No D_{train}: Model-Agnostic Counterfactual Explanations Using Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Machine learning (ML) methods have experienced significant growth in the past decade, yet their practical application in high-impact real-world domains has been hindered by their opacity. When ML methods are responsible for making critical decisions, stakeholders often require insights into how to alter these decisions. Counterfactual explanations (CFEs) have emerged as a solution, offering interpretations of opaque ML models and providing a pathway to transition from one decision to another. However, *most existing CFE methods require access to the model's training dataset, few methods can handle multivariate time-series, and none of model-agnostic CFE methods can handle multivariate time-series without training datasets*. These limitations can be formidable in many scenarios. In this paper, we present NTD-CFE, a novel reinforcement-learning-based model-agnostic CFE method that generates CFEs *when training datasets are unavailable*. NTD-CFE is suitable for both static and multivariate time-series datasets with continuous and discrete features. Users have the flexibility to specify non-actionable, immutable, and preferred features, as well as causal constraints . We demonstrate the performance of NTD-CFE against four baselines on several datasets and find that, despite not having access to a training dataset, NTD-CFE finds CFEs that make significantly fewer and significantly smaller changes to the input time-series. These properties make CFEs more actionable, as the magnitude of change required to alter an outcome is vastly reduced.

1 Introduction

After receiving a negative decision—a denial of a loan, a poor performance review, or a rejection from a prestigious conference—a very natural question to ask is "What could I have done differently?" When the decision is made by a person, that question can be answered directly by the person. Indeed, peer reviews represent the reasons why a paper is accepted or rejected from a conference and, in the best case, give authors *actionable* feedback which may lead to an acceptance in the future (assuming the underlying decision algorithm remains unchanged). But when a decision is made or influenced by a black-box model, it can be much harder to provide insights. Telling a loan applicant they have a "poor credit score" does not tell them how they might approach getting approved at a later date.

Counterfactual Explanations (CFEs) (Wachter et al., 2017) were introduced to fill this gap. Given an input to a model, a CFE is perturbed version of the input which yields a prescribed output from the model. For instance, if Bob's mortgage application is rejected, a CFE might suggest that Bob make \$20,000 more per year, or alternatively, make \$10,000 more per year *and* purchase a house in a different neighborhood.

CFEs also provide a method for examining how "fair" a model is, a concern that has become paramount in many real-world applications (Mehrabi et al., 2021; Angwin et al., 2022; Osoba et al., 2017). For instance, a CFE that shows if Bob's name were Alice his loan application would have been accepted points to potential discrimination on the basis of a protected characteristic. Classic models such as logistic regression and low-depth decision trees often called *inherently interpretable* (Verma et al., 2020; Murphy, 2012; Breiman, 2017) as the relative influence of input features can be read off from learned coefficients. But relations between learned parameters and input features are much harder to discern with more complex models. Indeed, even a logistic regression model with pairwise interaction terms can be complicated to reason about: Perhaps a woman who purchases a home in one neighborhood is less likely to have her application accepted than a man, but an additional $1,000 a year in income increases her chance of acceptance significantly more than a man's chance.

There exist many CFE methods for static datasets (Mothilal et al., 2020; Samoilescu et al., 2021; Verma et al., 2022). However, CFE methods for multivariate time-series data are less common due to the challenges posed by high dimensionality (Ates et al., 2021; Theissler et al., 2022). Additionally, to the best of our knowledge, *all existing model-agnostic CFE methods for multivariate time-series require access to large collection of samples from the training distribution of the model being explained.* This requirement can be infeasible in real-world domains especially due to privacy and other concerns.

In this paper, we introduce No-Training-Dataset reinforcement-learning-based CFE (NTD-CFE), a reinforcement learning (RL) based CFE method designed for both static and multivariate time-series data containing both continuous and discrete attributes. Remarkably, NTD-CFE operates without training datasets or similar data samples and is model-agnostic so that it is compatible with any (even non-differentiable) predictive models. NTD-CFE also allows the user to specify which features they prefer to change, as well as how changing a particular feature may affect another feature, thus allowing the user to express both what counterfactuals are *feasible* for them and any causal relationships between those features. While NTD-CFE works for both static and time-series data, we focus on the harder setting of multivariate time-series data in this paper. We compare NTD-CFE to four state-of-the-art CFE methods on eight real-world multivariate time-series datasets. We find that NTD-CFE yields CFEs with significantly better proximity (the total magnitude of change proposed by the CFE) and sparsity (how many features the CFE proposes to alter) compared to the baselines.

The paper is structured as follows. We discuss related works and preliminaries in Section 2 and Section 3, respectively. Section 4 describes the proposed algorithm NTD-CFE. Qualitative examples and quantitative experiments with 8 real-world datasets and 5 predictive models are given in Section 5. Finally, we conclude in Section 6.

2 Related Works

Counterfactual explanations belong to a much broader category of methods often called *Explainable AI* (XAI). XAI techniques can be broadly classified into two buckets (Verma et al., 2020): (a) inherently interpretable machine learning techniques and (b) post-hoc explanatory techniques. The first category is primarily a restriction on the types of models that a practitioner can employ to model a phenomenon. However, models often held out as interpretable (e.g., linear regression and low-depth decision trees (Murphy, 2012; Breiman, 2017)) may not have the capacity to capture complex phenomena. In the latter category, practitioners attempt to "model the model" with subsequent techniques (Sun et al., 2020). These methods can be further subdivided into *global* and *local* methods (Ates et al., 2021; Islam et al., 2021). Global methods attempt to simulate the opaque, complex logic of the original model using interpretable methods. On the other hand, local methods aim to explain the rationale behind a specific prediction made for a specific input. For instance, *feature-based methods* such as SHAP(Lundberg & Lee, 2017), identify features that have the most significant influence on a prediction. On the other hand, *sample-based methods* attempt to identify relevant samples to clarify a prediction (Kim et al., 2016; Mothilal et al., 2020). CFEs are an example of a post-hoc, local, sample-based explanation technique.

CFEs were introduced by Wachter et al. (2017) to explore an optimization-based technique for differentiable predictive models. Building upon this foundation, DiCE (Mothilal et al., 2020) noted that there were potentially many different CFEs proximal to any particular input and whether one was "better" than another was really a matter for the CFE's user to decide. As such, DiCE returns several *diverse* CFEs for any given sample. However, the base algorithm of DiCE is not guaranteed to return CFEs which satisfy causal constraints, and so DiCE introduced an expensive pruning step to filter "non-feasible" CFEs, i.e., those that do not satisfy causal constraints. A method that directly incorporated these causal constraints into the exploration phase would likely improve the time to generate CFEs.

These optimization-based methods are, unfortunately, not model-agnostic, as they require the underlying predictive model to be differentiable and thus exclude popular predictive models such as random forests and k-nearest neighbors. To overcome this limitation, methods such as those of Tsirtsis et al. (2021) leverage dynamic programming to identify an optimal counterfactual policy and subsequently utilize this policy to find CFEs. However, due to the high memory requirements of these methods, this technique is best suited for low-dimensional data with discrete features.

CFRL (Samoilescu et al., 2021) and FastAR (Verma et al., 2022) take a different tactic, employing techniques from reinforcement learning (RL) to generate CFEs. CFRL first encodes samples into latent space using autoencoders (Kingma & Welling, 2013), then an RL agent is trained to find a CFE in the latent space. Finally, a decoder converts the latent CFE back to the input space. Similarly, FastAR converts the CFE problem to a Markov decision process (MDP) (Sutton & Barto, 2018) and then uses proximal policy optimization (PPO) (Schulman et al., 2017) to solve the MDP. However, the action candidates in FastAR are discrete and fixed, making it impractical for complex multivariate time-series data. Moreover, both CFRL and FastAR require access to training datasets.

These and several other CFE techniques cater to static data, but methods for handling multivariate time-series data are much less prevalent (Theissler et al., 2022). For time-series data with k-nearest neighbor or random shapelet forest models, Karlsson et al. (2020) introduce one approach. Wang et al. (2021a) focus on univariate time-series, seeking CFEs in a latent space before decoding them back to the input space. Native-Guide (Delaney et al., 2021) finds the nearest unlike neighbor of an original univariate time-series sample, identifies the most influential subsequence of the neighbor, and substitutes it for the corresponding region in the original sample. On the other hand, CoMTE (Ates et al., 2021) can handle multivariate time-series data. First, it searches for distractor candidates, which are the original sample's neighbors in the training dataset that are predicted as the target class. Then, it identifies the best substitution parts on each distractor candidate. Finally, it gives a CFE by replacing a corresponding part of the original sample with the best substitution of the best distractor candidate.

We also note that, on the surface, generating adversarial examples seems quite similar to generating CFEs: both create proximal samples that yield distinct predictions from the original inputs. However, while adversarial learning considers proximity, essential CFE properties such as actionability, feasibility, and causal constraints are mostly ignored (Wachter et al., 2017; Verma et al., 2020; Sulem et al., 2022).

3 Preliminaries

CFEs aim to solve the following task: Given a user input $\boldsymbol{x}^*$, a predictive model f and a target prediction Y' such that $f(\boldsymbol{x}^*) \neq Y'$, the goal is to find a transformation from $\boldsymbol{x}^*$ to a new sample $\boldsymbol{x'}$ (i.e. CFE) such that (Verma et al., 2020; Karimi et al., 2020a; Guidotti, 2022):

- *valid*: $f(\boldsymbol{x'}) = Y'$,
- *actionable*: if the user cannot modify feature j, then $\boldsymbol{x}^*_j = \boldsymbol{x'}_j$,
- *sparse*: the CFE $\boldsymbol{x'}$ should differ in as few features from $\boldsymbol{x}^*$ as possible,
- *proximal*: the distance between $\boldsymbol{x}^*$ and $\boldsymbol{x'}$ should be small in some metric, and
- *plausible*: $\boldsymbol{x'}$ should satisfy all causal constraints on the input.

where x_j denotes the j-th feature of x. Our method for finding CFEs will rely on reinforcement learning (RL). In RL, an agent and an environment interact with each other (Sutton & Barto, 2018). The agent takes an action a_t on a state s_t at time step t. The environment receives a_t and s_t from the agent and returns the next state s_{t+1} and a reward R_{t+1} to the agent. The goal of the agent is to maximize the expected cumulative (discounted) reward. RL can be categorized as model-free RL and model-based RL. In model-free RL (Mnih et al., 2015; Haarnoja et al., 2018), the agent learns a policy from real experience when a model of the environment is not available to the agent. In model-based RL (Silver et al., 2017; Ha & Schmidhuber,

Algorithm 1 NTD-CFE. Best viewed in color. Typical RL code is colored in gray.

Input: the original user input $\boldsymbol{x}^*$, a predictive model f, a target class Y', a reward function R, a state transition function F_p, a proximity measure D_{pxmt}, a proximity weight λ_{pxmt}, feature feasibility weights $\boldsymbol{W}_{fsib}$, maximum number of episodes M_E, maximum number of interventions per episode M_T, discrete feature indicators $\boldsymbol{D}_{\text{dis}}$, numbers of possible values of the discrete features $\{N_{\text{dis},d} | d \in \boldsymbol{D}_{\text{dis}}\}$.
Optional Input: non-actionable feature indicators $\boldsymbol{D}_{\text{non-act}}$, immutable feature indicators $\boldsymbol{D}_{\text{immu}}$, causal constraints C_{SCM}, feature range constraints C_{range}, in-distribution detector $F_{\text{in\_dist}}$, a discount factor γ, a learning rate α, a regularization weight λ_{WD}
Output: a CFE $\boldsymbol{O^*}$
$\boldsymbol{O} = \{\emptyset\}$
$E := 0$
while $E < M_E$ **do**
    $\boldsymbol{\tau} = \{\emptyset\}$ *# Keep a record of (state, action, reward) pairs*
    $t := 0$
    $\boldsymbol{x_t} := \boldsymbol{x}^*$
    while $t < M_T$ **do**
        $\boldsymbol{a}_t \sim \pi_{\boldsymbol{\theta}}(\cdot | \boldsymbol{x_t})$ *# Sample an action from the RL policy network*
        $\boldsymbol{x_{t+1}} := F_p(\boldsymbol{x_t}, \boldsymbol{a}_t)$ *# State transition from the current $\boldsymbol{x}$ to the next $\boldsymbol{x}$*
        (Optionally, update $\boldsymbol{x_{t+1}}$ according to C_{range}, C_{SCM} and $\boldsymbol{D}_{\text{immu}}$)
        $r_{t+1} := R\left(f(\boldsymbol{x_{t+1}}), Y', D_{pxmt}(\boldsymbol{x}^*, \boldsymbol{x_{t+1}}, \boldsymbol{W}_{fsib}), \lambda_{pxmt}\right)$ *# Compute the reward*
        $\boldsymbol{\tau} := \boldsymbol{\tau} \cup (\boldsymbol{x}_t, \boldsymbol{a}_t, r_{t+1})$ *# Add the pair to the record*
        if $f(\boldsymbol{x_{t+1}}) = Y'$ and $\boldsymbol{x_{t+1}} \notin \boldsymbol{O}$ **then** *# If a new valid CFE is found*
            $\boldsymbol{O} := \boldsymbol{O} \cup \boldsymbol{x_{t+1}}$ (Optionally, if also $F_{\text{in\_dist}}(\boldsymbol{x_{t+1}}) = \text{True}$)
            $t := t + 1$
            Break
        end if
        $t := t + 1$
    end while
    $T := t$
    for $t = 0, 1, \ldots, T - 1$ **do** *# Update network parameters*
        $G := \sum_{t'=t+1}^{T} \gamma^{t'-t-1} \cdot r_{t'}$
        $\theta := \theta + \alpha \cdot \gamma^t \cdot G \cdot \nabla \ln \pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t | \boldsymbol{x_t})$
    end for
    $E := E + 1$
end while
$\boldsymbol{O^*} := \min_i D_{pxmt}(\boldsymbol{x}^*, \boldsymbol{O_i}, \boldsymbol{W}_{fsib})$ *# Return the valid CFE with the lowest proximity*

2018), the agent plans a policy from simulated experience generated by a model of the environment. This model of the environment is either learned or given. Furthermore, a policy-based RL algorithm (Williams, 1992; Schulman et al., 2017; 2015) typically samples actions from a policy network $\pi_{\boldsymbol{\theta}}$ parameterized by neural networks with parameters $\boldsymbol{\theta}$. Given a state, $\pi_{\boldsymbol{\theta}}$ is trained to return the best action that maximizes the expected cumulative reward.

4 The proposed method: NTD-CFE

In this work, we propose No-Training-Dataset reinforcement-learning-based CFE (NTD-CFE), formulating CFE as an RL problem. In this setup, the RL environment is the CFE predictive model f. The RL state s is a sequence of CFEs beginning at the original user input $\boldsymbol{x}^*$ and ending in a final generated CFE $\boldsymbol{O^*}$. An action taken by the RL agent represents a small perturbation on the way from $\boldsymbol{x}^*$ to $\boldsymbol{O^*}$. A reward is a function of the predictive model and other objectives we introduce to maintain the properties discussed in Section 3 (more details below).

One-hot encoding is applied for categorical features. We assume that the prediction function of f computes quickly, which is a common assumption in model-based RL (Sutton & Barto, 2018). We also assume that continuous features in the original user input $\boldsymbol{x}^*$ are standardized to have mean 0 and variance 1.

NTD-CFE pseudocode is given in Algorithm 1. Let $\boldsymbol{x}^* \in \mathbb{R}^{K\times D}$ denote a user input sample, where K and D denote the total number of time steps and features, respectively. To provide for plausibility, the D features can optionally be divided into actionable features $\boldsymbol{D}_{\text{act}}$, which the user can directly change; non-actionable features $\boldsymbol{D}_{\text{non-act}}$, which may change due to causal constraints but which the user cannot directly change; and immutable features $\boldsymbol{D}_{\text{immu}}$, which may be used by the predictive model but which cannot change. $\boldsymbol{x}^*$ is static if $K = 1$ or temporal if $K > 1$. The time complexity of the algorithm is $O(M_E \cdot M_T)$.

Action (Line 11 of Algorithm 1) $\pi_{\boldsymbol{\theta}}$ represents an RL policy network parameterized by neural networks with parameters $\boldsymbol{\theta}$. Each action $\boldsymbol{a}$ sampled from $\pi_{\boldsymbol{\theta}}$ is 3-dimensional $\boldsymbol{a} = \{a_{\text{time}}, a_{\text{feat}}, a_{\text{stre}}\}$, where a_{time} denotes the time step of the intervention, a_{feat} denotes which feature to intervene on, and a_{stre} corresponds to the strength of the intervention. To be more specific, let DC and DD denote the numbers of actionable continuous and discrete features, respectively, where $DC + DD = |\boldsymbol{D}_{\text{act}}|$ and $|\boldsymbol{D}_{\text{act}}|$ denotes the total number of actionable features. Given an $\boldsymbol{x}$, the neural network parameterized by θ produces parameters to define four distributions $\{\boldsymbol{p}_{\text{time}}, \boldsymbol{p}_{\text{feat}}, \boldsymbol{\mu}_{\text{DC}}, \boldsymbol{\sigma}_{\text{DC}}, \boldsymbol{p}_{\boldsymbol{N}_{\text{dis}}}\} := \theta(\boldsymbol{x})$, such that $\boldsymbol{p}_{\text{time}} \in \mathbb{R}^K, \boldsymbol{p}_{\text{feat}} \in \mathbb{R}^{|\boldsymbol{D}_{\text{act}}|}, \boldsymbol{\mu}_{\text{DC}} \in \mathbb{R}^{DC}, \boldsymbol{\sigma}_{\text{DC}} \in \mathbb{R}^{DC}$ and $\boldsymbol{p}_{\boldsymbol{N}_{\text{dis}}} \in \mathbb{R}^{N_{\text{dis}}}$, where $N_{\text{dis}} = \sum_{i=1}^{DD} N_{\text{dis},i}$ and each $\boldsymbol{p}$ vector contains non-negative probabilities that sum to 1. The parameters $\boldsymbol{p}_{\text{time}}$ and $\boldsymbol{p}_{\text{feat}}$ define two categorical distributions from which a_{time} and a_{feat} are sampled, e.g. $a_{\text{time}} \sim \text{Cat}(\boldsymbol{p}_{\text{time}})$ and $a_{\text{feat}} \sim \text{Cat}(\boldsymbol{p}_{\text{feat}})$. When a_{feat} is a continuous feature, the corresponding mean and standard deviation parameters $\mu_{\text{DC},a_{\text{feat}}}$ and $\sigma_{\text{DC},a_{\text{feat}}}$ define a normal distribution from which we sample how strong the intervention is for this continuous feature, e.g. $a_{\text{stre}} \sim N(\mu_{\text{DC},a_{\text{feat}}}, \sigma^2_{\text{DC},a_{\text{feat}}})$. When a_{feat} is a discrete feature, the corresponding parameters $\boldsymbol{p}_{\boldsymbol{N}_{\text{dis}},\boldsymbol{a}_{\text{feat}}}$ define a categorical distribution from which we sample what the interventional value is for this discrete feature, e.g. $a_{\text{stre}} \sim \text{Cat}(\boldsymbol{p}_{\boldsymbol{N}_{\text{dis}},\boldsymbol{a}_{\text{feat}}})$.

State Transition (Line 12 of Algorithm 1) The state transition function F_p can be any appropriate function for an application domain. In Section 5, we define $F_p(\boldsymbol{x_t}, \boldsymbol{a}_t = \{a_{\text{time}}, a_{\text{feat}}, a_{\text{stre}}\})$ as:

$$x_{t+1}^{\{k,d\}} := \begin{cases} x_t^{\{k,d\}} + a_{\text{stre}} & \text{for } k \geq a_{\text{time}} \text{ and } d = a_{\text{feat}} \text{ (when feature } d \text{ is continuous)} \\ a_{\text{stre}} & \text{for } k \geq a_{\text{time}} \text{ and } d = a_{\text{feat}} \text{ (when feature } d \text{ is discrete)} \\ x_t^{\{k,d\}} & \text{otherwise} \end{cases}$$

where $x_t^{\{k,d\}}$ denotes the d-th feature of the t-th $\boldsymbol{x}$ at the time step k.

Constraints (Line 13 of Algorithm 1) Regarding causal constraints C_{SCM}, many existing works require a complete causal graph or complete structural causal model (SCM) (Peters et al., 2017; Karimi et al., 2020b; 2021). However, complete SCMs are often unavailable in practice (Verma et al., 2020; 2022). NTD-CFE works with partial SCMs. An (partial) SCM can be encoded as a set of rules. After the state transition function F_p takes place, NTD-CFE checks for whether the new state violates any rules in C_{SCM}. If a rule is violated, NTD-CFE acts accordingly. For example, it may choose to discard the change or set the corresponding value of the new state to a limiting value.

Reward and Proximity (Line 14 of Algorithm 1) We define the reward function R as:

$$r := R\left(f(\boldsymbol{x}), Y', D_{pxmt}(\boldsymbol{x^*}, \boldsymbol{x}, \boldsymbol{W}_{fsib}), \lambda_{pxmt}\right) = \begin{cases} 1 - \lambda_{pxmt} \cdot D_{pxmt}(\boldsymbol{x^*}, \boldsymbol{x}, \boldsymbol{W}_{fsib}) & \text{if } f(\boldsymbol{x}) = Y' \\ 0 & \text{otherwise} \end{cases}$$

It combines a prediction reward (1 or 0) and a weighted proximity loss D_{pxmt}. D_{pxmt} is 0 when $f(\boldsymbol{x}) \neq Y'$. Otherwise, in difficult settings where $f(\boldsymbol{x}) \neq Y'$ dominates over $f(\boldsymbol{x}) = Y'$, the RL agent would learn to produce CFEs that are too close to the original user input, which results in invalid CFEs. λ_{pxmt} ensures that the reward is positive when $f(\boldsymbol{x}) = Y'$.

The proximity measure D_{pxmt} can be any suitable measures for the application domain. In Section 5, we define D_{pxmt} as the L_1-norm for continuous features and as the L_0-norm for discrete features, weighted by $\boldsymbol{W}_{fsib}$:

$$D_{pxmt}(\boldsymbol{x^d}, \boldsymbol{x'^d}, W^d_{fsib}) = \begin{cases} \sum_{k=1}^{K} |x^{\{k,d\}} - x'^{\{k,d\}}| \cdot W^d_{fsib} & \text{(if feature } d \text{ is continuous)} \\ \sum_{k=1}^{K} I(x^{\{k,d\}} \neq x'^{\{k,d\}}) \cdot W^d_{fsib} & \text{(if feature } d \text{ is discrete)} \end{cases} \tag{1}$$

W^d_{fsib} denotes the feasibility to change the d-th feature, which encodes the user's preference on altering this feature. $\boldsymbol{x}^d$ denotes the d-th feature of $\boldsymbol{x}$, and $x^{\{k,d\}}$ denotes the d-th feature of $\boldsymbol{x}$ at the time step k. NTD-CFE prefers to generate CFEs by altering features associated with small W_{fsib}. If a user does not specify preference on features, $\boldsymbol{W}_{fsib}$ is 1 for all features.

Classification and Regression (Line 16 of Algorithm 1) Algorithm 1 describes NTD-CFE for a classification model f. As a model-agnostic method, NTD-CFE supports not only classification but also regression models. To work with a regression predictive model f, one can replace the first condition of Line 16 by $Y'_{lower} \leq f(\boldsymbol{x_{t+1}}) \leq Y'_{upper}$, where Y'_{lower} and Y'_{upper} represent the lower and upper bounds for the target regression value, respectively.

Output (Lines 17 and 30 of Algorithm 1) On Line 17, if a valid CFE is reached and is not already in the set $\boldsymbol{O}$, then it is added to $\boldsymbol{O}$. Optionally, an additional condition can be imposed such that a valid CFE is added to $\boldsymbol{O}$ only if it is also plausible. Finally, on Line 30, NTD-CFE returns the CFE with the lowest proximity from the set $\boldsymbol{O}$.

4.1 Limitation

Without training datasets, standardizing the data becomes impractical. One approach is to leverage W_{fsib} to mitigate the impact of continuous features at different scales. Assuming domain knowledge about the ranges of feature values (which is often available in practice), one can assign smaller W_{fsib} to continuous features at larger scales and larger W_{fsib} to continuous features at smaller scales. This approach relaxes the standardization assumption. However, in our evaluation, we always assume that the continuous features have a mean of 0 and a variance of 1. We leave the evaluation of this approach or a more sophisticated approach for future work.

The performance of the proposed model is unaffected by the size of the predictive model. Instead, its performance depends on the frequency with which the predictive model returns the desired prediction. We assume that the desired prediction returned by the predictive model is not sparse. This is a realistic assumption in many real-world domains. For example, numerous applicants with diverse personal characteristics applying for credit cards receive approval from the decision model of a bank. This assumption ensures that the RL agent receives enough reward signals to improve its performance. Without both training datasets and reward signals, it would be extremely challenging for ML methods to solve meaningful tasks.

5 Evaluation

In this section, we provide qualitative examples and quantitative experiment results to demonstrate the effectiveness of NTD-CFE for multivariate data-series data. The details about datasets and hyperparameters are provided in Appendix A and Appendix C, respectively.

5.1 Qualitative Examples

We illustrate qualitative examples generated by NTD-CFE with two interpretable rule-based predictive models and an interpretable Life Expectancy dataset. Please refer to Appendix B.1 for the definitions of the interpretable rule-based models. All examples in this section are generated using the first sample, which represents the country Albania, in the Life Expectancy dataset.

5.1.1 Equal feature feasibility weights W_{fsib}

In Figure 1, all features have equal feasibility weights ($W_{fsib} = 1.0$) as no user preferences are set. In Figure 1a, following the definition of rule-based model 1 (Appendix B.1), Albania's prediction is 0 (i.e. undesired), because "*GDP-per-capita*" *and* "*health-expenditure*" in the last 5 years are below 0. Accordingly, NTD-CFE generates a CFE by increasing these values above 0. In Figure 1b, if at least one of "*GDP-per-capita*" *or* "*health-expenditure*" is above 0 in the last 5 years, then rule-based model 2 predicts 1 (i.e. desired). NTD-CFE generates a valid CFE for rule-based model 2 by raising "*GDP-per-capita*" above 0.

5.1.2 Different feature feasibility weights W_{fsib}

However, changing "*GDP-per-capita*" for Albania may be impractical. An alternative way to make rule-based model 2 predict 1 is to increase "*health-expenditure*" above 0. NTD-CFE can achieve this in three different ways: (1) marking "*GDP-per-capita*" as non-actionable; (2) assigning a small feasibility weight to "*health-expenditure*;" or (3) assigning a large feasibility weight to "*GDP-per-capita*." (1) is straightforward with NTD-CFE. Hence, we only present the results for (2) and (3). In Appendix Figure 2a, the feasibility weight W_{fsib} for "*health-expenditure*" is set to 0.1, ten times smaller than that of "*GDP-per-capita*," which remains unchanged as 1. With the reduced feasibility weight for "*health-expenditure*," NTD-CFE generates a valid CFE by modifying "*health-expenditure*." In Appendix Figure 2b, the feasibility weight for "*GDP-per-capita*" is set to 10, ten times greater than other features. With the high feasibility weight for "*GDP-per-capita*," NTD-CFE preserves "*GDP-per-capita*" and looks for other features to achieve the desired prediction. As a result, NTD-CFE learns to alter "*health-expenditure*".

Next, we show that setting small feasibility weights on irrelevant features does not affect the CFEs generated by NTD-CFE. In Appendix Figure 2c, although the feasibility weights for irrelevant features "*CO2-emissions*," "*electric-power-consumption*," and "*forest-area*" are set to be ten times smaller than others, NTD-CFE still alters the relevant feature "*GDP-per-capita*". Additional results for different feasibility weights are provided in Appendix Figure 3.

5.2 Quantitative Experiments

In this section, we compare NTD-CFE to 4 baseline methods in 40 experiments, which correspond to 8 real-world datasets each evaluated with 5 predictive models. Additional experiments are provided and analyzed in the appendix.

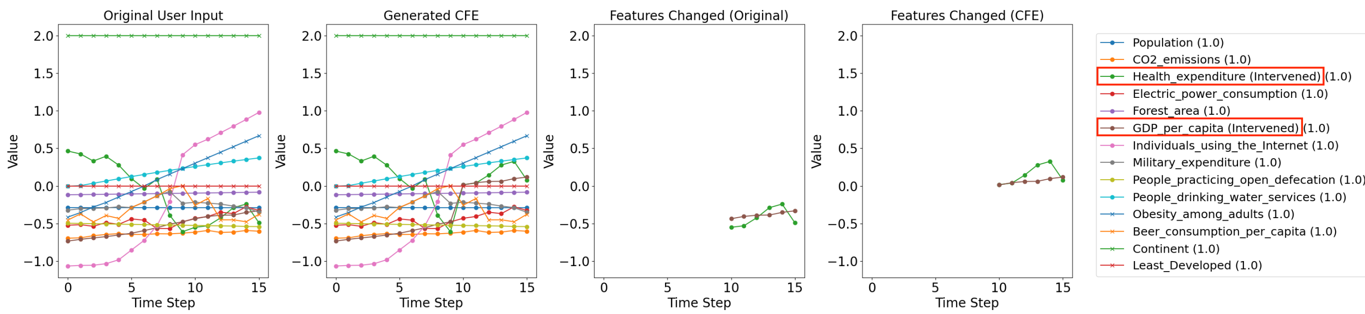
Datasets Eight real-world multivariate time-series datasets are used for evaluation: Life Expectancy, NATOPS, Heartbeat, Racket Sports, Basic Motions, eRing, Japanese Vowels, and Libras. Please see Appendix A for details.

Baseline Methods We benchmark NTD-CFE against four model-agnostic baseline methods: CoMTE (Ates et al., 2021), Native-Guide (Delaney et al., 2021), CFRL (Samoilescu et al., 2021), and FastAR (Verma et al., 2022). Optimization-based methods (Mothilal et al., 2020; Sulem et al., 2022; Hsieh et al., 2021) are excluded from the comparison, because our predictive models are not restricted to differentiable models. For Native-Guide, we follow the approach of Bahri et al. (2022) by concatenating multivariate time-series samples into univariate time-series samples. Unlike the proposed NTD-CFE, the baselines require training datasets, i.e. either $(\boldsymbol{X}_{train}, \boldsymbol{Y}_{train})$ or $(\boldsymbol{X}_{train})$. We omit comparisons with other methods and use these popular methods as the representative baselines.

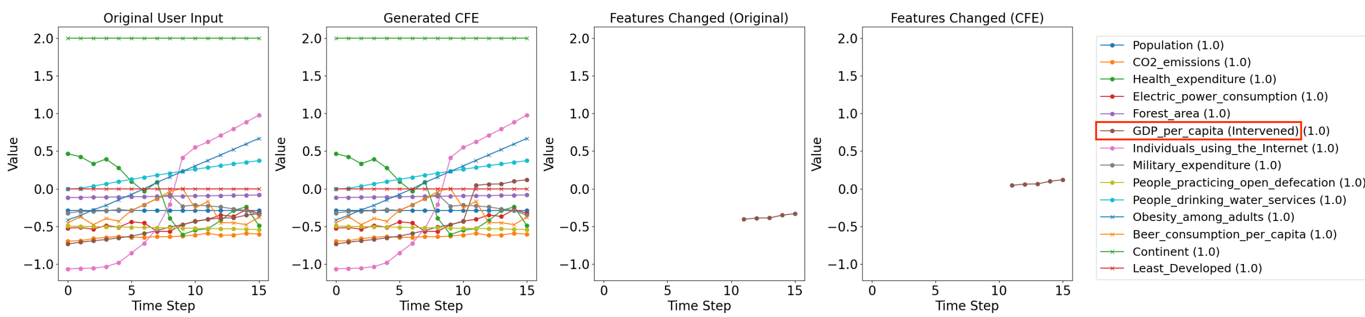
Predictive Models Five predictive models are employed per dataset: a long short-term memory (LSTM) neural network, a k-nearest neighbor (KNN), a random forest and two interpretable rule models; see Appendix B for details. Given eight datasets, there is a total of 40 predictive models.

5.2.1 Results

The methods are evaluated using five metrics. Let N_{inv} denote the total number of invalid samples, i.e., those classified as the undesired class by a predictive model, N_{inv_val} denote the number of invalid samples for



(a) Rule-based model 1



(b) Rule-based model 2

Figure 1: Qualitative examples with rule-based models and equal feature feasibility weights W_{fsib}. "Original User Input" shows the original input with the original feature values. "Generated CFE" shows the generated CFE with the modified features. The other two plots, "Features Changed (Original)" and "Features Changed (CFE)", omit most features and only show the modified features.

which a CFE method generates valid CFEs, N_{val} denote the number of valid CFEs generated by a CFE method, N_{CFE} denote the number of CFEs generated by a CFE method, and N_{plau_val} denote the number of plausible and valid CFEs generated by a CFE method. We set feature feasibility weights $W^d_{fsib} = 1$ for all the features $d \in D$.

Table 1 shows the results with the eRing dataset as an example. The proposed NTD-CFE consistently has the lowest proximity and sparsity. The results for all other datasets are given in Appendix Tables 3 to 9. We analyze and compare the *full results* in this section.

Results dimmed in gray in the tables are skipped from analysis and comparison. For Plausibility Rate, Proximity and Sparsity, we only compare the methods under 100% success rates. The reason is that if a method always generates CFEs that are close to the original *invalid* user input $\boldsymbol{x}^*$, even though these CFEs may often be invalid due to their closeness to $\boldsymbol{x}^*$, the plausibility rate, proximity and sparsity will always be superior. In the extreme case, if a method always returns the original but invalid $\boldsymbol{x}^*$, it would achieve a perfect plausibility rate, proximity and sparsity, but at the same time fail completely as a CFE method in terms of success rate.

Success Rate: $\frac{N_{inv\_val}}{N_{inv}}$ There are two scenarios for a CFE method to fail: 1) no valid CFEs are generated; 2) no CFEs (either valid or invalid) are generated. For RL-based baselines, CFRL and FastAR fail with a 0% success rate in 28/40 and 34/40 cases, respectively. NTD-CFE outperforms CFRL in 29/40 cases, and is on par with CFRL in 10/40 cases. In contrast, NTD-CFE underperforms CFRL in 1/40 case. NTD-CFE outperforms FastAR in all 40/40 cases. Compared to the other baselines, Native-Guide, CoMTA and NTD-CFE fail with a 0% success rate in 0/40, 3/40 and 0/40 cases, respectively. NTD-CFE outperforms Native-Guide in 8/40 cases, and is on par with it in 25/40 cases. NTD-CFE underperforms Native-Guide in 7/40 cases.

Table 1: Quantitative results with the eRing dataset for different predictive models and methods. Our proposed method NTD-CFE consistently achieves significantly better proximity and sparsity.

Predictive Model	N_{inv}	Methods	Success Rate	Validity Rate	Plausibility Rate	Proximity	Sparsity
LSTM	14	CoMTE	**100.0%**	**100.0%**	**100.0%**	346.746	260.0
		Native-Guide	**100.0%**	**100.0%**	**100.0%**	316.502	260.0
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	**100.0%**	**54.682**	**31.214**
KNN	16	CoMTE	**100.0%**	**100.0%**	**100.0%**	338.141	260.0
		Native-Guide	**100.0%**	**100.0%**	93.75%	3.789×10^{11}	229.938
		CFRL	**100.0%**	**100.0%**	**100.0%**	321.046	260.0
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	**100.0%**	**144.937**	**86.688**
Random Forest	15	CoMTE	**100.0%**	**100.0%**	**100.0%**	340.338	260.0
		Native-Guide	**100.0%**	**100.0%**	93.333%	3.275×10^{12}	246.467
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	**100.0%**	**68.882**	**46.733**
Rule Based 1	29	CoMTE	0.0%	—	—	—	—
		Native-Guide	**100.0%**	**100.0%**	44.828%	9.928×10^{14}	256.552
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	**100.0%**	**103.953**	**63.345**
Rule Based 2	25	CoMTE	**100.0%**	**100.0%**	**100.0%**	347.295	260.0
		Native-Guide	**100.0%**	**100.0%**	88.0%	5.075×10^{13}	245.4
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	**100.0%**	**31.301**	**14.76**

However, the minimum success rate of Native-Guide is 30.855%, which is better than that of NTD-CFE (0.68%). NTD-CFE underperforms CoMTE in success rate. NTD-CFE produces lower success rates than CoMTE in 11/40 cases, achieves the same success rates in 26/40 cases, and outperforms it in only 3/40 cases. Please see below for more about CoMTE.

It is important to note that: 1) We provide training datasets to the baselines (because they require training datasets to operate), but not to NTD-CFE. This additional information provided only to the baselines gives them an advantage over NTD-CFE. Without training datasets, the methods stop working except NTD-CFE. 2) In Appendix Table 18 we show that the success rate of NTD-CFE can be further improved, e.g. from 0.68% to 76.87%.

Validity Rate: $\frac{N_{val}}{N_{CFE}}$ Both NTD-CFE and CoMTE ensure perfect validity rates by design; they either produce a valid CFE or do not produce a CFE at all. In contrast, the other three baselines may return invalid CFEs; therefore, their validity rates may not be perfect. Furthermore, there are 3 cases where CoMTE fails completely with a 0% success rate (Tables 1, 4 and 8, explanations below). This results in undefined validity rates for CoMTE because $N_{CFE} = 0$. Hence, NTD-CFE outperforms all baselines in the experiments in terms of validity rate (i.e., 100% for all 40 cases). However, if there were cases where NTD-CFE fail with a 0% success rate, the validity rate for NTD-CFE would also be undefined.

Plausibility Rate: $\frac{N_{plau\_val}}{N_{val}}$ The comparisons to CFRL and FastAR are skipped because more than half of the experiments yield 0% success rates, and therefore, undefined plausibility rates. NTD-CFE outperforms and is on par with Native-Guide in 14/24 and 1/24 cases, respectively. NTD-CFE underperforms Native-Guide in

9/24 cases. NTD-CFE is on par with CoMTE in 8/26 cases and underperforms it in 18/26 cases. In summary, in terms of plausibility rate, CoMTE outperforms the proposed NTD-CFE, and NTD-CFE outperforms Native-Guide. Again, the baselines have the advantage by utilizing additional training information that is not provided to NTD-CFE.

Additionally, one can enforce plausibility in NTD-CFE (Line 17 of Algorithm 1). NTD-CFE achieves 100% plausibility rates at the cost of lower success rates and higher proximity and sparsity. Please see Appendix D for details.

Proximity and Sparsity Proximity and Sparsity are defined as the L_1-norm or L_0-norm, respectively, of the difference between a CFE and the original $\boldsymbol{x}^*$ (Verma et al., 2022; Samoilescu et al., 2021). Due to the aforementioned reason, proximity and sparsity are computed only with valid CFEs. Therefore, comparison with FastAR is skipped. *NTD-CFE outperforms all the baselines in proximity and sparsity in all cases. It also surpasses the baselines by a large margin.* For example, there are 3, 7 or 14 cases where the proximity of NTD-CFE is at least 20 times, 10 times or 5 times lower than that of all the baselines, respectively (e.g., 16.183 vs. 220.552 in Appendix Table 6). Similarly, there are 2, 4 or 20 cases where the sparsity of NTD-CFE is at least 50 times, 20 times or 10 times lower than that of all the baselines, respectively (e.g., 20.587 vs. 1224.0 in Appendix Table 4).

Comparison with RL-based methods. *NTD-CFE outperforms the two RL-based methods, CFRL and FastAR, in all the metrics.* CFRL and FastAR often fail to generate valid CFEs for complex multivariate time-series data. Please note that this is not a criticism of CFRL or FastAR, because they are not designed for multivariate time-series data.

Comparison with CoMTE Although CoMTE outperforms NTD-CFE in success rate and in plausibility rate, it is important to highlight that: 1) *CoMTE requires a training dataset, while NTD-CFE does not.* The better performance of CoMTE over NTD-CFE comes at the cost of needing more information and reduced versatility in practical applications. 2) CoMTE relies on finding distractors correctly classified as the target class. Tables 1, 4 and 8 for rule-based model 1 show that when the predictive models classify all training samples as the undesired class, CoMTE fails completely with a 0% success rate. In contrast, NTD-CFE is more versatile and can operate in such difficult situations. 3) Appendix Table 18 shows that the success rates of NTD-CFE can be improved by increasing the maximum number of episodes M_E or the maximum number of interventions per episode M_T, which correspond to more exhaustive RL search.

6 Conclusion

In this paper, we introduce NTD-CFE, a model-agnostic reinforcement learning based method that generates counterfactual explanations for static and multivariate time-series data. NTD-CFE operates without requiring a training dataset, is compatible with both classification and regression predictive models, handles continuous and discrete features, and offers functionalities such as feature feasibility (i.e. user preference), feature actionability and causal constraints. We illustrate the effectiveness of NTD-CFE through qualitative examples and benchmark it against four state-of-the-art methods using eight real-world multivariate time-series datasets. Our results consistently show that NTD-CFE produces counterfactual explanations with significantly better proximity and sparsity. Future research includes extending the work to large language models, exploring more advanced reinforcement learning algorithms to potentially enhance performance, relaxing assumptions such as data standardization, and exploring alternation solutions other than reinforcement learning for counterfactual explanation without training datasets.

References

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. In *Ethics of data and analytics*, pp. 254–264. Auerbach Publications, 2022.

Emre Ates, Burak Aksar, Vitus J Leung, and Ayse K Coskun. Counterfactual explanations for multivariate time series. In *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, pp. 1–8. IEEE, 2021.

Omar Bahri, Peiyu Li, Soukaina Filali Boubrahimi, and Shah Muhammad Hamdi. Temporal rule-based counterfactual explanations for multivariate time series. In *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1244–1249. IEEE, 2022.

Leo Breiman. *Classification and regression trees*. Routledge, 2017.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pp. 93–104, 2000.

Eoin Delaney, Derek Greene, and Mark T Keane. Instance-based counterfactual explanations for time series classification. In *International Conference on Case-Based Reasoning*, pp. 32–47. Springer, 2021.

Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pp. 1–55, 2022.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Chihcheng Hsieh, Catarina Moreira, and Chun Ouyang. Dice4el: interpreting process predictions using a milestone-aware counterfactual approach. In *2021 3rd International Conference on Process Mining (ICPM)*, pp. 88–95. IEEE, 2021.

Sheikh Rabiul Islam, William Eberle, Sheikh Khaled Ghafoor, and Mohiuddin Ahmed. Explainable artificial intelligence approaches: A survey. *arXiv preprint arXiv:2101.09429*, 2021.

Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Hiroki Arimura. Dace: Distribution-aware counterfactual explanation by mixed-integer linear optimization. In *IJCAI*, pp. 2855–2862, 2020.

Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050*, 2020a.

Amir-Hossein Karimi, Julius Von Kügelgen, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. *Advances in neural information processing systems*, 33:265–277, 2020b.

Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pp. 353–362, 2021.

Isak Karlsson, Jonathan Rebane, Panagiotis Papapetrou, and Aristides Gionis. Locally and globally explainable time series tweaking. *Knowledge and Information Systems*, 62(5):1671–1700, 2020.

Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pp. 607–617, 2020.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Osonde A Osoba, William Welser IV, and William Welser. *An intelligence in our image: The risks of bias and errors in artificial intelligence*. Rand Corporation, 2017.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Robert-Florian Samoilescu, Arnaud Van Looveren, and Janis Klaise. Model-agnostic and scalable counterfactual explanations via reinforcement learning. *arXiv preprint arXiv:2106.02597*, 2021.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Deborah Sulem, Michele Donini, Muhammad Bilal Zafar, Francois-Xavier Aubet, Jan Gasthaus, Tim Januschowski, Sanjiv Das, Krishnaram Kenthapadi, and Cedric Archambeau. Diverse counterfactual explanations for anomaly detection in time series. *arXiv preprint arXiv:2203.11103*, 2022.

Xiangyu Sun, Jack Davis, Oliver Schulte, and Guiliang Liu. Cracking the black box: Distilling deep sports analytics. In *Proceedings of the 26th acm sigkdd international conference on knowledge discovery & data mining*, pp. 3154–3162, 2020.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Andreas Theissler, Francesco Spinnato, Udo Schlegel, and Riccardo Guidotti. Explainable ai for time series classification: a review, taxonomy and research directions. *IEEE Access*, 10:100700–100724, 2022.

Stratis Tsirtsis, Abir De, and Manuel Rodriguez. Counterfactual explanations in sequential decision making under uncertainty. *Advances in Neural Information Processing Systems*, 34:30127–30139, 2021.

Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan E Hines, John P Dickerson, and Chirag Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

Sahil Verma, Keegan Hines, and John P Dickerson. Amortized generation of sequential algorithmic recourses for black-box models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8512–8519, 2022.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

Zhendong Wang, Isak Samsten, Rami Mochaourab, and Panagiotis Papapetrou. Learning time series counterfactuals via latent space representations. In *Discovery Science: 24th International Conference, DS 2021, Halifax, NS, Canada, October 11–13, 2021, Proceedings 24*, pp. 369–384. Springer, 2021a.

Zhendong Wang, Isak Samsten, and Panagiotis Papapetrou. Counterfactual explanations for survival prediction of cardiovascular icu patients. In *Artificial Intelligence in Medicine: 19th International Conference on Artificial Intelligence in Medicine, AIME 2021, Virtual Event, June 15–18, 2021, Proceedings*, pp. 338–348. Springer, 2021b.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

A Datasets

Eight real-world multivariate time-series datasets are used for evaluation in Section 5.2: Life Expectancy [1], eRing [2], NATOPS [3], Heartbeat [4], Racket Sports [5], Basic Motions [6], Japanese Vowels [7], and Libras [8].

Life Expectancy. The Life Expectancy dataset has 119 samples. Each sample has 16 time steps (from 2000 to 2015) and 17 features per time step. All the features are interpretable. Please see Table 2 for feature names and types. We remove "*Country Name*" and "*Year*" from the list of input features and use "*Life Expectancy*" in 2015 as the label. Therefore, the dataset has $K = 16$ and $D = 14$ in our notation. We set $Y = 1$ if "*Life Expectancy*" in 2015 is greater or equal to 75 as the target class and otherwise $Y = 0$ as the undesired class.

eRing The eRing dataset has 30 samples. Each sample has 65 time steps and 4 features per time step, i.e. $K = 65$ and $D = 4$. All the features in this dataset are continuous. There are six classes and we use the last three classes as the target classes.

NATOPS. The NATOPS dataset contains sensory data on hands, elbows, wrists and thumbs to classify movement types. It has 180 samples. Each sample has 51 time steps and 24 features per time step, i.e. $K = 51$ and $D = 24$. All the features in this dataset are continuous. There are 6 classes of different movements. We set class 4 to 6 as the target classes.

Heartbeat. The Heartbeat dataset has 204 samples. Each sample has 405 time steps and 61 features per time step, i.e. $K = 405$ and $D = 61$. All the features in this dataset are continuous. There are two classes: *normal* heartbeat (target class) and *abnormal* heartbeat (undesired class).

Racket Sports The Racket Sports dataset has 151 samples. Each sample has 30 time steps and 6 features per time step, i.e. $K = 30$ and $D = 6$. All the features in this dataset are continuous. There are four classes: "*Badminton Smash*", "*Badminton Clear*", "*Squash Forehand Boast*" and "*Squash Backhand Boast*". We use the last two classes as the target classes.

Basic Motions The Basic Motions dataset has 40 samples. Each sample has 100 time steps and 6 features per time step, i.e. $K = 100$ and $D = 6$. All the features in this dataset are continuous. There are four classes: "*Badminton*", "*Running*", "*Standing*" and "*Walking*". We use "*Standing*" as the target class.

Japanese Vowels The Japanese Vowels dataset has 270 samples. Each sample has 29 time steps and 12 features per time step, i.e. $K = 29$ and $D = 12$. All the features in this dataset are continuous. There are nine classes and we use the last five classes as the target classes.

Libras The Libras dataset has 180 samples. Each sample has 45 time steps and 2 features per time step, i.e. $K = 45$ and $D = 2$. All the features in this dataset are continuous. There are 15 classes and we use the last eight classes as the target classes.

All the categorical features are one-hot encoded. All the continuous features are standardized to have mean 0 and variance 1.

[1] https://www.kaggle.com/datasets/vrec99/life-expectancy-2000-2015
[2] https://www.timeseriesclassification.com/description.php?Dataset=ERing
[3] http://www.timeseriesclassification.com/description.php?Dataset=NATOPS
[4] http://www.timeseriesclassification.com/description.php?Dataset=Heartbeat
[5] https://www.timeseriesclassification.com/description.php?Dataset=RacketSports
[6] https://www.timeseriesclassification.com/description.php?Dataset=BasicMotions
[7] https://www.timeseriesclassification.com/description.php?Dataset=JapaneseVowels
[8] https://www.timeseriesclassification.com/description.php?Dataset=Libras

B Predictive Models Used in Section 5.2

Since NTD-CFE is model-agnostic, we expect it to work for predictive models with arbitrary hyperparameter values and architectures. In Section 5.2, we evaluate the methods using five predictive models: two interpretable rule- based models, a long short-term memory (LSTM) neural network, a K-nearest neighbor (KNNs), and a random forest.

B.1 Rule-based predictive models

Two interpretable rule-based predictive models are implemented for each dataset.

Rule-based models for the Life Expectancy dataset

We employed two interpretable rule-based predictive models for the Life Expectancy dataset in Section 5.1 and Section 5.2. Let d_1, d_2, d_3, d_4, d_5 denote the features "*least-developed*", "*GDP-per-capita*", "*health-expenditure*", "*people-using-at-least-basic-drinking-water-services*" and "*people-practicing-open-defecation*", respectively. We define the first rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} = 0 \wedge x^{\{k,d_2\}} > 0 \wedge x^{\{k,d_3\}} > 0 \wedge x^{\{k,d_4\}} > 0 \wedge x^{\{k,d_5\}} < 0 \text{ for } K-4 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

and the second rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} = 0 \wedge \left(x^{\{k,d_2\}} > 0 \vee x^{\{k,d_3\}} > 0\right) \wedge x^{\{k,d_4\}} > 0 \wedge x^{\{k,d_5\}} < 0 \text{ for } K-4 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

where Y' is the target class and Y is the undesired class.

Rule-based models for the NATOPS dataset

Let d_1 and d_2 denote the features "*Hand tip left, X coordinate*" and "*Hand tip right, X coordinate*", respectively. We define the first rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} > 0 \wedge x^{\{k,d_2\}} > 0 \text{ for } K-9 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

and the second rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} > 0 \vee x^{\{k,d_2\}} > 0 \text{ for } K-9 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

where Y' is one of the target classes and Y is one of the undesired classes.

Rule-based models for the Heartbeat dataset

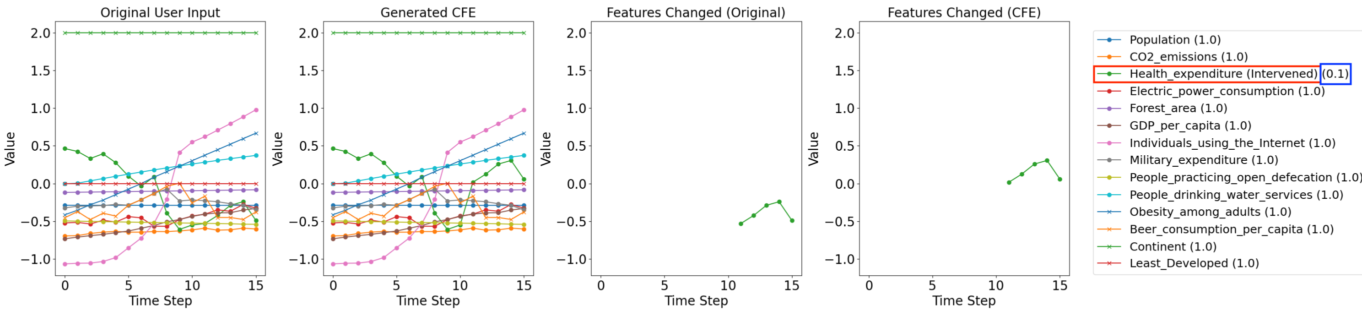
Let d_i denote the i-th feature. We define the first rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} > 0 \wedge x^{\{k,d_2\}} > 0 \wedge x^{\{k,d_3\}} > 0 \text{ for } K-4 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

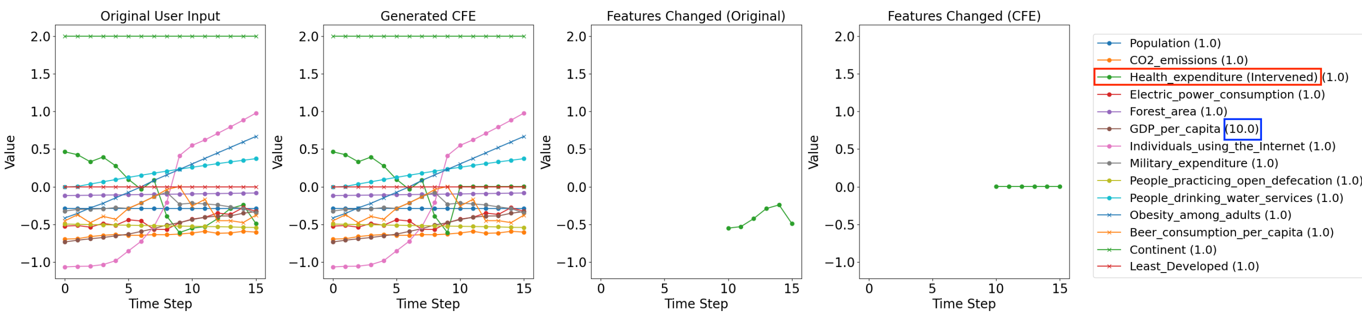
and the second rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} > 0 \vee x^{\{k,d_2\}} > 0 \vee x^{\{k,d_3\}} > 0 \text{ for } K-4 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

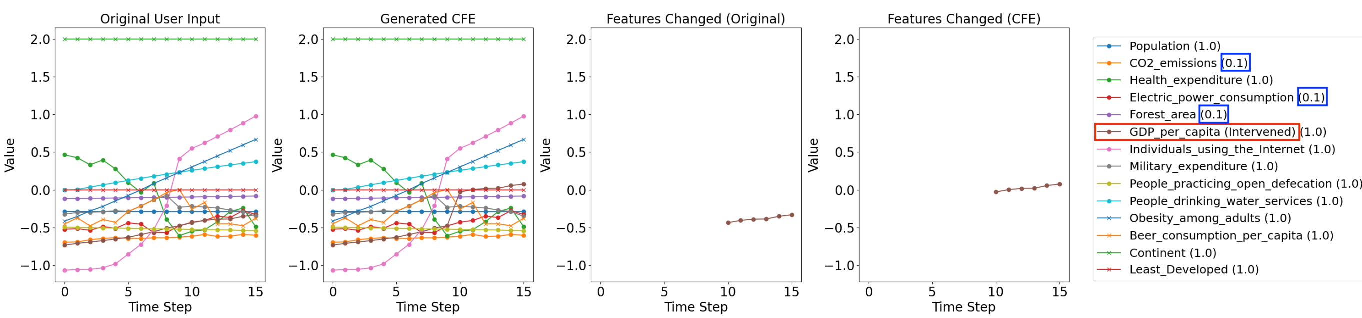
where Y' is the target class and Y is the undesired class.



(a) Rule-based model 2 Weight health 0.1



(b) Rule-based model 2 Weight GDP 10



(c) Rule-based model 2 Weight CO2 Elec Forest 0.1

Figure 2: Qualitative examples with rule-based model 2 and different feature feasibility weights W_{fsib}.

Rule-based models for the Racket Sports dataset

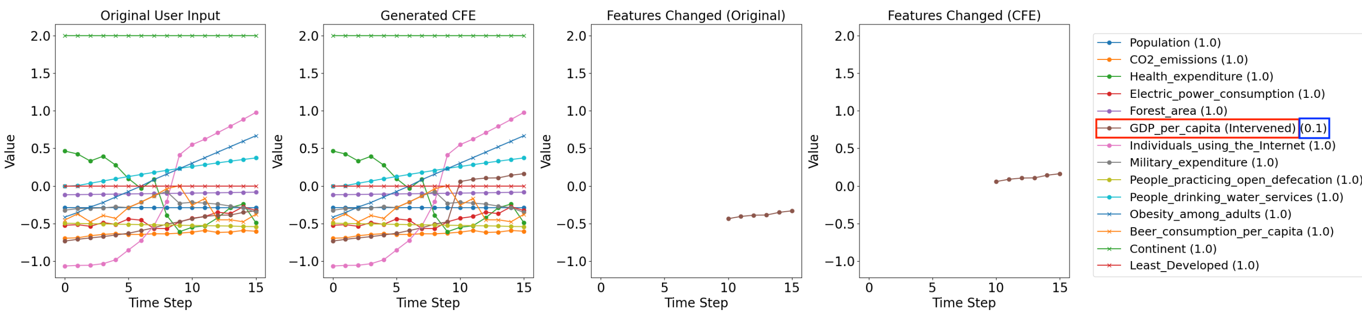
Let d_i denote the i-th feature. We define the first rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} > 0 \wedge x^{\{k,d_5\}} > 0 \text{ for } K-4 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

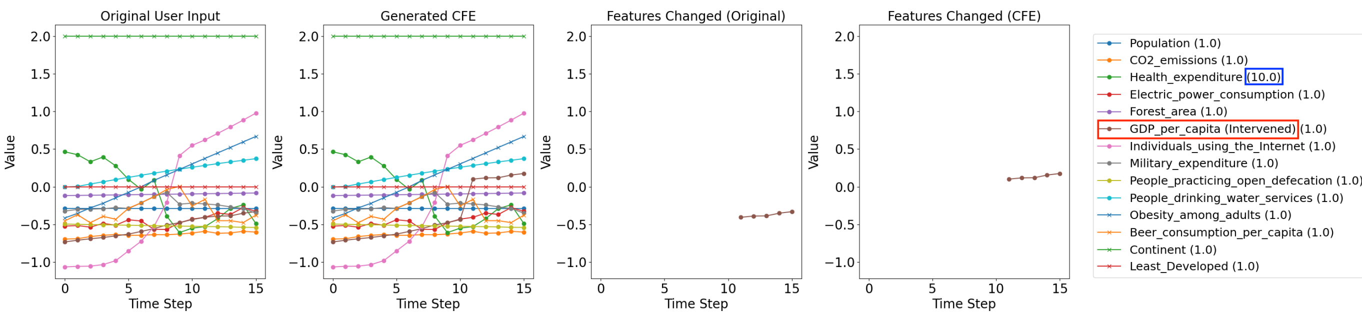
and the second rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} > 0 \vee x^{\{k,d_5\}} > 0 \text{ for } K-4 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

where Y' is the target class and Y is the undesired class.



(a) Rule-based model 2 Weight GDP 0.1



(b) Rule-based model 2 Weight health 10

Figure 3: Qualitative examples with rule-based model 2 and different feature feasibility weights W_{fsib}. Among relevant features to the predictive model prediction, NTD-CFE prefers the features with smaller W_{fsib}.

Rule-based models for the Basic Motions dataset

Let d_i denote the i-th feature. We define the first rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} > 0 \land x^{\{k,d_3\}} > 0 \land x^{\{k,d_6\}} > 0 \text{ for } K-9 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

and the second rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} > 0 \lor x^{\{k,d_3\}} > 0 \lor x^{\{k,d_6\}} > 0 \text{ for } K-9 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

where Y' is the target class "*Standing*", and Y is the undesired class.

Rule-based models for the eRing dataset

Let d_i denote the i-th feature. We define the first rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_2\}} > 0 \land x^{\{k,d_3\}} > 0 \text{ for } K-9 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

and the second rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_2\}} > 0 \lor x^{\{k,d_3\}} > 0 \text{ for } K-9 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

where Y' is the target class and Y is the undesired class.

Rule-based models for the Japanese Vowels dataset

Let d_i denote the i-th feature. We define the first rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} > 0 \wedge x^{\{k,d_6\}} > 0 \wedge x^{\{k,d_12\}} > 0 \text{ for } K-19 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

and the second rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} > 0 \vee x^{\{k,d_6\}} > 0 \vee x^{\{k,d_12\}} > 0 \text{ for } K-19 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

where Y' is the target class and Y is the undesired class.

Rule-based models for the Libras dataset

Let d_i denote the i-th feature. We define the first rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} > 0 \wedge x^{\{k,d_2\}} > 0 \text{ for } K-19 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

and the second rule-based model as:

$$f(\boldsymbol{x}) = \begin{cases} Y' & \text{if } x^{\{k,d_1\}} > 0 \vee x^{\{k,d_2\}} > 0 \text{ for } K-19 \leq k \leq K \\ Y & \text{otherwise} \end{cases}$$

where Y' is the target class and Y is the undesired class.

B.2 Other Predictive Models

Besides the rule-based models, each of the following predictive models is also used for each dataset.

long short-term memory (LSTM). The first layer of the neural network is a LSTM layer with 30 hidden states, followed by two linear layers. The first linear layer takes input of dimension of 30 and produces an output of dimension 60, then passes the output to a ReLU activation function. The second linear layer takes input of dimension of 60 and produces an output, then passes the output to a sigmoid activation function. We train the LSTM with learning rate 0.001 and weight decay 0.001 for 5000 epochs.

K-nearest neighbor (KNN). The number of neighbors to use for prediction is $\sqrt{N}$, where N denotes the number of samples in the dataset.

Random Forest. The number of trees is 100. The minimum number of samples required to split an internal node is 2. The minimum number of samples required to be at a leaf node is 1.

C Hyperparameters used in Section 5

We use a unique set of hyperparameter values for NTD-CFE throughout the paper, unless otherwise stated, without fine-tuning them:

- proximity weight $\lambda_{pxmt} = 0.001$
- maximum number of interventions per episode $M_T = 100$
- maximum number of episodes $M_E = 100$
- discount factor $\gamma = 0.99$

- learning rate $\alpha = 0.0001$
- regularization weight $\lambda_{WD} = 0.0$

The RL policy network contains two hidden linear layers with 1000 and 100 neurons, respectively. Adam (Kingma & Ba, 2014) is used as the optimizer.

It is important to note that NTD-CFE is not a supervised learning algorithm. As a reinforcement learning method, NTD-CFE does not rely on training datasets. Instead, it interacts with the environment (i.e., the predictive model f) for optimization. We select the set of hyperparameter values that works best for the Life Expectancy dataset from ten candidate sets of hyperparameter values. Although a more exhaustive hyperparameter search could potentially find another set of hyperparameters that produces better results, we leave this for future work. For the baseline methods, we use the hyperparameter values and architectures that are provided as default in their code [9]. All the experiments are conducted on CPU and with 32GB of RAM.

D Quantitative Experiments: NTD-CFE Versus NTD-CFE$_{\text{in_dist}}$

In this section, we compare NTD-CFE, which does not enforce plausibility, with NTD-CFE$_{\text{in_dist}}$, which enforces plausibility. For NTD-CFE$_{\text{in_dist}}$, $F_{\text{in\_dist}}(\boldsymbol{x}) = \text{True}$ is applied (Line 17 of Algorithm 1) while any other hyperparameter values remain unchanged. We employ local outlier factor (LOF) (Breunig et al., 2000) as the (optional) oracular in-distribution detector $F_{\text{in\_dist}}$. LOF is a common method for assessing plausibility in CFE literature (Kanamori et al., 2020; Delaney et al., 2021; Wang et al., 2021b). It employs KNNs to measure the degree to which a data point is unusual compared to others.

As shown in Tables 10 to 17, NTD-CFE$_{\text{in_dist}}$ achieves plausibility rates of 100%. However, its success rates are lower than those of NTD-CFE in 22/39 cases. We further compare their proximity and sparsity under the same success rates. Although NTD-CFE$_{\text{in_dist}}$ gets higher proximity in 8/17 cases and higher sparsity in 7/17 cases, the changes in the values are small.

[9]The code is publicly available at:
CoMTE: https://github.com/peaclab/CoMTE
Native-Guide: https://github.com/e-delaney/Instance-Based_CFE_TSC
CFRL: https://docs.seldon.io/projects/alibi/en/stable/methods/CFRL.html
FastAR: https://github.com/vsahil/FastAR-RL-for-generating-AR

Table 2: Features in the Life Expectancy dataset.

Features	**Type**
Country Name	Categorical
Year	Categorical
Continent	Categorical
Least Developed	Categorical
Population	Continuous
CO2 Emissions	Continuous
Health Expenditure	Continuous
Electric Power Consumption	Continuous
Forest Area	Continuous
GDP per Capita	Continuous
Individuals Using the Internet	Continuous
Military Expenditure	Continuous
People Practicing Open Defecation	Continuous
People Using at Least Basic Drinking Water Services	Continuous
Obesity Among Adults	Continuous
Beer Consumption per Capita	Continuous
Label: *Life Expectancy*	Categorical

Table 3: Quantitative results with the Life Expectancy dataset.

Predictive Model	N_{inv}	**Methods**	**Success Rate**	**Validity Rate**	**Plausibility Rate**	**Proximity**	**Sparsity**
LSTM	63	CoMTE	**100.0%**	**100.0%**	**100.0%**	37.475	201.54
		Native-Guide	**100.0%**	**100.0%**	85.714%	**30.674**	**190.238**
		CFRL	0.0%	0.0%	—	—	—
		FastAR	1.587%	1.587%	100.0%	0.45	1.0
		NTD-CFE	98.413%	**100.0%**	80.645%	10.893	64.065
KNN	68	CoMTE	**100.0%**	**100.0%**	**100.0%**	46.366	204.176
		Native-Guide	**100.0%**	**100.0%**	48.529%	1.452×10^{12}	**200.574**
		CFRL	**100.0%**	**100.0%**	**100.0%**	**44.264**	206.824
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	85.294%	**100.0%**	58.621%	19.756	82.879
Random Forest	62	CoMTE	**100.0%**	**100.0%**	**100.0%**	33.752	199.468
		Native-Guide	**100.0%**	**100.0%**	79.032%	2.834×10^{13}	200.548
		CFRL	**100.0%**	**100.0%**	**100.0%**	44.684	207.484
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	96.774%	**8.724**	**49.661**
Rule-Based 1	87	CoMTE	**100.0%**	**100.0%**	**100.0%**	**47.542**	**203.747**
		Native-Guide	65.517%	65.517%	66.667%	8.580×10^{13}	193.526
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	82.759%	**100.0%**	88.889%	10.566	49.25
Rule-Based 2	55	CoMTE	**100.0%**	**100.0%**	**100.0%**	**47.108**	**203.327**
		Native-Guide	72.727%	72.727%	60.0%	7.749×10^{13}	183.025
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	81.818%	**100.0%**	86.667%	11.238	52.667

Table 4: Quantitative results with the NATOPS dataset.

Predictive Model	N_{inv}	Methods	Success Rate	Validity Rate	Plausibility Rate	Proximity	Sparsity
LSTM	90	CoMTE	**100.0%**	**100.0%**	**100.0%**	1285.262	1224.0
		Native-Guide	**100.0%**	**100.0%**	63.333%	1.270×10^{13}	1158.133
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	28.889%	**227.184**	**135.1**
KNN	93	CoMTE	**100.0%**	**100.0%**	**100.0%**	**1284.259**	**1224.0**
		Native-Guide	**100.0%**	**100.0%**	55.914%	6.332×10^{13}	1213.172
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	6.452%	**100.0%**	50.0%	588.817	496.333
Random Forest	90	CoMTE	**100.0%**	**100.0%**	**100.0%**	1285.888	1224.0
		Native-Guide	**100.0%**	**100.0%**	28.889%	3.193×10^{12}	927.722
		CFRL	**100.0%**	**100.0%**	0.0%	1277.502	1224.0
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	45.556%	**228.323**	**157.733**
Rule-Based 1	178	CoMTE	0.0%	—	—	—	—
		Native-Guide	66.854%	66.854%	17.647%	1.349×10^{14}	1207.563
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**96.629%**	**100.0%**	86.047%	188.263	144.105
Rule-Based 2	126	CoMTE	**100.0%**	**100.0%**	**100.0%**	1294.181	1224.0
		Native-Guide	93.651%	93.651%	72.881%	1.445×10^{14}	1208.458
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	**100.0%**	**33.756**	**20.587**

Table 5: Quantitative results with the Heartbeat dataset.

Predictive Model	N_{inv}	**Methods**	**Success Rate**	**Validity Rate**	**Plausibility Rate**	**Proximity**	**Sparsity**
LSTM	136	CoMTE	**100.0%**	**100.0%**	**100.0%**	**621.692**	609.926
		Native-Guide	**100.0%**	**100.0%**	77.206%	1.075×10^{11}	**591.346**
		CFRL	2.941%	2.941%	100.0%	627.946	610.0
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	97.794%	**100.0%**	88.722%	16.825	12.12
KNN	192	CoMTE	**100.0%**	**100.0%**	**100.0%**	**626.788**	**609.948**
		Native-Guide	97.396%	97.396%	60.963%	4.749×10^{12}	600.824
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	72.396%	**100.0%**	30.935%	145.611	132.288
Random Forest	147	CoMTE	**100.0%**	**100.0%**	**100.0%**	**622.636**	**609.864**
		Native-Guide	65.986%	65.986%	79.381%	1.877×10^{12}	600.68
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	0.68%	**100.0%**	0.0%	57.664	48.0
Rule Based 1	171	CoMTE	**100.0%**	**100.0%**	**100.0%**	**624.442**	**609.883**
		Native-Guide	99.415%	99.415%	78.235%	5.192×10^{12}	571.535
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	70.175%	**100.0%**	43.333%	173.931	162.692
Rule Based 2	120	CoMTE	**100.0%**	**100.0%**	**100.0%**	619.454	609.917
		Native-Guide	**100.0%**	**100.0%**	82.5%	2.236×10^{11}	589.658
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	90.833%	**13.842**	**9.008**

Table 6: Quantitative results with the Racket Sports dataset.

Predictive Model	N_{inv}	Methods	Success Rate	Validity Rate	Plausibility Rate	Proximity	Sparsity
LSTM	78	CoMTE	**100.0%**	**100.0%**	**100.0%**	214.707	180.0
		Native-Guide	**100.0%**	**100.0%**	75.641%	202.626	161.59
		CFRL	0.0%	0.0%	—	—	—
		FastAR	14.103%	14.103%	100.0%	1.786	1.091
		NTD-CFE	**100.0%**	**100.0%**	98.718%	**16.734**	**8.295**
KNN	112	CoMTE	**100.0%**	**100.0%**	**100.0%**	217.73	180.0
		Native-Guide	**100.0%**	**100.0%**	76.786%	5.320×10^{12}	172.723
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	66.964%	**54.974**	**29.366**
Random Forest	82	CoMTE	**100.0%**	**100.0%**	**100.0%**	214.105	180.0
		Native-Guide	98.78%	98.78%	77.778%	3.529×10^{12}	167.222
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	90.244%	**43.695**	**26.232**
Rule Based 1	111	CoMTE	**100.0%**	**100.0%**	**100.0%**	**222.762**	**180.0**
		Native-Guide	94.595%	94.595%	67.619%	1.170×10^{14}	172.486
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	98.198%	**100.0%**	93.578%	24.46	13.385
Rule Based 2	16	CoMTE	**100.0%**	**100.0%**	**100.0%**	220.552	180.0
		Native-Guide	**100.0%**	**100.0%**	75.0%	228.065	170.125
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	**100.0%**	**16.183**	**9.438**

Table 7: Quantitative results with the Basic Motions dataset.

Predictive Model	N_{inv}	**Methods**	**Success Rate**	**Validity Rate**	**Plausibility Rate**	**Proximity**	**Sparsity**
LSTM	14	CoMTE	**100.0%**	**100.0%**	**100.0%**	782.188	600.0
		Native-Guide	**100.0%**	**100.0%**	85.714%	699.261	527.429
		CFRL	**100.0%**	**100.0%**	**100.0%**	802.065	600.0
		FastAR	7.143%	7.143%	100.0%	2.85	1.0
		NTD-CFE	**100.0%**	**100.0%**	**100.0%**	**87.828**	**38.429**
KNN	19	CoMTE	**100.0%**	**100.0%**	**100.0%**	761.556	600.0
		Native-Guide	**100.0%**	**100.0%**	78.947%	706.719	562.526
		CFRL	**100.0%**	**100.0%**	**100.0%**	795.277	600.0
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	94.737%	**206.984**	**121.421**
Random Forest	20	CoMTE	**100.0%**	**100.0%**	**100.0%**	751.741	600.0
		Native-Guide	**100.0%**	**100.0%**	80.0%	721.01	574.9
		CFRL	**100.0%**	**100.0%**	**100.0%**	773.104	600.0
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	50.0%	**305.677**	**182.25**
Rule Based 1	35	CoMTE	**100.0%**	**100.0%**	**100.0%**	**896.874**	**600.0**
		Native-Guide	97.143%	97.143%	88.235%	804.033	584.971
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	97.143%	**100.0%**	73.529%	133.66	80.559
Rule Based 2	8	CoMTE	**100.0%**	**100.0%**	**100.0%**	703.79	600.0
		Native-Guide	**100.0%**	**100.0%**	75.0%	687.975	562.0
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	**100.0%**	**44.01**	**19.25**

Table 8: Quantitative results with the Japanese Vowels dataset.

Predictive Model	N_{inv}	**Methods**	**Success Rate**	**Validity Rate**	**Plausibility Rate**	**Proximity**	**Sparsity**
LSTM	121	CoMTE	**100.0%**	**100.0%**	**100.0%**	330.771	300.0
		Native-Guide	**100.0%**	**100.0%**	88.43%	3.725×10^{12}	281.05
		CFRL	**100.0%**	**100.0%**	**100.0%**	335.712	300.0
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	99.174%	**35.3**	**19.413**
KNN	124	CoMTE	**100.0%**	**100.0%**	**100.0%**	331.845	300.0
		Native-Guide	**100.0%**	**100.0%**	85.484%	3.574×10^{12}	283.419
		CFRL	**100.0%**	**100.0%**	**100.0%**	334.799	300.0
		FastAR	1.613%	1.613%	100.0%	1.6	1.0
		NTD-CFE	**100.0%**	**100.0%**	71.774%	**104.147**	**62.073**
Random Forest	120	CoMTE	**100.0%**	**100.0%**	**100.0%**	331.467	300.0
		Native-Guide	**100.0%**	**100.0%**	89.167%	9.824×10^{12}	290.233
		CFRL	**100.0%**	**100.0%**	**100.0%**	334.707	300.0
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	90.0%	**83.856**	**52.55**
Rule Based 1	269	CoMTE	0.0%	—	—	—	—
		Native-Guide	30.855%	30.855%	78.313%	3.032×10^{13}	300.0
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**53.903%**	**100.0%**	57.241%	166.07	123.869
Rule Based 2	149	CoMTE	**100.0%**	**100.0%**	**100.0%**	330.597	300.0
		Native-Guide	67.114%	67.114%	77.0%	6.965×10^{12}	289.92
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	97.987%	**38.206**	**18.436**

Table 9: Quantitative results with the Libras dataset.

Predictive Model	N_{inv}	**Methods**	**Success Rate**	**Validity Rate**	**Plausibility Rate**	**Proximity**	**Sparsity**
LSTM	89	CoMTE	**100.0%**	**100.0%**	**100.0%**	120.329	89.494
		Native-Guide	**100.0%**	**100.0%**	64.045%	111.969	88.382
		CFRL	0.0%	0.0%	—	—	—
		FastAR	59.551%	59.551%	64.151%	2.037	1.094
		NTD-CFE	**100.0%**	**100.0%**	32.584%	**21.421**	**12.73**
KNN	89	CoMTE	**100.0%**	**100.0%**	**100.0%**	118.379	88.989
		Native-Guide	**100.0%**	**100.0%**	51.685%	126.318	89.213
		CFRL	**100.0%**	**100.0%**	**100.0%**	139.54	90.0
		FastAR	1.124%	1.124%	0.0%	4.4	3.0
		NTD-CFE	**100.0%**	**100.0%**	14.607%	**45.306**	**24.112**
Random Forest	84	CoMTE	**100.0%**	**100.0%**	**100.0%**	108.074	88.393
		Native-Guide	**100.0%**	**100.0%**	75.0%	111.086	87.143
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	13.095%	**50.151**	**27.024**
Rule Based 1	116	CoMTE	**100.0%**	**100.0%**	**100.0%**	144.053	88.836
		Native-Guide	**100.0%**	**100.0%**	87.069%	135.861	88.836
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	6.034%	**74.67**	**39.647**
Rule Based 2	53	CoMTE	**100.0%**	**100.0%**	**100.0%**	131.461	90.0
		Native-Guide	**100.0%**	**100.0%**	98.113%	135.119	90.0
		CFRL	0.0%	0.0%	—	—	—
		FastAR	0.0%	0.0%	—	—	—
		NTD-CFE	**100.0%**	**100.0%**	20.755%	**45.037**	**23.962**

Table 10: Compare NTD-CFE and NTD-CFE$_{\text{in_dist}}$ with the Life Expectancy dataset.

Predictive Model	N_{inv}	Methods	Success Rate	Validity Rate	Plausibility Rate	Proximity	Sparsity
LSTM	63	NTD-CFE	98.413%	100.0%	80.645%	10.893	64.065
		NTD-CFE$_{\text{in_dist}}$	93.651%	100.0%	100.0%	12.546	57.695
KNN	68	NTD-CFE	85.294%	100.0%	58.621%	19.756	82.879
		NTD-CFE$_{\text{in_dist}}$	79.412%	100.0%	100.0%	25.653	82.241
Random Forest	62	NTD-CFE	100.0%	100.0%	96.774%	8.724	49.661
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	8.984	49.242
Rule Based 1	87	NTD-CFE	82.759%	100.0%	88.889%	10.566	49.25
		NTD-CFE$_{\text{in_dist}}$	79.31%	100.0%	100.0%	10.063	46.014
Rule Based 2	55	NTD-CFE	81.818%	100.0%	86.667%	11.238	52.667
		NTD-CFE$_{\text{in_dist}}$	76.364%	100.0%	100.0%	9.329	46.69

Table 11: Compare NTD-CFE and NTD-CFE$_{\text{in_dist}}$ with the NATOPS dataset.

Predictive Model	N_{inv}	Methods	Success Rate	Validity Rate	Plausibility Rate	Proximity	Sparsity
LSTM	90	NTD-CFE	100.0%	100.0%	28.889%	227.184	135.1
		NTD-CFE$_{\text{in_dist}}$	40.0%	100.0%	100.0%	192.739	125.5
KNN	93	NTD-CFE	6.452%	100.0%	50.0%	588.817	496.333
		NTD-CFE$_{\text{in_dist}}$	3.226%	100.0%	100.0%	109.59	67.0
Random Forest	90	NTD-CFE	100.0%	100.0%	45.556%	228.323	157.733
		NTD-CFE$_{\text{in_dist}}$	63.333%	100.0%	100.0%	212.14	156.333
Rule Based 1	178	NTD-CFE	96.629%	100.0%	86.047%	188.263	144.105
		NTD-CFE$_{\text{in_dist}}$	90.449%	100.0%	100.0%	133.034	95.646
Rule Based 2	126	NTD-CFE	100.0%	100.0%	100.0%	33.756	20.587
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	33.756	20.587

Table 12: Compare NTD-CFE and NTD-CFE$_{\text{in_dist}}$ with the Heartbeat dataset.

Predictive Model	N_{inv}	Methods	Success Rate	Validity Rate	Plausibility Rate	Proximity	Sparsity
LSTM	136	NTD-CFE	97.794%	100.0%	88.722%	16.825	12.12
		NTD-CFE$_{\text{in_dist}}$	97.794%	100.0%	100.0%	19.104	14.714
KNN	192	NTD-CFE	72.396%	100.0%	30.935%	145.611	132.288
		NTD-CFE$_{\text{in_dist}}$	25.521%	100.0%	100.0%	87.665	77.245
Random Forest	147	NTD-CFE	0.68%	100.0%	0.0%	57.664	48.0
		NTD-CFE$_{\text{in_dist}}$	0.0%	—	—	—	—
Rule Based 1	171	NTD-CFE	70.175%	100.0%	43.333%	173.931	162.692
		NTD-CFE$_{\text{in_dist}}$	30.994%	100.0%	100.0%	83.472	75.906
Rule Based 2	120	NTD-CFE	100.0%	100.0%	90.833%	13.842	9.008
		NTD-CFE$_{\text{in_dist}}$	99.167%	100.0%	100.0%	14.205	9.613

Table 13: Compare NTD-CFE and NTD-CFE$_{\text{in_dist}}$ with the Racket Sports dataset.

Predictive Model	N_{inv}	Methods	Success Rate	Validity Rate	Plausibility Rate	Proximity	Sparsity
LSTM	78	NTD-CFE	100.0%	100.0%	98.718%	16.734	8.295
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	16.961	8.423
KNN	112	NTD-CFE	100.0%	100.0%	66.964%	54.974	29.366
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	64.028	36.42
Random Forest	82	NTD-CFE	100.0%	100.0%	90.244%	43.695	26.232
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	44.611	27.768
Rule Based 1	111	NTD-CFE	98.198%	100.0%	93.578%	24.46	13.385
		NTD-CFE$_{\text{in_dist}}$	98.198%	100.0%	100.0%	25.226	13.633
Rule Based 2	16	NTD-CFE	100.0%	100.0%	100.0%	16.183	9.438
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	16.183	9.438

Table 14: Compare NTD-CFE and NTD-CFE$_{\text{in_dist}}$ with the Basic Motions dataset.

Predictive Model	N_{inv}	**Methods**	**Success Rate**	**Validity Rate**	**Plausibility Rate**	**Proximity**	**Sparsity**
LSTM	14	NTD-CFE	100.0%	100.0%	100.0%	87.828	38.429
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	87.828	38.429
KNN	19	NTD-CFE	100.0%	100.0%	94.737%	206.984	121.421
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	223.681	128.368
Random Forest	20	NTD-CFE	100.0%	100.0%	50.0%	305.677	182.25
		NTD-CFE$_{\text{in_dist}}$	95.0%	100.0%	100.0%	321.474	199.895
Rule Based 1	35	NTD-CFE	97.143%	100.0%	73.529%	133.66	80.559
		NTD-CFE$_{\text{in_dist}}$	97.143%	100.0%	100.0%	157.419	104.824
Rule Based 2	8	NTD-CFE	100.0%	100.0%	100.0%	44.01	19.25
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	44.01	19.25

Table 15: Compare NTD-CFE and NTD-CFE$_{\text{in_dist}}$ with the eRing dataset.

Predictive Model	N_{inv}	**Methods**	**Success Rate**	**Validity Rate**	**Plausibility Rate**	**Proximity**	**Sparsity**
LSTM	14	NTD-CFE	100.0%	100.0%	100.0%	54.682	31.214
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	54.682	31.214
KNN	16	NTD-CFE	100.0%	100.0%	100.0%	144.937	86.688
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	144.937	86.688
Random Forest	15	NTD-CFE	100.0%	100.0%	100.0%	68.882	46.733
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	68.882	46.733
Rule Based 1	29	NTD-CFE	100.0%	100.0%	100.0%	103.953	63.345
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	103.953	63.345
Rule Based 2	25	NTD-CFE	100.0%	100.0%	100.0%	31.301	14.76
		NTD-CFE$_{\text{in_dist}}$	100.0%	100.0%	100.0%	31.301	14.76

Table 16: Compare NTD-CFE and NTD-CFE$_{\text{in_dist}}$ with the Japanese Vowels dataset.

Predictive Model	N_{inv}	**Methods**	**Success Rate**	**Validity Rate**	**Plausibility Rate**	**Proximity**	**Sparsity**
LSTM	121	NTD-CFE	100.0%	100.0%	99.174%	35.3	19.413
		NTD-CFE$_{\text{in_dist}}$	99.174%	100.0%	100.0%	35.753	19.833
KNN	124	NTD-CFE	100.0%	100.0%	71.774%	104.147	62.073
		NTD-CFE$_{\text{in_dist}}$	94.355%	100.0%	100.0%	114.437	74.667
Random Forest	120	NTD-CFE	100.0%	100.0%	90.0%	83.856	52.55
		NTD-CFE$_{\text{in_dist}}$	98.333%	100.0%	100.0%	85.103	54.314
Rule Based 1	269	NTD-CFE	53.903%	100.0%	57.241%	166.07	123.869
		NTD-CFE$_{\text{in_dist}}$	34.201%	100.0%	100.0%	135.475	98.674
Rule Based 2	149	NTD-CFE	100.0%	100.0%	97.987%	38.206	18.436
		NTD-CFE$_{\text{in_dist}}$	99.329%	100.0%	100.0%	38.038	18.926

Table 17: Compare NTD-CFE and NTD-CFE$_{\text{in_dist}}$ with the Libras dataset.

Predictive Model	N_{inv}	**Methods**	**Success Rate**	**Validity Rate**	**Plausibility Rate**	**Proximity**	**Sparsity**
LSTM	89	NTD-CFE	100.0%	100.0%	32.584%	21.421	12.73
		NTD-CFE$_{\text{in_dist}}$	93.258%	100.0%	100.0%	39.937	23.831
KNN	89	NTD-CFE	100.0%	100.0%	14.607%	45.306	24.112
		NTD-CFE$_{\text{in_dist}}$	88.764%	100.0%	100.0%	63.915	37.215
Random Forest	84	NTD-CFE	100.0%	100.0%	13.095%	50.151	27.024
		NTD-CFE$_{\text{in_dist}}$	88.095%	100.0%	100.0%	61.098	40.635
Rule Based 1	116	NTD-CFE	100.0%	100.0%	6.034%	74.67	39.647
		NTD-CFE$_{\text{in_dist}}$	50.862%	100.0%	100.0%	87.461	50.695
Rule Based 2	53	NTD-CFE	100.0%	100.0%	20.755%	45.037	23.962
		NTD-CFE$_{\text{in_dist}}$	84.906%	100.0%	100.0%	75.344	38.289

Table 18: This table shows that success rates are improved with increased maximum number of episodes M_E and maximum number of interventions per episode M_T, which correspond to more exhaustive RL search. We only test against the cases from Tables 1 and 3 to 9 in which the default hyperparameters give success rates below 50%.

NTD-CFE Hyperparameters	**Success Rate**	**Validity Rate**	**Plausibility Rate**	**Proximity**	**Sparsity**
$M_E = 100, M_T = 100$	0.68%	100.0%	0.0%	57.664	48.0
$M_E = 1000, M_T = 100$	10.204%	100.0%	20.0%	267.045	253.2
$M_E = 1000, M_T = 1000$	76.87%	100.0%	4.425%	444.993	417.336

(a) Dataset: Heartbeat. Predictive model: random forest.

NTD-CFE Hyperparameters	**Success Rate**	**Validity Rate**	**Plausibility Rate**	**Proximity**	**Sparsity**
$M_E = 100, M_T = 100$	6.452%	100.0%	50.0%	588.817	496.333
$M_E = 1000, M_T = 100$	12.903%	100.0%	33.333%	711.761	595.583
$M_E = 10000, M_T = 100$	26.882%	100.0%	16.0%	782.599	627.96

(b) Dataset: NATOPS. Predictive model: KNN.