Model-Agnostic Counterfactual Explanations for Consequential Decisions

Amir-Hossein Karimi
MPI-IS
amir@tue.mpg.de

Gilles Barthe
MPI-SP
IMDEA Software Institute
gjbarthe@gmail.com

Borja Balle
borja.balle@gmail.com

Isabel Valera
MPI-IS
isabel.valera@tue.mpg.de

Abstract

Predictive models are being increasingly used to support consequential decision making at the individual level in contexts such as pretrial bail and loan approval. As a result, there is increasing social and legal pressure to provide explanations that help the affected individuals not only to understand why a prediction was output, but also how to act to obtain a desired outcome. To this end, several works have proposed methods to generate counterfactual explanations. However, they are often restricted to a particular subset of models (e.g., decision trees or linear models), and cannot directly handle the mixed (numerical and nominal) nature of the features describing each individual. In this paper, we propose a model-agnostic algorithm to generate counterfactual explanations that builds on the standard theory and tools from formal verification. Specifically, our algorithm solves a sequence of satisfiability problems, where a wide variety of predictive models and distances in mixed feature spaces, as well as natural notions of plausibility and diversity, are represented as logic formulas. Our experiments on real-world data demonstrate that our approach can flexibly handle widely deployed predictive models, while providing meaningfully closer counterfactuals than existing approaches.

1 Introduction

Data-driven predictive models are ubiquitously being used to support or even substitute humans in decision making in a wide variety of real-world contexts including, e.g., selection process for hiring, loan approval, or pretrial bail. However, as algorithmic methods are increasingly used to make *consequential decisions* at the individual-level – i.e., decisions that may have significant consequences for the individuals they decide about – the debate about their lack of transparency and explainability becomes more heated. To make things worse, while the verdict is still out as to what constitutes a *good explanations* [7, 10, 16, 19, 23, 24], there already exists clearly defined legal requirements for explanations in the context of consequential decision making. For example, the EU General Data Protection Regulation ("GDPR") grants individuals the *right-to-explanation* [30, 31], via requiring institutions to provide explanations to individuals that are subject to their (semi-)automated decision making systems.

A growing number of works on interpretable machine learning have recently focused on the definitions of, and mechanisms for providing, good explanations for predictor-based decision making systems. In the context of consequential decision making, it is widely agreed that a good explanation should provide answers to the following two questions [7, 12, 32]: (i) "*why the model outputs a certain*

Preprint. Under review.

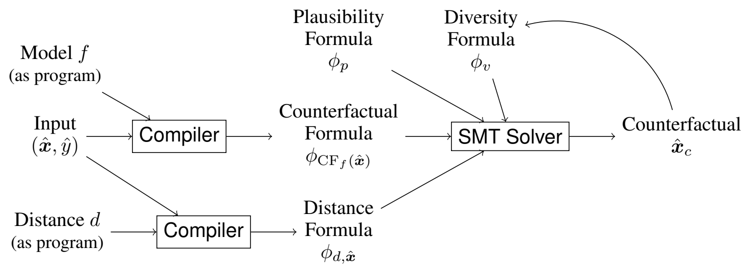

Model f (as program)
Input $(\hat{\boldsymbol{x}}, \hat{y})$
Distance d (as program)
Compiler
Compiler
Counterfactual Formula $\phi_{\mathrm{CF}_f(\hat{\boldsymbol{x}})}$
Distance Formula $\phi_{d,\hat{\boldsymbol{x}}}$
Plausibility Formula ϕ_p
Diversity Formula ϕ_v
SMT Solver
Counterfactual $\hat{\boldsymbol{x}}_c$

Figure 1: Architecture Overview for Model-Agnostic Counterfactual Explanations (MACE)

prediction for a given individual?"; and, (ii) "*what features describing the individual would need to change in order to achieve the desired output?*."

Here, we focus on answering the second question, or equivalently, on generating *counterfactual explanations*. Of specific importance is the problem of finding the *nearest counterfactual explanation* – i.e., identifying the set of features resulting in the desired prediction while remaining at minimum distance from the original set of features describing the individual. Previous works tackling this problem either propose solutions that are tailored to particular models, e.g., decision trees [28]; rely on classical optimization tools, thus being restricted to convex predictive models and distances [25, 29]; or, solve a relaxed version of the original optimization problem using gradient-based approaches, and therefore, are restricted to differentiable models and distance functions [32]. In addition to being model-specific, most of these approaches suffer from one or both of the following limitations: i) they do not provide any guarantees that the resulting counterfactual explanation is at *minimum distance* from the factual set of features; ii) they have limited *coverage* as they do not guarantee that the generated counterfactual is plausible, i.e, they do not ensure that the set of features in the counterfactual agree in data type and range with the input feature space of the predictive model.

Our contributions. In this paper, we propose a *model-agnostic* approach to generate nearest *counterfactual explanations*, namely MACE, under any given distance function (or convex combinations thereof); while, at the same time, easily supporting additional plausibility constraints. Moreover, our approach readily encodes natural notions of distance for *heterogeneous feature* spaces, which are common in consequential decision making system (e.g., loan approval) and consist of mixed numerical (e.g., age and income) and nominal features (e.g., gender and education level). To this end, in MACE we map the nearest counterfactual problem into a sequence of *satisfiability* problems, by expressing both the predictive model and the distance function (as well as the plausibility and diversity constraints) as logic formulae. Each of these satisfiability problems aims to verify if there exists a counterfactual explanation at a distance smaller than a given threshold, and can be solved using standard SMT (satisfiability modulo theories) solvers. Moreover, we rely on a binary search strategy on the distance threshold to find an approximation to the nearest (plausible) counterfactual with an arbitrary degree of accuracy, and a lower bound on distance such that no counterfactual provably exists at a smaller distance. Finally, once nearest counterfactuals are found, diversity constraints may be added to the satisfiability problems and the same procedure is used to find alternative counterfactuals. The overall architecture of MACE is illustrated in Figure 1.

Our experimental validation on real-world datasets show that MACE not only achieves 100% coverage by design, but also generates explanations that are significantly closer than previous approaches [28, 29]. We also provide qualitative examples showcasing the flexibility of our approach to generate actionable counterfactuals by extending our plausibility constraints to restrict changes to a subset of (non-protected) features. The Python implementation of our algorithms and the datasets used in our experiments are available at `https://github.com/???`.

2 Background on first-order predicate logic

In this section, we briefly recall basic concepts of first-order predicate logic, which MACE builds upon. We distinguish between *function symbols* (for instance, addition $+$ and multiplication $\times$) and *predicate symbols* (for instance, equality $=$ or lesser than $<$). Function symbols are used to build *expressions*, and predicate symbols are used to build *atomic formulae*. Examples of valid expressions are x, $x+2$, $(-x)+2$ and $(x+2)\times(y+3)$. Examples of valid atomic formulae are $e < e'$, $e \leq e'$ or

$e = e'$. A (quantifier-free) *formula* is a Boolean combinations of atomic formulae. That is, a formula is built from atomic formulae using conjunction $\wedge$, disjunction $\vee$, and negation $\neg$. Formulae have an *interpretation* over their intended domain. For instance, a formula about real-valued expressions has a natural interpretation as a subset of $\mathbb{R}^n$, where n denotes the number of variables that appear in the formula. The interpretation is obtained by mapping every variable into a value, e.g. a real number. For example, $(2, 1)$ belongs in the interpretation of $(x + 2) \times (y + 3) \leq x \times y + 16$ since the mapping $x \mapsto 2, y \mapsto 1$ assigns true because $16 \leq 18$. We say that a formula is *satisfiable* if its interpretation as a subset of $\mathbb{R}^n$ is non-empty.

The *satisfiability problem* consists in checking whether or not a formula is satisfiable. Satisfiability problems can be verified automatically using SMT (Satisfiability Modulo Theories) solvers like Z3 [5] or CVC4 [3]. We refer to [17] for an exposition of the basic algorithms used by SMT solvers. For the purpose of the next sections, it suffices to assume a given *satisfiability oracle* SAT. For our experiments, we use state-of-the-art SMT solvers to realize the oracle. We use SMT solvers as black-box, but it is interesting to note that our formulae fall in the linear fragment of the theory of reals (i.e. all formulae that only contain expressions of degree 1 when viewed as multi-variate polynomials over variables), which can be decided efficiently using the Fourier-Motzkin algorithm.

3 Counterfactual spaces for predictive models

This section defines a logical representation of counterfactual explanations for predictive models, which are functions mapping input feature vectors $\boldsymbol{x} \in \mathcal{X}$ into decisions $y \in \{0, 1\}$. [1] Given a predictive model $f : \mathcal{X} \to \{0, 1\}$, we can define the *set of counterfactual explanations* for a (factual) input $\hat{\boldsymbol{x}} \in \mathcal{X}$ as $\mathrm{CF}_f(\hat{\boldsymbol{x}}) = \{\boldsymbol{x} \in \mathcal{X} \mid f(\boldsymbol{x}) \neq f(\hat{\boldsymbol{x}})\}$. In words, $\mathrm{CF}_f(\hat{\boldsymbol{x}})$ contains all the inputs x for which the model f returns a prediction different from $f(\hat{\boldsymbol{x}})$. We also remark that $\mathrm{CF}_f(\hat{\boldsymbol{x}})$ is the set of preimages of $1 - f(\hat{\boldsymbol{x}})$ under f.

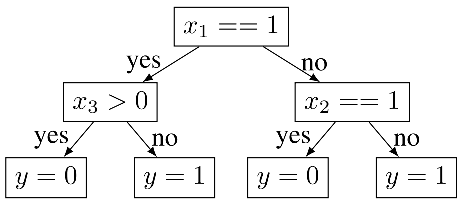
For a broad class of predictive models, it is possible to construct *counterfactual formulae* capturing membership in CF_f. We do so by computing the characteristic formula ϕ_f of the model. For a predictive model $f : \mathcal{X} \to \{0, 1\}$, and pair of input and output values $\boldsymbol{x}$ and y, the *characteristic formula* ϕ_f verifies that $\phi_f(\boldsymbol{x}, y)$ is valid if and only if $f(\boldsymbol{x}) = y$. Thus, given a factual input $\hat{\boldsymbol{x}}$ with $f(\hat{\boldsymbol{x}}) = \hat{y}$ and ϕ_f we define the *counterfactual formula* as

$$\phi_{\mathrm{CF}_f(\hat{\boldsymbol{x}})}(\boldsymbol{x}) = \phi_f(\boldsymbol{x}, 1 - \hat{y}) \tag{1}$$

Intuitively, the formula on the right hand side of (1) says that "$\boldsymbol{x}$ is a counterfactual for $\hat{\boldsymbol{x}}$ if either $f(\hat{\boldsymbol{x}}) = 0$ and $f(\boldsymbol{x}) = 1$, or $f(\hat{\boldsymbol{x}}) = 1$ and $f(\boldsymbol{x}) = 0$". It is thus clear from the definition that an input $\boldsymbol{x}$ satisfies $\phi_{\mathrm{CF}_f(\hat{\boldsymbol{x}})}$ if and only if $\boldsymbol{x} \in \mathrm{CF}_{f(\hat{\boldsymbol{x}})}$. Moreover, (1) shows that, to construct counterfactual formulae $\phi_{\mathrm{CF}_f(\hat{\boldsymbol{x}})}$, we only require the characteristic formulae of the corresponding predictive models, ϕ_f, and the value of $\hat{y}$. To obtain such characteristic formulae we assume that predictive models are represented by programs in a core programming language with assignments, conditionals, sequential composition, syntactically bounded loops and return statements. This allows us to use techniques from the program verification literature. Specifically, we use the so-called predicate transformers [6, 13, 9, 8]. The description of the general procedure is provided in Appendix A. For ease of exposition, we illustrate the construction of characteristic formulae through two examples, a decision tree and a multilayer perceptron.

As a first example, consider the decision tree from Figure 2a which takes as input $(x_1, x_2, x_3) \in \{0, 1\}^2 \times \mathbb{R}$ and returns a binary output in $\{0, 1\}$. Figure 2b provides the programming language description of this decision tree. To construct a formula representing the function $f(x) = y$ computed by this tree we first build a clause for each leaf in the tree by taking the conjunction of all the conditions encountered in the path from the root to the leaf. For example, the clause corresponding to the leftmost leaf on the tree in Figure 2a is $(x_1 = 1 \wedge x_3 > 0 \wedge y = 0)$. Once all these clauses are constructed, the characteristic formula $\phi_f(\boldsymbol{x}, y)$ corresponding to the full tree is obtained by taking the conjunction of all said clauses, as shown in Figure 2c. As a second example we consider a feed-forward neural network with one hidden layer followed by a ReLU activation function, as depicted in Figure 3a. This model implements a function $f : \mathbb{R}^3 \to \{0, 1\}$, where the binary decision is taken by thresholding the value of the last hidden node. The programming language representation

[1] While here we assume binary predictor models, i.e., classifiers, our approach generalizes to regression problems where $y \in \mathbb{R}$ and more generally any other output domain.



(a) Graphical representation

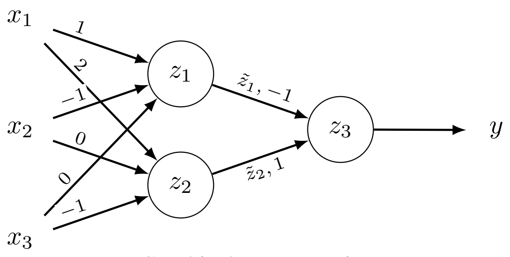
```
if x_1 == 1
    y = 0 if x_3 > 0 else 1
else
    y = 0 if x_2 == 1 else 1
return y
```

(b) Program (in Python)

$$\phi_f(\boldsymbol{x}, y) = (x_1 = 1 \wedge x_3 > 0 \wedge y = 0) \vee (x_1 = 1 \wedge x_3 \leq 0 \wedge y = 1) \\ \vee (x_1 = 0 \wedge x_2 = 1 \wedge y = 0) \vee (x_1 = 0 \wedge x_2 = 0 \wedge y = 1)$$

(c) Characteristic formula

Figure 2: Decision tree: model, program and characteristic formula



(a) Graphical representation

```
z_1 = x_1 - x_2
z_2 = 2x_1 - x_3
z̃_1 = z_1 if z_1 >= 0 else 0
z̃_2 = z_2 if z_2 >= 0 else 0
z_3 = -z̃_1 + z̃_2
y = 1 if z_3 >= 0 else 0
return y
```

(b) Program (in Python)

$$\phi_f(\boldsymbol{x}, y) = (z_1 = x_1 - x_2) \wedge (z_2 = 2x_1 - x_3) \wedge ((\tilde{z}_1 = z_1 \wedge z_1 \geq 0) \vee (\tilde{z}_1 = 0 \wedge z_1 < 0)) \\ \wedge ((\tilde{z}_2 = z_2 \wedge z_2 \geq 0) \vee (\tilde{z}_2 = 0 \wedge z_2 < 0)) \wedge (z_3 = -\tilde{z}_1 + \tilde{z}_2) \\ \wedge ((z_3 \geq 0 \wedge y = 1) \vee (z_3 < 0 \wedge y = 0))$$

(c) Characteristic formula

Figure 3: Multilayer perceptron: model, program and characteristic formula

of this model is given in Figure 3b. In this case, the characteristic formula predicates over inputs $\boldsymbol{x}$, output y and program variables z_i and $\tilde{z}_i$ for each hidden node i representing the values on that node before and after the non-linear ReLU transformation, respectively. The characteristic formula is a conjunction, and each conjunct corresponds to one instruction of the program. For example, for the leftmost hidden node in the first layer of the network in Figure 3a the variable z_1 is associated with the clause $(z_1 = x_1 - x_2)$; and the variable $\tilde{z}_1$ corresponds to the value of z_1 after the ReLU, which can be written as the disjunction $(\tilde{z}_1 = z_1 \wedge z_1 \geq 0) \vee (\tilde{z}_1 = 0 \wedge z_1 < 0)$. For the output node – in this case, z_3 – we introduce a pair of clauses representing the thresholding operation, i.e. $(y = 1 \wedge z_3 \geq 0) \vee (y = 0 \wedge z_3 < 0)$. Taking the conjunction of the formulas for each node we obtain the characteristic formula in Figure 3c.

4 Finding the nearest counterfactual

Based on the counterfactual space $\mathrm{CF}_f(\hat{\boldsymbol{x}})$ defined in the previous section, we would like to produce counterfactual explanations for the output of a model f on a given input $\hat{\boldsymbol{x}}$ by trying to find a *nearest counterfactual*, which is defined as:

$$\boldsymbol{x}^* \in \underset{\boldsymbol{x} \in \mathrm{CF}_f(\hat{\boldsymbol{x}})}{argmin}\ d(\boldsymbol{x}, \hat{\boldsymbol{x}}) \ . \tag{2}$$

For the time being, we assume that a notion of distance between instances, d, is given. For convenience, and without loss of generality, we also assume that d takes values in the interval $[0, 1]$.

Algorithm 1: Binary Search for Nearest Counterfactuals with Satisfiability Oracle

```
Input: Factual $\hat{x}$, counterfactual formula $\phi_{\mathrm{CF}_f(\hat{x})}$, distance formula $\phi_{d,\hat{x}}$, constraints formula $\phi_{g,\hat{x}}$,
        accuracy $\epsilon$
Output: Counterfactual $\hat{x}_\epsilon$, distance $\delta_{\max} = d(\hat{x}_\epsilon, \hat{x})$, lower bound $\delta_{\min}$ on (2)
Let $\delta_{\min} \leftarrow 0$ and $\delta_{\max} \leftarrow 1$
while $\delta_{\max} - \delta_{\min} > \epsilon$ do
    Let $\delta \leftarrow \frac{\delta_{\min}+\delta_{\max}}{2}$
    Let $\phi_{\hat{x},\delta}(x) \leftarrow \phi_{\mathrm{CF}_f(\hat{x})}(x) \wedge \phi_{d,\hat{x}}(x, \delta) \wedge \phi_{g,\hat{x}}$
    Let $x \leftarrow \mathsf{SAT}(\phi_{\hat{x},\delta})$
    if $x$ is "unsatisfiable" then
        Let $\delta_{\min} \leftarrow \delta$
    else
        Let $\hat{x}_\epsilon \leftarrow x$ and $\delta_{\max} \leftarrow \delta$
return $\hat{x}_\epsilon$, $\delta_{\min}$, $\delta_{\max}$
```

4.1 Main algorithm

Our goal now is to leverage the representation of $\mathrm{CF}_f(\hat{x})$ in terms of a logic formula to solve (2). To this end, we map the optimization problem in (2) into a sequence of satisfiability problems, which can be verified or refuted by standard SMT solvers. We do so by first converting the expression $d(x, \hat{x}) \leq \delta$, where $\delta \in [0, 1]$, into a logic formula $\phi_{d,\hat{x}}(x, \delta)$, which is valid if and only if $d(x, \hat{x}) \leq \delta$. We assume here that the distance d function is expressed by a program in the same language that we used to represent the models in Section 3. In particular, we can leverage the procedure detailed in Appendix A to automatically construct $\phi_{d,\hat{x}}$. Then, both the counterfactual formula $\phi_{\mathrm{CF}_f(\hat{x})}(x)$ and the distance formula $\phi_{d,\hat{x}}(x, \delta)$ are combined into the logic formula:

$$\phi_{\hat{x},\delta}(x) = \phi_{\mathrm{CF}_f(\hat{x})}(x) \wedge \phi_{d,\hat{x}}(x, \delta) \ ,$$

which is satisfiable if and only if there exists a counterfactual $x \in \mathrm{CF}_f(\hat{x})$ such that $d(x, \hat{x}) \leq \delta$. To check whether the above formula is satisfiable we use the satisfiability oracle $\mathsf{SAT}(\psi(x))$ which returns either an instance x such that $\psi(x)$ is valid, or "unsatisfiable" if no such x exists.

Note that, while the oracle SAT allows us to verify if there exist counterfactual explanations at distance smaller or equal than a given threshold δ, solving optimization (2) requires finding a nearest counterfactual. To do so, we apply a binary search strategy on the distance threshold $\delta \in [0, 1]$ that allows us to find *approximately* nearest counterfactuals with a pre-specified degree of accuracy. This is implemented in Algorithm 1, which for an accuracy parameter $\epsilon > 0$ makes at most $O(\log(1/\epsilon))$ calls to SAT and returns a counterfactual $\hat{x}_\epsilon \in \mathrm{CF}_f(\hat{x})$ such that $d(\hat{x}_\epsilon, \hat{x}) \leq d(x^*, \hat{x}) + \epsilon$, where x^* is some solution of the optimization problem in (2). This mild dependence on the accuracy ϵ allows Algorithm 1 to find arbitrarily accurate solutions of (2) while only making a moderate number of calls to the satisfiability oracle. Note that Algorithm 1 may also account for potential plausibility or diversity constraints (refer to next section for further details).

We remark here our approach to find nearest counterfactuals is agnostic to the details of the model and distance being used; the only requirement is that they must be expressable in a fairly general programming language. As a consequence, we can handle a wide variety of predictive models, including both differentiable – such as, logisitic regression and multilayer perceptron – and non-differentiable predictive models – e.g., decision trees and random forest–; as well as a wide variety of distance functions (refer to next section for further details). Moreover, the bound $\delta_{\min}$ returned by Algorithm 1 provides a certificate that any solution x^* to (2) must satisfy $d(x^*, \hat{x}) > \delta_{\min}$. This is because whenever $\mathsf{SAT}(\psi(x))$ returns "unsatisfiable" it does so by internally constructing a proof that the formula $\psi(x)$ is not valid.

4.2 Distance, Plausibility, and Diversity

Next we discuss additional criteria in the form of logic clauses that guide the satisfiability problem towards generating a counterfactual explanation with desired properties.

Distance. We first discuss several forms for the distance function $d(x, \hat{x})$ that can be used to define the notion of nearest counterfactual. To this end, we first remark that in consequential decision making the input feature space $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_J$ is often heterogeneous – for example, gender is categorical, education level is ordinal, and income is a numerical variable. We define an appropriate distance metric for every kind of variable in the input feature space of the model as:

$$\delta_j(x_j, \hat{x}_j) = \begin{cases} |x_j - \hat{x}_j|/R_j & \text{if } x_j \text{ is numerical} \\ \mathbb{I}[x_j \neq \hat{x}_j] & \text{if } x_j \in \{1, 2, \ldots, R_j\} \text{ is categorical} \\ |x_j - \hat{x}_j|/R_j & \text{if } x_j \in \{1, 2, \ldots, R_j\} \text{ is ordinal} \end{cases} ,$$

where R_j corresponds to the range of the feature x_j and is used to normalize the distances for all input features, such that $\delta_j : \mathcal{X}_j \times \mathcal{X}_j \to [0, 1]$ for all j, independently on the feature type. By defining the distance vector $\boldsymbol{\delta} = (\delta_1, \cdots, \delta_J)$ (being J the total number of input features), one can now write the distance between instances as:

$$d(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \alpha||\boldsymbol{\delta}||_0 + \beta||\boldsymbol{\delta}||_1 + \gamma||\boldsymbol{\delta}||_\infty \ , \tag{3}$$

where $|| \cdot ||_p$ is the p-norm of a vector, and $\alpha, \beta, \gamma \geq 0$ such that $(\alpha + \beta)/J + \gamma = 1$. Intuitively, 0-norm is used to restrict the number of features that changes between the initial instance $\hat{\boldsymbol{x}}$ and the generated counterfactual $\hat{\boldsymbol{x}}_\epsilon$; the 1-norm is used to restrict the average change distance between $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{x}}_\epsilon$; and ∞-norm is used to restrict maximum change across features. Any distance of this type can easily be expressed as a program.

Plausibility. Up to this point, we have only considered minimum distance as the only requirement for generating a counterfactual. However, this might result in unrealistic counterfactuals, such as e.g., decrease the age or change the gender of a loan applicant. To avoid unrealistic counterfactuals, one may introduce additional *plausibility constraints* in the optimization problem in Eq. (2). This is equivalent to adding a conjunction in the constraint formula $\phi_{g,\hat{\boldsymbol{x}}}$ in Algorithm 1 that accounts for any additional plausibility formulae ϕ_p, which ensure that: i) only actionable features [29] are changed in the resulting counterfactual; and ii) each feature in the counterfactual only takes plausible values. Specifically, we account for a non-actionable feature x_j, i.e., a feature whose value in the counterfactual explanation should match its initial value, by setting ϕ_p to be $(x_j = \hat{x}_j)$. Similarly, we account for variables that only allow for increasing values by setting $\phi_p = (x_j \geq \hat{x}_j)$.

Importantly, since here we are working with heterogeneous feature spaces, we require all the features in the counterfactual to agree in both the data-types (categorical, ordinal, etc.) and the data-ranges with the input space to the predictive model. In particular, if a categorical (ordinal) feature is one-hot (thermometer) encoded to be used as input to the predictive model, e.g., a logistic regression classifier, we make sure that the generated counterfactual provides a valid one-hot vector (thermometer) for such feature. Likewise, for any numerical feature we ensure that its value in the counterfactual falls into observed range in the original data used to train the predictive model.

Diversity. Finally, one might be interested in generating a (small) set of diverse counterfactual explanations for the same instance $\hat{\boldsymbol{x}}$. To this end, we iteratively call Algorithm 1 with a constraints formula ϕ_v that includes diversity clauses to ensure that the newly generated explanation is substantially different from all the previous ones. We can encode diversity by forcing that the distance between every pair of counterfactual explanations is greater than a given value – for example, enforcing the distance vector between counterfactuals to have 0-norm at least 1 will make sure that the set of input features, where the factual and the generated counterfactuals differ, is different for each explanation.

5 Experiments

In this section, we empirically evaluate MACE at generating counterfactual explanations on three real-world datasets in the context of loan approval (Adult [1] and Credit [34] datasets) and pretrial bail (COMPAS dataset [18]). Appendix B provides complete details on these datasets.

Experimental set-up. We consider as predictive models decision trees, random forest, logistic regression, and multilayer perceptron; which we train on the three datasets using the Python library scikit-learn [21], with default parameters.[2] Furthermore, we built MACE atop the open-source PySMT library [11] with the Z3 [5] backend, and the sole hyperparamater ϵ was set to 10^{-5}. Additionally, following the motivation in the previous section, we experiment with the ℓ_0, ℓ_1, ℓ_∞ norms to identify the nearest counterfactuals.

[2]For the multilayer perceptron, we used two hidden layers with 10 neurons each to avoid overfitting.

Table 1: Comparison of approaches for generating counterfactual explanations, based on the supported model types, data types, distance types, and plausibility constraints (actionability, data type & range).

Approach	Models	Data-types	Distances	Plausibility
Proposed (MACE)	tree, forest, lr, mlp	heterogeneous	$\ell_p \ \forall\, p$	✓
Minimum Observable (MO)[3]	-	heterogeneous	$\ell_p \ \forall\, p$	✓
Feature Tweaking (FT) [28]	tree, forest	heterogeneous	$\ell_p \ \forall\, p$	x
Actionable Recourse (AR) [29]	lr	numeric, binary	ℓ_1, ℓ_∞	x

Table 2: Coverage Ω computed on $N = 500$ factual samples. For comparison, MO and MACE always have 100% coverage by definition and by design, respectively. Cells are shaded when tests are not supported. The higher the %, the higher the coverage (better performance).

		Adult			Credit			COMPAS		
		ℓ_0	ℓ_1	ℓ_∞	ℓ_0	ℓ_1	ℓ_∞	ℓ_0	ℓ_1	ℓ_∞
tree	PFT	0%	0%	0%	68%	68%	68%	74%	74%	74%
forest	PFT	0%	0%	0%	99%	99%	99%	100%	100%	100%
lr	AR	▨	18%	0.4%	▨	100%	100%	▨	100%	100%

Metrics. To assess the performance of our approach, we recall the criteria of good explanations for consequential decisions: i) the returned counterfactual should be as near as possible to the factual sample corresponding to the individual's features; ii) the returned counterfactual must be plausible (refer to Section 4.2). Hence, we quantitatively compare the performance of MACE with the above approaches in terms of i) the *normalized distance* $\boldsymbol{\delta}$; and ii) *coverage* Ω indicating the percentage of factual samples for which the approach generates plausible (in type and range) counterfactuals.

Baselines. We compare the performance (in terms for the metrics above) of MACE at generating the nearest counterfactual explanations with: the *Minimum Observable* (MO) approach,[3] which seeks the closest sample present in the dataset that flips the prediction; the *Feature Tweaking* (FT) approach [28], which searches for the nearest counterfactual lying close to the decision boundary of a Random Forest; and the *Actionable Recourse* (AR) [29], which solves a mixed integer linear program to obtain counterfactual explanations for Linear Regression models. Table 1 summarizes the main properties of all the considered approaches to generate counterfactuals.

For each combination of model, dataset, distance, and approach, we generate the nearest counterfactual explanations for a held-out set of 500 instances classified as negative by the corresponding model. Unfortunately, we found that FT not once returned a plausible counterfactual. As a consequence, we modified the original implementation of FT, to ensure that the generated counterfactuals are plausible. The resulting *Plausible Feature Tweaking* (PFT) projects each candidate counterfactual into a plausible domain before selecting the nearest counterfactual amongst them.

Results. Table 2 shows the coverage Ω of all the approaches based only on data-range and data-type plausibility. Note that, since by definition both MACE and MO have 100% coverage, we have not depicted these values in the table. In contrast, PFT fails to return counterfactuals for roughly 15% of the Credit and COMPASS datasets, while both PFT and AR achieve minimal coverage on the Adult dataset – see Appendix B for a description of the failure points of Actionable Recourse, as reported by the authors. After arriving at a set of plausible counterfactual explanations for the same factual samples, we are able to compute the relative distance reductions achieved when using MACE as compared to other approaches, as shown in Table 3 (additionally, Figure 4 in Appendix B shows the distribution of the distance of the generated plausible counterfactual for all models, datasets, distances and approaches). Here, we observe that MACE results in significantly closer counterfactual explanations than competing approaches. As a consequence, the counterfactuals generated by MACE would require significantly less effort on behalf of the affected individual in order to achieve the desired prediction.

Finally, we qualitatively analyze the generated counterfactuals to ensure that they are meaningful. However, we observe that many of the counterfactuals require changes in features that are often protected by law such as, age, race, and gender [2]. As an example, for a trained random forest, the counterfactuals generated by both the MACE and MO approaches required individuals to change

[3]Identical to the deployed What If tool by Google's PAIR team: `https://pair-code.github.io/what-if-tool/walkthrough.html`

Table 3: Percentage of improvement in distances, computed as $100 * \mathbb{E}[1 - \boldsymbol{\delta}_{\text{MACE}}/\boldsymbol{\delta}_{\text{Other}}]$. $N = \Omega_{\text{MACE}} \cap \Omega_{\text{Other}}$ factual samples. Cells are shaded when tests are not supported. The higher the % the better the improvement.

		Adult			Credit			COMPAS		
		ℓ_0	ℓ_1	ℓ_∞	ℓ_0	ℓ_1	ℓ_∞	ℓ_0	ℓ_1	ℓ_∞
tree	MACE vs MO	47%	81%	72%	67%	97%	94%	1%	5%	5%
	MACE vs PFT				53%	97%	96%	15%	56%	54%
forest	MACE vs MO	51%	82%	71%	68%	97%	96%	1%	6%	6%
	MACE vs PFT				53%	96%	96%	4%	28%	27%
lr	MACE vs MO	62%	93%	88%	80%	82%	81%	3%	7%	6%
	MACE vs AR		5%	91%		41%	71%		10%	38%
mlp	MACE vs MO	85%	99%	98%	89%	99%	99%	58%	92%	88%

Table 4: Percentage of factual samples for which the nearest counterfactual sample requires a change in age for a random forest trained on the Adult dataset, and the corresponding increase in distance to nearest counterfactual when restricting the approaches not to change age: $100 \times \mathbb{E}[\delta_{\text{restr.}}/\delta_{\text{unrestr.}} - 1]$. The higher %, the greater the increase in distance.

	ℓ_0		ℓ_1		ℓ_∞	
	% age-change	rel. dist. increase	% age-change	rel. dist. increase	% age-change	rel. dist. increase
MACE	13.2%	9.0%	20.4%	100.3%	84.4%	32.8%
MO	78.8%	50.9%	92.0%	245.7%	95.6%	193.3%

their age. Worse yet, for a substantial portion of the counterfactuals, a reduction in age was required, which is not even possible. To further study this effect, we regenerate counterfactual explanations for those samples for which age-change was required, with an additional plausibility constraint ensuring that the age shall not change (results with constraints to ensure non-decreasing age are shown in Appendix B). The results presented in Table 4 show interesting results. First, we observe that the additional plausibility constraint for the age incurs significant increases in the distance of the nearest counterfactual – being, as expected, more pronounced for the ℓ_1 and the ℓ_∞ norms, since the ℓ_0 norm only accounts for the number of features that change in the counterfactual but not for how much they change. For the ℓ_0 norm, we find that for the 66 factual samples (i.e., $13.2\% \times 500$) for which the unrestricted MACE required age-change, the addition of the no-age-change constraint results in counterfactuals at very similar distance. In fact, of the newly generated counterfactuals, $8/66$ only require a change in Occupation, and $19/66$ only require a change in Capital Gains, therefore remaining at the same distance as the original counterfactual. In contrast, for the ℓ_1 and the ℓ_∞ norms we find that the restricted counterfactual incurs a significant increase in the distance (cost) with respect to the unrestricted counterfactual. These results suggest that the predictions of the random forest trained on the Adult data are strongly correlated to the age, which is often legally and socially considered as unfair. Thus, our results show that counterfactuals may assist in qualitatively ascertaining if other desiderata, such as fairness, are met [7, 33].

6 Conclusions

In this work, we have presented a novel approach for generating counterfactual explanations in the context of consequential decisions. Building on theory and tools from formal verification, we demonstrated that a large class of predictive models can be compiled to formulae which can be verified by standard SMT-solvers. By conjuncting the model formulae with formulae corresponding to distance, plausibility, and diversity constraints, we demonstrated on three real-world datasets and four popular predictive models that the proposed method not only achieves perfect coverage, but also generates counterfactuals at more favorable distances than existing approaches. Furthermore, we showed that the proposed method can not only provide explanations for individuals subject to automated decision making systems, but also inform system administrators regarding the potentially unfair reliance of the model on protected attributes.

There are a number of interesting directions for future work. First, MACE can naturally be extended to support counterfactual explanations for multi-class classification models, as well as regression scenarios. Second, in regards to the multi-faceted notion of plausibility defined in Section 4.2, it would be interesting to account for statistical correlations and causal relationships among the features when generating counterfactual explanations. Third, we would like also to explore how different notions of diversity may help generating meaningful and useful counterfactuals. Finally, in our experiments we noticed that the running time of MACE directly depends on the efficiency of the SMT solver. As future work we aim to explore the scalability of the proposed method by investigating ideas that have been developed in the context of formal verification of deep neural networks [14, 15, 27] and optimization modulo theories [20, 26].

References

[1] Adult data. https://archive.ics.uci.edu/ml/datasets/adult, 1996.

[2] S. Barocas and A. D. Selbst. Big data's disparate impact. *SSRN Electronic Journal*, 01 2016.

[3] C. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanovic, T. King, A. Reynolds, and C. Tinelli. CVC4. In G. Gopalakrishnan and S. Qadeer, editors, *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV '11)*, volume 6806, pages 171–177. Springer, jul 2011.

[4] R. Cytron, J. Ferrante, B. K. Rosen, M. N. Wegman, and F. K. Zadeck. Efficiently computing static single assignment form and the control dependence graph. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 13(4):451–490, 1991.

[5] L. M. de Moura and N. Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and J. Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008*, volume 4963, pages 337–340. Springer, 2008.

[6] E. W. Dijkstra. A constructive approach to the problem of program correctness. *BIT Numerical Mathematics*, 8(3):174–186, Sep 1968.

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

[8] C. Flanagan and J. B. Saxe. Avoiding exponential explosion: Generating compact verification conditions. In *ACM SIGPLAN Notices*, volume 36, pages 193–205. ACM, 2001.

[9] R. W. Floyd. *Assigning Meanings to Programs*, pages 65–81. Springer Netherlands, Dordrecht, 1993.

[10] A. A. Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD explorations newsletter*, 15(1):1–10, 2014.

[11] M. Gario and A. Micheli. Pysmt: a solver-agnostic library for fast prototyping of smt-based algorithms. In *SMT Workshop 2015*, 2015.

[12] D. Gunning. Darpa's explainable artificial intelligence (xai) program. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*, pages ii–ii. ACM, 2019.

[13] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[14] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety verification of deep neural networks. In R. Majumdar and V. Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV*, volume 10426, pages 3–29. Springer, 2017.

[15] G. Katz, C. W. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In R. Majumdar and V. Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV*, volume 10426, pages 97–117. Springer, 2017.

[16] Y. Kodratoff. The comprehensibility manifesto. *KDD Nugget Newsletter*, 94(9), 1994.

[17] D. Kroening and O. Strichman. *Decision Procedures: An Algorithmic Point of View*. Springer Publishing Company, Incorporated, 1 edition, 2008.

[18] J. Larson, S. Mattu, L. Kirchner, and J. Angwin. https://github.com/propublica/compas-analysis, 2016.

[19] Z. C. Lipton. The mythos of model interpretability. *Queue*, 16(3):30:31–30:57, June 2018.

[20] R. Nieuwenhuis and A. Oliveras. On SAT modulo theories and optimization problems. In A. Biere and C. P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT*, volume 4121, pages 156–169. Springer, 2006.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[22] B. K. Rosen, M. N. Wegman, and F. K. Zadeck. Global value numbers and redundant computations. In *Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 12–27. ACM, 1988.

[23] C. Rudin. Please stop explaining black box models for high stakes decisions. *arXiv preprint arXiv:1811.10154*, 2018.

[24] S. Rüping. *Learning interpretable models*. PhD dissertation, Technical University of Dortmund, 2006.

[25] C. Russell. Efficient search for diverse coherent explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 20–28. ACM, 2019.

[26] R. Sebastiani and S. Tomasi. Optimization in SMT with $\mathcal{LA}(\mathbb{Q})$ cost functions. In B. Gramlich, D. Miller, and U. Sattler, editors, *Automated Reasoning - 6th International Joint Conference, IJCAR*, volume 7364, pages 484–498. Springer, 2012.

[27] G. Singh, T. Gehr, M. Püschel, and M. T. Vechev. An abstract domain for certifying neural networks. *PACMPL*, 3(POPL):41:1–41:30, 2019.

[28] G. Tolomei, F. Silvestri, A. Haines, and M. Lalmas. 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*, pages 465–474. ACM, 2017.

[29] B. Ustun, A. Spangher, and Y. Liu. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 10–19. ACM, 2019.

[30] P. Voigt and A. Von dem Bussche. The EU general data protection regulation (GDPR). 2017.

[31] S. Wachter, B. Mittelstadt, and L. Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2):76–99, 2017.

[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*, 31(2), 2017.

[33] A. Weller. Challenges for transparency. In *Workshop on Human Interpretability in Machine Learning (ICML)*, 2017.

[34] I.-C. Yeh and C.-h. Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2):2473–2480, 2009.

A Background on programming language and program verification

Programs We assume given a set of function symbols with their arity. For simplicity, we consider the case where operators are untyped and have arity 0 (constants), 1 (unary functions), and 2 (binary functions). We let c, c_1, and c_2 range over constants, unary functions and binary functions respectively. Expressions are built from function symbols and variables. The set of expressions is defined inductively by the following grammar:

$$\begin{array}{rcll} e & ::= & x & \text{variable} \\ & | & c & \text{constant} \\ & | & c_1(e) & \text{unary function} \\ & | & c_2(e_1, e_2) & \text{binary function} \end{array}$$

We next assume given a set of atomic predicates. For simplicity, we also consider that predicates have arity 1 or 2, and let P_1 and P_2 range over unary and binary predicates respectively. We define guards using the following grammar:

$$\begin{array}{rcll} b & ::= & P_1(e) & \text{unary predicate} \\ & | & P_2(e_1, e_2) & \text{binary predicate} \\ & | & b_1 \& b_2 & \text{conjunction} \\ & | & b_1 \parallel b_2 & \text{disjunction} \\ & | & \neg b & \text{negation} \end{array}$$

We next define commands. These include assignments, conditionals, bounded loops and return expressions. The set of commands is defined inductively by the following grammar:

$$\begin{array}{rcll} c & ::= & \mathsf{skip} & \text{no-op} \\ & | & x := e & \text{assignment} \\ & | & c_1; c_2 & \text{sequential composition} \\ & | & \mathsf{if}\ b\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2 & \text{conditionals} \\ & | & \mathsf{for}\ (i = 1, \ldots, n)\ \mathsf{do}\ c & \text{for loop} \\ & | & \mathsf{return}\ e & \text{return statement} \end{array}$$

We assume that programs satisfy a well-formedness condition. The condition requires that return expressions have no successor instruction, i.e. we do not allow commands of the form $\mathsf{return}\ e;\ c$ or $\mathsf{if}\ b\ \mathsf{then}\ c;\ \mathsf{return}\ e\ \mathsf{else}\ c'; c''$. This is without loss of generally, since commands can always be transformed into functionally equivalent programs which satisfy the well-formedness condition.

Single assignment form Our first step to construct characteristic formulae is to transform programs in an intermediate form that is closer to logic. Without loss of generality, we consider loop-free commands, since loops can be fully unrolled. The intermediate form is called a variant of the well-known SSA form [22, 4] from compiler optimization. Concretely, we transform programs into some weak form of single assignment. This form requires that every non-input variable is defined before being used, and assigned at most once during execution for any fixed input. The main difference with SSA form is that we do not use so-called ϕ-nodes, as we require that variables are assigned at most once for any fixed input. More technically, our transformation can be seen as a composition of SSA transform with a naive de-SSA transform where ϕ-nodes are transformed into assignments in the branches of the conditionals.

Path formulae and characteristic formulae Our second step is to define the set of path formulae. Informally, a path formula represents a possible execution of the program. Fix a distinguished variable y for return values. Then the path formulae of a command c is defined inductively by the clauses:

$$\begin{array}{rcl} \mathrm{PF}_{z:=e}(y) & = & \{z = e\} \\ \mathrm{PF}_{c_1;c_2}(y) & = & \{\phi_1 \wedge \phi_2 \mid \phi_1 \in \mathrm{PF}_{c_1}(y) \wedge \phi_2 \in \mathrm{PF}_{c_2}(y)\} \\ \mathrm{PF}_{\mathrm{if}\ b\ \mathrm{then}\ c_1\ \mathrm{else}\ c_2}(y) & = & \{b \wedge \phi_1 \mid \phi_1 \in \mathrm{PF}_{c_1}(y)\} \cup \{\neg b \wedge \phi_2 \mid \phi_2 \in \mathrm{PF}_{c_2}(y)\} \\ \mathrm{PF}_{\mathsf{return}\ e}(y) & = & \{y = e\} \end{array}$$

The characteristic formula ϕ_c of a command c is then defined as:

$$\bigvee_{\phi \in \mathrm{PF}_c(y)} \phi$$

One can prove that for every inputs $x_1, \ldots, x_n$, the formula $\phi_y(x_1, \ldots, x_n, v)$ is valid iff the execution of c on inputs $x_1, \ldots, x_n$ returns v. Note that, strictly speaking, the formula ϕ_y contains as free variables the distinguished variable y, the inputs $x_1, \ldots, x_n$ of the program, *and* all the program variables, say $z_1 \ldots z_m$. However, the latter are fully defined by the characteristic formula so validity of $\phi_y(x_1, \ldots, x_n, v)$ is equivalent to validity of $\exists z_1 \ldots z_m.\ \phi_y(x_1, \ldots, x_n, v)$

B Experiment Details

In this section we provide further details on the detasets and methods used in or experiments, together with some additional results.

B.1 Datasets

First, we detail the different types of variables present in each dataset. We used the default features for the Adult and COMPAS datasets, and applied the same preprocessing used in [29] for the Credit dataset. All samples with missing data were dropped.

Adult dataset includes $45,222$ samples with 12 features, consisting of the following feature types:

- Integer: Age, Education Number, Hours Per Week
- Real: Capital Gain, Capital Loss
- Categorical: Sex, Native Country, Work Class, Marital Status, Occupation, Relationship
- Ordinal: Education Level

Credit dataset includes $29,623$ samples with 14 features, consisting of the following feature types:

- Integer: Total Overdue Counts, Total Months Overdue, Months With Zero Balance Over Last 6 Months, Months With Low Spending Over Last 6 Months, Months With High Spending Over Last 6 Months
- Real: Max Bill Amount Over Last 6 Months, Max Payment Amount Over Last 6 Months, Most Recent Bill Amount, Most Recent Payment Amount
- Categorical: Is Male, Is Married, Has History Of Overdue Payments
- Ordinal: Age Group, Education Level

COMPAS dataset includes $5,278$ samples with 5 features, consisting of the following feature types:

- Integer: -
- Real: Priors Count
- Categorical: Race, Sex, Charge Degreee
- Ordinal: Age Group

B.2 Handling Mixed Data Types

While the proposed method naturally handles mixed data types, other methods do not. Specifically, the Feature Tweaking method generates counterfactual explanations for Random Forest models trained on non-hot embeddings of the dataset, meaning that the resulting counterfactuals will not have multiple categories of the same variable activated at the same time. However, because this method is only restricted to working with real-valued variables, the resulting counterfactual is must undergo a post-processing step to ensure integer-, categorical-, and ordinal-based variables are plausible in the counterfactual. The Actionable Recourse method, on the other hand, explanations for Logistic Regression models trained on one-hot embeddings of the dataset, hence requiring additional constraints to ensure that multiple categories of a categorical variable are not simultaneously activated in the counterfactual. While the authors suggest how this can be supported using their method, their open-source implementation *converts categorical columns to binary where possible and drops other more complicated categorical columns*, postponing to future work. Furthermore, the authors state that *the question of mutually exclusive features will be revisited in later releases* [4]. Moreover, ordinal

[4]`https://github.com/ustunb/actionable-recourse/blob/master/examples/ex_01_quickstart.ipynb`

variables are not supported using this method. The overcome these shortcomings, the counterfactuals generated by both methods is post-processed to ensure correctness of variable types by rounding integer-based variables, and taking the maximally activated category as the counterfactual category.

B.3 Additional Results

Following the study of counterfactuals that change or reduce age (Section 5), we regenerate counterfactual explanations for those samples for which age-reduction was required, with an additional plausibility constraint ensuring that the age shall not decrease. The results presented in Table 5 show interesting results. Once again, we observe that the additional plausibility constraint for the age incurs significant increases in the distance of the nearest counterfactual – being, as expected, more pronounced for the ℓ_1 and the ℓ_∞ norms. For the ℓ_0 norm, we find that for the 18 factual samples (i.e., $3.6\% \times 500$) for which the unrestricted MACE required age-reduction, the addition of the no-age-reduction constraint results in counterfactuals at the same distance, while suggesting a change in work class $(5/18)$ or education level $(4/18)$ instead of changing age.

Finally, Figure 4 provides a visual comparison of the nearest distances found for a fixed held-out set of 500 factual samples, using all combinations of models, datasets, distances, and approaches.

Table 5: Percentage of factual samples for which the nearest counterfactual sample requires a reduction in age for a random forest trained on the Adult dataset, and the corresponding increase in distance to nearest counterfactual when restricting the approaches not to reduce age: $100 \times \mathbb{E}[\delta_{\text{restr.}}/\delta_{\text{unrestr.}} - 1]$.

	ℓ_0		ℓ_1		ℓ_∞	
	% age-red.	rel. dist. increase	% age-red.	rel. dist. increase	% age-red.	rel. dist. increase
MACE	3.6%	0%	7.4%	61.3%	34.2%	13.9%
MO	24.6%	29.7%	34.6%	94.6%	34.2%	66.6%

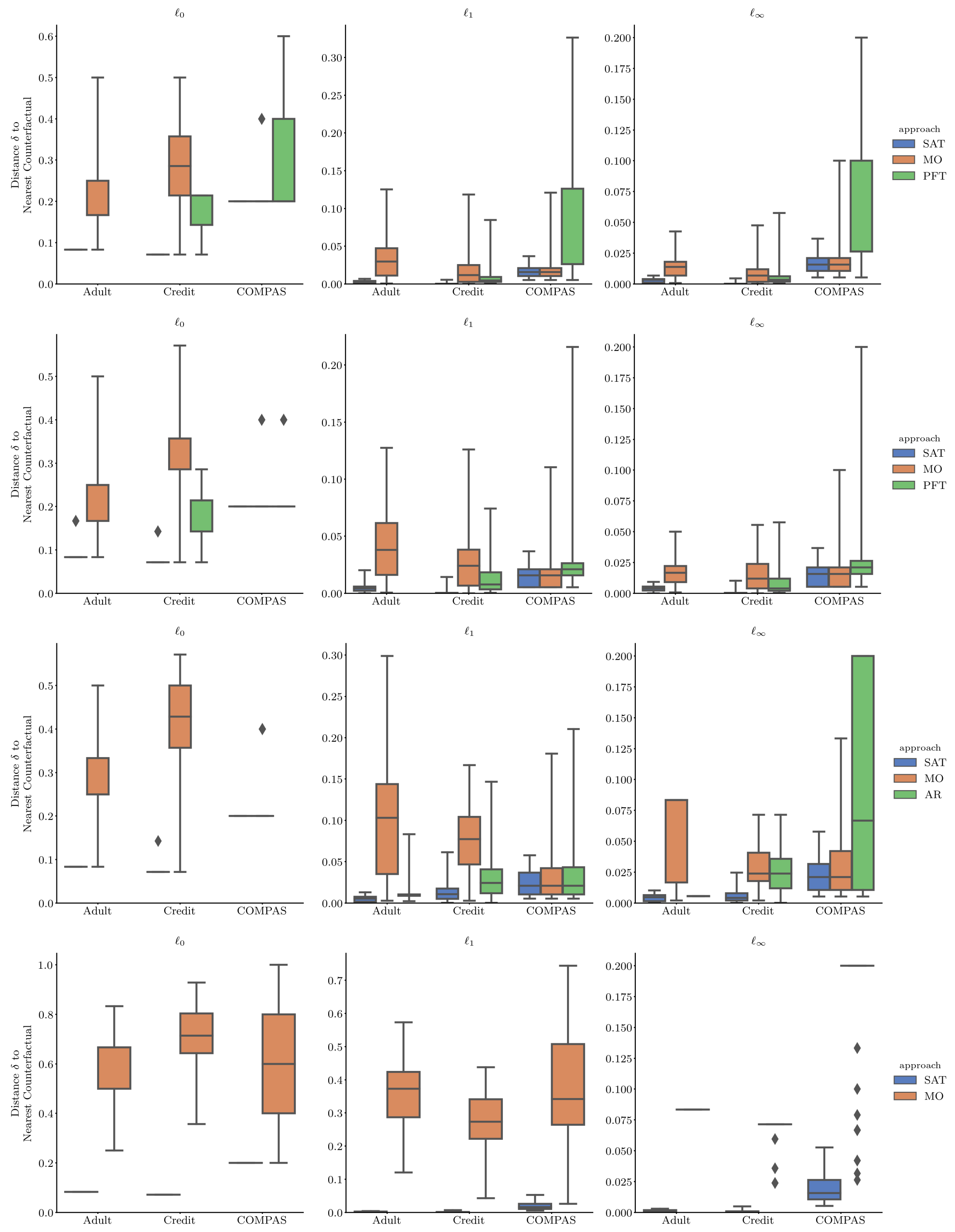


Figure 4: Comparison of approaches for generating counterfactual explanations for a (top to bottom) trained decision tree, random forest, logistic regression, and multilayer perceptron model. Lower distance is better. For each bar, $N = 500 \times \Omega$ from Table 2, and absent bars refer to $\Omega = 0$.