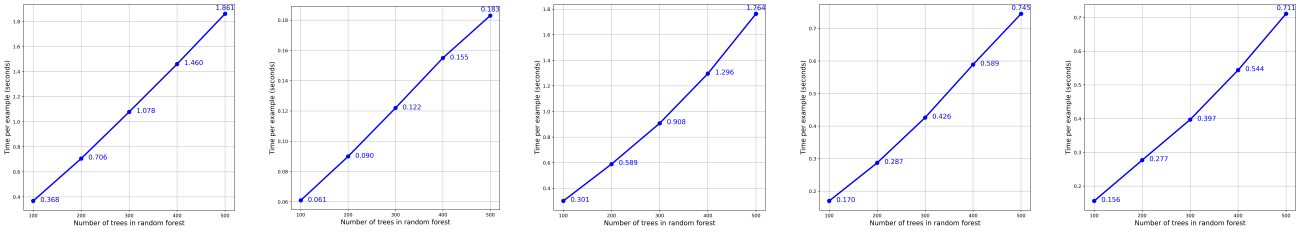


(f) German (g) Heart (h) Heloc (i) Liver (j) Magic



(k) Mammographic (l) Phishing (m) Pima (n) Spam (o) Wine

Figure 4: The efficiency of generating counterfactual examples with respect to the depth of trees in the forest.



(a) Adult (b) Compas (c) Heloc (d) Phishing (e) Spam

Figure 5: The efficiency of generating counterfactual examples for large datasets with respect to the larger number of trees in the forest. Each tree is trained to its maximum depth.

all datasets, particularly excelling on large datasets where many data points fall into the same active region, reducing redundant predictions compared to MO. DisCERN is only slightly less efficient than MO because we leveraged global feature importance in our implementation instead of instance-level feature importance evaluation. Our method EECE, which generates counterfactual examples from all leaves in the forest and active regions of LIRE, tends to be slower. However, on large dataset of Adult, EECE is slightly faster than MO with the help of pre-filtering based on LIRE method and parallelization. In summary, our method can generate a counterfactual example in less than 0.5 seconds, ensuring practical usability for large-scale random forests and datasets.

4.2. Effectiveness of generated counterfactual examples

In this experiment, we compared our proposed method with the MO, DisCERN, and LIRE methods in terms of effectiveness, i.e., the quality of the generated counterfactual examples, including proximity, sparsity, and plausibility. Since all methods inherently generate valid counterfactual examples, validity was not assessed. As the introduction of actionability constraints may lead to no-availability of counterfactual examples, the evaluation of actionability was not conducted.

4.2.1. Experiment setting

In this experiment, we trained random forests consisting of 100 decision trees on all datasets, where all trees were

Table 4
Required time to generate a counterfactual example (seconds) in forests of 100 tress trained to the maximum depth.

Data	MO	DisCERN	LIRE	EECE
Adult	0.4447 ± 0.0135	0.4687 ± 0.0229	**0.0988 ± 0.0083**	0.4061 ± 0.0300
Banknote	0.0070 ± 0.0024	0.0107 ± 0.0011	**0.0040 ± 0.0013**	0.0426 ± 0.0020
Biodeg	0.0118 ± 0.0007	0.1165 ± 0.0024	**0.0078 ± 0.0008**	0.0674 ± 0.0019
Breast-cancer	0.0070 ± 0.0018	0.0910 ± 0.0050	**0.0049 ± 0.0010**	0.0392 ± 0.0022
Compas	0.0253 ± 0.0014	0.0301 ± 0.0009	**0.0062 ± 0.0007**	0.0559 ± 0.0027
German	0.0129 ± 0.0009	0.0805 ± 0.0043	**0.0081 ± 0.0008**	0.0753 ± 0.0014
Heart	0.0050 ± 0.0030	0.0235 ± 0.0026	**0.0032 ± 0.0012**	0.0512 ± 0.0033
Heloc	0.1305 ± 0.0011	0.1680 ± 0.0020	**0.0635 ± 0.0014**	0.3331 ± 0.0388
Liver	0.0050 ± 0.0014	0.0117 ± 0.0015	**0.0031 ± 0.0012**	0.0497 ± 0.0041
Magic	0.0253 ± 0.0005	0.1636 ± 0.0018	**0.0132 ± 0.0006**	0.0969 ± 0.0011
Mammographic	0.0099 ± 0.0015	0.0135 ± 0.0021	**0.0024 ± 0.0011**	0.0440 ± 0.0020
Phishing	0.0972 ± 0.0013	0.1575 ± 0.0014	**0.0259 ± 0.0005**	0.1485 ± 0.0027
Pima	0.0072 ± 0.0024	0.0165 ± 0.0012	**0.0038 ± 0.0010**	0.0520 ± 0.0021
Spam	0.0493 ± 0.0012	0.1773 ± 0.0016	**0.0222 ± 0.0005**	0.1266 ± 0.0024
Wine	0.0152 ± 0.0006	0.0264 ± 0.0006	**0.0052 ± 0.0006**	0.0648 ± 0.0055

trained to their maximum depths. We tested the generation of counterfactual examples by optimizing the L_2^v and L_1^{mad} distance metrics separately and recorded the number of modified features to obtain counterfactual examples of desired predictions. Plausibility was also measured based on Eq. (10) in the same way. Note that in this experiment, we used 10-fold cross-validation (Kohavi et al., 1995) to address the randomness introduced by data splitting. Therefore, in Tables 4, 5, 6 and 7, we report the mean and standard deviation of each metric. This experiment simplified the consideration of actionability, thus we assumed that all features could be arbitrarily changed. The main reason for this consideration is that we have illustrated in example of Table 2: for MO and DisCERN methods, they often fail to generate counterfactual examples when considering actionability restriction.

4.2.2. Result analysis

Table 5 presents a comparison of different counterfactual generation methods in terms of proximity, measured using both L_2^v distance and L_1^{mad} distance. Proximity is a critical property in counterfactual explanations, as it ensures that the generated counterfactual instance remains as close as possible to the original input while still altering the predicted outcome (Wachter et al., 2017), i.e., requiring minimum effort to obtain desired predictions. Across all datasets, our proposed EECE method consistently achieves the lowest proximity values, significantly outperforming MO, DisCERN, and LIRE. Specifically:

- For L_2^v distance, EECE yields the smallest perturbations across all datasets, indicating that it generates counterfactual examples with minimal modifications. The reductions in distance are particularly notable for datasets such as German, Magic, and Spam, where EECE achieves up to 60% lower proximity values compared to the next best method.
- Similarly, for L_1^{mad} distance, EECE again demonstrates substantial improvements. The gains are most pronounced in datasets such as Heloc and Phishing, where EECE reduces the modification distance by over 80% relative to MO and DisCERN.

Similarly, Table 6 compares the sparsity of counterfactual explanations generated by different methods. Sparsity is a desirable property, as it ensures that only a minimal number of features are modified, improving interpretability and actionability. This is crucial as the human brain has a limited load capacity and we typically prefer explanations involving around three modified features (Miller, 2019). These results highlight that EECE generates closer counterfactual examples with fewer modifications, enhancing interpretability while reducing the risk of generating counterfactual examples that deviate too far from real-world distributions. We also note that, counterfactual examples optimized based on L_1^{mad} involve fewer modified features modified than those optimized by L_2^v, which is consistent with what we observed for the other three methods. However, on the Mammographic dataset, EECE shows the opposite behavior. This is because we used plausibility as a precondition when selecting counterfactual examples, which may limit the sparsity benefits of L_1^{mad}.

Table 7 presents the plausibility of counterfactual explanations generated by different methods, measured as the percentage of counterfactual examples that remain within a reasonable and realistic data distribution using LOF defined in Eq. (10). A high plausibility score indicates that the generated counterfactual examples resemble real-world instances, ensuring meaningful and realistic explanations.

Table 5
Comparison of different counterfactual generation approaches in terms of proximity (L_2^v and L_1^{mad}).

L_2^v distance evaluation				
Data	MO	DisCERN	LIRE	EECE
Adult	0.885 ± 0.060	0.850 ± 0.061	0.684 ± 0.049	**0.467 ± 0.043**
Banknote	1.097 ± 0.013	1.049 ± 0.014	1.036 ± 0.016	**0.723 ± 0.020**
Biodeg	3.618 ± 0.315	3.352 ± 0.320	2.644 ± 0.146	**1.974 ± 0.168**
Breast-cancer	4.130 ± 0.171	3.952 ± 0.145	3.341 ± 0.126	**3.104 ± 0.150**
Compas	0.375 ± 0.070	0.351 ± 0.071	0.280 ± 0.063	**0.115 ± 0.016**
German	3.477 ± 0.065	3.165 ± 0.094	2.269 ± 0.035	**0.945 ± 0.084**
Heart	2.862 ± 0.119	2.517 ± 0.120	1.987 ± 0.071	**0.860 ± 0.063**
Heloc	2.360 ± 0.270	2.110 ± 0.301	1.950 ± 0.202	**0.743 ± 0.140**
Liver	1.093 ± 0.117	0.902 ± 0.091	0.957 ± 0.099	**0.332 ± 0.075**
Magic	4.933 ± 0.320	4.670 ± 0.355	3.576 ± 0.184	**1.499 ± 0.121**
Mammographic	0.710 ± 0.366	0.608 ± 0.109	0.398 ± 0.083	**0.262 ± 0.089**
Phishing	3.305 ± 0.059	3.210 ± 0.058	1.767 ± 0.029	**0.935 ± 0.013**
Pima	1.424 ± 0.044	1.274 ± 0.063	1.321 ± 0.044	**0.680 ± 0.052**
Spam	4.693 ± 0.263	4.468 ± 0.237	3.497 ± 0.136	**1.311 ± 0.175**
Wine	1.484 ± 0.062	1.300 ± 0.070	1.463 ± 0.067	**0.678 ± 0.055**
L_1^{mad} distance evaluation				
Data	MO	DisCERN	LIRE	EECE
Adult	2.213 ± 0.217	2.082 ± 0.191	1.752 ± 0.172	**1.189 ± 0.155**
Banknote	2.209 ± 0.025	2.036 ± 0.037	1.975 ± 0.036	**1.027 ± 0.036**
Biodeg	23.519 ± 2.134	20.732 ± 2.035	14.896 ± 0.895	**9.515 ± 0.922**
Breast-cancer	24.543 ± 1.106	22.378 ± 0.961	17.291 ± 0.963	**15.840 ± 0.978**
Compas	1.139 ± 0.286	1.046 ± 0.287	0.892 ± 0.271	**0.243 ± 0.048**
German	11.193 ± 0.430	8.975 ± 0.456	7.594 ± 0.284	**1.815 ± 0.238**
Heart	7.938 ± 0.386	6.318 ± 0.427	5.780 ± 0.244	**1.596 ± 0.180**
Heloc	12.289 ± 1.901	10.026 ± 1.889	9.055 ± 1.131	**2.051 ± 0.421**
Liver	3.059 ± 0.336	2.043 ± 0.166	2.628 ± 0.290	**0.681 ± 0.181**
Magic	49.603 ± 4.332	46.957 ± 4.377	31.812 ± 2.104	**11.509 ± 1.379**
Mammographic	1.367 ± 0.445	1.316 ± 0.445	0.876 ± 0.102	**0.607 ± 0.091**
Phishing	9.714 ± 0.396	9.160 ± 0.357	5.311 ± 0.205	**1.720 ± 0.075**
Pima	4.020 ± 0.101	3.329 ± 0.148	3.632 ± 0.100	**1.310 ± 0.128**
Spam	48.076 ± 3.431	45.433 ± 3.360	31.813 ± 1.355	**10.124 ± 1.244**
Wine	5.396 ± 0.221	4.304 ± 0.266	5.123 ± 0.222	**1.285 ± 0.104**

- EECE achieves the highest plausibility scores across 13 out of 15 datasets.
- MO and DisCERN perform well overall, but their plausibility scores show variability. DisCERN, in particular, has a significantly lower plausibility score in the Breast-cancer dataset (80.2%), indicating that it sometimes generates unrealistic counterfactual examples.
- LIRE struggles with plausibility as well, performing worse than MO in most datasets, with particularly low scores in Adult (88.8%), Breast-cancer (88.8%), and Compas (86.5%).

The experimental results demonstrate that EECE consistently outperforms existing methods in terms of proximity, sparsity, and plausibility, i.e., the method effectively generates counterfactual explanations that are closer to the original instance, require fewer feature modifications, and remain within the realm of realistic data distributions. These improvements are primarily attributed to EECE's novel integration of decision tree structures, active regions, and plausibility filtering mechanisms. By leveraging the leaf structure of decision trees, EECE efficiently identifies counterfactual candidates while maintaining computational feasibility. Additionally, its actionability constraints ensure that the generated explanations provide meaningful guidance for end-users. Overall, EECE sets a new benchmark for counterfactual explanation generation for random forests, offering

Table 6
Comparison of different counterfactual generation approaches in terms of sparsity when optimizing L_2^v and L_1^{mad}, respectively.

When optimizing L_2^v				
Data	MO	DisCERN	LIRE	EECE
Adult	2.835 ± 0.167	2.485 ± 0.149	3.037 ± 0.184	**2.255 ± 0.130**
Banknote	3.999 ± 0.003	3.511 ± 0.055	3.771 ± 0.039	**1.845 ± 0.073**
Biodeg	20.343 ± 0.310	14.694 ± 0.263	18.759 ± 0.280	**10.349 ± 0.886**
Breast-cancer	29.979 ± 0.019	26.806 ± 0.717	20.660 ± 0.773	**18.050 ± 1.429**
Compas	1.690 ± 0.031	1.440 ± 0.020	1.766 ± 0.051	**1.374 ± 0.043**
German	6.984 ± 0.141	4.832 ± 0.271	7.550 ± 0.097	**2.670 ± 0.157**
Heart	6.581 ± 0.226	4.812 ± 0.272	7.079 ± 0.206	**2.323 ± 0.186**
Heloc	15.533 ± 0.300	11.916 ± 0.630	15.457 ± 0.383	**3.656 ± 0.209**
Liver	5.258 ± 0.106	3.475 ± 0.214	5.232 ± 0.084	**1.873 ± 0.158**
Magic	16.355 ± 0.640	13.700 ± 0.561	17.001 ± 0.523	**5.473 ± 0.671**
Mammographic	1.710 ± 0.092	1.581 ± 0.166	1.816 ± 0.105	**1.242 ± 0.069**
Phishing	3.705 ± 0.072	3.414 ± 0.070	3.996 ± 0.088	**1.765 ± 0.028**
Pima	6.760 ± 0.092	5.289 ± 0.196	6.733 ± 0.108	**3.015 ± 0.188**
Spam	16.223 ± 0.310	13.642 ± 0.275	16.761 ± 0.294	**4.938 ± 0.278**
Wine	10.271 ± 0.046	7.774 ± 0.237	10.056 ± 0.032	**3.123 ± 0.129**
When optimizing L_1^{mad}				
Data	MO	DisCERN	LIRE	EECE
Adult	2.604 ± 0.162	2.327 ± 0.133	2.775 ± 0.173	**2.235 ± 0.117**
Banknote	3.998 ± 0.003	3.506 ± 0.038	3.293 ± 0.092	**1.404 ± 0.031**
Biodeg	19.775 ± 0.332	14.286 ± 0.228	18.121 ± 0.255	**7.985 ± 0.743**
Breast-cancer	29.972 ± 0.025	26.613 ± 0.751	19.139 ± 0.677	**16.480 ± 1.240**
Compas	1.533 ± 0.051	1.343 ± 0.043	1.620 ± 0.045	**1.256 ± 0.032**
German	6.624 ± 0.142	4.553 ± 0.271	6.849 ± 0.144	**2.113 ± 0.155**
Heart	6.321 ± 0.219	4.743 ± 0.330	6.475 ± 0.199	**2.014 ± 0.161**
Heloc	14.315 ± 0.264	10.787 ± 0.454	14.157 ± 0.219	**2.573 ± 0.074**
Liver	4.956 ± 0.138	3.112 ± 0.218	4.960 ± 0.106	**1.577 ± 0.126**
Magic	14.277 ± 0.559	12.342 ± 0.554	13.990 ± 0.375	**4.560 ± 0.457**
Mammographic	1.458 ± 0.102	1.367 ± 0.109	1.670 ± 0.091	**1.333 ± 0.088**
Phishing	3.564 ± 0.089	3.285 ± 0.083	3.692 ± 0.086	**1.699 ± 0.044**
Pima	6.546 ± 0.101	5.086 ± 0.181	6.556 ± 0.080	**2.015 ± 0.084**
Spam	14.086 ± 0.285	12.182 ± 0.279	13.832 ± 0.294	**4.235 ± 0.230**
Wine	10.068 ± 0.047	7.616 ± 0.245	9.829 ± 0.040	**1.952 ± 0.085**

a more interpretable, efficient, and effective approach to understanding complex tree ensemble decisions.

4.3. Diversity of generated counterfactual examples

Since in some situations, we may want to obtain several diverse counterfactual examples as possible choices for a given instance. In this experiment, we compared our proposed method with the MO, DisCERN, and LIRE methods in terms of the diversity of generated counterfactual examples, i.e., the number of unique feature-changed subsets and the average pairwise Jaccard distance, defined in Eq. (11) and Eq. (12), respectively.

4.3.1. Experiment setting

In this experiment, we trained random forests with 100 decision trees on all datasets, with each tree grown to its maximum depth. For each test instance, three counterfactual examples were generated by optimizing the L_1^{mad} distance, which encourages sparser counterfactual examples. Generating three examples strikes a balance between informativeness and interpretability: it provides multiple plausible alternatives without overwhelming the end user with excessive information (Pu & Chen, 2007). Diversity was then assessed within each local set of explanations. We performed 10-fold cross-validation and reported both the mean and standard deviation of the diversity scores.

Table 7
Plausibility of generated counterfactuals (%).

Data	MO	DisCERN	LIRE	EECE
Adult	95.8 ± 2.1	95.5 ± 2.1	88.8 ± 3.9	**98.6 ± 0.9**
Banknote	98.7 ± 0.9	90.9 ± 2.9	96.1 ± 2.1	**99.9 ± 0.2**
Biodeg	**95.6 ± 1.9**	94.7 ± 1.7	93.1 ± 1.9	95.2 ± 1.7
Breast-cancer	**99.1 ± 1.3**	80.2 ± 19.4	88.8 ± 6.0	90.5 ± 5.7
Compas	89.7 ± 3.5	89.2 ± 3.4	86.5 ± 2.3	**92.0 ± 2.3**
German	99.7 ± 0.5	98.5 ± 1.0	99.9 ± 0.3	**100 ± 0.0**
Heart	99.2 ± 1.4	98.8 ± 1.6	98.8 ± 2.2	**99.5 ± 1.6**
Heloc	97.4 ± 3.1	95.6 ± 2.8	97.6 ± 2.6	**100 ± 0.0**
Liver	96.8 ± 2.7	94.5 ± 4.0	95.7 ± 3.2	**98.7 ± 1.7**
Magic	95.6 ± 2.1	93.8 ± 2.3	94.5 ± 2.7	**98.9 ± 0.8**
Mammographic	98.8 ± 1.5	98.6 ± 1.8	98.9 ± 1.9	**99.4 ± 1.5**
Phishing	96.8 ± 1.0	96.4 ± 1.0	94.6 ± 2.0	**97.8 ± 0.8**
Pima	99.0 ± 1.0	98.4 ± 1.6	98.9 ± 0.9	**99.5 ± 0.6**
Spam	97.0 ± 0.6	94.8 ± 1.5	94.6 ± 2.2	**98.7 ± 1.0**
Wine	99.0 ± 0.7	98.5 ± 0.8	98.8 ± 0.6	**99.8 ± 0.3**

Table 8
Diversity of generated counterfactuals in terms of Jaccard distance.

Data	MO	DisCERN	LIRE	EECE
Adult	0.371 ± 0.013	**0.411 ± 0.021**	0.399 ± 0.010	0.403 ± 0.034
Banknote	0.001 ± 0.001	0.039 ± 0.011	0.155 ± 0.014	**0.508 ± 0.007**
Biodeg	0.201 ± 0.012	0.362 ± 0.017	0.310 ± 0.009	**0.473 ± 0.019**
Breast-cancer	0.001 ± 0.001	0.071 ± 0.015	0.300 ± 0.014	**0.326 ± 0.022**
Compas	0.240 ± 0.025	0.286 ± 0.034	0.408 ± 0.032	**0.484 ± 0.030**
German	0.475 ± 0.012	**0.627 ± 0.026**	0.478 ± 0.013	0.578 ± 0.018
Heart	0.341 ± 0.015	0.468 ± 0.042	0.367 ± 0.022	**0.563 ± 0.032**
Heloc	0.275 ± 0.008	0.425 ± 0.011	0.305 ± 0.009	**0.583 ± 0.020**
Liver	0.202 ± 0.024	0.409 ± 0.052	0.225 ± 0.017	**0.607 ± 0.015**
Magic	0.147 ± 0.012	0.214 ± 0.022	0.328 ± 0.011	**0.532 ± 0.022**
Mammographic	0.303 ± 0.052	0.338 ± 0.049	**0.351 ± 0.047**	0.347 ± 0.063
Phishing	0.258 ± 0.022	0.265 ± 0.024	0.586 ± 0.027	**0.594 ± 0.031**
Pima	0.143 ± 0.008	0.281 ± 0.018	0.170 ± 0.010	**0.534 ± 0.017**
Spam	0.129 ± 0.011	0.233 ± 0.015	0.302 ± 0.014	**0.624 ± 0.018**
Wine	0.116 ± 0.006	0.230 ± 0.013	0.159 ± 0.007	**0.571 ± 0.012**

4.3.2. Result analysis

In terms of Jaccard diversity, which measures the dissimilarity between the feature sets changed in different counterfactuals, EECE achieves the highest values across nearly all datasets, except for the German, Adult, and Mammographic dataset, see Table 8. For instance, EECE significantly outperforms the other methods in datasets such as banknote (0.508), spam (0.624), and pima (0.534), indicating that the counterfactuals it generates tend to modify different sets of features, providing more diverse decision paths. This is particularly important in user-facing applications where actionable and varied options are desired. In contrast, MO and DisCERN often show lower Jaccard diversity scores, suggesting that they generate counterfactuals with overlapping changed features. LIRE typically performs moderately, usually ranking second or third across datasets.

The findings for subset diversity in Table 9, which measures the number of differing features between pairs of counterfactuals, reveal a more nuanced picture. Contrary to its performance in Jaccard diversity, EECE does not consistently achieve the highest subset diversity scores. Specifically, EECE achieves the highest subset diversity in 10 out of 15 datasets, but is only slightly outperformed by LIRE across other datasets. For example, in datasets such as adult, biodeg, and breast-cancer, LIRE produces slightly more diverse changes in terms of feature count, resulting in marginally higher subset diversity scores. Nevertheless, the differences are generally small, and EECE remains highly competitive.

Table 9
Diversity of generated counterfactuals in terms of the number of unique subsets changed.

Data	MO	DisCERN	LIRE	EECE
Adult	2.401 ± 0.048	2.369 ± 0.064	**2.482 ± 0.034**	2.446 ± 0.097
Banknote	1.008 ± 0.006	1.179 ± 0.045	1.785 ± 0.047	**2.905 ± 0.021**
Biodeg	2.860 ± 0.035	2.874 ± 0.040	**2.973 ± 0.012**	2.862 ± 0.033
Breast-cancer	1.042 ± 0.029	2.231 ± 0.151	**2.898 ± 0.048**	2.825 ± 0.065
Compas	1.722 ± 0.074	1.712 ± 0.078	2.087 ± 0.092	**2.289 ± 0.096**
German	2.976 ± 0.021	2.917 ± 0.030	2.979 ± 0.017	**2.992 ± 0.013**
Heart	2.900 ± 0.038	2.860 ± 0.044	2.915 ± 0.048	**2.961 ± 0.042**
Heloc	2.947 ± 0.027	2.960 ± 0.028	2.982 ± 0.017	**3.000 ± 0.000**
Liver	2.299 ± 0.105	2.497 ± 0.100	2.420 ± 0.112	**2.980 ± 0.032**
Magic	2.305 ± 0.083	2.392 ± 0.070	2.763 ± 0.046	**2.844 ± 0.045**
Mammographic	1.929 ± 0.165	1.928 ± 0.154	**2.056 ± 0.147**	2.026 ± 0.175
Phishing	1.712 ± 0.064	1.679 ± 0.063	**2.928 ± 0.027**	2.821 ± 0.046
Pima	2.245 ± 0.077	2.552 ± 0.057	2.379 ± 0.086	**2.962 ± 0.022**
Spam	2.495 ± 0.056	2.700 ± 0.070	2.850 ± 0.051	**2.984 ± 0.015**
Wine	2.194 ± 0.059	2.408 ± 0.082	2.582 ± 0.054	**2.989 ± 0.016**

It offers a strong balance between subset-level variability and the disjointness of feature changes, as reflected by its consistently superior Jaccard scores. This makes EECE a robust method for generating counterfactuals that are not only diverse in terms of the degree of modification, but also relatively competitive in how many features are altered across instances.

Taken together, these findings suggest that EECE offers a strong compromise between disjointness and richness of feature variation, providing a robust baseline for generating informative, multi-faceted counterfactual explanations.

5. Conclusion

In this paper, we introduced the Efficient and Effective Counterfactual Explanation (EECE) generator, a novel method for generating counterfactual explanations for random forests. Addressing the inherent challenges posed by the model's complex classification boundaries and non-differentiability, EECE enhances the Feature-Tweaking approach by leveraging an improved tree representation and a more effective counterfactual search strategy. By integrating active regions from LIRE alongside leaf nodes, our method ensures the availability and validity of counterfactual examples for all input instances while maintaining computational efficiency. EECE improves essential properties such as proximity, sparsity, plausibility and diversity. Furthermore, it can easily integrate the considerations of actionability and diversity into the generation of counterfactual examples. Our results demonstrate that EECE strikes a balance between efficiency and effectiveness, making it a promising solution for generating counterfactual explanations that are more proximal, plausible, and actionable than existing methods for random forests.

Despite its advantages, EECE has certain limitations that open avenues for future research:

1. Beyond random forests: While EECE is tailored for random forests (where trees are independent), we plan to extend it to other tree-based models where trees are dependent such as XGBoost, LightGBM, CatBoost, etc.
2. Integration of causal constraints: Introducing causality constraints into counterfactual generation could further enhance plausibility and actionability.
3. User-centric evaluation: Beyond technical metrics, future work could investigate the human interpretability of EECE-generated counterfactual examples through user studies in real-world applications.

CRediT authorship contribution statement

Haifei Zhang: Conceptualization, Methodology, Writing - Original draft preparation, Writing - Review and Editing, Software. **Jinfeng Zhong:** Methodology, Writing - Review and Editing, Software.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partly done while H. Zhang was a research assistant at the University of Technology of Compiègne, and we thank for its support. This work also received support from the Jean Monnet University Foundation for New Lecturers.

Data availability

All code, datasets and experimental results of this paper are available on https://github.com/Haifei-ZHANG/EECE.

References

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, *6*, 52138–52160. doi:10.1109/ACCESS.2018.2870052.

Alam, S., Sonbhadra, S. K., Agarwal, S., & Nagabhushan, P. (2020). One-class support vector classifiers: A survey. *Knowledge-Based Systems*, *196*, 105754. doi:10.1016/j.knosys.2020.105754.

Apley, D. W., & Zhu, J. (2020). Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *82*, 1059–1086. doi:10.1111/rssb.12377.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R. et al. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, *58*, 82–115. doi:10.1016/j.inffus.2019.12.012.

Blanchart, P. (2021). An exact counterfactual-example-based approach to tree-ensemble models interpretability. *arXiv preprint*, . doi:10.48550/arXiv.2105.14820.

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5–32. doi:10.1023/A:1010933404324.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (pp. 93–104). doi:10.1145/342009.335388.

Carreira-Perpinán, M. A., & Hada, S. S. (2023). Very fast, approximate counterfactual explanations for decision forests. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 6935–6943). volume 37. doi:10.1609/aaai.v37i6.25848.

Cui, Z., Chen, W., He, Y., & Chen, Y. (2015). Optimal action extraction for random forests and boosted trees. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 179–188). doi:10.1145/2783258.2783281.

Deng, H. (2019). Interpreting tree ensembles with intrees. *International Journal of Data Science and Analytics*, *7*, 277–287. doi:10.1007/s41060-018-0144-8.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint*, . doi:10.48550/arXiv.1702.08608.

Dua, D., & Graff, C. (2017). UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

Fernández, R. R., De Diego, I. M., Aceña, V., Fernández-Isabel, A., & Moguerza, J. M. (2020). Random forest explainability using counterfactual sets. *Information Fusion*, *63*, 196–207. doi:10.1016/j.inffus.2020.07.001.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, *29*, 1189–1232. doi:10.1214/aos/1013203451.

Goldstein, M., & Uchida, S. (2016). A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, *11*, e0152173. doi:10.1371/journal.pone.0152173.

Guidotti, R. (2024). Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, *38*, 2770–2824. doi:10.1007/s10618-022-00831-6.

Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., & Giannotti, F. (2018). Local rule-based explanations of black box decision systems. *arXiv preprint*, . doi:10.48550/arXiv.1805.10820.

Gunning, D., Vorm, E., Wang, Y., & Turek, M. (2021). Darpa's explainable ai (xai) program: A retrospective. *Authorea Preprints*, . doi:10.1002/ail2.61.

Haddouchi, M., & Berrado, A. (2019). A survey of methods and tools used for interpreting random forest. In *2019 1st International Conference on Smart Systems and Data Science (ICSSD)* (pp. 1–6). IEEE. doi:10.1109/ICSSD47982.2019.9002770.

Kanamori, K., Takagi, T., Kobayashi, K., & Arimura, H. (2020). Dace: Distribution-aware counterfactual explanation by mixed-integer linear optimization. In *IJCAI* (pp. 2855–2862). doi:10.24963/ijcai.2020/395.

Karimi, A.-H., Barthe, G., Balle, B., & Valera, I. (2020). Model-agnostic counterfactual explanations for consequential decisions. In *International conference on artificial intelligence and statistics* (pp. 895–905). PMLR. URL: https://proceedings.mlr.press/v108/karimi20a.html.

Kim, B., Khanna, R., & Koyejo, O. O. (2016). Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, *29*. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/5680522b8e2bb01943234bce7bf84534-Paper.pdf.

Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai* (pp. 1137–1145). Montreal, Canada volume 14. URL: https://www.ijcai.org/Proceedings/95-2/Papers/016.pdf.

Kozielski, M., Sikora, M., & Wawrowski, Ł. (2025). Towards consistency of rule-based explainer and black box model–fusion of rule induction and xai-based feature importance. *Knowledge-Based Systems*, (p. 113092). doi:10.1016/j.knosys.2025.113092.

Li, J., & Zaiane, O. R. (2017). Exploiting statistically significant dependent rules for associative classification. *Intelligent data analysis*, *21*, 1155–1172. doi:10.3233/IDA-163141.

Lipton, Z. C. (2018). The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, *16*, 31–57. doi:10.1145/3236386.3241340.

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *2008 eighth ieee international conference on data mining* (pp. 413–422). IEEE. doi:10.1109/ICDM.2008.17.

Lucic, A., Oosterhuis, H., Haned, H., & de Rijke, M. (2022). Focus: Flexible optimizable counterfactual explanations for tree ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 5313–5322). volume 36. doi:10.1609/aaai.v36i5.20468.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (p. 4768–4777). Curran Associates, Inc. volume 30. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, *267*, 1–38. doi:10.1016/j.artint.2018.07.007.

Molnar, C. (2020). *Interpretable machine learning*. Lulu. com. URL: https://christophm.github.io/interpretable-ml-book/.

Motallebi, M., Anik, M. T. A., & Zaïane, O. R. (2023). Explaining decisions of black-box models using barbe. In *International Conference on Database and Expert Systems Applications* (pp. 82–97). Springer. doi:10.1007/978-3-031-39821-6_6.

Parmentier, A., & Vidal, T. (2021). Optimal counterfactual explanations in tree ensembles. In *International conference on machine learning* (pp. 8422–8431). PMLR. URL: https://proceedings.mlr.press/v139/parmentier21a.html.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. URL: https://scikit-learn.org/stable/index.html.

Pu, P., & Chen, L. (2007). Trust-inspiring explanation interfaces for recommender systems. *Knowledge-Based Systems*, *20*, 542–556. doi:10.1016/j.knosys.2007.04.004.

Ribeiro, M. T., Singh, S., & Guestrin, C. (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* (pp. 1135–1144). doi:10.1145/2939672.2939778.

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., & Zhong, C. (2022). Interpretable machine learning: Fundamental principles and 10

grand challenges. *Statistic Surveys*, *16*, 1–85. doi:10.1214/21-SS133.

Saeed, W., & Omlin, C. (2023). Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. *Knowledge-based systems*, *263*, 110273. doi:10.1016/j.knosys.2023.110273.

Sagi, O., & Rokach, L. (2020). Explainable decision forest: Transforming a decision forest into an interpretable tree. *Information Fusion*, *61*, 124–138. doi:10.1016/j.inffus.2020.03.013.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2020). Grad-cam: visual explanations from deep networks via gradient-based localization. *International journal of computer vision*, *128*, 336–359. doi:10.1007/s11263-019-01228-7.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint*, . doi:10.48550/arXiv.1312.6034.

Tolomei, G., Silvestri, F., Haines, A., & Lalmas, M. (2017). Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 465–474). doi:10.1145/3097983.3098039.

Verma, S., Boonsanong, V., Hoang, M., Hines, K., Dickerson, J., & Shah, C. (2024). Counterfactual explanations and algorithmic recourses for machine learning: A review. *ACM Computing Surveys*, *56*. doi:10.1145/3677119.

Voigt, P., & Von dem Bussche, A. (2017). The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, *10*, 10–5555. doi:10.1007/978-3-319-57959-7.

Wachter, S., Mittelstadt, B., & Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, *31*, 841. doi:10.2139/ssrn.3063289.

Wijekoon, A., & Wiratunga, N. (2023). A user-centred evaluation of discern: Discovering counterfactuals for code vulnerability detection and correction. *Knowledge-Based Systems*, *278*, 110830. doi:10.1016/j.knosys.2023.110830.

Wiratunga, N., Wijekoon, A., Nkisi-Orji, I., Martin, K., Palihawadana, C., & Corsar, D. (2021). Discern: Discovering counterfactual explanations using relevance features from neighbourhoods. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)* (pp. 1466–1473). IEEE. doi:10.1109/ICTAI52525.2021.00233.