ReLACE: Reinforcement Learning Agent for Counterfactual Explanations of Arbitrary Predictive Models

Ziheng Chen†, Fabrizio Silvestri‡, Gabriele Tolomei‡, He Zhu††, Jia Wang⋆⋆, Hongshik Ahn⋆

⋆Department of Applied Mathematics and Statistics, Stony Brook University, NY, USA
†Department of Computer, Control and Management Engineering, Sapienza University of Rome, Italy
‡Department of Computer Science, Sapienza University of Rome, Italy
††Computer Science Department, Rutgers University, NJ, USA
⋆⋆Department of Intelligent Science, Xi’an Jiaotong-Liverpool University, China

Abstract

The demand for explainable machine learning (ML) models has been growing rapidly in recent years. Amongst the methods proposed to associate ML model predictions with human-understandable rationale, counterfactual explanations are one of the most popular. They consist of post-hoc rules derived from counterfactual examples (CFs), i.e., modified versions of input samples that result in alternative output responses from the predictive model to be explained. However, existing CF generation strategies either exploit the internals of specific models (e.g., random forests or neural networks), or depend on each sample’s neighborhood, which makes them hard to be generalized for more complex models and inefficient for larger datasets. In this work, we aim to overcome these limitations and introduce a model-agnostic algorithm to generate optimal counterfactual explanations. Specifically, we formulate the problem of crafting CFs as a sequential decision-making task and then find the optimal CFs via deep reinforcement learning (DRL) with discrete-continuous hybrid action space. Differently from other techniques, our method is easily applied to any black-box model, as this resembles the environment that the DRL agent interacts with. In addition, we develop an algorithm to extract explainable decision rules from the DRL agent’s policy, so as to make the process of generating CFs itself transparent. Extensive experiments conducted on several datasets have shown that our method outperforms existing CF generation baselines.

1 Introduction

Recent years have witnessed surprising advances in machine learning (ML), which in turn have led to the pervasive application of ML models across several domains. The excitement for the astonishing performance obtained by ML systems has created the need for really understanding the rationale behind ML model predictions. Indeed, knowing why an ML model returns a certain output in response to a given input is pivotal for a variety of reasons, such as model debugging, aiding decision-making, or fulfilling legal requirements (EU 2016). This applies especially to social areas like finance (Bracke et al. 2019) and healthcare (Ahmad, Eckert, and Teredesai 2018), where human-to-human relationships still play a significant role.

To properly achieve model transparency, a new initiative named eXplainable AI (XAI) has emerged (Samek et al. 2019). A large body of work on XAI has flourished in recent years (Guidotti et al. 2018), and approaches to XAI can be broadly categorized into two classes (Verma, Dickerson, and Hines 2020): (i) native and (ii) post-hoc. The former leverages ML models that are inherently interpretable and transparent, such as linear/logistic regression, decision trees, association rules, etc. The latter aims at generating ex post explanations for predictions made by opaque or black-box models like random forests and (deep) neural networks.

In this work, we focus on specific post-hoc explanations called counterfactual explanations, which are used to interpret predictions of individual instances in the form: “If A had been different, B would not have occurred” (Stepin et al. 2021). They work by generating modified versions of input samples that result in alternative output responses from the predictive model, i.e., counterfactual examples (CFs). Typically, the problem of creating CFs is formulated as an optimization task, whose goal is to find the “closest” data point to a given instance, which crosses the decision boundary induced by a trained predictive model. Depending on the level of access to the underlying predictive model, different CF generation methods have been proposed. CF generators based on mixed integer programming have proven effective (Kanamori et al. 2020; Karimi et al. 2020a,b; Russell 2019; Ustun, Spangher, and Liu 2019) once they have complete access to the model’s internals. Gradient-based CF generators need access to the model’s gradients, but they are limited to differentiable models only (Wachter, Mittelstadt, and Russell 2017; Pawelczyk, Broelemann, and Kasneci 2020). Gradient-free optimization-based algorithms (McGrath et al. 2018; Laugel et al. 2018; Looveren and Klaas 2019; Dandl et al. 2020) generate CFs without the knowledge of the model at all. However, existing CF generators either exploit the internals of specific models (e.g., random forests or neural networks), or depend on each sample’s neighborhood, which makes them hard to be generalized for more complex models and inefficient for larger datasets.

To overcome these limitations, we introduce a novel CF generation method called ReLACE that takes a different
Counterfactual Explanations for Machine Learning. A naïve approach to counterfactual explanation is called NEAREST-CF. Instead of generating a synthetic CF for a given input sample, this approach selects the nearest CF data point from the training set. More sophisticated counterfactual explanation methods can be broadly classified into model-specific and model-agnostic; as the names suggest, the former are tailored for a particular ML model (e.g., random forest), whereas the latter are able to generate explanations for any model.

Model-specific. One of the first counterfactual explanation method FEATWTEAK is proposed by (Tolomei et al. 2017), which is specifically designed for random forests, and exploits the internal structure of the learned trees to generate synthetic counterfactual instances. To accommodate for non-differentiable models, such as tree ensembles, FOCUS (Lucic et al. 2019) frames the problem of finding counterfactual explanations as an optimization task and uses probabilistic model approximations in the optimization framework. Meanwhile, the rise of deep learning has given way to more complex and opaque neural networks (NNs). In this regard, DEEPFOOL (Moosavi-Dezfooli, Fawzi, and Frossard 2016) – which was originally designed for crafting adversarial examples to undermine the robustness of NNs – has proven effective also as a CF generation method. However, CFs obtained from adversarial techniques often require changing almost all the features of the original instances, making them unrealistic to implement. Thus, (Le, Wang, and Lee 2020) propose GRACE: a novel technique that explains NN model’s predictions using sparser CFs, which therefore are suitable also for high-dimensional datasets.

Model-agnostic. (Guidotti et al. 2018) introduce LORE, a model-agnostic explainer. LORE first trains a local interpretable predictor on a synthetic sample’s neighborhood generated by a genetic algorithm. Then it derives from the logic of the local interpretable predictor a meaningful explanation consisting of: a decision rule, which explains the reasons of the decision; and a set of counterfactual rules, suggesting the changes to make to the instance’s features. More recently, (Karimi et al. 2020a) propose another model-agnostic CF generator called MACE, built on standard theory and tools from formal verification. In particular, MACE introduces a novel algorithm that solves a sequence of satisfiability problems, where both the distance function (objective) and predictive model (constraints) are represented as logic formulae.

Similarly to LORE and MACE, our RELACE method is also model-agnostic and aims to be as general and flexible as possible. Differently from them, though, RELACE is much more efficient to generate optimal CFs. Indeed, RELACE requires to train a DRL agent that makes use only of the input/output nature of the target predictive model to explain, regardless of its internal complexity. On the one hand, our method better scales to high-dimensional, large datasets than LORE: the genetic algorithm used by LORE to build each synthetic sample’s neighborhood may be unfeasible for large feature spaces. Plus, LORE also requires to train a locally-interpretable decision tree that is tight to each generated neighborhood, and therefore may be prone to overfitting. On
the other hand, RELACE can seamlessly handle more complex models (e.g., deeper NNs) than MACE, which needs to construct a first-order logic characteristic formula from the predictive model and test for its satisfiability. This may be intractable when the formula (i.e., the model to explain) is too big. Finally, as opposed to RELACE, both LORE and MACE do not consider nor control over the sparsity of the generated CFs; moreover, LORE does not even take into account their actionability.

**Reinforcement Learning with Hybrid Action Space.** Many problems contain the action space with a set of discrete actions and corresponding continuous action-parameter, which is very challenging for RL. The straightforward RL methods either discretize the continuous action space into a large discrete set (Sherstov and Stone 2005), or convert a discrete action into a continuous action method (Hausknecht et al. 2016). But they significantly increase the problem complexity. Recently, Masson et al. (Masson, Kanchod, and Komodakis 2016) propose a learning framework Q-PAMDP that alternatively learns the discrete action selection via Q-learning and employs policy search to get continuous action-parameters. Also, Wei et al. (Wei, Wicke, and Luke 2018) propose a hierarchical approach to deal with the parameterized action space, where the parameter policy is conditioned on the discrete policy.

**Explainable Reinforcement Learning.** As with traditional ML, the RL community has also been asked for post-hoc explanation methods that extract explainable policies from trained agents. Prior work has applied program synthesis to extract a high-level program from a DRL policy to retain the RL policy’s performance while rendering the policy’s decision logic interpretable in the program (Verma et al. 2018 2019 Zhu et al. 2019). As opposed to these methods, our approach does not need to know a suitable context-free grammar to specify a search space a priori. Decision trees were previously used by various post-hoc explanation algorithms (Frosst and Hinton 2017 Bastani, Pu, and Solar-Lezama 2018 Vasc et al. 2019) to visualize a DRL policy’s decision rules succinctly. These algorithms in general distill the decision knowledge of an RL agent, guided by the agent’s Q-value function, into a discrete decision tree or a soft decision tree (Frosst and Hinton 2017) following the DAgGER imitation learning technique (Ross, Gordon, and Bagnell 2011). Unlike our approach, these methods do not consider RL agents in parameterized action spaces with both discrete actions and continuous parameters.

### 3 Problem Definition

Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a feature space and $\mathcal{Y}$ a label space, where $\mathcal{Y} \in \{1, \ldots, K\}$ or $\mathcal{Y} \subseteq \mathbb{R}$. Suppose there exists a predictive model $h_\omega : \mathcal{X} \mapsto \mathcal{Y}$, parametrized by $\omega$, which accurately maps any input feature vector $x = (x_1, \ldots, x_n) \in \mathcal{X}$ to its label $h_\omega(x) = y \in \mathcal{Y}$.

The idea of counterfactual explanations is to reveal the rationale behind predictions made by $h_\omega$ on individual inputs $x$ by means of counterfactual examples (CFs). More specifically, for an instance $x$, a CF $\tilde{x}$ $\neq x$ according to $h_\omega$ is found by perturbing a subset of the features of $x$, chosen from the set $\mathcal{F} \subseteq \{1, \ldots, n\}$. The ultimate goal of such perturbation may vary depending on whether $h_\omega$ is a classifier or a regressor. In the former case, the objective would be to transform $x$ into $\tilde{x}$ so as to change the original predicted class into another class, such that $h_\omega(\tilde{x}) \neq h_\omega(x)$ (Wachter, Mittelstadt, and Russell 2017).

In the case $h_\omega$ is a regressor, instead, the goal is trickier: one possible approach to specifying the validity of a counterfactual example $\tilde{x}$ is to set a threshold $\delta \in \mathbb{R} \setminus \{0\}$ and let $| h_\omega(\tilde{x}) - h_\omega(x) | \geq \delta$.

Either way, as long as the CF classification or regression goal is met, several CFs can be generally found for a given input $x$. Amongst all the possible CFs, we search for the optimal $\tilde{x}$ as the “closest” one to $x$. This is to favor CFs that require the minimal perturbation of the original input.

More formally, let $g_\theta : \mathcal{X} \mapsto \mathcal{X}$ be a counterfactual generator, parametrized by $\theta$, that takes as input $x$ and produces as output a counterfactual example $\tilde{x} = g_\theta(x)$. For a given sample $D$ of i.i.d. observations drawn from a probability distribution, i.e., $D \sim P_{data}(\mathcal{X})$, we can measure the cost of generating CFs with $g_\theta$ for all the instances in $D$, using the following loss function:

$$L_{CF}(g_\theta; D, h_\omega) = \frac{1}{|D|} \sum_{x \in D} \ell_{pred}(x, g_\theta(x); h_\omega) + \lambda \ell_{dist}(x, g_\theta(x)).$$

(1)

The first component ($\ell_{pred}$) penalizes when the CF prediction goal is not satisfied. Let $S \subseteq \mathcal{X}$ be the set of inputs which do not meet the CF prediction goal, e.g., in the case of classification, $S = \{x \in \mathcal{X} \mid h_\omega(x) = h_\omega(g_\theta(x))\}$. Once the set $S$ is defined in terms of the desired CF prediction goal, we can compute $\ell_{pred}$ as follows:

$$\ell_{pred} = \ell_{pred}(x, g_\theta(x); h_\omega) = \mathbb{I}_S(x),$$

(2)

where $\mathbb{I}_S(x)$ is the indicator function.

The second component $\ell_{dist} : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}_{>0}$ is any arbitrary distance function that discourages $\tilde{x}$ to be too far away from $x$. For example, $\ell_{dist} = \ell_{dist}(x, g_\theta(x)) = ||x - g_\theta(x)||_p$, where $|| \cdot ||_p$ is the $L^p$-norm. In this work, without loss of generality, we set $\ell_{dist}$ equal to $L^1$-norm (Guidotti et al. 2018).

In addition, $\lambda$ serves as a scaling factor to trade off between $\ell_{dist}$ and $\ell_{pred}$. Notice, though, that not every input feature can always be modified to generate a CF (e.g., the “age” of an individual cannot be changed). Therefore, we must restrict $\mathcal{F}$ to the set of actionable features only. Plus, the total number of perturbed features must also be limited, i.e., $|\mathcal{F}| \leq c$ for some value $1 \leq c \leq |\mathcal{F}|$.

Eventually, we can find the optimal CF generator $g^* = g_{\theta^*}$ as the one whose parameters $\theta^*$ minimize Equation 1 i.e., by solving the following constrained objective:

$$\theta^* = \arg \min_\theta \left\{ L_{CF}(g_\theta; D, h_\omega) \right\}$$

subject to: $|\mathcal{F}| \leq c$.

(3)

This in turn allows us to generate the optimal CF $\tilde{x}^*$ for any $x$, as $\tilde{x}^* = g^*(x)$. Finally, the resulting optimal counterfactual explanation can be simply computed as $e_x = \tilde{x}^* - x$ (Tolomei and Silvestri 2021).
4 Proposed Framework: RELACE

In this work, we propose to find the optimal CF generator for an arbitrary predictive model—i.e., to solve the constrained optimization problem defined in Equation 3—via deep reinforcement learning (DRL).

4.1 Markov Decision Process Formulation

We consider the problem of computing the optimal counterfactual example \( \hat{x}^* \) from \( x \in \mathcal{D} \)—i.e., the optimal CF generator \( g^* \) defined in Equation 3—as a sequential decision-making task. More precisely, we refer to the standard reinforcement learning setting, where at each time step an agent: (i) takes an action (i.e., selects a feature of the original sample \( x \)) to tweak and the magnitude of such change and (ii) receives an observation (i.e., the prediction output by \( h_\omega \) on the input just modified according to the action taken before) along with a scalar reward from the environment. The process continues until the agent eventually meets the specified CF prediction goal and the optimal \( \hat{x}^* \) is found.

We formulate this process as a standard Markov Decision Process (MDP) \( \mathcal{M} = \{ \mathcal{S}, \mathcal{A}, T, p_0, r, \gamma \} \). In the following, we describe each component of this framework, separately.

**States (S)** At each time step \( t \), the agent’s state is \( s_t = s_t \), where \( s_t = (x_t, f_t) \in \mathcal{S} \) represents the current modified sample (\( x_t \)) along with the set of features changed so far (\( f_t \)). More specifically, \( f_t \in \{0,1\}^{2|\mathcal{F}|} \) is an \(|\mathcal{F}|\)-dimensional binary indicator vector, where \( f_t[k] = 1 \) iff actionable feature \( k \) has been modified in one of the actions taken by the agent before time \( t \). Initially, when \( t = 0 \), \( x_0 = x \) and \( f_0 = 0^{2|\mathcal{F}|} \).

If the prediction goal is met, e.g., \( h_\omega(x_t) \neq h_\omega(x) \), the agent reaches the end of the episode and the process terminates returning \( \hat{x}^* = x_t \) as the CF for \( x \). Otherwise, the agent must select an action \( A_t = a_t \) so as to: (i) pick a feature to change amongst those which have not been modified yet and (ii) decide the magnitude of that change.

**Discrete-Continuous Hybrid Actions (A)** To mimic the two-step behavior discussed above, differently from completely discrete or continuous actions, we consider a discrete-continuous hybrid action space, according to the two-tier hierarchical structure detailed below.

For an arbitrary step \( t \), we maintain the set of feature identifiers that the agent is allowed to modify \( F_t \); initially, when \( t = 0 \), \( F_0 = \mathcal{F} \) as the agent can pick any of the actionable features to change. Then, at each time step \( t > 0 \), the agent first chooses a high level action \( k_t \) from the discrete set \( \mathcal{F}_t \subset \mathcal{F} = \mathcal{F} \setminus \bigcup_{j=0}^{t-1} k_j \). This is to allow each feature to be selected at most in one action. Upon choosing \( k_t \in \mathcal{F}_t \), the agent must further select a low level parameter \( v_{k_t} \in \mathbb{R} \), which specifies the magnitude of the change to feature \( k_t \). Overall, \( a_t = (k_t, v_{k_t}) \), and we define our discrete-continuous hybrid action space as follows:

\[
A_t = \{(k_t, v_{k_t}) \mid k_t \in \mathcal{F}_t, v_{k_t} \in \mathbb{R} \}.
\]

**Transition Function (T)** Let \( a_t = (k_t, v_{k_t}) \in A_t \) be the generic action that the agent can take at time \( t \). The action \( a_t \) deterministically moves the agent from state \( s_t \) to state \( s_{t+1} \), by operating on \( x_t \) and \( f_t \) as follows:

\[
x_{t+1}[k_t] = x_t[k_t] + v_{k_t}; \quad f_{t+1}[k_t] = 1.
\]

This corresponds to increasing the value of feature \( k_t \) by the magnitude \( v_{k_t} \), and updating the binary indicator vector \( f_t \) accordingly.

**Reward (r)** The reward is defined in terms of the objective function defined in Equation 3 and has the following form:

\[
r(s_t, a_t) = \left\{ \begin{array}{ll}
1 - \lambda(\ell_{\text{dist}}^t - \ell_{\text{dist}}^{t-1}), & \text{if } h_\omega(x_t) \neq h_\omega(x) \\
-\lambda(\ell_{\text{dist}}^t - \ell_{\text{dist}}^{t-1}), & \text{otherwise}
\end{array} \right.
\]

where \( \ell_{\text{dist}}^t = \ell_{\text{dist}}(x, x_t) \).

In other words, we make a trade-off between achieving the CF prediction goal and the distance of the counterfactual from the original input sample \( x \).

**Policy (πθ)** We define a parametrized policy \( \pi_\theta \) to maximize the expected reward in the MDP problem. Our ultimate goal, however, is to find an optimal policy \( \pi^* = \pi_\theta^* \) that minimizes Equation 3. The equivalency of these two formulations is guaranteed by the following equation:

\[
\theta^* = \arg \max_\theta \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \sum_{t=1}^T r(s_t, \pi_\theta(s_t)).
\]

Here, \( T(x) \) illustrates the terminal step for sample \( x \). The details of the proof can be found in Appendix 1.

4.2 Policy Optimization

We use the P-DQN framework (Xiong et al. 2018) to find the optimal policy \( \pi^* \). As mentioned before, at each time step the agent takes a hybrid action \( a_t \in \mathcal{A}_t \) to perturb the currently modified \( x_t \), obtained from the original input \( x \).

In Q-learning, one aims at finding the optimal Q-value function, representing the expected discounted reward for taking action \( a_t \) at a given state \( s_t \). Inside the Q-value function, the continuous parameter \( v_{k_t} \) is associated with the discrete action \( k_t \), which means \( v_{k_t} \) is the optimal action given state \( s_t \) and \( k_t \): \( v_{k_t} = \arg \max_{v_{k_t}} Q(s_{t+1}, k_t, v_{k_t}) \). We therefore cast this as a function \( v_{k_t}(s_t) \). Thus, the Bellman equation can be written as:

\[
Q(s_t, k_t, v_{k_t}) = \mathbb{E}_{(i_t, s_{t+1}) \sim P_{r, s_{t+1}}}(r_t + \gamma \max_{k_{t+1}} Q(s_{t+1}, k_{t+1}, v_{k_{t+1}})|s_t, a_t = (k_t, v_{k_t}))
\]

Similar to DQN, a deep neural network (DNN) \( \mathcal{Q}_\theta^1(s_t, k_t, v_{k_t}) \) is used to approximate the Q-value function, and we fit \( v_{k_t} \) with another deterministic policy network \( \mathcal{Q}_\theta^2(s_t) \), where \( \theta_1 \) and \( \theta_2 \) are the parameters of the two DNNs. To find the optimal \( \theta_1^*, \theta_2^* \), we minimize the following loss functions via stochastic gradient methods:

\[
\mathcal{L}_{\mathcal{Q}}(\theta_1) = |\mathcal{Q}_\theta^1(s_t, k_t, v_{k_t}) - y_t|^2
\]

\[
\mathcal{L}_{\mathcal{Q}}(\theta_2) = - \sum_{k_t \in \mathcal{F}} \mathbb{E}_{s_t \sim \mathcal{D}} Q(s_t, k_t, v_{k_t}^\theta(s_t)),
\]

\[
\mathcal{L}_{\mathcal{Q}}(\theta_1^*, \theta_2^*) = \min_{\theta_1^*, \theta_2^*} \mathcal{T}(\theta_1^*, \theta_2^*)
\]

\[
\mathcal{T}(\theta_1^*, \theta_2^*) = \mathbb{E}_{(s_t, k_t, v_{k_t}) \sim \mathcal{D}} [\mathcal{Q}_\theta^1(s_t, k_t, v_{k_t}) - y_t]^2
\]
where the $y_t$ is the n-step target (Sutton and Barto 1998). Theoretically, as the error diminishes, the $Q$ network converges to the optimal $Q$-value function and the policy network outputs the optimal continuous action. However, this method is unstable in practice. This is because DQN only sample transition pairs uniformly from the replay buffer. Therefore, we adopt prioritized experience replay (Schaul et al. 2015) to effectively learn from pairs with high expected learning value. As a measurement for learning potential, prioritized experience replay samples transition pairs with probability $p_j$ based on their TD error:

$$p_j \propto R_j + \gamma Q_{\text{target}}(s_j, Q(s_j, a_j)) - Q(s_{j-1}, a_{j-1})$$

where $\beta$ is a parameter to determine how much prioritization is used. During the training procedure, the transition pairs are stored into the replay buffer with their priority, and a new data structure “sum tree” is employed to efficiently update the priority and sample pairs from the replay buffer.

### 4.3 Global vs. Local Policy

So far, we consider one single agent as responsible for generating the CFs for all the instances of a given dataset, and hereinafter we refer to it as RELACE-GLOBAL. This allows us to learn a generalized policy that is able to produce counterfactuals, which are not tailored to a specific input sample. Still, in some cases, the CF generation process should in fact capture the peculiarities of each individual original instance. To accommodate such a need, we introduce a variant to our proposed method, called RELACE-LOCAL.

Differently from RELACE-GLOBAL, RELACE-LOCAL does not leverage the whole training instances; instead, it trains a dedicated agent for crafting the optimal CF of a single target example. To achieve that, RELACE-LOCAL first needs to randomly generate synthetic training data points around the target example, using an approach similar to (Guidotti et al. 2018). More specifically, it uniformly samples data points whose Euclidean distance from the target example is at most one. To speed up the training procedure of the dedicated agent, RELACE-LOCAL may initialize its policy with one learned by a pretrained RELACE-GLOBAL agent. Notice that this approach, known as transfer learning, is very common in several domains, where highly complex models pretrained on huge datasets are used as an initialization or a fixed feature extractor for the downstream task of interest (Zhuang et al. 2021).

### 5 Policy Explanations

The complex network structure of a DRL policy learned for CF generation poses a challenge for understanding and reasoning about the decision logic of the agent. To explain the decision process of a learned policy, we distill knowledge from the policy to a much smaller decision tree (DT). The DT model approximates the policy’s operation and resembles its internal logic. The model’s tree-based logic structure is highly interpretable because it reveals simple but comprehensive decision rules made by the DRL policy.

Extracting a decision tree to interpret a complicated neural network is an established practice, often conducted by a teacher-student training process (Bastani, Pu, and Solar-Lezama 2018; Frosst and Hinton 2017; Zhang et al. 2019). However, we cannot simply adopt this strategy because the parameterized action space of our CF generation policy consists of both discrete actions and continuous parameters. Our algorithm addresses this challenge and consists in the following key steps.

#### Trajectories Collection

We sample trajectories of state-action pairs generated by a teacher DRL policy to train a student DT. The student model may encounter states that are not possible under the teacher’s induced state distribution due to imperfect approximation. Similar to previous work (Ross, Gordon, and Bagnell 2011), our solution is data augmentation. We iteratively sample a set of additional states by simulating the current DT model, and add the teacher’s decision on these samples to the training dataset.

#### Decision Tree Training

At each training step of a student DT, following (Bastani, Pu, and Solar-Lezama 2018), we resample state-action pairs from the current (augmented) training dataset with probability proportional to state importance:

$$p((s, a)) \propto (Q^*(s, \pi^*(s)) - \min_{a' \in A} Q^*(s, a')) |\mathcal{D}((s, a))|$$

where $\pi^*$ is the DRL teacher policy and $A$ is the policy’s parameterized action space. An action $a \in A$ includes both a discrete action and its continuous parameter. A state-action pair has a higher priority to be sampled if the action on the state significantly outweighs other actions, deemed by the Q-value function of the teacher policy $\pi^*$.

We train a student decision tree in two steps to approximate discrete actions and their continuous parameters of a teacher policy, separately. Firstly, we learn a DT to predict a DRL policy’s discrete actions, where each leaf node denotes a discrete action prediction. Once the DT’s structure is learned, we apply logistic regression to learn a function at each leaf node that takes as input a state and predicts the continuous parameter of the state’s discrete action. More details can be found in Appendix 2. Our experiment results demonstrate that a learned decision tree by our algorithm can produce transparent policies with small performance degradation within 5% of the original teacher policy’s result.

### 6 Experiments

In this section, we describe the experiments we conduct to validate our CF generation method. To demonstrate the effectiveness, efficiency, and flexibility of RELACE, we assess the quality of explanations extracted from several models learned either for classification or regression tasks.

#### Datasets

We test with five public datasets: Breast Cancer, Diabetes, Sonar, Wave, and Boston Housing. The first four are associated with classification tasks, whereas the last one with regression. A complete overview of the main properties of these data collections is available in Appendix 3.

---

1Notice that all our input samples are normalized unit vectors.

2We note that directly learning such tree models is infeasible as a tree structure cannot be updated by gradient-based optimization.
Known as Predictive Models. Each dataset is split into 70% training, 20% validation, and 10% test portions. Thus, for each task and the associated dataset, we train the suitable set of predictive models chosen amongst the following: Random Forest (RF), Adaptive Boosting (ADABOOST), Gradient Boosting (XGBOOST), Multi-Layer Perceptron (MLP), and Multi-Layer Perceptron for Regression (MLP-REG). Notice that some combinations do not apply, e.g., MLP-REG is only trained on Boston Housing. Eventually, we consider only the best performing model(s) for each task/dataset pair.

**Baselines.** We compare RELACE with all the baselines described in NEAREST-CF is considered the naive approach. Model-specific methods are FEATTWEAK and FOCUS (tree-ensemble-specific), and DeepFOOL and GRACE (NN-specific). Finally, LORE and MACE are model-agnostic approaches.

**Evaluation Metrics.** We generate a separate CF for each input sample in the test set of every dataset above, and evaluate their quality according to the following four metrics: Validity, Proximity, Sparsity, and Generation Time. Validity measures the ratio of CFs that actually meet the prediction goal to the total number of CFs generated. Proximity computes the distance of a counterfactual from the original input sample. In this work, we use $L^2$-norm to measure proximity. Sparsity is equivalent to the $L^0$-norm between a CF and the corresponding original input sample. It encourages for CFs that ask to change a small number of features, as they likely lead to more human-interpretable explanations. Generation Time computes the time required to generate CFs. All the metrics above are averaged across all the test input samples.

### 6.1 Results

In Figure 2 we plot the number of perturbed features (i.e., sparsity) versus the validity of obtained counterfactuals when using different CF generation methods. We compare the set of suitable CF generation techniques for each combination of dataset and model to explain. More specifically, we can always apply RELACE (in its two variants) and LORE, regardless of the target model. The other model-agnostic CF generator, i.e., MACE, unfortunately, turned out to be highly time-consuming when used to explain NN models like MLP. Of course, results for model-specific methods only appear when applied to the model to be explained (e.g., GRACE is NN-specific, and therefore is shown only in combination with MLP). Results show that both RELACE-GLOBAL and RELACE-LOCAL outperform sensitively all the baselines in terms of validity. More importantly, we obtain these results only when few features are modified. That is, RELACE generates sparser, thereby more interpretable, counterfactuals, which is the essential requirement for any CF generation technique. We can observe from the results that RELACE works well even when we set a very restrictive threshold on the sparsity. Indeed, if we removed such a cap on the number of features allowed to change, other methods like LORE could eventually match the performance of our method or even reach higher validity scores. We expect this behavior.

In fact, not controlling for sparsity will make every CF generation method behave similarly. Allowing perturbation on every single feature likely results in many valid CFs, but perhaps those will not be realizable and thus useless. This behavior is what we observe when we test with DEEPROOL or the naive baseline NEAREST-CF, which by design tend to generate valid CFs only if a significant fraction of the input features get modified. Due to this behavior, both DEEPROOL and NEAREST-CF are not visible in the plots.

In Table 1, we show the average proximity and generation time obtained by all the model-agnostic CF generation methods considered in this work. We observe that our method takes up to 42% less time than other approaches to produce valid counterfactuals. Moreover, those counterfactuals are generally closer to the original input sample. Similar trends also show up when comparing RELACE with other baselines. The complete results are reported in Appendix 4.

### 6.2 Ablation Study

To study how the parameters take effect on RELACE’s performance, we conduct the analysis on the Sonar dataset with MLP as the classifier and limit the maximal sparsity to 5 features. Typically, we first investigate how the scaling factor $\lambda$ affects the performance. We pick $\lambda$ from 10, 1, 0.1, 0.01, 0.001. As shown in Figure 2, one can find that $\lambda$ controls the balance of validity and sparsity. Higher $\lambda$ forces the agent to prefer lower sparsity and lower cost over validity, and vice versa. Then, we investigate the robustness over different NN architectures, whose results are shown in Appendix 5. We found that GRACE is more sensitive to the NN size since it is based on the gradient of the output function, which is associated with the complexity of the NN’s structure. However, RELACE and LORE are more robust toward different NNs, since these two methods treat the target model as a black-box regardless of its complexity. Finally, we show how pretrained RELACE-GLOBAL improves the performance of RELACE-LOCAL. More specifically, we trained RELACE-LOCAL as described in Section 4.3 and compare it with another agent learned from scratch. We observe that initializing RELACE-LOCAL with pretrained RELACE-GLOBAL significantly reduces its CF generation time.

### 7 Policy Explanations for Healthcare

We extend our experiments on an ADABOOST classifier trained on the Diabetes dataset. We aim to extract transparent decision rules from the learned agent’s policy (i.e., neural network). The dataset contains 768 instances and each one has a label, which is either 1 (positive for diabetes) or 0 (negative for diabetes). As our task is to find the optimal counterfactual explanation for patients with diabetes, we focus on converting positive samples into negative ones. Note that we only perturb actionable features. Specifically, features like “Pregnancy”, “Pedigree” and “Age” cannot be modified, since they are properties that are historical.

We first train a specific agent via RELACE-GLOBAL to generate CFs for positive samples (patients with diabetes). Its validity can reach 0.9 with sparsity 1.86. Then, we distill

---

4 Some work consider the complementary metric, which is known as Fidelity and is equal to (1-Validity).
Figure 1: Sparsity vs. Validity on every dataset/model combination.

| Metric | Dataset [Models] | CF Methods |
|--------|------------------|------------|
|        | Breast Cancer [RF,MLP] | LORE [4.63, 5.63] | MACE [4.47, N/A] | RELACE-GLOBAL [4.46, 5.92] | RELACE-LOCAL [4.49, 5.87] |
|        | Diabetes [ADABOOST] | [4.76] | [N/A] | [4.41] | [4.50] |
|        | Sonar [MLP] | [7.30] | [N/A] | [7.32] | [7.66] |
|        | Wave [XGBOOST, MLP] | [6.60, 6.41] | [N/A, N/A] | [5.93, 6.38] | [6.02, 6.50] |
|        | Boston Housing [MLP-REG] | [N/A] | [N/A] | [5.10] | [5.36] |

Table 1: Comparison of Proximity and Avg. Generation Time for model-agnostic CF methods.

Figure 2: The impact of $\lambda$ on Sparsity vs. Validity.

the agent to a hybrid decision tree (see Section 5) to generate CFs. The validity degradation of the tree model is within 5% of the original neural model.

We present the top-4 layers of the decision tree in Appendix 6. The decision variable of each node contains Glucose (G), Blood-pressure (BP), Skin (S), BMI (BM), Pedigree (P), Age (A), and their indicator variables: $I_G$, $I_{BP}$, $I_S$, $I_{BM}$, $I_P$ and $I_A$, which denote if the corresponding variable has been used. Also, we show the frequency of final decisions of each tree path on the fourth layer.

We explain the reasons behind the superior performance of RELACE-GLOBAL using this case study in two directions. (i) Capturing existing knowledge: similar to (Tolomei et al. 2017), modifying the four main features $G$, $BM$, $S$, $BP$ can highly likely change a positive sample into a negative one. As depicted in the decision tree, these four features play a significant role in the decision nodes. (ii) Discovering new rules: the decision tree takes multiple variables into consideration, rather than relying on a single variable like (Tolomei et al. 2017). Moreover, although only focusing on actionable features, the tree model also illustrates how non-actionable features affect a final decision.

8 Conclusion

In this work, we presented RELACE, i.e., the first method for generating model-agnostic counterfactual examples based on deep reinforcement learning. We implemented two variants of it: RELACE-GLOBAL and RELACE-LOCAL. Extensive experiments run on five datasets demonstrated that RELACE outperforms sensitively all the baselines in all the metrics. It is scalable with respect to the number of features and the size of the dataset, and can explain any black-box model, regardless of its internal complexity and prediction task (classification or regression). Finally, we showed that RELACE is itself transparent by distilling the agent’s policy into an interpretable decision tree.
Acknowledgments. This research was supported by the Italian Ministry of Education, University and Research (MIUR) under the grant “Dipartimenti di eccellenza 2018–2022” of the Department of Computer Science and the Department of Computer Engineering at Sapienza University of Rome. Partially supported by the ERC Advanced Grant 788893 AMDROMA “Algorithmic and Mechanism Design Research in Online Markets”, the EC H2020RIIA project “SoBigData++” (871042), and the MIUR PRIN project ALGADIMAR “Algorithms, Games, and Digital Markets”. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References
Ahmad, M. A.; Eckert, C.; and Teredesai, A. 2018. Interpretable Machine Learning in Healthcare. In Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics, BCB ’18, volume 899 of CEUR Workshop Proceedings.

Bastani, O.; Pu, Y.; and Solar-Lezama, A. 2018. Verifiable reinforcement learning via policy extraction. arXiv preprint arXiv:1805.08328

Bracke, P.; Datta, A.; Jung, C.; and Sen, S. 2019. Machine learning explainability in finance: an application to default risk analysis.

Dandl, S.; Molnar, C.; Binder, M.; and Bischl, B. 2020. Multi-Objective Counterfactual Explanations. In Bück, T.; Preuss, M.; Deutz, A. H.; Wang, H.; Doerr, C.; Emmrich, M. T. M.; and Trautmann, H., eds., Parallel Problem Solving from Nature - PPSN XVI - 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part I, volume 12269 of Lecture Notes in Computer Science, 448–469. Springer.

EU. 2016. Regulation (EU) 2016/679 of the European Parliament (GDPR). Official Journal of the European Union, L119: 1–88.

Frosst, N.; and Hinton, G. E. 2017. Distilling a Neural Network Into a Soft Decision Tree. In Besold, T. R.; and Kutz, O., eds., Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017), Bari, Italy, November 16th and 17th, 2017, volume 2071 of CEUR Workshop Proceedings. CEUR-WS.org.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Pedreschi, D.; Turini, F.; and Giannotti, F. 2018. Local Rule-Based Explanations of Black Box Decision Systems. arXiv preprint arXiv:1805.10820

Hausknecht, M.; Mupparaju, P.; Subramanian, S.; Kalyanakrishnan, S.; and Stone, P. 2016. Half field offense: An environment for multiagent learning and ad hoc teamwork. In AAMAS Adaptive Learning Agents (ALA) Workshop. sn.

Kanamori, K.; Takagi, T.; Kobayashi, K.; and Arimura, H. 2020. DACE: Distribution-Aware Counterfactual Explanations by Mixed-Integer Linear Optimization. In Bessiere, C., ed., Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, 2855–2862. ijcai.org.

Karimi, A.; Barthe, G.; Balle, B.; and Valera, I. 2020a. Model-Agnostic Counterfactual Explanations for Consequential Decisions. In Chiappa, S.; and Calandra, R., eds., Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

Laugel, T.; Lesot, M.; Marsala, C.; Renard, X.; and Deyniecki, M. 2018. Comparison-Based Inverse Classification for Interpretability in Machine Learning. In Medina, J.; Ojeda-Aciego, M.; Galdeano, J. L. V.; Pelta, D. A.; Cabrera, I. P.; Boucheon-Meunier, B.; and Yager, R. R., eds., Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations - 17th International Conference, IPMU 2018, Cádiz, Spain, June 11-15, 2018, Proceedings, Part I, volume 853 of Communications in Computer and Information Science, 100–111. Springer.

Le, T.; Wang, S.; and Lee, D. 2020. GRACE: Generating Concise and Informative Contrastive Sample to Explain Neural Network Model’s Prediction. In Gupta, R.; Liu, Y.; Tang, J.; and Prakash, B. A., eds., KDD ’20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020, 238–248. ACM.

Looveren, A. V.; and Klaize, J. 2019. Interpretable Counterfactual Explanations Guided by Prototypes. CoRR, abs/1907.02584.

Lucic, A.; Oosterhuis, H.; Haned, H.; and de Rijke, M. 2019. FOCUS: Flexible optimalizible counterfactual explanations for tree ensembles. arXiv preprint arXiv:1911.12199

Masson, W.; Ranchod, P.; and Konidaris, G. 2016. Reinforcement learning with parameterized actions. In Thirtieth AAAI Conference on Artificial Intelligence.

McGrath, R.; Costabello, L.; Van, C. L.; Sweeney, P.; K amiab, F.; Shen, Z.; and Lécué, F. 2018. Interpretable Credit Application Predictions With Counterfactual Explanations. CoRR, abs/1811.05245.

Moosavi-Dezfooli, S.; Fawzi, A.; and Frossard, P. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, 2574–2582. IEEE Computer Society.

Pawelczyk, M.; Broelemann, K.; and Kasneci, G. 2020. Learning Model-Agnostic Counterfactual Explanations for
Appendix 1
In this section, we show the equivalence between the traditional formulation of the optimal counterfactual generation problem (i.e., the minimization of the objective function as defined in Equation 3 in the main paper) and the maximization of the deep reinforcement learning agent’s reward:

\[
\theta^* = \arg \min_{\theta} \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \ell_{\text{pred}}(x, g_\theta(x); h_\omega) + \lambda \ell_{\text{dist}}(x, g_\theta(x))
\]

= \arg \max_{\theta} \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} -\ell_{\text{pred}}(x, g_\theta(x); h_\omega) - \lambda \ell_{\text{dist}}(x, g_\theta(x))

= \arg \max_{\theta} \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \left\{ 1 - \lambda(\ell_{\text{dist}} - \ell_{\text{dist}}^{t-1}), \text{if } h_\omega(x_i) \neq h_\omega(x) \right\}, \text{otherwise,}

where \( \ell_{\text{dist}} = \ell_{\text{dist}}(x, x_i) \).

= \arg \max_{\theta} \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \sum_{t=1}^{T(x)} r((x_t, f_t), \pi_\theta(s_t)).

= \arg \max_{\theta} \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \sum_{t=1}^{T(x)} r(s_t, \pi_\theta(s_t)).

Appendix 2
We adopt the teacher-student training strategy to train the decision tree. The maximum depth of the distilled decision tree is 8. Our training algorithm is shown in Algorithm 1

Algorithm 1: Extracting rules from agent with hybrid action
1: \( \mathcal{D} \leftarrow \text{initialize dataset with empty set} \)
2: \( \pi_0 \leftarrow \text{initialize the policy with agent’s policy } \pi^* \)
3: for \( i \leq N \) do
4: \( \text{Sample } B \text{ trajectories } \mathcal{D}_i \leftarrow \{(s, \pi^*(s)) \sim d^{\mathcal{X}^i-1}\} \)
5: \( \text{Update dataset } \mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i \)
6: \( \text{Resample from the dataset } \mathcal{D}' \leftarrow \{(s, k) \sim p(s, k)\}, \)
\quad where \( k \) is the discrete action.
7: \( \text{Train decision tree } \tilde{\pi}_i \)
8: end for
9: Best policy \( \tilde{\pi} \in \left\{ \tilde{\pi}_1, \ldots, \tilde{\pi}_N \right\} \)
10: for each leaf \( \omega \) in the \( \tilde{\pi} \) do
11: \( \text{Train a logistic regression } l_\omega \text{ to fit the continuous action } v_k \text{ with } s \text{ as the input.} \)
12: end for
13: return \( \tilde{\pi}, \{l_\omega : \omega \in \tilde{\pi}\} \quad \triangleright \text{return the decision tree } \tilde{\pi} \text{ for discrete action and a group of } \{l_\omega\} \text{ for continuous action.} \)

Appendix 3
Table 2 describes the main properties of the five public datasets used in our experiments. Moreover, in Table 3 we show additional information for each predictive model used in combination with those datasets.

Appendix 4
In this paper, we implement multiple experiments for one setting. In each experiment, we control the total number of perturbed features (i.e., maximal sparsity). In Table 4 we present the maximal sparsity that we used for all experiments. In Table 5 we report the complete set of results of all our experiments. Notice, though, that NEAREST-CF and DEEPFOOL may very often obtain very high validity scores. However, counterfactuals generated by those methods require changing almost every feature, and therefore they turn out to be useless. For this reason, we do not highlight any score obtained by NEAREST-CF nor DEEPFOOL in Table 5. Besides, the authors of MACE apply its counterfactual generation method to Random Forest and (simple) MLP. However, running MACE on MLPs with complex structures is really time-consuming, and therefore unfeasible. Thus, we did not test MACE in combination with our (complex) MLP models. Finally, we incorporate the training time of RELACE into the counterfactual generation time. We show the main algorithm for our proposed RELACE-GLOBAL counterfactual generation method, in Algorithm 2

Algorithm 2: Training RELACE-GLOBAL Agent
1: \( \theta_1 \leftarrow \text{initialize the deep Q-network } Q_{\theta_1} \)
2: \( \theta_2 \leftarrow \text{initialize the deterministic policy network } \psi_{\theta_2} \)
3: \( M \leftarrow \text{initialize the replay buffer} \)
4: \( i \leftarrow 1 \)
5: while \( i \leq \text{max_epochs} \) do
6: \( x \sim \mathcal{D} \quad \triangleright \text{sample a training instance } x \text{ from } \mathcal{D} \)
7: \( x_0 \sim x \)
8: \( s_0 = (x_0, f_0) \quad \triangleright \text{initial state} \)
9: \( t \leftarrow 0 \)
10: for \( t \leq T \) do
11: \( v_{k_t} \leftarrow v^{\theta_2}_{k_t}(s_t) \)
12: \( \triangleright \text{compute the continuous parameter} \)
13: \( a_t \leftarrow (k_t, v_{k_t}) \)
14: \( \triangleright \text{select the discrete action by } \varepsilon\text{-greedy} \)
15: \( r_t, s_{t+1} \leftarrow T(s_t, a_t) \)
16: \( \triangleright \text{the agent gets the reward and observes the next state} \)
17: \( p_t \leftarrow \text{compute the importance } p_t \)
18: \( M \leftarrow \{(s_{t}), \{a_t\}, \{r_t\}, \{s_{t+1}\}, \{p_t\}) \)
19: \( \triangleright \text{store transition into the replay buffer} \)
20: \( B \sim M \quad \triangleright \text{randomly sample batch } B \text{ from } M \)
21: \( \theta_1 \leftarrow \theta_1 - \eta_1 \nabla L_Q(\theta_1) \)
22: \( \theta_2 \leftarrow \theta_2 - \eta_2 \nabla L_\pi(\theta_2) \)
23: \( \triangleright \text{update both networks parameters via SGD} \)
24: \( t \leftarrow t + 1 \)
25: end for
26: \( i \leftarrow i + 1 \)
27: end while
28: return \( \theta_1, \theta_2 \) \( \triangleright \text{return the optimal networks parameters} \)

Appendix 5
In the ablation study, we conduct the analysis on the Sonar dataset with MLP as the classifier, and limit the maximal sparsity to 5 features. In Table 6 we present the impact of different MLP architectures on the validity and sparsity of counterfactuals generated by our method in comparison with two competitors: GRACE (NN-specific) and LORE (model-agnostic). We also present the gener-
| Dataset         | N. of Instances | N. of Features | Task          | URL                                                                 |
|-----------------|-----------------|----------------|---------------|----------------------------------------------------------------------|
| Breast Cancer   | 699             | 10 (numerical) | classification | /breast-cancer-wisconsin-%28original%29†                               |
| Diabetes        | 768             | 8 (numerical)  | classification | /uciml/pima-indians-diabetes-database‡                               |
| Sonar           | 208             | 60 (numerical) | classification | /Connectionist+Bench+(Sonar,+Mines+vs.+Rocks)†                        |
| Wave            | 5,000           | 21 (numerical) | classification | /waveform+database+generator+(version+1)†                              |
| Boston Housing  | 506             | 14 (mixed)     | regression     | /vikrishnan/boston-house-prices‡                                     |

Table 2: Main characteristics of the five public datasets used.

(†=https://archive.ics.uci.edu/ml/datasets/; ‡=https://www.kaggle.com/)

| Dataset [Model]          | Model Information |
|-------------------------|-------------------|
| Wave [MLP]              | {100, 200}        |
| Sonar [MLP]             | {256, 256}        |
| Breast Cancer [MLP]     | {64, 128}         |
| Diabetes [ADABOOST]     | {100 trees}       |
| Breast Cancer [RF]      | {100 trees}       |
| Wave [XGBOOST]          | {100 trees}       |
| Boston Housing [MLP-REG]| {50, 128}         |

Table 3: Model’s information for each setting.

| Dataset [Models]      | Maximal Sparsity |
|----------------------|------------------|
| Wave [MLP]           | {3, 5, 7}        |
| Sonar [MLP]          | {3, 5, 7, 9}     |
| Breast Cancer [MLP]  | {2, 3, 5}        |
| Diabetes [ADABOOST]  | {1, 2, 3, 4}     |
| Breast Cancer [RF]   | {2, 3, 5}        |
| Wave [XGBOOST]       | {3, 5, 7}        |
| Boston Housing [MLP-REG] | {3, 5}    |

Table 4: Maximal sparsity for each setting.

In Figure 3, we show the top-4 layers of the distilled decision tree. Each path from the root to the leaf is a transparent rule. Here, we present one complete rule in Figure 4. From these plots, we give a transparent explanation of how to find the optimal counterfactual example.

Appendix 6

In Figure 3, we show the top-4 layers of the distilled decision tree. Each path from the root to the leaf is a transparent rule. Here, we present one complete rule in Figure 4. From these plots, we give a transparent explanation of how to find the optimal counterfactual example.
Figure 3: Top-4 layers of the distilled decision tree.

Figure 4: Example of one transparent rule extracted from the RELACE agent’s policy.
| CF Method       | Dataset [Models]       | Validity | Sparsity | Proximity | Generation Time |
|-----------------|------------------------|----------|----------|-----------|-----------------|
|                 | Breast Cancer [RF, MLP] | [1, 1]   | [29.98, 29.52] | [18.05, 10.78] | [0.10, 0.08]    |
| Nearest-CF      | Diabetes [ADABOOST]    | [0]      | [Inf]    | [Inf]     | [Inf]           |
|                 | Sonar [MLP]            | [1]      | [59.76]  | [30.23]   | [0.12]          |
|                 | Wave [XGBOOST, MLP]    | [1, 1]   | [20.86, 19.83] | [14.03, 12.02] | [0.08, 0.08]    |
|                 | Boston Housing [MLP-REG] | [N/A]   | [N/A]    | [N/A]     | [N/A]           |
|                 |                        |          |          |           |                 |
| FeatTweak       | Breast Cancer [RF, MLP] | [0.70, N/A] | [2.42, N/A] | [4.20, N/A] | [2200, N/A]     |
|                 | Diabetes [ADABOOST]    | [N/A]    | [N/A]    | [N/A]     | [N/A]           |
|                 | Sonar [MLP]            | [N/A]    | [N/A]    | [N/A]     | [N/A]           |
|                 | Wave [XGBOOST, MLP]    | [N/A, N/A] | [N/A, N/A] | [N/A, N/A] | [N/A, N/A]      |
|                 | Boston Housing [MLP-REG] | [N/A]    | [N/A]    | [N/A]     | [N/A]           |
|                 |                        |          |          |           |                 |
| Focus           | Breast Cancer [RF, MLP] | [0.66, N/A] | [2.60, N/A] | [4.39, N/A] | [120, N/A]      |
|                 | Diabetes [ADABOOST]    | [N/A]    | [N/A]    | [N/A]     | [N/A]           |
|                 | Sonar [MLP]            | [N/A]    | [N/A]    | [N/A]     | [N/A]           |
|                 | Wave [XGBOOST, MLP]    | [N/A, 0.88] | [N/A, 20.30] | [N/A, 16.60] | [N/A, 0.02]     |
|                 | Boston Housing [MLP-REG] | [N/A]   | [N/A]    | [N/A]     | [N/A]           |
|                 |                        |          |          |           |                 |
| DeepFool        | Breast Cancer [RF, MLP] | [N/A, 0.58] | [N/A, 20.74] | [N/A, 5.09] | [N/A, 1.20]     |
|                 | Diabetes [ADABOOST]    | [N/A]    | [N/A]    | [N/A]     | [N/A]           |
|                 | Sonar [MLP]            | [N/A]    | [N/A]    | [N/A]     | [N/A]           |
|                 | Wave [XGBOOST, MLP]    | [0.68, 0.56] | [2.74, 3.19] | [6.60, 6.41] | [2000, 1800]    |
|                 | Boston Housing [MLP-REG] | [N/A]    | [N/A]    | [N/A]     | [N/A]           |
|                 |                        |          |          |           |                 |
| Grace           | Breast Cancer [RF, MLP] | [0.65, 0.58] | [2.05, 2.02] | [4.63, 5.63] | [2200, 2100]    |
|                 | Diabetes [ADABOOST]    | [0.52]   | [1.61]   | [4.76]    | [1900]          |
|                 | Sonar [MLP]            | [0.56]   | [3.35]   | [7.36]    | [2700]          |
|                 | Wave [XGBOOST, MLP]    | [0.68, 0.56] | [2.74, 3.19] | [6.60, 6.41] | [2000, 1800]    |
|                 | Boston Housing [MLP-REG] | [N/A]    | [N/A]    | [N/A]     | [N/A]           |
|                 |                        |          |          |           |                 |
| Lore            | Breast Cancer [RF, MLP] | [0.75, N/A] | [2.38, N/A] | [4.17, N/A] | [2200, N/A]     |
|                 | Diabetes [ADABOOST]    | [N/A]    | [N/A]    | [N/A]     | [N/A]           |
|                 | Sonar [MLP]            | [N/A]    | [N/A]    | [N/A]     | [N/A]           |
|                 | Wave [XGBOOST, MLP]    | [N/A, N/A] | [N/A, N/A] | [N/A, N/A] | [N/A, N/A]      |
|                 | Boston Housing [MLP-REG] | [N/A]    | [N/A]    | [N/A]     | [N/A]           |
|                 |                        |          |          |           |                 |
| MACE            | Breast Cancer [RF, MLP] | [0.70, 0.75] | [2.39, 2.14] | [4.46, 5.92] | [2100, 1200]    |
|                 | Diabetes [ADABOOST]    | [0.70]   | [1.50]   | [4.41]    | [2000]          |
|                 | Sonar [MLP]            | [0.80]   | [2.79]   | [7.32]    | [1400]          |
|                 | Wave [XGBOOST, MLP]    | [0.84, 0.83] | [2.62, 2.69] | [5.93, 6.38] | [1300, 1200]    |
|                 | Boston Housing [MLP-REG] | [0.74]  | [2.41]   | [5.10]    | [1300]          |
|                 |                        |          |          |           |                 |
| RelACE-Global   | Breast Cancer [RF, MLP] | [0.78, 0.84] | [2.57, 2.22] | [4.49, 5.87] | [1900, 1100]    |
|                 | Diabetes [ADABOOST]    | [0.76]   | [1.49]   | [4.50]    | [1800]          |
|                 | Sonar [MLP]            | [0.97]   | [3.04]   | [7.66]    | [1000]          |
|                 | Wave [XGBOOST, MLP]    | [0.88, 0.91] | [2.67, 2.71] | [6.02, 6.50] | [1100, 1340]    |
|                 | Boston Housing [MLP-REG] | [0.81]  | [2.57]   | [5.36]    | [1000]          |
|                 |                        |          |          |           |                 |
| RelACE-Local    | Breast Cancer [RF, MLP] | [0.78, 0.84] | [2.57, 2.22] | [4.49, 5.87] | [1900, 1100]    |
|                 | Diabetes [ADABOOST]    | [0.76]   | [1.49]   | [4.50]    | [1800]          |
|                 | Sonar [MLP]            | [0.97]   | [3.04]   | [7.66]    | [1000]          |
|                 | Wave [XGBOOST, MLP]    | [0.88, 0.91] | [2.67, 2.71] | [6.02, 6.50] | [1100, 1340]    |
|                 | Boston Housing [MLP-REG] | [0.81]  | [2.57]   | [5.36]    | [1000]          |

Table 5: Complete overview of the experimental results.

| MLP size | Predictive Accuracy | RELACE-Global | RELACE-Local | GRACE | LORE |
|----------|---------------------|---------------|--------------|-------|------|
| [256, 256] | 0.88               | 0.76 (2.95)   | 0.90 (3.19)  | 0.62 (3.32) | 0.60 (3.65) |
| [128, 128] | 0.85               | 0.80 (2.69)   | 0.90 (2.88)  | 0.73 (2.85) | 0.76 (2.31) |
| [64, 64]   | 0.79               | 0.90 (1.88)   | 0.95 (1.93)  | 0.86 (1.90) | 0.90 (2.32) |

Table 6: Impact of different MLP architectures over the Validity (Sparsity) of counterfactuals.

| Setting     | Validity (Sparsity) | Avg. Generation Time (secs.) |
|-------------|----------------------|-----------------------------|
| without pretrained | 0.80 (3.27)        | 1132                        |
| with pretrained   | 0.90 (3.19)         | 500                         |

Table 7: Performance of RELACE-LOCAL with/without pretrained RELACE-GLOBAL.