UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

# Actionable Interpretability through Optimizable Counterfactual Explanations for Tree Ensembles

Lucic, A.; Oosterhuis, H. ; Haned, H.; de Rijke, M.

**Publication date**
2019

**Document Version**
Submitted manuscript

**License**
Unspecified

Link to publication

**Citation for published version (APA):**
Lucic, A., Oosterhuis, H., Haned, H., & de Rijke, M. (2019). *Actionable Interpretability through Optimizable Counterfactual Explanations for Tree Ensembles*. (v1 ed.) ArXiv.

**General rights**
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

**Disclaimer/Complaints regulations**
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

*UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)*

Download date:10 Oct 2026

# Actionable Interpretability through Optimizable Counterfactual Explanations for Tree Ensembles

**Ana Lucic**[1] and **Harrie Oosterhuis**[2] and **Hinda Haned**[3] and **Maarten de Rijke**[4]

arXiv:1911.12199v1 [cs.LG] 27 Nov 2019

**Abstract.** Counterfactual explanations help users understand why machine learned models make certain decisions, and more specifically, how these decisions can be changed. In this work, we frame the problem of finding counterfactual explanations – the minimal perturbation to an input such that the prediction changes – as an optimization task. Previously, optimization techniques for generating counterfactual examples could only be applied to differentiable models, or alternatively via query access to the model by estimating gradients from randomly sampled perturbations. In order to accommodate non-differentiable models such as tree ensembles, we propose using probabilistic model approximations in the optimization framework. We introduce a novel approximation technique that is effective for finding counterfactual explanations while also closely approximating the original model. Our results show that our method is able to produce counterfactual examples that are closer to the original instance in terms of Euclidean, Cosine, and Manhattan distance compared to other methods specifically designed for tree ensembles.

## 1 INTRODUCTION

Model interpretability has become an important problem in machine learning [7]. As machine learned models are prominently applied and their behavior has a substantial effect on the general population, there is an increased demand for understanding what contributes to their predictions. Furthermore, regulations on algorithmic decision making have been introduced, such as the General Data Protection Regulation (GDPR) in Europe, which gives an individual the right to an explanation for algorithmic decisions [9]. Consequently, this makes the interpretability problem crucial for organizations that wish to adopt more data-driven decision-making processes.

Explanations can be provided in a global or local manner. Global explanations give insight into how the model that generated the prediction performs as a whole, while local explanations are specific to individual predictions [13] – in this paper, we focus on the latter. Prominent methods for generating local explanations include feature importances [20, 24], influential training points [17, 26], decision sets [12, 18] and counterfactual examples [11, 29, 32].

The advantage of the latter is that the explanations are *actionable*: they inform users of what contributed to an individual prediction such that they understand how to change the input in order to change the prediction. For instance, a user may be denied a loan based on the prediction of a machine learning (ML) model used by their bank. A counterfactual explanation for this user informs them about the changes required in their profile that would result in an approval for the loan (i.e., change the outcome of the ML model). A counterfactual explanation could, e.g., be "*Had your income been €*1000 *higher, you would have been approved for the loan*." In this work, we focus on generating *optimal* counterfactual explanations – the *minimal* changes required to change the outcome. We do not want to present explanations that suggest the user should change more than what is minimally necessary to change the outcome. Counterfactual explanations naturally provide users with an understanding that is actionable, and therefore can be an effective tool for dealing with algorithmic decision making systems.

Counterfactual explanations are based on counterfactual examples: generated instances that are close to an existing instance but have a different prediction. The difference between the original instance and the counterfactual example is the counterfactual explanation. Previous work on generating counterfactual explanations involves perturbing features based on how much they contribute to the prediction [23], solving a series of satisfiability problems [15], framing the problem as an inverse classification task [19], using mixed integer linear programming [25] or perturbing features along a desired outcome path (for tree ensembles) [29]. Our work differs from these since it casts the problem as an optimization task – we want to find the best feature values for a counterfactual example (from which we obtain a counterfactual explanation) rather than rely on heuristics.

Previous work on generating counterfactual explanations via optimization assumes that the underlying machine learning models are differentiable [11, 32], which excludes an important class of widely applied and highly effective non-differentiable models: tree ensembles. Our work relaxes this assumption by probabilistically approximating tree ensembles in order to make them differentiable. Given a trained tree-based model $\mathcal{M}$, we probabilistically approximate $\mathcal{M}$ by replacing each split in each tree with a sigmoid function centred at the splitting threshold. This results in a differentiable version of $\mathcal{M}$ from which we find the feature values required for counterfactual examples via gradient descent. Given an instance $x$ and a trained tree-based model $\mathcal{M}$, our approximation allows us to generate a counterfactual example $\bar{x}$ based on a minimal perturbation of $x$ such that the prediction changes: $y_x \neq y_{\bar{x}}$, where $y_x$, $y_{\bar{x}}$ are the labels $\mathcal{M}$ assigns to $x$ and $\bar{x}$, respectively.

This leads us to our main research question:

*Are counterfactual examples generated by our approximation method closer to the original input instances than those generated by existing methods?*

We want to minimize the distance between the counterfactual example and the original input in order to ensure our explanation is

[1] University of Amsterdam, The Netherlands, email: a.lucic@uva.nl
[2] University of Amsterdam, The Netherlands, email: oosterhuis@uva.nl
[3] Ahold Delhaize and University of Amsterdam, The Netherlands, email: hinda.haned@aholddelhaize.com
[4] University of Amsterdam, The Netherlands, email: derijke@uva.nl

the minimal perturbation required in order to change the outcome. The notion of *minimality* is not uniquely defined and may differ depending on the problem setting and end-user [4]. For some tasks it may be more important to perturb the smallest proportion of features, whereas for other tasks it might be preferable to produce the smallest possible perturbations per feature, regardless of the number of features. In our method, these preferences are made explicit by the choice of distance function in the optimization framework; this flexibility allows us to generate actionable explanations tailored to the end-users' needs. For each model, we generate three types of counterfactual examples, corresponding to different distance metrics. The main contributions of this work are:

- We propose *Optimizable Counterfactual Explanations* (OCE), a method for generating counterfactual explanations based on counterfactual examples learned through gradient descent. Our results indicate that our examples are considerably closer to the original instance compared to other methods that generate counterfactual examples for tree ensembles.
- We propose *Differentiable Approximations of Tree Ensembles* (DATE), a method for obtaining differentiable versions of tree ensembles. We show that these approximations are faithful to the original model.

Together these methods provide a reliable and easily-adaptable approach to counterfactual explanations for non-differentiable tree-based models.

## 2 METHOD

A *counterfactual explanation* for an instance $x$ and a model $\mathcal{M}$, $\Delta_x$, is the minimal perturbation of $x$ that changes the prediction of $\mathcal{M}$. A perturbed example $\bar{x}$ is an alteration of an instance $x$; if the prediction of $\mathcal{M}$ for $\bar{x}$ is different than for $x$, then $\bar{x}$ is a counterfactual example. We presume that $\mathcal{M}$ is a probabilistic classifier, where $\mathcal{M}(y \mid x)$ is the probability of $x$ belonging to class $y$ according to $\mathcal{M}$. The prediction of $\mathcal{M}$ for $x$ is the most probable class label $y_x = \arg\max_y \mathcal{M}(y \mid x)$. Thus in order for $\bar{x}$ to be a counterfactual example, $y_x \neq y_{\bar{x}}$, in other words:

$$\arg\max_y \mathcal{M}(y \mid x) \neq \arg\max_{y'} \mathcal{M}(y' \mid \bar{x}). \tag{1}$$

Besides changing the prediction of $\mathcal{M}$, $\bar{x}$ should be the result of a minimal perturbation to $x$, meaning that the distance between $x$ and $\bar{x}$ is minimized. We presume there is an appropriate differentiable distance function $d(x, \bar{x})$. The *optimal counterfactual example* $\bar{x}^*$ can then be defined as:

$$\bar{x}^* := \arg\min_{\bar{x}} d(x, \bar{x}) \text{ such that } y_x \neq y_{\bar{x}}. \tag{2}$$

The corresponding *optimal counterfactual explanation* of $x$, $\Delta_x^*$, is:

$$\Delta_x^* = \bar{x}^* - x. \tag{3}$$

This definition aligns with previous ML work on counterfactual explanations [15, 19, 29]. It should be noted that it is possible to have more than one optimal counterfactual explanation if the loss space is non-convex. In our work, we assume a convex loss space.

A counterfactual explanation provides users with an example of how an instance can be altered to result in an alternative prediction. Minimizing the distance between $x$ and $\bar{x}$ should ensure that $\bar{x}$ is as close to the decision boundary as possible. Moreover, the distance should indicate the effort it takes to apply the perturbation in practice, therefore the optimal counterfactual explanation shows how a prediction can be changed with the least amount of modification. The optimal explanation provides the user with interpretable and actionable feedback related to understanding the predictions of model $\mathcal{M}$.

In Section 2.1, we propose a method for finding $\bar{x}^*$ if $\mathcal{M}$ is differentiable or if a differentiable approximation of $\mathcal{M}$ is available. Then, in Section 2.2, we introduce a method for creating such differentiable approximations of tree-based models. Together, these contributions enable us to find counterfactual explanations through optimization for non-differentiable models, specifically tree ensembles.

### 2.1 Optimizable Counterfactual Explanations (OCE)

Wachter et al. [32] recognized that counterfactual examples can be found through gradient descent if the task is cast as a differentiable optimization problem. Specifically, they use a loss consisting of two components: (i) a prediction loss to change the prediction of $\mathcal{M}$: $\mathcal{L}_{\mathcal{M}}(y_x, \bar{x})$, and (ii) a distance loss to minimize the distance $d$: $\mathcal{L}_d(x, \bar{x})$. The complete loss is a linear combination of these two parts, with a weight $\beta \in \mathbb{R}_{>0}$:

$$\mathcal{L}(x, \bar{x} \mid \mathcal{M}, d) = \mathcal{L}_{\mathcal{M}}(y_x, \bar{x}) + \beta \cdot \mathcal{L}_d(x, \bar{x}). \tag{4}$$

The assumption here is that $\bar{x}^*$ can be found by minimizing the overall loss:

$$\bar{x}^* = \arg\min_{\bar{x}} \mathcal{L}(x, \bar{x} \mid \mathcal{M}, d). \tag{5}$$

Wachter et al. [32] propose a prediction loss $\mathcal{L}_{\mathcal{M}}$ based on the mean-squared-error. In contrast, we introduce a hinge-loss since we assume a classification task:

$$\mathcal{L}_{\mathcal{M}}(y, \bar{x}) = \mathbb{1}[y \neq \arg\max_{y'} \mathcal{M}(y' \mid \bar{x})] \cdot \mathcal{M}(y \mid \bar{x}). \tag{6}$$

Under the assumption that the distance function is differentiable, our choice of the distance loss $\mathcal{L}_d(x, \bar{x})$ is simply:

$$\mathcal{L}_d(x, \bar{x}) = d(x, \bar{x}). \tag{7}$$

The weight $\beta$ should not be too large, otherwise $\bar{x}$ will not deviate from $x$ and the prediction $y_{\bar{x}}$ will not change, thus not leading to a valid counterfactual example. However, because $\mathcal{L}_{\mathcal{M}}$ is a hinge-loss, we expect that the choice of $\beta$ does not need to be fine-tuned.

An obvious limitation of our approach so far is that it is based on the assumption that $\mathcal{M}$ is differentiable. This excludes many commonly used ML models, including tree-based models, on which we focus in this paper. We propose a solution through differentiable approximations of these models; an approximation $\widetilde{\mathcal{M}}$ should match the original model closely: $\widetilde{\mathcal{M}}(y \mid x) \approx \mathcal{M}(y \mid x)$. We define the prediction loss for this approximation as follows:

$$\widetilde{\mathcal{L}_{\mathcal{M}}}(y, \bar{x}) = \mathbb{1}[y \neq \arg\max_{y'} \mathcal{M}(y' \mid \bar{x})] \cdot \widetilde{\mathcal{M}}(y \mid \bar{x}). \tag{8}$$

We note that this loss is both based on the original model $\mathcal{M}$ and the approximation $\widetilde{\mathcal{M}}$: the loss is active as long as the prediction according to $\mathcal{M}$ has not changed, but its gradient is based on the differentiable $\widetilde{\mathcal{M}}$. The approximation of the complete loss becomes:

$$\widetilde{\mathcal{L}}(x, \bar{x} \mid \mathcal{M}, d) = \widetilde{\mathcal{L}_{\mathcal{M}}}(y_x, \bar{x}) + \beta \cdot \mathcal{L}_d(x, \bar{x}). \tag{9}$$

Since we assume that it approximates the complete loss:

$$\widetilde{\mathcal{L}}(x, \bar{x} \mid \mathcal{M}, d) \approx \mathcal{L}(x, \bar{x} \mid \mathcal{M}, d), \tag{10}$$

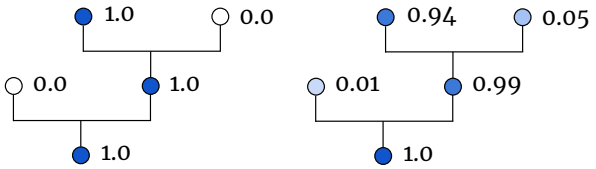


**Figure 1**: Left: A decision tree $\mathcal{T}$ and per node activations for a single instance. Right: a differentiable approximation of the same tree $\widetilde{\mathcal{T}}$ and activations for the same instance.

we also assume that the optimal counterfactual example can be found by minimizing it:

$$\bar{x}^* \approx \arg\min_{\bar{x}} \widetilde{\mathcal{L}}(x, \bar{x} \mid \mathcal{M}, d). \tag{11}$$

The *Optimizable Counterfactual Explanations* (OCE) approach we propose performs gradient descent on $\mathcal{L}(x, \bar{x} \mid \mathcal{M}, d)$ or $\widetilde{\mathcal{L}}(x, \bar{x} \mid \mathcal{M}, d)$ if $\mathcal{M}$ is not differentiable. The gradient is taken w.r.t. the perturbation $\bar{x}$, i.e., for the approximated loss $\nabla_{\bar{x}}\widetilde{\mathcal{L}}(x, \bar{x} \mid \mathcal{M}, d)$. Thus, if $\widetilde{\mathcal{M}}$ closely approximates $\mathcal{M}$ and $\widetilde{\mathcal{L}}$ is convex, then OCE is guaranteed to find the optimal counterfactual example $\bar{x}^*$, from which we obtain the optimal counterfactual explanation $\Delta_x^*$.

## 2.2 Differentiable Approximations of Tree Ensembles (DATE)

For a non-differentiable model $\mathcal{M}$, OCE requires a differentiable approximation $\widetilde{\mathcal{M}}$ and minimizes a loss based on both: $\widetilde{\mathcal{L}}$ (Equation 9). As an effective choice for $\widetilde{\mathcal{M}}$, we introduce *Differentiable Approximations of Tree Ensembles* (DATE), which can be used in conjunction with OCE to generate counterfactual examples for any tree ensemble.

Tree ensembles are based on decision trees; a single decision tree $\mathcal{T}$ uses a binary-tree structure to make predictions about an instance $x$ based on its features. Figure 1 shows a simple decision tree consisting of five nodes in a hierarchical structure. A node $j$ is activated if its parent node $p_j$ is activated and feature $x_{f_j}$ is on the correct side of the threshold $\theta_j$; which side is the correct side depends on whether $j$ is a *left* or *right* child, let $t_j(x)$ indicate if node $j$ is activated:

$$t_j(x) = \begin{cases} 1, & \text{if } j \text{ is the root,} \\ t_{p_j}(x) \cdot \mathbb{1}[x_{f_j} > \theta_j], & \text{if } j \text{ is a left child,} \\ t_{p_j}(x) \cdot \mathbb{1}[x_{f_j} \leq \theta_j], & \text{if } j \text{ is a right child,} \end{cases} \tag{12}$$

with the exception of the root node, which is always activated. Nodes that have no children are called *leaf nodes*; an instance $x$ always ends up in a single leaf node. Every leaf node $j$ has its own predicted distribution $\mathcal{T}(y \mid j)$, the prediction of the full tree is given by its activated leaf node. Let $\mathcal{T}_{leaf}$ be the set of leaf nodes in $\mathcal{T}$, then:

$$(j \in \mathcal{T}_{leaf} \wedge t_j(x) = 1) \rightarrow \mathcal{T}(y \mid x) = \mathcal{T}(y \mid j). \tag{13}$$

Alternatively, we can reformulate this as a sum over leaves:

$$\mathcal{T}(y \mid x) = \sum_{j \in \mathcal{T}_{leaf}} t_j(x) \cdot \mathcal{T}(y \mid j). \tag{14}$$

Generally, tree ensembles are deterministic; let $\mathcal{M}$ be an ensemble of $M$ many trees with weights $\omega_m \in \mathbb{R}$, then:

$$\mathcal{M}(y \mid x) = \mathbb{1}[y = \arg\max_{y'} \sum_{m=1}^{M} \omega_m \cdot \mathcal{T}_m(y' \mid x)]. \tag{15}$$

This formulation fits adaptive or gradient boosted ensembles, random forests (i.e., $\forall m, \omega_m = 1$), or decision trees (i.e., $M = 1$).

Since $\mathcal{M}$ is not differentiable, we are unable to calculate its gradient w.r.t. input $x$ and thus OCE cannot be applied to it. However, the non-differentiable operations in our formulation are (i) the indicator function, and (ii) a maximum operation, both of which can be approximated by differentiable functions. First, we introduce the $\widetilde{t}_j(x)$ function that approximates the activation of node $j$: $\widetilde{t}_j(x) \approx t_j(x)$, using a sigmoid function with parameter $\sigma \in \mathbb{R}_{>0}$:

$$sig(z) = \frac{1}{1 + \exp(\sigma \cdot z)},$$

$$\widetilde{t}_j(x) = \begin{cases} 1, & \text{if } j \text{ is the root,} \\ \widetilde{t}_{p_j}(x) \cdot sig(\theta_j - x_{f_j}), & \text{if } j \text{ is left child,} \\ \widetilde{t}_{p_j}(x) \cdot sig(x_{f_j} - \theta_j), & \text{if } j \text{ is right child.} \end{cases} \tag{16}$$

As $\sigma$ increases, $\widetilde{t}_j$ approximates $t_j$ more closely. Second, we introduce a tree approximation:

$$\widetilde{\mathcal{T}}(y \mid x) = \sum_{j \in \mathcal{T}_{leaf}} \widetilde{t}_j(x) \cdot \mathcal{T}(y \mid j). \tag{17}$$

The approximation $\widetilde{\mathcal{T}}$ uses the same tree structure and thresholds as $\mathcal{T}$. However, its activations are no longer deterministic but instead are dependent on the distance between the feature values $x_{f_j}$ and the thresholds $\theta_j$. Figure 1 displays an example of a decision tree and its approximation, along with the corresponding per node activations.

Lastly, we replace the maximum operation of $\mathcal{M}$ by a softmax with temperature $\tau \in \mathbb{R}_{>0}$, and introduce the approximation:

$$\widetilde{\mathcal{M}}(y \mid x) = \frac{\exp\left(\tau \cdot \sum_{m=1}^{M} \omega_m \cdot \widetilde{\mathcal{T}}_m(y \mid x)\right)}{\sum_{y'} \exp\left(\tau \cdot \sum_{m=1}^{M} \omega_m \cdot \widetilde{\mathcal{T}}_m(y' \mid x)\right)}. \tag{18}$$

Thus, the output of our DATE method is the approximation $\widetilde{\mathcal{M}}$ based on an original model $\mathcal{M}$ and the parameters $\sigma$ and $\tau$. DATE is applicable to any tree-based model, and how well $\widetilde{\mathcal{M}}$ approximates $\mathcal{M}$ depends on the choice of $\sigma$ and $\tau$. Potentially, the approximation is perfect since:

$$\lim_{\sigma, \tau \to \infty} \widetilde{\mathcal{M}}(y \mid x) = \mathcal{M}(y \mid x). \tag{19}$$

In other words, it is possible for our approximation $\widetilde{\mathcal{M}}$ to be perfectly faithful to the original model $\mathcal{M}$, since increasing $\sigma$ eventually leads to exact approximations of the indicator functions, while increasing $\tau$ leads to a completely unimodal softmax distribution. In practice, we expect that choosing high values for $\sigma$ or $\tau$ leads to impractical near-zero gradients, and, consequently, that a better model approximation does not necessarily produce better counterfactual examples.

To intuitively illustrate DATE, Figure 2 shows a simple two-feature example ensemble consisting of three trees, each with a single leaf that predicts a positive label. On the left is the decision boundary for a tree ensemble; the middle visualizes the positive leaf nodes that form the decision boundary; on the right is the approximated loss $\widetilde{\mathcal{L}_{\mathcal{M}}}$ and its gradient w.r.t. $\bar{x}$. The gradients push features close to thresholds harder and in the direction of the decision boundary if $\widetilde{\mathcal{L}}$ is convex.

Finally, the approximation of DATE can be used with OCE in order to find counterfactual examples for any tree-based model. In the following sections, we will evaluate whether combining OCE and DATE leads to counterfactual examples that are closer to the original inputs in comparison to existing methods.

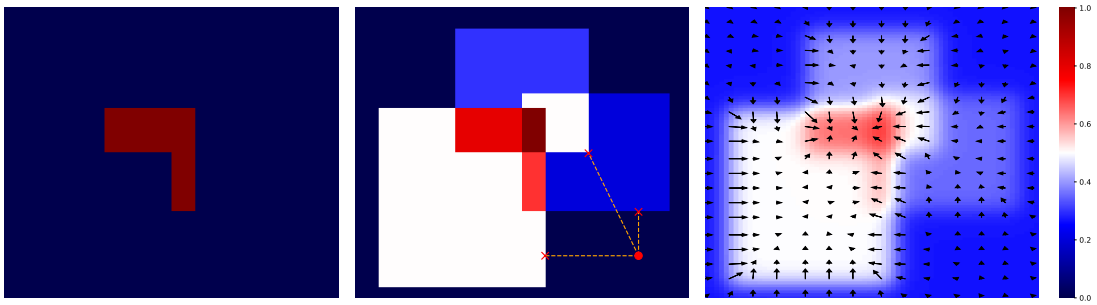


**Figure 2**: An example of how the FT baseline method (explained in Section 3.2) and our OCE method handle an adaptive boosting forest consisting of three trees. Left: the decision boundary of the forest. Middle: three positive leaves that determine the decision of the forest, an example instance and the perturbed examples suggested by FT. Right: a probabilistic approximation of the forest from DATE and the resulting gradients. The FT perturbed examples do not change the prediction of the forest, whereas the gradient of the probabilistic approximation leads toward the true decision boundary.

## 3 EXPERIMENTAL SETUP

To test OCE, we consider 27 experimental settings to find the best counterfactual explanations (3 datasets $\times$ 3 tree-based models $\times$ 3 distance functions) by jointly tuning the hyperparameters of OCE ($w_d, \alpha$) and DATE ($\sigma, \tau$). We choose the parameters that produce (i) a valid counterfactual example for every instance in the dataset, and (ii) the smallest mean distance between corresponding pairs ($x$, $\bar{x}$). We compare OCE to the FT method by Tolomei et al. [29] and to a Random Perturbation (RP) baseline in terms of (i) mean distance, (ii) mean relative distance, and (iii) proportion of counterfactual examples that are closer to the original input. We found that the Adam optimizer [16] produced the best results for our task, but it should be noted that any optimizer can be used. We also evaluate DATE in terms of fidelity – the agreement between the approximation $\widetilde{\mathcal{M}}$ and the original model $\mathcal{M}$.

### 3.1 Datasets and Models

We evaluate OCE on three datasets: *Wine Quality* [30], *HELOC* [10], and COMPAS [22]. The *Wine Quality* dataset is about predicting the quality of white wine on a 0–10 scale. We adapt this to a binary classification setting by labelling the wine as "high quality" if the quality is $\geq 7$. The *HELOC* set is from the Explainable Machine Learning Challenge at NeurIPS 2017, where the task is to predict whether or not a customer will default on their loan. The *COMPAS* dataset is widely used for detecting bias in machine learning systems, where the task is predicting whether or not a criminal defendant will reoffend upon release. The *Wine Quality* set has 4,897 instances and 11 features, the *HELOC* set has 10,459 instances and 23 features, and the *COMPAS* dataset has 4,320 instances and 6 features. In all cases, we scale the features such that their values are in the range $[0, 1]$ and remove any categorical features. We split the data into a training (70%) and validation (30%) set, where the validation set was used for hyper-parameter tuning. Since our method is not dependent on the performance of these models, we do not create a test set. Counterfactual explanations were found for the datapoints in the training set since this is larger than the validation set.

We test OCE on three types of tree-based models and use the scikit learn implementation of each: (i) Decision Trees (DTs); (ii) Random Forests (RFs); and (iii) Adaptive Boosting (AB) with DTs as the base learners.

### 3.2 Baselines

We compare OCE against the Feature Tweaking (FT) method by Tolomei et al. [29]. To the best of our knowledge, this is the only existing method for generating counterfactual examples specifically for tree ensembles. FT identifies the leaf nodes where the prediction of the leaf nodes do not match the current prediction $y_x$. In other words, it recognizes the set of leaves that if activated, $t_j(\bar{x}) = 1$, would change the prediction of a tree $\mathcal{T}$:

$$\mathcal{T}_{change} = \{j \mid j \in \mathcal{T}_{leaf} \wedge y_x \neq \arg\max_y T(y \mid j)\}. \quad (20)$$

For every $\mathcal{T}$ in $\mathcal{M}$, FT generates a perturbed example per node in $\mathcal{T}_{change}$ so that it is activated with at least an $\epsilon$ difference per threshold, and then selects the most optimal example. This means that for every feature threshold $\theta_j$ involved, the corresponding feature is perturbed accordingly: $\bar{x}_{f_j} = \theta_j \pm \epsilon$. The result is a perturbed example that was changed minimally to activate a leaf node in $\mathcal{T}_{change}$. The main problem with FT is that the perturbed examples are not necessarily counterfactual examples, since changing the prediction of a single tree $\mathcal{T}$ does not guarantee a change in the prediction of the full ensemble $\mathcal{M}$. Figure 2 shows all three perturbed examples generated by FT for a single instance. In this case, none of the generated examples change the model prediction and therefore none are counterfactual examples.

We also compare against a Random Perturbation (RP) baseline where noise is randomly sampled from a Gaussian $\mathcal{N}(0, 0.5)$ and added to the original input $x$. Since we perform 1,000 optimization steps in OCE, RP takes 1,000 samples and selects the $\bar{x}$ that minimizes the distance to $x$.

### 3.3 Distance Metrics

We use three distance functions: Euclidean, Cosine, and Manhattan, each covering a different interpretation of what *minimal* distance is. Euclidean distance measures the geometric displacement:

$$d_{Euclidean}(x, \bar{x}) = \sqrt{\sum_i (x_i - \bar{x}_i)^2}. \quad (21)$$

Cosine distance measures the angle by which $\bar{x}$ deviates from $x$ – whether $\bar{x}$ preserves the relationship between features in $x$:

$$d_{Cosine}(x, \bar{x}) = 1 - \frac{\sum_i (x_i \cdot \bar{x}_i)}{\|x\| \, \|\bar{x}\|}. \quad (22)$$

**Table 1**: Mean average distance and mean relative distance between counterfactual examples and original inputs. Significant improvements and losses over the baseline are denoted by $^{\blacktriangledown}$ ($p < 0.0001$), $^{\triangledown}$ ($p < 0.05$), and $^{\blacktriangle}$, $^{\triangle}$, respectively; $^{\circ}$ denotes no significant difference; $^{\otimes}$ denotes settings where the FT baseline cannot find a counterfactual example for every instance.

| | | | Euclidean | | | Cosine | | | Manhattan | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Dataset** | **Metric** | **Method** | **DT** | **RF** | **AB** | **DT** | **RF** | **AB** | **DT** | **RF** | **AB** |
| *Wine Quality* | $d_{mean}$ | *RP* | 0.600 | 0.536 | 0.622 | 0.064 | 0.064 | 0.067 | 1.564 | 1.441 | 1.639 |
| | | *FT* | 0.259 | 0.151 | 0.229$^{\otimes}$ | 0.036 | 0.016 | 0.029$^{\otimes}$ | 0.259 | **0.185** | 0.316$^{\otimes}$ |
| | | *OCE* | **0.258**$^{\blacktriangledown\circ}$ | **0.124**$^{\blacktriangledown\blacktriangledown}$ | **0.135**$^{\blacktriangledown\blacktriangledown}$ | **0.003**$^{\blacktriangledown\blacktriangledown}$ | **0.007**$^{\blacktriangledown\blacktriangledown}$ | **0.012**$^{\blacktriangledown\blacktriangledown}$ | **0.258**$^{\blacktriangledown\circ}$ | 0.199$^{\blacktriangledown\blacktriangle}$ | **0.301**$^{\blacktriangledown\triangledown}$ |
| | $d_{Rmean}$ | *OCE/RP* | 0.402 | 0.222 | 0.193 | 0.034 | 0.092 | 0.127 | 0.156 | 0.132 | 0.166 |
| | | *OCE/FT* | 0.991 | 0.982 | 0.506$^{\otimes}$ | 0.039 | 0.419 | 0.238$^{\otimes}$ | 0.991 | 1.283 | 0.867$^{\otimes}$ |
| | $\%_{closer}$ | *OCE <RP* | 100% | 100% | 100% | 100% | 99% | 99% | 100% | 100% | 100% |
| | | *OCE <FT* | 100% | 51% | 96%$^{\otimes}$ | 100% | 91% | 97%$^{\otimes}$ | 100% | 42% | 64%$^{\otimes}$ |
| *HELOC* | $d_{mean}$ | *RP* | 0.873 | 0.897 | 0.881 | 0.080 | 0.082 | 0.079 | 2.942 | 3.043 | 2.969 |
| | | *FT* | **0.113** | 0.213 | 0.192 | 0.002 | 0.008 | 0.007 | **0.126** | **0.291** | 0.214 |
| | | *OCE* | 0.128$^{\blacktriangledown\blacktriangle}$ | **0.182**$^{\blacktriangledown\blacktriangledown}$ | **0.135**$^{\blacktriangledown\blacktriangledown}$ | **0.002**$^{\blacktriangledown\blacktriangledown}$ | **0.005**$^{\blacktriangledown\blacktriangledown}$ | **0.003**$^{\blacktriangledown\blacktriangledown}$ | 0.148$^{\blacktriangledown\blacktriangle}$ | 0.311$^{\blacktriangledown\blacktriangle}$ | **0.205**$^{\blacktriangledown\blacktriangledown}$ |
| | $d_{Rmean}$ | *OCE/RP* | 0.146 | 0.197 | 0.151 | 0.018 | 0.052 | 0.040 | 0.050 | 0.099 | 0.069 |
| | | *OCE/FT* | 1.239 | 0.813 | 0.720 | 0.711 | 0.506 | 0.521 | 1.358 | 0.980 | 0.983 |
| | $\%_{closer}$ | *OCE <RP* | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% |
| | | *OCE <FT* | 33% | 90% | 90% | 76% | 85% | 86% | 45% | 53% | 56% |
| *COMPAS* | $d_{mean}$ | *RP* | 0.813 | 0.810 | 0.826 | 0.447 | 0.432 | 0.435 | 1.469 | 1.453 | 1.441 |
| | | *FT* | **0.085** | **0.084** | 0.088 | 0.015 | 0.019 | 0.020 | **0.091** | **0.088** | **0.092** |
| | | *OCE* | 0.097$^{\blacktriangledown\blacktriangle}$ | **0.084**$^{\blacktriangledown\circ}$ | **0.079**$^{\blacktriangledown\blacktriangledown}$ | **0.011**$^{\blacktriangledown\blacktriangledown}$ | **0.013**$^{\blacktriangledown\blacktriangledown}$ | **0.009**$^{\blacktriangledown\blacktriangledown}$ | 0.097$^{\blacktriangledown\triangle}$ | 0.097$^{\blacktriangledown\blacktriangle}$ | 0.096$^{\blacktriangledown\circ}$ |
| | $d_{Rmean}$ | *OCE/RP* | 0.114 | 0.088 | 0.085 | 0.027 | 0.025 | 0.022 | 0.066 | 0.059 | 0.060 |
| | | *OCE/FT* | 1.194 | 1.154 | 1.071 | 0.511 | 0.817 | 0.607 | 1.188 | 1.235 | 1.223 |
| | $\%_{closer}$ | *OCE <RP* | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% | 100% |
| | | *OCE <FT* | 23% | 30% | 40% | 78% | 87% | 84% | 65% | 29% | 52% |

The Manhattan distance (a.k.a. $L1$-norm) measures per feature differences, minimizing the number of features perturbed and therefore inducing sparsity:

$$d_{Manhattan}(x, \bar{x}) = \sum_i |x_i - \bar{x}_i|. \quad (23)$$

## 3.4 Evaluation Metrics

We evaluate the counterfactual examples generated by OCE with DATE in terms of (i) distance from the original input, (ii) relative distance from the original input based on the three distance functions, and (iii) the proportion of counterfactual examples that are closer to the original input. We also evaluate the approximations generated by DATE in terms of fidelity: how often the prediction of $\widetilde{\mathcal{M}}$ agrees with that of $\mathcal{M}$.

Let $X$ be the set of $N$ original instances and $\bar{X}$ be the corresponding set of $N$ generated counterfactual examples. The mean distance is defined as:

$$d_{mean}(X, \bar{X}) = \frac{1}{N} \sum_{n=1}^{N} d(x^{(n)}, \bar{x}^{(n)}). \quad (24)$$

To better interpret individual improvements over the baselines, we evaluate in terms of mean relative distance. Let $\bar{X}$ be the set of counterfactual examples produced by OCE and let $\bar{X}'$ be the set of counterfactual examples produced by a baseline. Then the mean relative distance is defined as:

$$d_{Rmean}(\bar{X}, \bar{X}') = \frac{1}{N} \sum_{n=1}^{N} \frac{d(x^{(n)}, \bar{x}^{(n)})}{d(x^{(n)}, \bar{x}'^{(n)})}. \quad (25)$$

If $d_{Rmean} < 1$, OCE counterfactual examples are on average closer to the original input compared to the baseline. We also look at the proportion of counterfactual examples that are closer to the original input.

In our experiments, only the FT baseline failed to find valid counterfactual explanations for every instance in the datasets. In these cases, we exclude invalid explanations from the computation of $d_{mean}$ and $d_{Rmean}$ for a fair comparison with OCE.

Lastly, we measure the fidelity of DATE: the percentage of instances in $X$ where the prediction of $\widetilde{\mathcal{M}}$ matches that of $\mathcal{M}$:

$$\textit{fidelity}(\widetilde{\mathcal{M}}, X) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}[y_{x^{(n)}} = \arg\max_{y'} \widetilde{\mathcal{M}}(y' \mid x^{(n)})]. \quad (26)$$

# 4 EXPERIMENTAL RESULTS

In this section, we evaluate (i) the counterfactual examples produced by OCE and DATE, (ii) the fidelity of the model approximations produced by DATE, and (iii) investigate the effect of different distance functions in a brief case-study.

## 4.1 Evaluating OCE

First, we consider whether OCE provides explanations that are more optimal compared to the FT and RP baselines, based on the distance functions. Table 1 shows the performance of OCE, the baselines and their relative differences. When comparing against RP, OCE counterfactual examples are closer to the original instance in all 27 experimental settings, and the difference in $d_{mean}$ is statistically significant

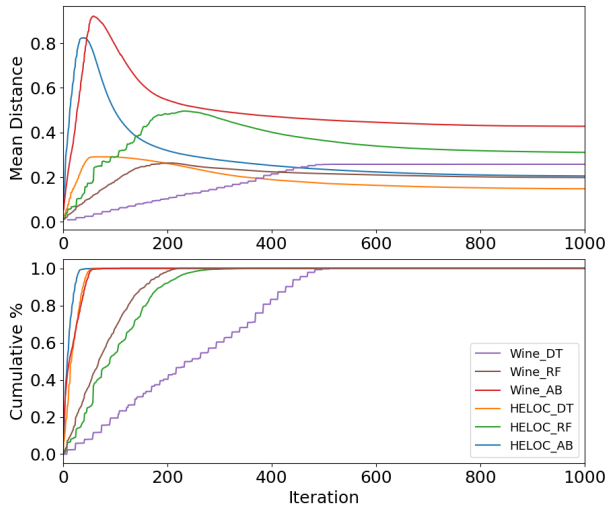


**Figure 3**: Top: Mean distances for counterfactual examples in each iteration of OCE for Manhattan explanations. Bottom: Cumulative percentage of counterfactual examples found in each iteration.

with $p < 0.0001$ in all cases. When comparing against FT in terms of $d_{mean}$, OCE counterfactual examples are significantly closer to the original instance in 16 settings, and are significantly further from the original instance in 7 settings. In the remaining 4 settings, the difference in $d_{mean}$ is not significant.

In our experimental setup, OCE always significantly outperforms FT in terms of $d_{mean}$ for Cosine distance. OCE also significantly outperforms FT in 5 out of 9 settings for Euclidean distance, while FT is only significantly better in 2 out of 9. The opposite is true for Manhattan distance, that is, FT is significantly better in 5 out of 9 settings while OCE is significantly better in 2 out of 9. This is most likely because in each iteration, OCE perturbs many of the features by a small amount, which is better suited for minimizing Cosine and Euclidean distance, whereas FT perturbs only the features associated with an individual leaf, which better aligns with minimizing Manhattan distance.

Overall, OCE always outperforms the RP baseline, and the FT baseline in the majority of cases. The relative improvements we observe are also quite substantial: in almost half of the cases (13 out of 27), the relative distance between OCE and the baselines is $< 0.9$.

In most settings, the method with the best $d_{mean}$ also has the best $d_{Rmean}$ as well as the larger proportion of counterfactual examples that are closer to the original input. However, this is not always the case, particularly with the *COMPAS* dataset. For example, although OCE significantly outperforms FT in terms of Euclidean distance for the AB model, it is not better in terms of $d_{Rmean}$ or $\%_{closer}$. The results for the DT model in terms of Manhattan distance are similar, but in this case, FT is significantly better in terms of $d_{mean}$ while OCE is better in terms of $d_{Rmean}$ or $\%_{closer}$. We speculate this is due to the small number of features in this dataset and plan to investigate this further in future work. Although we use $d_{mean}$ as our main evaluation metric, this highlights the importance of using multiple metrics to evaluate counterfactual examples.

Cases where the FT baseline is unable to generate a counterfactual example for every instance are marked with $^{\otimes}$ in Table 1. For instance, the FT baseline only generates valid counterfactual examples for 69% of the instances in the *Wine Quality* AB model. In contrast, OCE produces valid counterfactual examples for all instances in every setting.

Figure 3 shows the mean Manhattan distance of the perturbed examples in each iteration of OCE, along with the percentage of counterfactual examples found for the *Wine Quality* and *HELOC* datasets. The results are similar for the *COMPAS* dataset but we omit these due to space considerations. The trends we see here are indicative of all cases: the mean distance increases until a counterfactual example has been found for every instance, after which the mean distance starts to decrease. This appears to be a result of the hinge-loss OCE uses, which first prioritizes finding a valid counterfactual example and then decreases the distance between $x$ and $\bar{x}$.

In conclusion, our results show that OCE in combination with DATE is very efficient at finding counterfactual explanations for tree-based models. Unlike the FT baseline, OCE finds valid counterfactual explanations for every instance across all settings, making OCE a much more reliable method. In the majority of tested settings, OCE explanations are substantial improvements in terms of distance to the original inputs. Thus, we conclude that OCE in combination with DATE is a more reliable choice, and often the best choice for generating minimal counterfactual explanations.

## 4.2 Evaluating DATE

Table 2 shows the fidelity of the DATE approximations we used in our experiments. All of the approximations are faithful to the original model for the vast majority of predictions. The worst approximation matches $\mathcal{M}$ in 74.4% of instances (*COMPAS*: RF – Manhattan), while the best approximation matches $\mathcal{M}$ in 97.0% of instances (*Wine Quality*: RF – Euclidean). This confirms our expectation that the approximated models that produce the best counterfactual explanations are not necessarily the most exact approximations.

**Table 2**: Fidelity of approximations used in experiments.

| Dataset | Model | Euclid. | Cosine | Manhat. |
|---|---|---|---|---|
| *Wine Quality* | DT | 0.823 | 0.823 | 0.823 |
| | RF | 0.970 | 0.925 | 0.969 |
| | AB | 0.868 | 0.868 | 0.868 |
| *HELOC* | DT | 0.923 | 0.897 | 0.938 |
| | RF | 0.923 | 0.842 | 0.842 |
| | AB | 0.936 | 0.920 | 0.936 |
| *COMPAS* | DT | 0.816 | 0.877 | 0.793 |
| | RF | 0.799 | 0.778 | 0.744 |
| | AB | 0.942 | 0.942 | 0.844 |

## 4.3 Case Study: Actionable Explanations

The counterfactual explanations that OCE generates highly depend on the choice of distance function. Figure 4 shows three different explanations generated using different distance functions for the same input instance in the *HELOC* dataset for the AB model. Each explanation shows an individual who was denied a loan (negative class) and the minimal changes they could make in order to be approved for that loan (positive class). The Manhattan explanation only requires a few changes to the individual's profile, where each change is larger in comparison to the Euclidean explanation where the individual changes are smaller but there are more of them. These explanations are useful when the features in the model are mostly independent, as is usually the case in credit risk modeling [27]. In settings where there are significant dependencies between features, the Cosine explanations may be preferred since they are based on perturbations that try to preserve the relationship between features. For

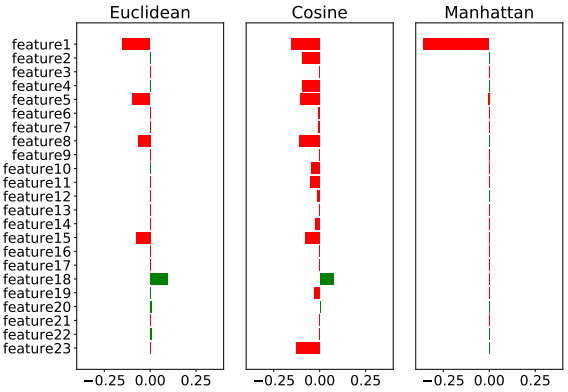


**Figure 4**: OCE explanations for the same input instance from the *HELOC* dataset based on different distance functions. Green and red indicate increases and decreases in feature values respectively.

instance, in the *Wine Quality* dataset, it would be difficult to change the amount of citric acid without affecting the pH level. Thus the choice of distance function in OCE allows users to choose what kind of explanation is best suited for their problem.

## 5 RELATED WORK

Some previous work on local explanations for tree-based models focuses on finding decision rules [31], influential training points [26], or prototypes [28] rather than counterfactual examples, which is the focus of our work.

Counterfactual examples have been used in a variety of ML areas, such as reinforcement learning [21], deep learning [1], and explainable AI (XAI) [2, 5, 15, 19, 29]. Previous XAI methods for generating counterfactual examples are either model-agnostic [5, 15, 19, 25] or model-specific [2, 29]. Model-agnostic approaches treat the original model as a "black-box" and only assume query access to the model, whereas model-specific approaches typically do not make this assumption and can therefore make use of the inner workings of the model.

Our work is a model-specific approach to tree ensembles for generating counterfactual examples through optimization. Previous model-specific works for generating counterfactual examples through optimization have solely been conducted on differentiable models [5, 11, 32]. Although we focus on tree ensembles, our method can be applied to any non-differentiable model that can be approximated probabilistically. Previous work on generating counterfactual examples for tree ensembles shares our objective of finding minimal perturbations based on the inner workings of the model and therefore we use it as one of our baselines. It involves an extensive search over many possible paths in the ensemble that could lead to an alternative prediction [29]. In our experimental results, this set of proposed possible paths sometimes did not include a valid counterfactual example.

Karimi et al. [15] provide model-agnostic counterfactual explanations based on formal verification, where the task is to solve a series of satisfiability problems that result in a perturbed version of the original input such that the prediction changes. Laugel et al. [19] also provide model-agnostic counterfactual explanations, but frame the problem as an inverse classification task, while Russell [25] uses mixed integer programming to find diverse counterfactual examples. Similar to our work, Dhurandhar et al. [6] learn counterfactual examples through gradient descent, but their work suffers from the same constraints as the previous three – they only assume query access to the model and therefore estimate the gradient through zero-order convex optimization [8]. Although a theoretical upper bound on the error for this estimate is provided, no analysis is conducted to determine how accurate these approximations actually are. Since we do not have this restriction on only having query access to the model, we can generate a probabilistic approximation of the full model to use in the optimization, as opposed to a finite number of gradient estimates.

One of the main components of our method, DATE, involves constructing a differentiable version of a tree ensemble to be used in optimization by replacing each splitting threshold with a sigmoid function. This can be seen as using a (very small) neural network to obtain a smooth approximation of the tree. Other work in a similar vein involves neural decision trees [3, 33], which use a full neural network instead of a simple sigmoid. However, neural decision tree methods do not optimize for approximating a given (already trained) model. Therefore, unlike DATE method, they are not an obvious choice for finding counterfactual examples for an existing model. Soft decision trees [14] are another example of differentiable tree approximations, which instead approximate a neural network with a decision tree. This can be seen as the inverse of our approximation.

## 6 CONCLUSION

We propose a local explanation method for classifiers, OCE, that casts the problem of finding counterfactual examples for individual predictions as an optimization task. Using gradient descent, OCE generates the minimal perturbation to an input instance $x$ which results in an alternative prediction from a model $\mathcal{M}$. Unlike previous methods, which assume $\mathcal{M}$ is differentiable, we propose a solution for when $\mathcal{M}$ is a non-differentiable tree-based model – DATE – that provides a differentiable approximation of $\mathcal{M}$ using which OCE can find counterfactual examples. We find that in the majority of our experimental settings, examples generated by OCE are significantly closer to the input instances both in terms of mean distance and mean realtive distance, in comparison to those generated by the baselines. We also show that the approximations generated by DATE are faithful to the original model in at least 74.4% of predictions.

The OCE method is easily extendable, leading to obvious opportunities for future work. For instance, different distance functions can be optimized or additional criteria for counterfactual examples could be added. Furthermore, an approach similar to DATE could potentially be applied to other non-differentiable models, thus enabling counterfactual explanations for a wider range of models.

While existing research from the cognitive sciences has shown that humans are able to interpret counterfactual explanations, the notion of what constitutes a *minimal* perturbation is currently not clear [4]. Further research into the interpretability of counterfactual explanations could help the field to better understand the appropriate criteria to optimize for.

## Reproducibility

Upon publication of the paper, we will share the code used to run our experiments, including all hyperparameter settings. All datasets used in this paper are publicly available.

## REFERENCES


[1] A. M. Alaa, M. Weisz, and M. van der Schaar. Deep counterfactual networks with propensity-dropout. *arXiv preprint arXiv:1706.05966*, June 2017.

[2] D. Alvarez Melis and T. Jaakkola. Towards robust interpretability with self-explaining neural networks. In *NeurIPS*, pages 7786–7795, 2018.

[3] R. Balestriero. Neural decision trees. *arXiv preprint arXiv:1702.07360*, Feb. 2017.

[4] R. M. Byrne. Counterfactual thought. *Annual Review of Psychology*, 67:135–157, 2016.

[5] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P. Ting, K. Shanmugam, and P. Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *arXiv preprint arXiv:1802.07623*, Feb. 2018.

[6] A. Dhurandhar, T. Pedapati, A. Balakrishnan, P.-Y. Chen, K. Shanmugam, and R. Puri. Model agnostic contrastive explanations for structured data. *arXiv preprint arXiv:1906.00117*, May 2019.

[7] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608v2*, 2017.

[8] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *arXiv preprint arXiv:1312.2139*, Dec. 2013. URL `http://arxiv.org/abs/1312.2139`.

[9] EU. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*, volume L119. 2016.

[10] FICO. Explainable machine learning challenge. `https://community.fico.com/s/explainable-machine-learning-challenge`, 2017.

[11] R. M. Grath, L. Costabello, C. L. Van, P. Sweeney, F. Kamiab, Z. Shen, and F. Lecue. Interpretable credit application predictions with counterfactual explanations. *arXiv preprint arXiv:1811.05245*, Nov. 2018.

[12] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, May 2018.

[13] R. Guidotti, A. Monreale, F. Turini, D. Pedreschi, and F. Giannotti. A survey of methods for explaining black box models. *arXiv preprint arXiv:1802.01933*, 2018.

[14] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *NIPS 2014 Deep Learning Workshop*, Mar. 2014.

[15] A.-H. Karimi, G. Barthe, B. Balle, and I. Valera. Model-agnostic counterfactual explanations for consequential decisions. *arXiv preprint arXiv:1905.11190*, May 2019.

[16] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, Jan. 2017. URL `http://arxiv.org/abs/1412.6980`. arXiv: 1412.6980.

[17] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. *arXiv preprint arXiv:1703.04730*, 2017.

[18] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec. Faithful and Customizable Explanations of Black Box Models. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society - AIES '19*, pages 131–138, Honolulu, HI, USA, 2019. ACM Press. ISBN 978-1-4503-6324-2. doi: 10.1145/3306618.3314229. URL `http://dl.acm.org/citation.cfm?doid=3306618.3314229`.

[19] T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki. Inverse classification for comparison-based interpretability in machine learning. *arXiv preprint arXiv:1712.08443*, Dec. 2017.

[20] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *NIPS*, pages 4765–4774. Curran Associates, Inc., 2017.

[21] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere. Explainable reinforcement learning through a causal lens. *arXiv preprint arXiv:1905.10958*, May 2019.

[22] D. Ofer. Compas dataset, 2017. URL `https://www.kaggle.com/danofer/compass`.

[23] S. Rathi. Generating counterfactual and contrastive explanations using SHAP. *arXiv preprint arXiv:1906.09293*, June 2019.

[24] M. T. Ribeiro, S. Singh, and C. Guestrin. Why should I trust you?: Explaining the predictions of any classifier. In *KDD*, pages 1135–1144. ACM, 2016.

[25] C. Russell. Efficient Search for Diverse Coherent Explanations. *arXiv:1901.04909 [cs, stat]*, Jan. 2019. URL `http://arxiv.org/abs/1901.04909`. arXiv: 1901.04909.

[26] B. Sharchilev, Y. Ustinovsky, P. Serdyukov, and M. de Rijke. Finding influential training samples for gradient boosted decision trees. In *ICML*, pages 4584–4592, July 2018.

[27] N. Siddiqi. *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*. Wiley, 2012.

[28] H. F. Tan, G. Hooker, and M. T. Wells. Tree space prototypes: Another look at making tree ensembles interpretable. *arXiv preprint arXiv:1611.07115*, Nov. 2016.

[29] G. Tolomei, F. Silvestri, A. Haines, and M. Lalmas. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *KDD*, pages 465–474, 2017.

[30] UCI Machine Learning Repository. Wine quality data set. `https://archive.ics.uci.edu/ml/datasets/Wine+Quality`, 2009.

[31] J. van der Waa, M. Robeer, J. van Diggelen, M. Brinkhuis, and M. Neerincx. Contrastive explanations with local foil trees. *arXiv preprint arXiv:1806.07470*, June 2018.

[32] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 2017.

[33] Y. Yang, I. G. Morillo, and T. M. Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, June 2018.