ACE: Adapting sampling for Counterfactual Explanations

Margarita A. Guerrero MAGS3@KTH.SE
Division of Decision and Control, KTH Royal Institute of Technology, Stockholm, Sweden

Cristian R Rojas CRRO@KTH.SE
Division of Decision and Control, KTH Royal Institute of Technology, Stockholm, Sweden

Editors: G. Sukhatme, L. Lindemann, S. Tu, A. Wierman, N. Atanasov

Abstract

Counterfactual Explanations (CFEs) interpret machine learning models by identifying the smallest change to input features needed to change the model's prediction to a desired output. For classification tasks, CFEs determine how close a given sample is to the decision boundary of a trained classifier. Existing methods are often sample-inefficient, requiring numerous evaluations of a black-box model, which can be impractical when access to the model is limited. We propose Adaptive sampling for Counterfactual Explanations (ACE), a sample-efficient algorithm combining Bayesian estimation and stochastic optimization to approximate the decision boundary with fewer queries. By prioritizing informative points, ACE minimizes evaluations while generating accurate and feasible CFEs. Across benchmarks, ACE delivers higher query efficiency than state-of-the-art methods, yielding minimal changes, and demonstrates effectiveness in a control-tuning application.

Keywords: Explainable AI, Counterfactual Explanations, Bayesian Optimization

1. Introduction

Today, Artificial Intelligence (AI) has become an integral part of our lives, impacting both personal and professional decisions. A major challenge arises when determining whether to trust these increasingly complex machine learning models. To comply with the General Data Protection Regulation (Voigt and von dem Bussche, 2017) and other AI-specific data protection laws, organizations must explain how data is handled (European Commission and Technology, 2019), and, even in the absence of specific laws, numerous recommendations and guidelines advocate for transparency and explainability in AI (Molnar, 2022; Gilpin et al., 2018). This need has driven the development of Explainable AI (XAI) approaches (Samek et al., 2019), which aim to make AI systems more transparent, trustworthy, and less biased, ensuring that their decisions can be understood and justified.

Questions such as "Why did the model reject my loan application?" or, in control systems, "How much should I adjust the controller parameters to satisfy given closed-loop specifications?" can be addressed by model-agnostic XAI methods that explain black-box predictions. Examples include feature-importance methods such as LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017), which identify the attributes that play a major role in a classifier prediction. These ideas have recently inspired *control-oriented* XAI, adapting LIME/SHAP-style explanations to analyze control actions and MPC policies (Allamaa et al., 2025; Porcari et al., 2025).

When addressing these questions, it is essential to identify the minimal input changes needed to achieve a different prediction. However, the methods discussed above fall short, as they do not directly handle this goal, limiting the actionability of their explanations (Karimi et al., 2021). They also overlook feature correlations, often producing unrealistic data points. To address these issues, Counterfactual Explanations (CFEs) (Wachter et al., 2017; Verma et al., 2024) have emerged as a

more actionable alternative. CFEs seek the smallest feasible changes that alter a model's prediction to a desired outcome. This has inspired a variety of methods targeting simplicity (Sadiku et al., 2025), fairness (Fragkathoulas et al., 2024), and diversity (Mothilal et al., 2020), supported by open-source frameworks like CARLA (Pawelczyk et al., 2021) and OmniXAI (Yang et al., 2022).

In the context of binary classification, CFEs require only the ability to query the black-box model and observe its binary output, h. Since they do not need access to the model internals, CFEs are particularly suitable for scenarios where the model is proprietary or inaccessible, as is often the case when the models are owned by third parties, e.g., a loan approval system where only $h{=}1$ (approve) or $h{=}0$ (reject) is observable and queries are limited or costly. CFEs then propose feasible changes to flip h with a few queries.

Current CFE methods explore the input space via geometric expansions (Laugel et al., 2018), multi-objective evolutionary algorithms (Dandl et al., 2020; Regenwetter et al., 2024; Deb et al., 2002), or constraint-guided search (Karimi et al., 2020; Poyiadzi et al., 2020; Lucic et al., 2022; Pawelczyk et al., 2020). Surrogate-based approaches such as BayCon (Romashov et al., 2022) and EI-CFX (Spooner et al., 2021) leverage Gaussian Processes or tree models to guide sample selection. Nevertheless, these methods often neglect the cost of querying h, resulting in inefficiencies in scenarios where model evaluations are limited or expensive. Additionally, most approaches rely on large datasets for effective calibration, which may not be practical in real-world applications.

To address the limitations of existing methods, we propose the Adaptive sampling for Counterfactual Explanations (ACE) algorithm for classification tasks. ACE uses Gaussian processes to construct surrogate models and leverages Bayesian optimization (Mockus, 1989; Jones et al., 1998) to reduce query counts. It avoids complex hyperparameter tuning via a Gaussian prior and ensures global convergence using the penalty method (Gardner et al., 2014; Picheny et al., 2016). ACE further incorporates Monte Carlo sampling (Wilson et al., 2018) to prioritize informative points based on their relation to the acquired data. A key strength of ACE is its ability to handle both continuous and categorical features, optimizing a cost function via a hybrid approach: Quasi-Newton methods (Broyden, 1972) for continuous variables and Branch-and-Bound (Land and Doig, 1960) for discrete ones. Finally, extensive benchmark simulations show that ACE scales well from low- to high-dimensional spaces, identifying CFEs for datasets with as few as 21 features and as many as 784 features. Moreover, we apply ACE to tune a lead–lag controller on an industrial heat-exchanger case, enforcing minimal, safety-conscious changes that meet the specifications in a few trials.
To the best of our knowledge, this is the first Bayesian estimation–based method for sample-efficient counterfactuals, delivering high-quality explanations with far fewer model queries.

In summary, the main contributions of this work are:

1. We propose ACE, a new and efficient method for generating counterfactual explanations that requires only limited access to the black-box model—via query responses—and yet produces plausible, feasible, and actionable explanations.
2. We conduct a comprehensive quantitative and qualitative evaluation of ACE on multiple real-world datasets and a control case study, benchmarking against state-of-the-art CFE methods.

The remainder of this paper is organized as follows: Section 2 formulates the CFE problem, Section 3 presents the ACE algorithm, Sections 4–6 provide the empirical evaluation (quantitative, control study, qualitative), and Section 7 concludes with final remarks.

Notation: Vectors and matrices are written in bold. $[n]$ represents the set of indices from 1 to n and $[a]_+ := \max\{a, 0\}$. $\boldsymbol{x}_{1:n}$ represents a set of data points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, where each $\boldsymbol{x}_i$ is a point in the input space $\mathcal{X}$. C^k denotes the set of functions that are k times continuously differentiable.

2. Problem Statement

Consider a trained classifier $h\colon \mathcal{X} \to \{0, 1\}$, where $\mathcal{X}$ is an input space endowed with a metric d, and let $\tilde{\boldsymbol{x}} \in \mathcal{X}$ be a fixed input, referred to as the *instance*, whose predicted value $\tilde{y} = h(\tilde{\boldsymbol{x}})$ we aim to "explain". In this context, we define a *counterfactual explanation* (CFE) as the solution to the following optimization problem:

$$\underset{x}{\text{minimize}} \quad d(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \quad \text{s.t.} \quad \boldsymbol{x} \in \mathcal{S}_{db}, \tag{1}$$

where $\mathcal{S}_{db} = \{\boldsymbol{x} \in \mathcal{X}\colon h(\boldsymbol{x}) \neq h(\tilde{\boldsymbol{x}})\}$ represents the decision boundary of the classifier. The solution of (1) corresponds to identifying the point $\boldsymbol{x}$ closest to $\tilde{\boldsymbol{x}}$ that lies on the decision boundary of the classifier ($\mathcal{S}_{db}$), representing the smallest change needed in the input features to induce a classification flip (i.e., $h(\boldsymbol{x}) \neq h(\tilde{\boldsymbol{x}})$).

3. ACE Algorithm

In this section we present our novel algorithm, Adaptive sampling for Counterfactual Explanations (ACE), which relies on Bayesian optimization. To construct ACE, we address two central challenges: formulating a cost function J that captures proximity and structural constraints within the input space $\mathcal{X}$, and approximating the behavior of the black-box classifier h through a continuous surrogate model.

Following standard Bayesian classification (Rasmussen and Williams, 2006), we assume that the binary classifier is a thresholded version of a smooth latent function:

$$h(\boldsymbol{x}) = \mathbb{H}(f(\boldsymbol{x})), \tag{2}$$

where $\mathbb{H}\colon \mathbb{R} \to \{0, 1\}$ is the Heaviside step function centered at 0.5 (i.e., $\mathbb{H}(a) = 1$ if $a \geq 0.5$, and $\mathbb{H}(a) = 0$ otherwise), and $f\colon \mathcal{X} \to \mathbb{R}$ is a smooth map that we call the *black-box target function*.

The next subsections detail the main components of ACE: (i) reformulating the constrained counterfactual problem via a penalized objective over f; (ii) modeling f with a Gaussian Process to account for the black-box nature of h; (iii) leveraging Expected Improvement to guide Bayesian Optimization; (iv) optimizing mixed-type inputs via gradient-based and combinatorial methods.

3.1. Lagrangian Cost Function

As introduced in the previous section, we express the classifier as $h(\boldsymbol{x}) = \mathbb{H}(f(\boldsymbol{x}))$, where f is a real-valued latent function. Since f will later be modeled as a probabilistic surrogate (see Section 3.2), we interpret the decision boundary as the set of points satisfying $f(\boldsymbol{x}) = 0.5$. Then, the CFE problem (1) can be written in terms of f as

$$\underset{\boldsymbol{x}}{\text{minimize}}\ d(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \quad \text{subject to } f(\boldsymbol{x}) = 0.5. \tag{3}$$

In order to solve this problem with Bayesian Optimization, we reformulate it as an unconstrained problem through a Lagrangian formulation. Let us define the cost function $J(\boldsymbol{x}) = d(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \lambda|f(\boldsymbol{x}) - 0.5|$, and let $\{\lambda_k\}_{k=1,2,\ldots}$ be a non-negative, increasing sequence tending to infinity. At each iteration k, we solve

$$\underset{x}{\text{minimize}} \quad d(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \lambda_k |f(\boldsymbol{x}) - 0.5|, \tag{4}$$

by employing an optimization algorithm to identify the minimizer for the current penalty value λ_k. Ideally, as $\lambda_k \to \infty$, the solution of problem (4) is expected to approach the solution of the original constrained problem (3). This is the so-called "penalty function method" (Luenberger and Ye, 2008, Sec. 13.1). The terms $d(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ and $|f(\boldsymbol{x}) - 0.5|$ jointly enforce *proximity*: the first term keeps the candidate point close to the instance in the feature space, while the second one drives queries toward the decision boundary where the label flips. Additional constraints are then incorporated to satisfy the key properties for counterfactual explanations discussed in Section 1.

Actionability, Sparsity, and Plausibility. To enhance interpretability and feasibility, we impose three structural constraints (see more details in Appendix A): (i) *Actionability*—only actionable features may change; let $\mathcal{A} \subseteq \mathcal{X}$ be the set where immutable features remain fixed, i.e., $\boldsymbol{x} \in \mathcal{A}$ enforces domain constraints (e.g., age, gender) (Poyiadzi et al., 2020). (ii) *Sparsity*—a penalty $g(\boldsymbol{x} - \tilde{\boldsymbol{x}})$ (typically ℓ_1 or ℓ_0) promotes minimal modifications. (iii) *Plausibility*—a hard plausibility filter, $l(\boldsymbol{x}; \boldsymbol{X})$, is implemented via the Local Outlier Factor (LOF) (Breunig et al., 2000), which measures how isolated a candidate is relative to the local density of its neighbors in the collected data $\boldsymbol{X}$. Thus, ACE keeps inliers and discards points that lie far from the observed data manifold. Beyond these constraints, ACE also satisfies key desiderata (Doshi-Velez and Kim, 2017) such as validity, diversity, and scalability (Vo et al., 2023; Guidotti, 2022) (cf. Appendix A).

Extended Optimization Problem. Building upon the three structural constraints introduced above, we define the extended optimization problem as:

$$\underbrace{\underset{\boldsymbol{x} \in \mathcal{A} \subset \mathcal{X}}{\arg\min}}_{\text{actionability}} \underbrace{d(\boldsymbol{x}, \tilde{\boldsymbol{x}}) + \lambda_k |f(\boldsymbol{x}) - 0.5|}_{\text{proximity}} + \Theta(\boldsymbol{x}) \,, \qquad \Theta(\boldsymbol{x}) = \underbrace{\beta\, g(\boldsymbol{x} - \tilde{\boldsymbol{x}})}_{\text{sparsity}} + \underbrace{l(\boldsymbol{x}; \boldsymbol{X})}_{\text{plausibility}}, \tag{5}$$

where the hyperparameter $\beta > 0$ controls the trade-off with sparsity.

3.2. Surrogate Model

To solve optimization problem (5), we need to approximate f based on samples. However, we may only have access to the classifier output $h(\boldsymbol{x}) \in \{0, 1\}$ at a given sample $\boldsymbol{x} \in \mathcal{X}$, rather than the underlying value $f(\boldsymbol{x})$ required to solve (5). Therefore, we approximate the true underlying function f with a surrogate $\hat{f}$. In our setting, inspired by Gaussian Process Classifiers (GPC) (Bishop, 2006, Sec. 6.4), $\hat{f}$ is a realization of a Gaussian process that returns the likelihood of Class 1 membership, taking values in $[0, 1]$.

For classification problems, we estimate the class probability $p(t = 1 | \boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*)$, where $t \in \{0, 1\}$ is the label of a test input $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{x}_* = \boldsymbol{x}_{1:n}$ are the training inputs, and $\boldsymbol{t}_* = \boldsymbol{t}_{1:n}$ their binary labels. We denote the observed dataset by $(\boldsymbol{X}, \boldsymbol{y}) = (\boldsymbol{x}_*, \boldsymbol{t}_*)$, the kernel matrix by $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ with entries $K_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and the latent values at the training inputs by $\boldsymbol{a}_* = \hat{f}(\boldsymbol{x}_*)$. The posterior distribution $p(\boldsymbol{a}_* | \boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*)$ can be replaced by the Laplace approximation (Rasmussen and Williams, 2006, Sec. 3.4)

$$p(\boldsymbol{a}_* | \boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*) \approx \mathcal{N}\left(\boldsymbol{a}_*; \hat{\boldsymbol{a}},\ (\boldsymbol{W}(\hat{\boldsymbol{a}}) + \boldsymbol{K}^{-1})^{-1}\right),$$

where $\boldsymbol{W}(\boldsymbol{a}) = \operatorname{diag}(\sigma_s(a_i)[1 - \sigma_s(a_i)])$, with $\sigma_s(a) = 1/(1 + \exp(-a))$ denoting the logistic sigmoid function, and $\hat{\boldsymbol{a}}$ is obtained by iterating the Newton update in (Rasmussen and Williams, 2006, Sec. 3.4, Eq. (3.18)) until convergence. Given this approximation for the latent values at the training inputs, we can now compute the posterior distribution over the latent function value

$a = \hat{f}(\boldsymbol{x})$ at a test input $\boldsymbol{x}$, which is approximately Gaussian, $\mathcal{N}(\mu_a, \sigma_a^2)$, where μ_a and σ_a^2 are the logit mean and the logit variance, respectively, defined as

$$\mu_a = \boldsymbol{k}_n^T(\boldsymbol{t}_* - \boldsymbol{\sigma_s}(\hat{\boldsymbol{a}})), \qquad \sigma_a^2 = \kappa(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_n^T \left(\boldsymbol{W}(\hat{\boldsymbol{a}})^{-1} + \boldsymbol{K}\right)^{-1} \boldsymbol{k}_n, \tag{6}$$

with $\boldsymbol{k}_n := [\kappa(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, \kappa(\boldsymbol{x}, \boldsymbol{x}_n)]^T$ denoting the kernel vector between x and the training inputs. The predictive class-1 probability is then approximated using the inverse probit transformation, i.e., $p(t = 1 \mid \boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*) = \sigma_s\big(\mu_a\big(1 + \frac{\pi\sigma_a^2}{8}\big)^{-1/2}\big)$ (cf. Appendix C).

The posterior mean μ_a in logit space is transformed into a probability via the sigmoid function, yielding $\mu = \sigma_s(\mu_a)$. The variance in probability space is then $\sigma^2 = \sigma_a^2\mu^2(1-\mu)^2$, obtained via the delta method (Casella and Berger, 2024, p. 240), as discussed in Appendix D.

3.3. Expected Improvement

Given the previous observations and the constructed surrogate model, we adaptively pick new points using Expected Improvement (EI), a Bayesian optimization acquisition function that selects the most informative sampling point. EI measures the gap between the current optimum and the surrogate function at a point $\boldsymbol{x} \in \mathcal{X}$, i.e.,

$$\mathrm{EI}_n(x) := \mathbb{E}_n\left\{[J_n^* - J(\boldsymbol{x})]_+\right\} \tag{7}$$

where $\mathbb{E}_n$ denotes expectation conditioned on the previously observed data, $J(\boldsymbol{x})$ is defined in Equation (5), and $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}_i \in \boldsymbol{x}_{1:n}} J(\boldsymbol{x}_i)$ is the cost minimizer among the previously observed inputs. To estimate $\mathrm{EI}_n(\boldsymbol{x})$, we use a correlated Monte Carlo sampling method (Bishop, 2006, Sec. 11.1). Accordingly, we define $\hat{q}(\boldsymbol{x})$ as

$$\hat{q}(\boldsymbol{x}) = \max\big(0, d(\boldsymbol{x}^*, \tilde{\boldsymbol{x}}) + \lambda|\hat{f}(\boldsymbol{x}^*) - 0.5| + \Theta\left(\boldsymbol{x}^*\right) - d(\boldsymbol{x}, \tilde{\boldsymbol{x}}) - \lambda|\hat{f}(\boldsymbol{x}) - 0.5| - \Theta(\boldsymbol{x})\big).$$

We approximate Eq. (7) by a Monte Carlo average over m draws of $\hat{q}(\boldsymbol{x})$. To obtain *correlated* GP samples of the latent values $\hat{f}(\boldsymbol{x})$ and $\hat{f}(\boldsymbol{x}^*)$, we use their joint Gaussian specified by the posterior means $\mu(\boldsymbol{x}), \mu(\boldsymbol{x}^*)$, variances $\sigma^2(\boldsymbol{x}), \sigma^2(\boldsymbol{x}^*)$, and cross-covariance $\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x}^*)$, and then form the 2×2 covariance matrix $\boldsymbol{\Sigma}_f$. To generate correlated samples, we apply Cholesky decomposition to $\boldsymbol{\Sigma}_f$, i.e., $\boldsymbol{\Sigma}_f = \boldsymbol{L}\boldsymbol{L}^T$, where $\boldsymbol{L}$ is a lower triangular matrix. Sampling $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we obtain correlated samples via $[\hat{f}(\boldsymbol{x}), \hat{f}(\boldsymbol{x}^*)]^T = [\mu(\boldsymbol{x}), \mu(\boldsymbol{x}^*)]^T + \boldsymbol{L}\mathbf{z}$. The derivation of the cross-covariance term $\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x}^*)$ is detailed in Appendix D. Finally, we approximate the maximizer of EI_n using a general-purpose optimization method such as the L-BFGS-B algorithm (Liu and Nocedal, 1989).

3.4. Branch and Bound Method for Mixed Variables

ACE handles mixed variables by combining L-BFGS-B for continuous parameters with a Branch-and-Bound (B&B) search over categorical ones. Starting from the continuous optimum (root), B&B fixes the categorical values and re-optimizes the remaining continuous variables to maximize the acquisition function; categorical features use ordinal/label encoding when appropriate. Full details and a schematic example appear in Appendix E.

3.5. Overall Algorithm

The ACE pseudo-code is outlined in Algorithm 1, and the code is available on ACE repository.
Initialization. In line 1 from Algorithm 1, we draw n_0 samples to fit a Gaussian process. We assume mild prior information (bounds/summary stats): truncated normal for continuous features and

uniform over categories, or alternatively select n_0 points from a known dataset.

Optimize Acquisition Function. After fitting the kernel to $(\boldsymbol{X}, \boldsymbol{y})$, we maximize the acquisition function with L-BFGS-B (line 5), initialized around $\tilde{\boldsymbol{x}}$ via a truncated normal. For high-dimensional data, we initialize in a PCA space (Bishop, 2006, Sec. 12.1) and project back before optimizing.

Filtered Monte-Carlo Expected Improvement. ACE calculates the EI using Monte Carlo sampling, cf. Section 3.3, with *MC* samples.

Algorithm 1: ACE algorithm

Input: Initial data $\boldsymbol{n_0}$, instance to explain $\tilde{\boldsymbol{x}}$; **Parameters**: λ_0 (init. penalty), $\lambda_{\max}$ (max penalty), κ (kernel), MC (MC samples), SS (Sobol), ϵ (tol.), p (growth); **Output**: CFE $\boldsymbol{x_s}$

1: $\boldsymbol{X}, \boldsymbol{y} \leftarrow$ Update Initial Data $(\boldsymbol{x}_{1:n_0}, h(\boldsymbol{x}_{1:n_0}))$
2: **while** $h(\tilde{\boldsymbol{x}}) = h(\boldsymbol{x}_s^n)$ **and** $\|\boldsymbol{x}_s^n - \tilde{\boldsymbol{x}}\| < \|\boldsymbol{x}_s^o - \tilde{\boldsymbol{x}}\|$ **do**
3: $\quad \boldsymbol{x}_s^o \leftarrow \boldsymbol{x}_s^n$, $\lambda_k \leftarrow \lambda_0$
4: $\quad$ **while** $\|\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\| > \epsilon$ **or** $\lambda_k < \lambda_{\max}$ **do**
5: $\quad\quad \boldsymbol{x}_k \leftarrow \arg\max_x \mathrm{EI}_k(\boldsymbol{X}, \boldsymbol{y}, \lambda_k, \kappa, MC)$
6: $\quad\quad$ Observe $h(\boldsymbol{x}_k)$; $\boldsymbol{X}, \boldsymbol{y} \leftarrow$ Update Data $(\boldsymbol{x}_k, h(\boldsymbol{x}_k))$
7: $\quad\quad k \leftarrow k{+}1$, $\lambda_k \leftarrow (\lambda_{k-1})^p$
8: $\quad$ **end while**
9: $\quad \boldsymbol{x}_s^n \leftarrow$ Sample Decision Boundary $(\boldsymbol{X}, \boldsymbol{y}, \kappa, SS)$
10: **end while**
11: **return** $\boldsymbol{x}_s^n$

Both $d(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ and $g(\boldsymbol{x} - \tilde{\boldsymbol{x}})$ in (5) are computed using feature-normalized norms, where each dimension is scaled by the standard deviation of the corresponding feature in the current dataset $\boldsymbol{X}$.

Best Posterior CFE. ACE evaluates the posterior mean on a Sobol low-discrepancy grid (Sobol', 1967) to find points with posterior probability closest to 0.5, as shown in line 9. Among these, the CFE is the one with minimal Euclidean distance to $\tilde{\boldsymbol{x}}$; sparsity is already encouraged by the cost in (5). If the selected candidate $(\boldsymbol{x}_s^n)$ has the desired label but is farther from $\tilde{\boldsymbol{x}}$ than the previous CFE found $(\boldsymbol{x}_s^o)$, the algorithm terminates (line 4); otherwise, the process continues until a closer CFE is identified.

4. Quantitative Evaluation

In this section, we define evaluation criteria based on the key properties of CFEs (Section 3.1, Appendix A) and introduce an aggregated score for comparison. Using these metrics, we benchmark ACE against three state-of-the-art methods across eight binary classification datasets.

Table 1: Summary of real-world datasets.

Dataset	Features (Num/Cat)	Samples
Diabetes	8/0	768
KC2	21/0	522
Breast Cancer	9/0	683
Blood	4/0	748
Tic-Tac-Toe	0/9	958
Nursery	0/8	12,961
CMC	2/7	1,473
Credit	4/5	1,000

4.1. Experimental Setup

We evaluate three state-of-the-art methods: BayCon (Romashov et al., 2022), MOC (Dandl et al., 2020), and Growing Spheres (GS) (Laugel, 2018), with the latter implemented using CARLA (Pawelczyk et al., 2021). Experiments use eight real-world datasets (Table 1); due to space constraints, we report in the main text only four datasets—one label per setting—while the remaining ones and complementary-label results appear in Appendix F. Datasets with only numerical features are labeled *continuous*, while those mixing numerical and categorical features are *heterogeneous*.

For each dataset, we perform two evaluations: (i) *fixed-instance*, where a single input instance is selected and each method is executed 100 times using different random seeds to assess robustness and variability; and (ii) *mixed-instance*, which involves generating one counterfactual explanation for each of 100 randomly sampled instances and target labels to evaluate general performance. For BayCon and MOC, which yield multiple candidates, we report the one minimizing the cost in (5), ensuring a fair comparison with ACE and GS, which return a single CFE per run.

Across all experiments, CFEs are generated for a Random Forest black-box model trained via bootstrap aggregation. Hyperparameter details are provided in Appendix G. All experiments were conducted in Python 3.12.7, using SciPy for optimization[1].

4.2. Evaluation Metrics

The main evaluation metric is the *number of black-box evaluations* $h_{\#}$, used as a proxy for sample efficiency. We measure *proximity* via $d_2 = \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2$, and *sparsity* by $g_1 = \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_1$. In the mixed-instance test setting, both metrics are normalized by the standard deviation of each input feature computed over the entire dataset to account for feature scale differences, and are denoted by d_{2_N} and g_{1_N}, respectively. *Plausibility* uses the affinity score $\alpha(\boldsymbol{x}) := \texttt{clip}\{\exp(1 + \text{LOF}_k(\boldsymbol{x}))\}$, where $\text{LOF}_k(\boldsymbol{x})$ denotes the Local Outlier Factor score of $\boldsymbol{x}$, computed with respect to its k nearest neighbors in the collected data $\boldsymbol{X}$; accordingly, values of $\alpha(\boldsymbol{x})$ close to 1 indicate inliers (cf. Appendix B). We also report *validity* $\mathcal{V}$, the proportion of successful counterfactual generations.

CFE Score

To compare methods, we define a scalar *CFE Score* $\mathcal{S}$ that aggregates all metrics into a single value using min-max normalization. Since affinity and validity lie in $[0, 1]$ and are to be maximized, we use $1 - \boldsymbol{x}_{ij}$ to align interpretation across terms, where lower values are preferred.

Based on the normalized metrics, the CFE Score is defined as: $\mathcal{S} = w_1 \cdot \tilde{h}_{\#} + w_2 \cdot \tilde{d}_2 + w_3 \cdot \tilde{g}_1 + w_4 \cdot (1 - \alpha) + w_5 \cdot (1 - \mathcal{V})$, where the tilde $\tilde{(\cdot)}$ denotes min-max normalization, and w_i are weights reflecting metric priorities. Since ACE prioritizes sample efficiency, we set $w_1 = 0.35$ for the number of queries $h_{\#}$. The remaining weights are 0.25 for d_2, 0.15 for g_1, and 0.125 for both affinity and validity, with $\sum w_i = 1$. A lower $\mathcal{S}$ indicates better performance.

4.3. Benchmark Results

Fixed Test Results

Tables 2 and 3 summarize the evaluation results for continuous and heterogeneous datasets. Each method explains two fixed instances 100 times to assess consistency (mean) and variability (standard deviation in parentheses). Non-actionable features—such as age or gender—are kept fixed throughout the optimization to guarantee feasibility and interpretability. ACE is initialized with $n_0 = 30$ points for all datasets (counted in the total evaluations). Across all four datasets, ACE consistently outperforms competing methods in terms of the number of black-box evaluations ($h_{\#}$), successfully generating counterfactuals in every experiment.

Continuous Datasets. On *Diabetes* and *Breast*, ACE achieves the lowest CFE Score, offering a better trade-off across all metrics. It outperforms BayCon and MOC in proximity while maintaining similar sparsity. Although GS obtains slightly better d_2, it requires over 28,000 and 40,883 queries—compared to around 71 and 60 for ACE—making it highly inefficient. Notably, ACE consistently achieves $\alpha(\boldsymbol{x}) \approx 1$ and is the only method to produce valid CFEs in all 200 queries (same for all 600 runs in Appendix F).

Heterogeneous Datasets. On *CMC* and *Nursery*, ACE identifies CFEs by changing only one feature by a single unit, while using far fewer evaluations—demonstrating the efficiency of its B&B strategy

1. The GP classifier in ACE was implemented from scratch, as standard libraries (e.g., `scikit-learn`) do not expose the latent posterior—i.e., the mean and variance in (6)—required to compute covariances between candidates $\boldsymbol{x}$ and the current best $\boldsymbol{x}^*$ (see Appendix D).

Table 2: Fixed test for continuous datasets with 100 tested points per experiment. Best CFE score are highlighted.

Dataset	Method	$\mathbf{h}_{\#}$ (std)	$\mathbf{d_2}$ (std)	$\mathbf{g_1}$ (std)	$\alpha(\mathbf{x})$ (std)	$\mathcal{V}$ (std)	$\mathcal{S}$
Diabetes	ACE	71 (16)	34.09 (14.30)	49.57 (22.34)	1 (0.06)	1 (0)	**0.24**
	BayCon	1211 (110)	37.13 (9.23)	46.87 (12.19)	0.25 (0.08)	1 (0)	0.36
	MOC	2038 (1065)	58.95 (18.46)	62.91 (16.77)	0.84 (0.19)	0.98 (0.14)	0.45
	CARLA$_{GS}$	28013 (4455)	5.51 (0.90)	12.46 (2.24)	0.78 (0.03)	1 (0)	0.38
Breast	ACE	60 (16)	9.33 (1.05)	27.89 (2.87)	1 (0.10)	1 (0)	**0.20**
	BayCon	3567 (1058)	9.60 (1.25)	21.12 (1.90)	0.31 (0.03)	0.03 (0.20)	0.35
	MOC	845 (748)	10.28 (1.19)	20.81 (3.11)	0.09 (0.06)	0.97 (0.17)	0.37
	CARLA$_{GS}$	40883 (3545)	9.08 (0.71)	21.41 (2.25)	0.77 (0.03)	1 (0)	0.39

Table 3: Fixed test for heterogeneous datasets with 100 tested points per experiment. Best CFE score are highlighted.

Dataset	Method	$h_{\#}$ (std)	d_2 (std)	g_1 (std)	$\alpha(x)$ (std)	$\mathcal{V}$ (std)	$\mathcal{S}$
CMC	ACE	65 (6)	1 (0)	1 (0)	1 (0.01)	1 (0)	**0**
	BayCon	1021 (102)	1 (0)	1 (0)	0.33 (0.02)	1 (0)	0.43
	MOC	590 (212)	1 (0.1)	1 (0.2)	0.9 (0.1)	1 (0)	0.2
	CARLA$_{GS}$	1005 (0)	3.87 (0)	7 (0)	1 (0)	1 (0)	0.74
Nursery	ACE	57 (7)	1 (0.04)	1.01 (0.1)	1 (0)	1 (0)	**0**
	BayCon	836 (29)	1 (0)	1 (0)	0.37 (0.01)	1 (0)	0.37
	MOC	220 (0)	- (-)	- (-)	- (-)	0 (0)	-
	CARLA$_{GS}$	1005 (0)	3.32 (0)	5 (0)	1 (0)	1 (0)	0.75

Table 4: Mixed test for continuous datasets. Best CFE scores are highlighted.

Dataset	Method	$h_{\#}$(std)	d_{2_N} (std)	g_{1_N} (std)	α(std)	$\mathcal{V}$ (std)	$\mathcal{S}$
Diabetes	ACE	72 (18)	2.46 (0.84)	2.7 (1.87)	0.98 (0.09)	1 (0)	**0.10**
	BayCon	1239 (238)	2.04 (0.97)	2.99 (1.89)	0.27 (0.09)	1 (0)	0.14
	MOC	1753 (1011)	2.09 (0.91)	2.62 (1.34)	0.58 (0.36)	0.91 (0.29)	**0.10**
	CARLA$_{GS}$	27463 (23054)	3.2 (2.66)	4.26 (3.41)	0.74 (0.08)	1 (0)	0.78
Breast	ACE	67 (22)	2.45 (1.37)	7.6 (3.95)	1 (0.01)	1 (0)	**0.16**
	BayCon	2045 (952)	2.42 (1.25)	4.25 (2.63)	0.30 (0.08)	0.60 (0.49)	**0.16**
	MOC	890 (804)	3.36 (1.19)	5.7 (2.51)	0.47 (0.33)	1 (0)	0.39
	CARLA$_{GS}$	36803 (20489)	2.62 (1.46)	6.56 (3.81)	0.76 (0.05)	1 (0)	0.54

Table 5: Mixed test for heterogeneous datasets. Best CFE scores are highlighted.

Dataset	Method	$h_{\#}$ (std)	d_{2_N} (std)	g_{1_N} (std)	α (std)	$\mathcal{V}$ (std)	$\mathcal{S}$
CMC	ACE	70 (12)	2.12 (0.92)	2.99 (1.8)	1 (0.01)	0.97 (0.17)	**0.07**
	BayCon	1024 (241)	1.85 (0.71)	2.39 (1.24)	0.32 (0.05)	0.98 (0.14)	0.12
	MOC	332 (124)	1.57 (0.68)	1.8 (1.01)	0.8 (0.24)	0.18 (0.38)	0.13
	CARLA$_{GS}$	330674 (469741)	5.04 (0.9)	9.19 (1.96)	1 (0)	0.67 (0.47)	0.79
Nursery	ACE	56 (7)	1.22 (0)	1.22 (0)	1 (0.01)	1 (0)	**0**
	BayCon	840 (31)	1.22 (0)	1.22 (0)	0.37 (0)	1 (0)	0.37
	MOC	226 (10.47)	- (-)	- (-)	- (-)	0 (0)	-
	CARLA$_{GS}$	1005 (0)	2.09 (0.5)	3.4 (1.27)	1 (0)	1 (0)	0.75

Once again, ACE maintains high plausibility and validity, confirming its robustness on mixed-variable datasets.

Mixed Test Results

Tables 4 and 5 present the results of the mixed-instance test, where as in the fixed test, ACE retains its sample-efficiency edge, matching or surpassing baselines with far fewer evaluations.

Continuous Datasets. ACE tops CFE Score across datasets, while BayCon attains slightly better proximity and sparsity. However, BayCon's CFEs are less plausible ($\alpha(\boldsymbol{x}) = 0.27 - 0.30$ vs. $0.98 - 1$ for ACE), indicating they lie farther from the data manifold. ACE also requires only 5.77% of BayCon's queries on average, highlighting its efficiency.

Heterogeneous Datasets. Again, ACE obtains the best CFE Score on all datasets. Nevertheless, a similar trade-off is observed: while BayCon achieves slightly better proximity or sparsity, its affinity score is markedly lower. Thus, ACE yields feasible, actionable CFEs under tighter query budgets.

5. Adaptive Control Tuning

In this section we apply ACE to iterative controller retuning. Starting from a controller-parameter baseline $\boldsymbol{\theta}_0$ that fails time–domain specs, we cast "meeting the specifications" as a binary oracle $h(\boldsymbol{\theta}) \in \{0, 1\}$ and seek the smallest actionable change $\boldsymbol{\theta} - \boldsymbol{\theta}_0$ that flips 0 (fail) to 1 (pass) under tight trial budgets.

5.1. Methodology

We consider a critical process that cannot be halted during operation; thus, parameter tuning must rely on short *closed-loop* trials with small, limited updates and without aggressive excitation.

Let $\boldsymbol{\theta}_i \in \mathbb{R}^p$ denote the controller parameters used in the i-th closed-loop trial. For a unit step, we compute the vector of time-domain metrics $\pi(\boldsymbol{\theta}_i) = \big(e_{\mathrm{ss}}(\boldsymbol{\theta}_i), t_{\mathrm{rise}}(\boldsymbol{\theta}_i), t_{\mathrm{settle}}(\boldsymbol{\theta}_i), \mathrm{OS}, \|u\|_{\mathrm{rms}}(\boldsymbol{\theta}_i)\big)$, where e_{ss} is the steady-state error, t_{rise} the rise time, t_{settle} the settling time (for a 5% band), OS the overshoot percentage, and $\|u\|_{\mathrm{rms}}$ is the control effort. The pass/fail binary oracle is

$$h(\boldsymbol{\theta}_i) = \mathbb{I}\{\pi(\boldsymbol{\theta}_i) \preceq \boldsymbol{\tau}\}, \quad i = 1, \ldots, N, \tag{8}$$

where $\boldsymbol{\tau} \in \mathbb{R}^5$ collects the per-metric thresholds and $h = 1$ iff $\pi(\boldsymbol{\theta}_i) \preceq \boldsymbol{\tau}$ (all specs met), and $h = 0$ otherwise. We treat (8) as the classifier from Section 3 (cf. (2)) and apply ACE via Algorithm 1. A detailed discussion on the tuning application is given in Guerrero et al. (2026).

5.2. Experimental Setting

Process & Controller. We study shell–and–tube heat–exchange temperature control in which steam flow is manipulated to regulate outlet–water temperature. The controller is a classical cascaded lead–lag compensator $F(s)$; the plant dynamics $G(s)$ are approximated by a standard second-order plus dead-time (SOPDT) model (Padhee, 2014):

$$G(s) = \frac{5\,e^{-1s}}{90s^2 + 33s + 1}, \qquad F(s) = K\,\frac{T_d s + 1}{\beta T_d s + 1}\,\frac{T_i s + 1}{T_i s + \gamma}, \quad 0 < \beta, \gamma < 1.$$

The tunable parameter vector is $\boldsymbol{\theta} = (K,\, T_d,\, \beta,\, T_i,\, \gamma)$. For ACE, $G(s)$ is unknown—the plant is treated as a black box (cf. (2)).

Spec thresholds. By Eq. (8), we set $\boldsymbol{\tau} = \left(\tau_e, \tau_{\text{rise}}, \tau_{\text{settle}}, \tau_{\text{OS}}, \tau_{\|u\|_{\text{rms}}}\right) = (0.01,\ 20\text{s},\ 50\text{s}, 20\%, 2)$.

Configurations. We initialize with 30 historical controllers identified from logged batches $(\boldsymbol{r}^{(i)}_{1:N_i}, \boldsymbol{u}^{(i)}_{1:N_i}, \boldsymbol{y}^{(i)}_{1:N_i})$. For each batch, the parameters are fitted via output–error NLS (Söderström and Stoica, 1989, Ch. 7) (see Appendix G). We add Gaussian noise σ_y^2=0.01 and use a unit step (r_{step}=1). Plant and controller are discretized via the bilinear transform with sampling period $T_s = 0.1\,\text{s}$.

5.3. Results

We perform 100 independent ACE runs, with different random seeds, from the same baseline $\boldsymbol{\theta}_0 = [4.7770, 2.4354, 0.0132, 27.5408, 0.3100]$. Figure 1 shows the component-wise deviations $\Delta = \boldsymbol{\theta}_{\text{cand}} - \boldsymbol{\theta}_0$, with $\boldsymbol{\theta}_{\text{cand}}$ collecting *all* candidate evaluations produced during the search—not only the final CFEs. The boxes show the interquartile range (25–75%), the orange line is the median, whiskers cover the non-outlier range, and circles are outliers. Across parameters—except T_I—typical magnitudes stay small ($|\Delta| \lesssim 0.45$), with only modest positive drifts in T_D and γ. The median trend indicates a mild increase in T_I, consistent with using a gentle lag to improve low-frequency behavior while keeping the phase-margin impact limited (per lead–lag guidelines (Ogata, 2010, Ch. 6)). Overall, the distributions are tightly centered around zero, showing that ACE searches in a compact neighborhood and proposes small, local retunings; on average, $\sim$16 proposals per run are needed before a CFE is found.

Figure 2 compares the baseline $\boldsymbol{\theta}$ (red) against a representative CFE $[3.4294, 2.2920, 0.0428, 28.9531, 0.1605]$ (blue). Before tuning, four specs were violated: $e_{\text{ss}} = 0.0128$, $t_{\text{settle}} \approx 797\,\text{s}$, OS $= 33\%$, and $\|u\|_{\text{rms}} = 2.8355$. After ACE, they improve to $e_{\text{ss}} = 0.0093$, $t_{\text{settle}} \approx 8.0\,\text{s}$, OS $= 18.56\%$, and $\|u\|_{\text{rms}} = 0.9620$, showing that a few targeted tweaks bring the loop in-spec; the baseline settling time is not visible in Figure 2, as it is caused by small late excursions outside the $\pm 5\%$ band.

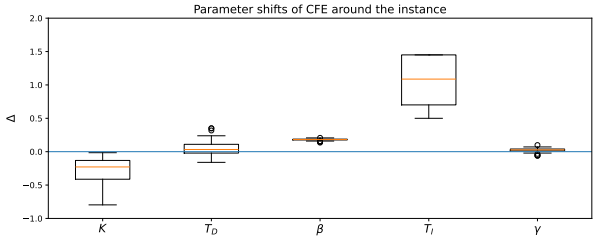

Figure 1: Box plot of component-wise shifts $\boldsymbol{\Delta} = \boldsymbol{\theta}_{\text{cand}} - \boldsymbol{\theta}_0$ across 100 ACE runs.

6. Qualitative Analysis

We qualitatively analyze ACE's outputs in low- and high-dimensional settings, as visual inspection is key to understanding CFEs.

6.1. Low-Dimensional Visualization

Synthetic Dataset. We compare ACE to Growing Spheres (GS) (Laugel et al., 2018) on a 2D `make_moons` example (Pedregosa et al., 2011). The black-box is an RBF-SVC with $\gamma = 1$, where $\gamma = 1/(2\sigma^2)$ controls the bias–variance trade-off (Smola and Schölkopf, 2002, p. 47).

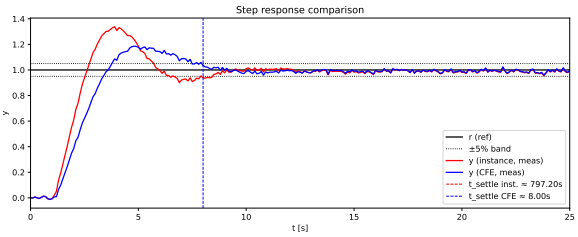

Figure 2: Step responses: baseline $\boldsymbol{\theta}_0$ (red) vs. one representative CFE (blue).

Figure 3 compares ACE (first figure) and GS (second figure) on `make_moons`. The black dashed 0.5 contour approximates the black-box boundary (red dotted). The instance $\tilde{x} = [0.55, 0.25]^T$ is green and CFEs are yellow. With $n_0 = 4$, ACE reaches $[1.23, 0.25]^T$ in 14 queries vs. GS's $[1.14, 0.20]^T$ in 501; distances are 0.68 vs. 0.592. ACE moves on one axis $[+0.68, 0]$ while GS perturbs two $[+0.59, -0.05]$, yielding a sparser, simpler change despite a slightly larger distance.

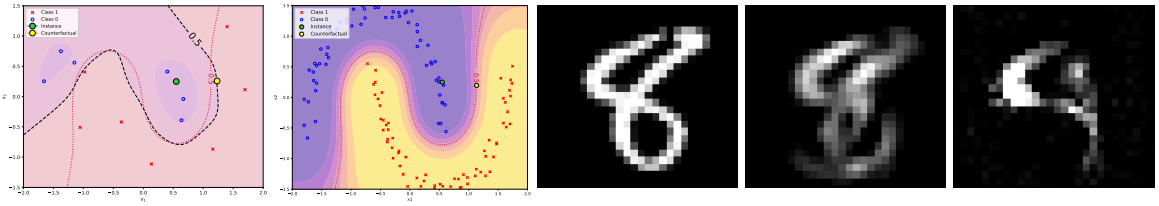

Figure 3: First: ACE on Make Moons. Second: GS on Make Moons. Third-Fifth: MNIST results (original, ACE CFE, and OmniXAI CFE).

6.2. High-Dimensional Visualization

MNIST. We use 28×28 grayscale digits (784-d) and focus on number 8 vs. 9. The task is to find the minimal ℓ_2 perturbation to turning an 8 into a 9. We compare ACE to OmniXAI (Wachter et al., 2017) a library that generates counterfactual examples using a CNN trained specifically for this task. The black-box model is a CNN trained for binary classification between 8 and 9.

In Fig. 3 (3 rightmost figures)—Class 0 corresponds to digit 8 and Class 1 to digit 9—ACE (with $n_0 = 50$) finds a valid CFE after only 9 extra queries (59 total), yielding a digit with an opened lower loop that visually resembles a 9 and is confidently classified as Class 1 by both the surrogate and black-box models. OmniXAI fails at 50 queries but succeeds after retraining on the same 59 points. ACE attains $\ell_2 = 5.47$ with posterior 55.19% (near the 50% boundary), vs. OmniXAI's $\ell_2 = 7.44$ and 64.93%. Overall, ACE demonstrates superior sample efficiency and plausibility, yielding concise and actionable explanations.

7. Conclusion

We propose the Adaptive sampling for Counterfactual Explanations (ACE) algorithm, designed to generate reliable and precise CFEs in a sample-efficient manner. Across real-world and synthetic datasets, ACE outperforms state-of-the-art methods by requiring fewer black-box evaluations while producing meaningful explanations. ACE further demonstrates effectiveness in controller-tuning applications, indicating deployability beyond standard machine learning benchmarks. Future work includes multi-objective diversity schemes and theoretical study of ACE's properties.

References

Jean Pierre Allamaa, Panagiotis Patrinos, and Tong Duy Son. ExAMPC: The data-driven explainable and approximate NMPC with physical insights. *arXiv preprint arXiv:2503.00654*, 2025.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

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*, page 93–104, 2000.

Charles George Broyden. Quasi-newton methods. In W. Murray, editor, *Numerical Methods for Unconstrained Optimization*, pages 87–106. Academic Press, 1972.

George Casella and Roger L. Berger. *Statistical Inference, 2nd Ed.* CRC Press, 2024.

Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. In *Parallel Problem Solving from Nature – PPSN XVI. PPSN 2020*, pages 448–469, 2020.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017. arXiv preprint 1702.08608, available at https://arxiv.org/abs/1702.08608.

Content European Commission, Directorate-General for Communications Networks and Technology. Ethics Guidelines for Trustworthy AI. Technical report, Publications Office, 2019. URL https://data.europa.eu/doi/10.2759/346720.

Christos Fragkathoulas, Vasiliki Papanikou, Evaggelia Pitoura, and Evimaria Terzi. Fgce: Feasible group counterfactual explanations for auditing fairness, 2024. arXiv preprint 2410.22591, available at https://arxiv.org/abs/2410.22591.

Jacob R. Gardner, Matt J. Kusner, Zhixiang Xu, Kilian Q. Weinberger, and John P. Cunningham. Bayesian optimization with inequality constraints. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 937–945, 2014.

Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89, 2018.

Margarita A. Guerrero, Rodrigo A. González, and Cristian R. Rojas. Sample-efficient counterfactual tuning for compressor pressure control. *Journal of Process Control*, 162:103723, 2026.

Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Min. Knowl. Discov.*, 38:2770–2824, 2022.

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 895–905, 2020.

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*, page 353–362, 2021.

Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

Thibault Laugel. Growing spheres: A python library for adversarial example generation, 2018. Available at https://github.com/thibaultlaugel/growingspheres.

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Mateusz Detyniecki. Comparison-based inverse classification for interpretability in machine learning. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations. IPMU 2018*, pages 100–111, 2018.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

Ana Lucic, Maartje A. Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 4499–4511, 2022.

David G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*. Springer, 2008.

Scott M. Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)*, pages 4765–4774, 2017.

Jonas Mockus. *The Bayesian Approach to Global Optimization*. Springer, 1989.

Christoph Molnar. *Interpretable machine learning: A guide for making Black Box models explainable*. Independently published, 2022.

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*, page 607–617, 2020.

Katsuhiko Ogata. *Modern control engineering*. Prentice hall, 2010.

Subhransu Padhee. Controller design for temperature control of heat exchanger system: simulation studies. *WSEAS Transactions on Systems and Control*, 9(1):485–491, 2014.

Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference 2020*, page 3126–3132, 2020.

Martin Pawelczyk, Sascha Bielawski, Johannes van den Heuvel, Tobias Richter, and Gjergji Kasneci. Carla: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, pages 1–15, 2021.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12: 2825–2830, 2011.

Victor Picheny, Robert B. Gramacy, Stefan Wild, and Sébastien Le Digabel. Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian. In *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)*, pages 1443–1451, 2016.

Federico Porcari, Donatello Materassi, and Simone Formentin. eXplainable AI for data driven control: an inverse optimal control approach. *arXiv preprint arXiv:2504.11446*, 2025.

Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. Face: Feasible and actionable counterfactual explanations. In *Proceedings of the 2020 AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, pages 344–350, 2020.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Lyle Regenwetter, Yazan Abu Obaideh, and Faez Ahmed. Mcd: A model-agnostic counterfactual search method for multi-modal design modifications. *Journal of Mechanical Design*, 147:1–18, 2024.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016)*, pages 1135–1144, 2016.

Piotr Romashov, Martin Gjoreski, Kacper Sokol, Maria Vanina Martinez, and Marc Langheinrich. Baycon: Model-agnostic bayesian counterfactual generator. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 740–746, 2022.

Shpresim Sadiku, Moritz Wagner, Sai Ganesh Nagarajan, and Sebastian Pokutta. S-cfe: Simple counterfactual explanations, 2025. arXiv preprint 2410.15723, available at https://arxiv.org/abs/2410.15723.

Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer, 2019.

Alexander J. Smola and Bernhard Schölkopf. *Learning with Lernels*. MIT Press, 2002.

Ilya M. Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7:86–112, 1967.

Torsten Söderström and Petre Stoica. *System identification*. Prentice-Hall, 1989.

Thomas Spooner, Danial Dervovic, Jason Long, Jon Shepard, Jiahao Chen, and Daniele Magazzeni. Counterfactual explanations for arbitrary regression models, 2021. arXiv preprint 2106.15212, available at https://arxiv.org/abs/2106.15212.

Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan Hines, John Dickerson, and Chirag Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. *ACM Computing Surveys*, 56:Article 312, 42 pages, 2024.

Vy Vo, Trung Le, Van Nguyen, He Zhao, Edwin V. Bonilla, Gholamreza Haffari, and Dinh Phung. Feature-based learning for diverse and privacy-preserving counterfactual explanations. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 2211–2222, 2023.

Paul Voigt and Axel von dem Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer, 2017.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31: 841–887, 2017.

James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for Bayesian optimization. In *Advances in Neural Information Processing Systems*, pages 9905–9916, 2018.

Wenzhuo Yang, Hung Le, Tanmay Laud, Silvio Savarese, and Steven C. H. Hoi. Omnixai: A library for explainable ai, 2022. arXiv preprint 2206.01612, available at https://arxiv.org/abs/2206.01612.

Appendix A. Desiderata for Counterfactual Explanations

As identified in prior literature (Vo et al., 2023; Guidotti, 2022), high-quality counterfactual explanations (CFEs) should satisfy the following key properties, which are effectively addressed by our proposed method:

Validity. *The counterfactual must change the predicted label.* ACE guarantees validity by explicitly searching for counterfactuals x such that the predicted outcome differs from the original instance $\tilde{x}$, i.e., $h(x) \neq h(\tilde{x})$.

Sparsity. *Minimal number of features should be altered.* To encourage sparse solutions, ACE incorporates an ℓ_1-norm penalty in its cost function, promoting counterfactuals that modify as few features as possible.

Proximity. *Counterfactuals should be close to the original input.* ACE measures proximity via a distance metric $d(x, \tilde{x})$, typically the Euclidean norm, though other metrics can be used depending on the data characteristics. This ensures that explanations remain within a small, interpretable neighborhood ε around $\tilde{x}$, i.e., $d(x, \tilde{x}) < \varepsilon$.

Actionability. *Only mutable features should be changed.* ACE supports user-defined feature constraints and restricts modifications to actionable features, leaving immutable attributes (e.g., age or gender) unchanged throughout the optimization process to ensure the generation of feasible CFEs.

Diversity. *Providing multiple distinct counterfactuals improves user choice.* ACE combines Monte Carlo sampling with GP uncertainty to explore the decision boundary globally, while prioritizing diverse candidates near $\tilde{x}$. This dual mechanism enables the generation of multiple, semantically distinct yet plausible CFEs.

Plausibility. *Counterfactuals should respect feature constraints and data distribution.* ACE enforces domain constraints and avoids unrealistic combinations by sampling within the input domain and promoting CFEs in high-density regions with respect to the training data.

Scalability. *Efficient generation across multiple instances.* Thanks to its Bayesian surrogate model, ACE reuses learned structures across similar queries and scales to both low- and high-dimensional datasets, supporting simultaneous generation of multiple CFEs.

Appendix B. Extended Cost Function

Plausibility Term. To ensure closeness to the data manifold, we penalize counterfactuals that lie in low-density regions relative to the collected data $\boldsymbol{X}$ using the *Local Outlier Factor (LOF)* (Breunig et al., 2000). LOF quantifies the degree to which a point is isolated from its neighbors, with scores typically interpreted as

$$\mathrm{LOF}_k(\boldsymbol{x}) := \frac{1}{|N_k(\boldsymbol{x})|} \sum_{\boldsymbol{z} \in N_k(\boldsymbol{x})} \frac{\mathrm{lrd}(\boldsymbol{z})}{\mathrm{lrd}(\boldsymbol{x})}, \tag{1}$$

where $N_k(\boldsymbol{x})$ is the k-nearest neighborhood of $\boldsymbol{x}$, and $\mathrm{lrd}(\boldsymbol{z})$ denotes the local reachability density. In our implementation, we adopt `scikit-learn`'s `LocalOutlierFactor` with `novelty = True`, which outputs negative LOF scores. A point $\boldsymbol{x}$ is considered an *inlier* if its LOF score exceeds a threshold τ (in ACE, we use the default threshold, typically $\tau = -1.5$); otherwise, it is deemed implausible. Thus, we define the penalty or plausibility term $l(\boldsymbol{x}; \boldsymbol{X})$ as

$$l(\boldsymbol{x}; \boldsymbol{X}) := \begin{cases} 0, & \text{if } \mathrm{LOF}_k(\boldsymbol{x}) > \tau \text{ (inlier)}, \\ \infty, & \text{otherwise (outlier)}, \end{cases}$$

which effectively acts as a hard constraint that discards implausible candidates.

Appendix C. Class-1 Posterior Calculations

For classification problems, we aim to estimate the probability $p(t = 1|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*)$, where $l \in \{0, 1\}$ is the label corresponding to input $\boldsymbol{x} \in \mathcal{X}$, and $\boldsymbol{t}_* \in \{0, 1\}^n$ is a vector of labels at the inputs in $\boldsymbol{x}_* \in \mathcal{X}^n$. This probability will be approximated as

$$p(t = 1|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*) = \int p(t = 1|a)p(a|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*)da,$$

where $a = \hat{f}(\boldsymbol{x})$, $p(t = 1|a) = \sigma(a) := 1/(1 + \exp(-a))$ is the logistic sigmoid function, and $\hat{f}$ is a Gaussian process. We are actually interested only on $p(a|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*)$, given by

$$p(a|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*) = \int p(a|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{a}_*)p(\boldsymbol{a}_*|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*)d\boldsymbol{a}_*,$$

where $p(a|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{a}_*)$ is a Gaussian distribution given by

$$p(a|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{a}_*) = \mathcal{N}\big(a; \boldsymbol{k}_n^T \boldsymbol{K}^{-1} \boldsymbol{a}_*, \kappa(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_n^T \boldsymbol{K}^{-1} \boldsymbol{k}_n\big).$$

The density $p(\boldsymbol{a}_*|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*)$, on the other hand, can be replaced by the Laplace approximation (Rasmussen and Williams, 2006, Sec. 3.4)

$$p(\boldsymbol{a}_*|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*) \approx \mathcal{N}\left(\boldsymbol{a}_*; \hat{\boldsymbol{a}},\ (\boldsymbol{W}(\hat{\boldsymbol{a}}) + \boldsymbol{K}^{-1})^{-1}\right),$$

where $\boldsymbol{W}(\boldsymbol{a}) = \text{diag}(\sigma(a_i)[1-\sigma(a_i)])$, and $\hat{\boldsymbol{a}}$ is obtained by iterating until convergence the equation

$$\hat{\boldsymbol{a}}_{m+1} = \boldsymbol{K}(\boldsymbol{I} + \boldsymbol{W}(\hat{\boldsymbol{a}}_m)\boldsymbol{K})^{-1}(\boldsymbol{t}_* - \boldsymbol{\sigma}(\hat{\boldsymbol{a}}_m) + \boldsymbol{W}(\hat{\boldsymbol{a}}_m)\hat{\boldsymbol{a}}_m),$$

where $\boldsymbol{\sigma}$ consists in the entry-wise application of σ. Finally,

$$p(a|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*) \approx \mathcal{N}\left(a;\ \mu_a,\ \sigma_a^2\right),$$

where μ_a and σ_a^2 are the logit mean and the logit variance, respectively, defined as

$$\mu_a = \boldsymbol{k}_n^T(\boldsymbol{t}_* - \boldsymbol{\sigma}(\hat{\boldsymbol{a}})), \tag{2}$$

$$\sigma_a^2 = \kappa(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_n^T(\boldsymbol{W}(\hat{\boldsymbol{a}})^{-1} + \boldsymbol{K})^{-1}\boldsymbol{k}_n. \tag{3}$$

The Class 1 probability at input $\boldsymbol{x}$ given $\boldsymbol{x_*}$ and $\boldsymbol{t_*}$ is calculated using the inverse probit function $\sigma(a) \simeq \Phi(\lambda a)$, obtaining the approximate predictive distribution of the form

$$p(t = 1|\boldsymbol{x}, \boldsymbol{x}_*, \boldsymbol{t}_*) = \sigma\big(\mu_a\big(1 + \frac{\pi\sigma_a^2}{8}\big)^{-1/2}\big).$$

The posterior mean of the GPC in the logit space, μ_a, gives us the expected value of the latent function at point $\boldsymbol{x}$. By applying the sigmoid function, we transform it into the probability space. This transformed value, $\mu = \sigma(\mu_a)$ represents the posterior mean probability. Finally, the variance in the probability space is given by

$$\sigma^2 = \sigma_a^2\, \mu^2(1 - \mu)^2,$$

which is obtained using the delta method (Casella and Berger, 2024, p. 240), as discussed in the next subsection.

Appendix D. Delta Method for $\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x}^*)$

First, let $\mu(\boldsymbol{x})$ and $\sigma^2(\boldsymbol{x})$ be the posterior mean and variance of the GP model at point $\boldsymbol{x}$, and similarly, let $\mu(\boldsymbol{x}^*)$ and $\sigma^2(\boldsymbol{x}^*)$ be the mean and variance at point x^*. The covariance between $\boldsymbol{x}$ and $\boldsymbol{x}^*$ is denoted by $\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x}^*)$. We can represent the joint distribution of $\hat{f}(\boldsymbol{x})$ and $\hat{f}(\boldsymbol{x}^*)$ as

$$\begin{bmatrix} \hat{f}(\boldsymbol{x}) \\ \hat{f}(\boldsymbol{x}^*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(\boldsymbol{x}) \\ \mu(\boldsymbol{x}^*) \end{bmatrix}, \begin{bmatrix} \sigma^2(\boldsymbol{x}) & \mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x}^*) \\ \mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x}^*) & \sigma^2(\boldsymbol{x}^*) \end{bmatrix} \right).$$

To generate correlated samples, we perform the Cholesky decomposition on the covariance matrix

$$\begin{bmatrix} \sigma^2(\boldsymbol{x}) & \mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x}^*) \\ \mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x}^*) & \sigma^2(\boldsymbol{x}^*) \end{bmatrix} = LL^T,$$

where L is a lower triangular matrix.

To calculate the covariance between x and x^*, we apply the delta method which uses the first-order Taylor expansion to approximate the expectation of a function of random variables, in particular, of $g(f(x))$ and $g(f(x^*))$, where g is a nonlinear function. Using the first-order Taylor expansion around the logit means $\mu_a(x)$ and $\mu_a(x^*)$ (cf. (2)) yields

$$\begin{aligned} g(f(\boldsymbol{x})) &\approx g(\mu_a(\boldsymbol{x})) + g'(\mu_a(\boldsymbol{x}))(\hat{f}(\boldsymbol{x}) - \mu_a(\boldsymbol{x})), \\ g(f(\boldsymbol{x}^*)) &\approx g(\mu_a(\boldsymbol{x}^*)) + g'(\mu_a(\boldsymbol{x}^*))(\hat{f}(\boldsymbol{x}^*) - \mu_a(\boldsymbol{x}^*)). \end{aligned}$$

The covariance is then approximated by

$$\begin{aligned} &\mathrm{Cov}(g(\hat{f}(\boldsymbol{x})), g(\hat{f}(\boldsymbol{x}^*))) \\ &\quad \approx \mathrm{Cov}\big(g(\mu_a(\boldsymbol{x})) + g'(\mu_a(\boldsymbol{x}))(\hat{f}(\boldsymbol{x}) - \mu_a(\boldsymbol{x})), \\ &\qquad\qquad g(\mu_a(\boldsymbol{x}^*)) + g'(\mu_a(\boldsymbol{x}^*))(\hat{f}(\boldsymbol{x}^*) - \mu_a(\boldsymbol{x}^*))\big) \\ &\quad = g'(\mu_a(\boldsymbol{x}))g'(\mu_a(\boldsymbol{x}^*))\,\mathrm{Cov}(\hat{f}(\boldsymbol{x}), \hat{f}(\boldsymbol{x}^*)), \end{aligned}$$

where $\mathrm{Cov}((\hat{f}(\boldsymbol{x}), \hat{f}(\boldsymbol{x}^*))$ is calculated using (3) as

$$\begin{aligned} &\mathrm{Cov}((\hat{f}(\boldsymbol{x}), \hat{f}(\boldsymbol{x}^*)) \\ &= \kappa(\boldsymbol{x}, \boldsymbol{x}^*) - \kappa(\boldsymbol{x}_{1:n}, \boldsymbol{x})^T(\boldsymbol{W}(\hat{\boldsymbol{a}})^{-1} + \boldsymbol{K})^{-1}\kappa(\boldsymbol{x}_{1:n}, \boldsymbol{x}). \end{aligned}$$

For GPC, $g(z)$ is represented by the sigmoid function σ, converting logits into probabilities. Consequently, the derivatives are given by $g'(z) = \sigma(z)(1 - \sigma(z))$. Finally,

$$\begin{aligned} &\mathrm{Cov}(\sigma(\hat{f}(\boldsymbol{x})), \sigma(\hat{f}(\boldsymbol{x}^*))) \\ &\approx \mu_x(1 - \mu_x)\mu_{x^*}(1 - \mu_{x^*})\,\mathrm{Cov}(\hat{f}(\boldsymbol{x}), \hat{f}(\boldsymbol{x}^*)) \\ &\approx \mathrm{Cov}(\boldsymbol{x}, \boldsymbol{x}^*). \end{aligned}$$

We then draw samples from standard normal distributions $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and transform them into correlated samples:

$$\begin{bmatrix} \hat{f}(\boldsymbol{x}) \\ \hat{f}(\boldsymbol{x}^*) \end{bmatrix} = \begin{bmatrix} \mu(\boldsymbol{x}) \\ \mu(\boldsymbol{x}^*) \end{bmatrix} + L\mathbf{z}.$$

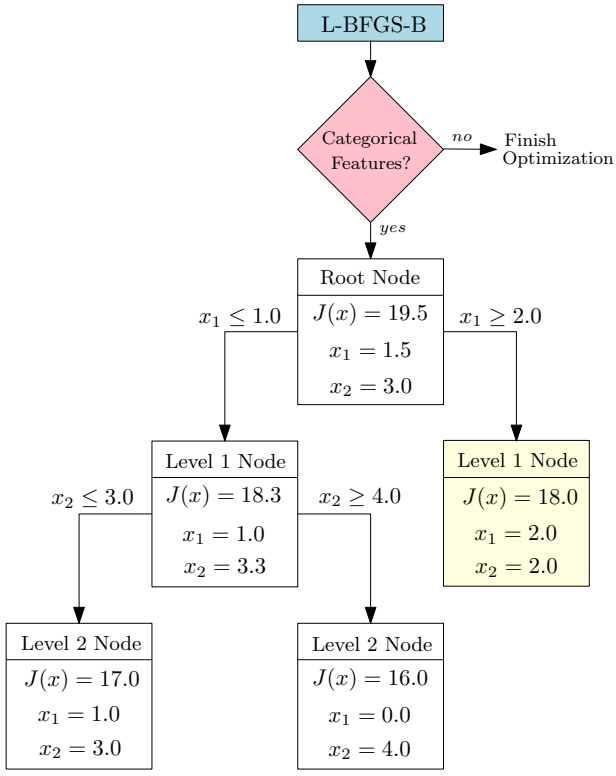

Figure 1: Branch and Bound method with L-BFGS-B Initialization.

Appendix E. Branch and Bound Example

The ACE algorithm employs a hybrid optimization strategy to handle both continuous and categorical variables. For continuous features, we apply the quasi-Newton L-BFGS-B algorithm to maximize the acquisition function, whereas for categorical variables, which are inherently discrete, ACE employs a Branch and Bound (B&B) strategy.

Categorical features are encoded using ordinal or label encoding (Pedregosa et al., 2011), depending on whether they exhibit a natural order. Starting from the solution obtained via continuous optimization (the *root node*), the B&B method systematically partitions the categorical space into smaller subproblems by introducing integer constraints. At each branch, the algorithm runs L-BFGS-B over the remaining continuous variables while fixing categorical ones, selecting the candidate that maximizes the acquisition function. For a visual example, see Appendix E.

Figure 1 illustrates the Branch and Bound algorithm for two categorical variables. Each node represents a constrained maximization, and the yellow-highlighted box corresponds to the best candidate found ($J(x) = 18.0$, $x_1 = 2.0$, $x_2 = 2.0$). This structured refinement ensures a principled and efficient exploration of the solution space, enabling the identification of near-global optima in problems involving discrete variables and non-convex structures, while effectively pruning suboptimal branches.

Appendix F. Datasets

The quantitative evaluation is conducted on eight real-world classification datasets sourced from OpenML (Vanschoren et al., 2014), Kaggle (Kaggle, 2010), and the UCI ML Repository (Dua and Graff, 2019). The selected datasets largely align with those used in the original evaluations of the compared methods. Growing Spheres was evaluated on MNIST (in the original paper) and on Make Moons (illustrative examples provided in its public repository). For the eight benchmark datasets, we included those used in the MOC and BayCon studies, which also compared against each other, and added Growing Spheres to enable both qualitative and quantitative comparisons. For OmniXAI, the MNIST dataset was included to complement the visual comparison with Growing Spheres.

The websites corresponding to the datasets used in Section 4 are the following: Diabetes · Breast · Blood · KC2 · Tic-Tac-Toe · Nursery · CMC · German Credit

F.1. Hyperparameter Search

An exhaustive grid search is performed over the ranges in Table 1. Continuous ranges ("to") are sampled at regular intervals, and bracketed values denote discrete candidate sets. Final values are selected from the subrange where further changes have negligible impact on the generated CFE, ensuring robustness and efficiency.

Table 1: Hyperparameter ranges explored.

Name	**Symbol**	**Range Tried**
Initial Penalty	λ_0	2 to 12
Kernel (ν)	κ	Matérn $[\frac{1}{2}, \frac{3}{2}, \frac{5}{2}]$
Monte-Carlo Samples	MC	800 to 2000
Penalty Growth	p	1.1 to 1.9
Sobol Samples	SS	1000 to 10000
Sparsity Trade-off	β	$[0.1, 1, 3, 5, 7, 10, 100]$

Notably, a key design choice is the selection of the kernel function for the Gaussian Process surrogate, which fundamentally shapes the quality of the acquisition strategy in Bayesian optimization. We employ the Matérn 5/2 kernel, which models C^2 functions—unlike the RBF kernel, which assumes infinite differentiability, or the linear kernel, which fails to capture non-linear structure—making it better suited to model complex and irregular decision boundaries in high-dimensional settings (Rasmussen and Williams, 2006, Sec. 4.2.1).

As noted above, we chose hyperparameters so that small perturbations would not materially change the resulting CFE, ensuring stable convergence within a well-defined region; the final values used in our experiments are reported in Table 2. The algorithm starts with an initial penalty value, λ_0, in the vicinity of 1—set to 10 in our runs—ensuring that the search begins near the instance to be explained and helps the algorithm efficiently find a nearby minimizer. After each acquisition function maximization, λ is scaled by $p = 1.5$, gradually increasing to guide the search toward the decision boundary. Sobol sampling ($SS = 8000$) is utilized to densely cover high-dimensional spaces, ensuring thorough exploration across all features in the dataset, while Monte Carlo sampling ($MC = 1000$) provides an efficient approximation of the Expected Improvement (EI) integral, balancing computational cost and precision.

F.2. Computing Infrastructure

All experiments are conducted on a system with an **AMD Ryzen 7 4800HS CPU @ 2.90 GHz**, **16 GB RAM**, running **Windows 11 Home Single Language**, with computation performed on CPU only and no

Table 2: Summary of Hyperparameters.

Name	Symbol	Value
Initial Penalty	λ_0	10
Maximum Penalty	λ_{max}	1e15
Kernel	κ	Matern $\frac{5}{2}$
Monte-Carlo Samples	MC	1000
Convergence Tolerance	ϵ	0.001
Penalty Growth	p	1.5
Sobol Samples	SS	8000
Sparsity trade-off parameter	β	5

Table 3: Fixed test for continuous datasets with 100 tested points per experiment. The best Score CFE values are highlighted.

Dataset	Method	Label 1						Label 2					
		$h_{\#}$ (std)	d_2 (std)	g_1 (std)	$\alpha(x)$ (std)	$\mathcal{V}$ (std)	$\mathcal{S}$	$h_{\#}$ (std)	d_2 (std)	g_1 (std)	$\alpha(x)$ (std)	$\mathcal{V}$ (std)	$\mathcal{S}$
Diabetes	ACE	71 (16)	34.09 (14.3)	49.57 (22.34)	1 (0.06)	1 (0)	**0.24**	70 (15)	37.10 (6.88)	70.70 (13.33)	0.97 (0.04)	1 (0)	**0.17**
	BayCon	1211 (110)	37.13 (9.23)	46.87 (12.19)	0.25 (0.08)	1 (0)	0.36	1458 (225)	52.92 (10.5)	77.56 (14.59)	0.24 (0.10)	1 (0)	0.34
	MOC	2038 (1065)	58.95 (18.46)	62.91 (16.77)	0.84 (0.19)	0.98 (0.14)	0.45	2269 (1247)	87.00 (70.22)	98.66 (82.49)	0.55 (0.16)	0.98 (0.14)	0.47
	CARLA$_{GS}$	28013 (4455)	5.51 (0.90)	12.46 (2.24)	0.78 (0.03)	1 (0)	0.38	68937 (7978)	13.70 (1.60)	30.20 (4.30)	0.80 (0)	1 (0)	0.38
Breast	ACE	60 (16)	9.33 (1.05)	27.89 (2.87)	1 (0.10)	1 (0)	**0.2**	78 (27)	15.05 (0.90)	40.03 (3.24)	1 (0.01)	1 (0)	0.34
	BayCon	3567 (1058)	9.60 (1.25)	21.12 (1.90)	0.31 (0.03)	0.03 (0.20)	0.35	2960 (488)	13.81 (0.76)	33.30 (4.61)	0.29 (0.03)	0.03 (0.20)	0.35
	MOC	845 (748)	10.28 (1.19)	20.81 (3.11)	0.09 (0.06)	0.97 (0.17)	0.37	372 (507)	15.65 (0.78)	29.97 (3.71)	0.52 (0.19)	0.99 (0.10)	**0.31**
	CARLA$_{GS}$	40883 (3545)	9.08 (0.71)	21.41 (2.25)	0.77 (0.03)	1 (0)	0.39	65703 (5583)	13.03 (1.12)	32.88 (3.53)	0.79 (0.02)	1 (0)	0.42
KC2	ACE	56 (12)	626.76 (170.35)	1436.83 (328.41)	0.90 (0.22)	1 (0)	**0.02**	57 (13)	662.05 (170.25)	1529.80 (322.74)	0.89 (0.23)	1 (0)	**0.12**
	BayCon	3715 (1498)	2448.56 (2672.30)	2843.70 (3083.79)	0.13 (0.15)	0.98 (0.14)	0.14	3575 (1177)	3441.04 (2359.45)	3988.32 (2674.31)	0.05 (0.12)	1 (0)	0.53
	MOC	457 (672)	62570.15 (165673.31)	64514.02 (167510.11)	0.02 (0.08)	0.19 (0.39)	0.62	1277 (1093)	1652.42 (1597.79)	2069.07 (1808.94)	0.09 (0.19)	0.13 (0.34)	0.42
	CARLA$_{GS}$	139093 (22089)	27.73 (4.42)	101.57 (18.79)	0.17 (0.02)	1 (0)	0.45	102423 (42302)	20.39 (8.45)	73.65 (30.27)	0.16 (0.02)	1 (0)	0.46
Blood	ACE	54 (15)	2035.74 (596.52)	2072.76 (595.77)	0.99 (0.12)	1 (0)	**0.14**	57 (15)	78.15 (207.30)	70.63 (207.07)	1 (0.02)	1 (0)	0.38
	BayCon	919 (159)	2017.03 (104.66)	2046.13 (107.42)	0 (0.03)	1 (0)	0.31	772 (105)	75.42 (8.98)	78.77 (10.57)	0 (0.01)	1 (0)	0.6
	MOC	1790 (970)	5999.42 (1407.57)	6044.28 (1418.61)	0.05 (0.20)	0.85 (0.36)	0.63	632 (396)	8.12 (43.56)	8.27 (43.94)	0.99 (0.05)	0.66 (0.47)	**0.15**
	CARLA$_{GS}$	6313 (1239)	1.15 (0.23)	1.91 (0.42)	0.64 (0.16)	1 (0)	0.4	3083 (271)	0.55 (0.06)	0.76 (0.16)	0.8 (0.03)	1 (0)	0.38

GPU acceleration (integrated or discrete) used. The implementation is in **Python 3.12.7** with the following key libraries: NumPy (v1.26.4), SciPy (v1.13.1), scikit-learn (v1.5.1), pandas (v2.2.2), matplotlib (v3.9.2), scikit-image (v0.24.0), and threadpoolctl (v3.5.0).

F.3. Quantitative Evaluation - Real-World datasets

Complete results for all eight real-world datasets—using two randomly selected labels per dataset—are reported in Tables 3, 4, 5 and 6. Distances for categorical features use consistent encodings (label or ordinal), and identical random seeds ensure reproducibility. Non-actionable features—such as age or gender— are held fixed throughout the optimization to guarante feasibility and interpretability

ACE is initialized with $n_0 = 30$ points for all datasets, except for the low-dimensional Blood Test dataset (4 features), where $n_0 = 15$ is used. These initial samples are included in the total evaluation count. Across all eight datasets, ACE consistently outperforms competing methods in terms of the number of black-box evaluations ($h_{\#}$), successfully generating counterfactuals in every experiment.

Table 4: Fixed test for heterogeneous datasets with 100 tested points per experiment. The best Score CFE values are highlighted.

Dataset	Method	Label 1						Label 2					
		$h_{\#}$ (std)	d_2 (std)	g_1 (std)	$\alpha(x)$ (std)	$\mathcal{V}$ (std)	$\mathcal{S}$	$h_{\#}$ (std)	d_2 (std)	g_1 (std)	$\alpha(x)$ (std)	$\mathcal{V}$ (std)	$\mathcal{S}$
CMC	ACE	65 (6)	1 (0)	1 (0)	1 (0.01)	1 (0)	**0**	63 (7)	1 (0)	1 (0)	1 (0.01)	1 (0)	**0**
	BayCon	1021 (102)	1 (0)	1 (0)	0.33 (0.02)	1 (0)	0.43	925 (46)	1 (0)	1 (0)	0.29 (0.04)	1 (0)	0.41
	MOC	590 (212)	1 (0.1)	1 (0.2)	0.9 (0.1)	1 (0)	0.2	528 (184)	1 (0)	1 (0)	0.7 (0)	1 (0)	0.21
	CARLA$_{GS}$	1005 (0)	3.87 (0)	7 (0)	1 (0)	1 (0)	0.74	1005 (0)	4.24 (0)	8 (0)	1 (0)	1 (0)	0.75
Nursery	ACE	57 (7)	1 (0.04)	1.01 (0.1)	1 (0)	1 (0)	**0**	60 (8)	1 (0)	1 (0)	1 (0.01)	1 (0)	**0**
	BayCon	836 (29)	1 (0)	1 (0)	0.37 (0.01)	1 (0)	0.37	822 (29)	1 (0)	1 (0)	0.37 (0.01)	1 (0)	0.36
	MOC	220 (0)	- (-)	- (-)	- (-)	0 (0)	-	220 (0)	- (-)	- (-)	- (-)	0 (0)	-
	CARLA$_{GS}$	1005 (0)	3.32 (0)	5 (0)	1 (0)	1 (0)	0.75	1005 (0)	3.32 (0)	5 (0)	1 (0)	1 (0)	0.75
German Credit	ACE	68 (26)	7.48 (9.05)	12.88 (10.44)	0.94 (0.08)	1 (0)	**0.01**	54 (10)	6.11 (2.82)	11.4 (4.67)	1 (0.01)	1 (0)	**0**
	BayCon	973 (98)	1 (0)	1 (0)	0.23 (0.05)	1 (0)	0.33	1013 (69)	1 (0)	1 (0)	0.23 (0.08)	1 (0)	0.3
	MOC	1436 (753)	4159.51 (1039.12)	4200.26 (1047.8)	0.5 (0.14)	0.35 (0.48)	0.89	1739 (702)	3993.84 (473.19)	4013.8 (472.05)	0.26 (0.06)	0.05 (0.22)	0.96
	CARLA$_{GS}$	1005 (0)	2.13 (0.27)	3.49 (0.5)	1 (0.01)	1 (0)	0.24	1005 (0)	1.41 (0)	2.01 (0.01)	0.99 (0.07)	1 (0)	0.2
Tic-Tac-Toe	ACE	67 (11)	1 (0.01)	1 (0.03)	1 (0.01)	1 (0)	**0**	61 (9)	1 (0.01)	1 (0.02)	1 (0.01)	1 (0)	**0**
	BayCon	1526 (274)	1.41 (0)	2 (0)	0.27 (0.02)	1 (0)	0.44	930 (42)	1 (0)	1 (0)	0.28 (0.02)	1 (0)	0.44
	MOC	304 (71)	1.47 (0.17)	2.19 (0.59)	0.25 (0.03)	0.21 (0.41)	0.59	289 (81)	1.73 (0)	3 (0)	0.28 (0.06)	0.02 (0.14)	0.7
	CARLA$_{GS}$	890111 (312576)	1.22 (0.25)	1.55 (0.66)	1 (0)	0.11 (0.31)	0.65	1000001 (0)	- (-)	- (-)	- (-)	0 (0)	-

Table 5: Mixed test for continuous datasets with 100 tested points. The best Score CFE values are highlighted.

Dataset	Method	$h_{\#}$ (std)	d_{2_N} (std)	g_{1_N} (std)	α (std)	$\mathcal{V}$ (std)	$\mathcal{S}$
Diabetes	ACE	72 (18)	2.46 (0.84)	2.7 (1.87)	0.98 (0.09)	1 (0)	**0.1**
	BayCon	1239 (238)	2.04 (0.97)	2.99 (1.89)	0.27 (0.09)	1 (0)	0.14
	MOC	1753 (1011)	2.09 (0.91)	2.62 (1.34)	0.58 (0.36)	0.91 (0.29)	**0.1**
	CARLA$_{GS}$	27463 (23054)	3.2 (2.66)	4.26 (3.41)	0.74 (0.08)	1 (0)	0.78
Breast	ACE	67 (22)	2.45 (1.37)	7.6 (3.95)	1 (0.01)	1 (0)	**0.16**
	BayCon	2045 (952)	2.42 (1.25)	4.25 (2.63)	0.30 (0.08)	0.60 (0.49)	**0.16**
	MOC	890 (804)	3.36 (1.19)	5.7 (2.51)	0.47 (0.33)	1 (0)	0.39
	CARLA$_{GS}$	36803 (20489)	2.62 (1.46)	6.56 (3.81)	0.76 (0.05)	1 (0)	0.54
KC2	ACE	57 (13)	4.05 (1.06)	14.37 (4.06)	0.85 (0.23)	1 (0)	**0.12**
	BayCon	3730 (1408)	2.57 (1.26)	6.43 (4.01)	0.14 (0.14)	0.97 (0.17)	0.13
	MOC	1104 (1165)	11.87 (9.46)	24.49 (22.74)	0.12 (0.28)	0.23 (0.42)	0.61
	CARLA$_{GS}$	79763 (39687)	9.59 (6.57)	15.67 (8.8)	0.18 (0.09)	1 (0)	0.72
Blood	ACE	63 (23)	1.83 (0.72)	2.9 (1.24)	0.85 (0.32)	0.99 (0.1)	0.42
	BayCon	1091 (741)	0.82 (0.65)	1.02 (0.9)	0.32 (0.1)	0.91 (0.29)	**0.16**
	MOC	504 (402)	1.8 (2.75)	2.48 (4.28)	0.37 (0.43)	0.09 (0.29)	0.56
	CARLA$_{GS}$	30633 (28509)	0.65 (0.7)	0.85 (0.87)	0.33 (0.3)	1 (0)	0.43

Appendix G. System Identification

We begin by considering a noise source solely in the plant output, as in Figure 2. An approach to estimating $\{C_i(q)\}_{i=1}^K$ is to minimize the input prediction error:

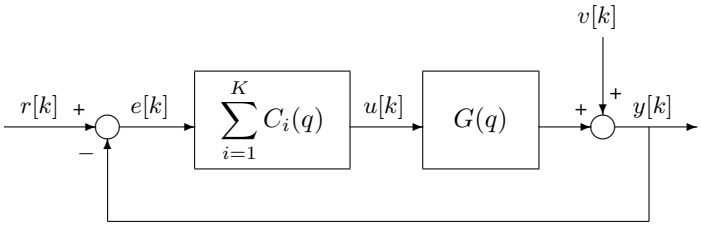

Figure 2: Closed-loop system setting, noiseless input, noisy output case.

$$\hat{\beta}_{\text{LS}} = \arg\min_{\beta} \frac{1}{N} \sum_{k=1}^{N} \left(u[k] - \sum_{i=1}^{K} C_i(q, \theta_i) e[k] \right)^2, \tag{4}$$

Table 6: Mixed test for heterogeneous datasets with 100 tested points. The best Score CFE values are highlighted.

Dataset	Method	$h_{\#}$ (std)	d_{2_N} (std)	g_{1_N} (std)	α (std)	$\mathcal{V}$ (std)	$\mathcal{S}$
CMC	ACE	70 (12)	2.12 (0.92)	2.99 (1.8)	1 (0.01)	0.97 (0.17)	**0.07**
	BayCon	1024 (241)	1.85 (0.71)	2.39 (1.24)	0.32 (0.05)	0.98 (0.14)	0.12
	MOC	332 (124)	1.57 (0.68)	1.8 (1.01)	0.8 (0.24)	0.18 (0.38)	0.13
	CARLA_{GS}	330674 (469741)	5.04 (0.9)	9.19 (1.96)	1 (0)	0.67 (0.47)	0.79
Nursery	ACE	56 (7)	1.22 (0)	1.22 (0)	1 (0.01)	1 (0)	**0**
	BayCon	840 (31)	1.22 (0)	1.22 (0)	0.37 (0)	1 (0)	0.37
	MOC	226 (10.47)	- (-)	- (-)	- (-)	0 (0)	-
	CARLA_{GS}	1005 (0)	2.09 (0.5)	3.4 (1.27)	1 (0)	1 (0)	0.75
German Credit	ACE	73 (24)	2.61 (1)	2.47 (2.24)	0.94 (0.2)	1 (0)	0.21
	BayCon	1006 (143)	2.01 (0.57)	2.11 (0.76)	0.23 (0.1)	0.95 (0.22)	**0.2**
	MOC	1502 (763)	2.83 (0.93)	4.35 (2.02)	0.27 (0.32)	0.4 (0.49)	0.71
	CARLA_{GS}	3575 (12320)	2.7 (0.83)	4.28 (1.6)	0.95 (0.16)	1 (0)	0.71

where $e[k] := y[k] - r[k]$, and where we have made explicit the dependence on θ_i of $C_i(q)$. Under technical conditions on reference and noise stationarity and closed-loop stability (Ljung, 1978), the cost being minimized in (4) converges uniformly to its expected value on a compact set of parameters β. Thus, $\hat{\beta}_{\text{LS}}$ converges with probability one as $N \to \infty$ to

$$\hat{\beta}_{\text{LS}} \xrightarrow{\text{w.p.1}} \hat{\beta}_{\text{LS}}^{\infty} = \arg\min_{\beta} \mathbb{E}\left\{ \left(u[k] - \sum_{i=1}^{K} C_i(q, \theta_i) e[k] \right)^2 \right\} \text{ as } N \to \infty. \tag{5}$$

Due to the input satisfying

$$u[k] = \sum_{i=1}^{K} C_i(q) e[k],$$

the cost function being minimized in (5) can reach 0, the global optimum, when $e[k]$ is persistently exciting of sufficient order. After some algebraic procedures, a short derivation yields

$$\sum_{i=1}^{K} [C_i(q) - C_i(q, \hat{\theta}_i^{\infty})] = 0,$$

that is, $C_i(q, \hat{\theta}_i^{\infty}) = C_i(q)$ for all $i = 1, \ldots, K$, or in other words, $\hat{\beta}_{\text{LS}} \xrightarrow{\text{w.p.1}} \beta$.

In summary, a consistent estimator of the controller transfer function can be obtained without knowledge of the plant dynamics. This requires solving the nonlinear least squares problem in (4), which can be done using, for example, the `oe` command in MATLAB, with $e[k] = r[k] - y[k]$ as input and $u[k]$ as output.

References

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*, page 93–104, 2000.

George Casella and Roger L. Berger. *Statistical Inference, 2nd Ed.* CRC Press, 2024.

Dheeru Dua and Casey Graff. Uci machine learning repository, 2019. URL https://archive.ics.uci.edu/.

Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Min. Knowl. Discov.*, 38:2770–2824, 2022.

Kaggle. Kaggle: Your machine learning and data science community, 2010. URL https://www.kaggle.com.

L. Ljung. Convergence analysis of parametric identification methods. *IEEE Transactions on Automatic Control*, 23(5):770–783, 1978.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *SIGKDD Explor. Newsl.*, 15:49–60, 2014.

Vy Vo, Trung Le, Van Nguyen, He Zhao, Edwin V. Bonilla, Gholamreza Haffari, and Dinh Phung. Feature-based learning for diverse and privacy-preserving counterfactual explanations. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 2211–2222, 2023.