WHEN FEATURES DON'T MOVE ALONE: RELATIONAL CONSTRAINT SATISFACTION FOR COUNTERFACTUAL EXPLANATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Counterfactual explanations (CEs) explain model predictions by identifying input changes that alter the predicted outcome. CE quality spans multiple properties, including validity, proximity, confidence, sparsity, and constraints on feature changes. While boundary and immutability constraints have received considerable attention, relational constraints encoding dependencies among features remain largely unaddressed. We introduce relational constraints as a third constraint type and propose MARBLE, a gray-box method that assembles candidates from observed nearest unlike neighbors values, deterministically repairs constraints, and refines them through an ε-constrained multi-objective evolutionary search, returning only CEs satisfying all stated constraints and abstaining otherwise. We evaluate MARBLE on four high-stakes real-world use cases: URL phishing detection, credit scoring, ICU survival prediction, and botnet attack detection. On the hardest dataset, MARBLE returns a valid CE satisfying all stated constraints for 91% of queries, compared with at most 4% for five state-of-the-art methods. Across the benchmark, MARBLE outperforms these methods in validity and constraint satisfaction while remaining competitive in confidence, sparsity, and proximity.

1 INTRODUCTION

Machine learning models increasingly support high-stakes decisions over tabular data, from credit scoring to medical triage. Yet their often-opaque decisions must be understood, audited, and, under regulations, explained to affected individuals (Wachter et al., 2017). This motivates post-hoc explanations of individual decisions without requiring access to the model internals (Miller, 2019).

CEs identify input changes that alter the model output (Verma et al., 2024), providing local explanations and actionable alternatives without full model disclosure. Wachter et al. (2017) formulated CE generation around *(i) validity*, achieving a different class, and *(ii) proximity* to the original instance. Subsequent work added further objectives (Paredes Cetina et al., 2026): *(iii) confidence* in the target class; *(iv) sparsity*, the number of modified features; *(v) boundary constraints*, keeping values within admissible ranges; and *(vi) immutability constraints*, preserving immutable features.

While *immutability* and *boundary constraints* have received considerable attention in the CE literature (Liu et al., 2025), a third type remains largely unaddressed: *relational constraints*, which encode dependencies among features through linear or non-linear (in)equalities or logical combinations. To address this gap, we introduce MARBLE, a gray-box method that satisfies relational constraints while optimizing the remaining CE properties. We evaluate MARBLE on four high-stakes real-world use cases: URL phishing detection, credit scoring (LCLD), ICU survival prediction (WiDS), and botnet attack detection (CTU). Figure 1 illustrates, on LCLD – *one of the more challenging datasets for CE generation* – how MARBLE and the two strongest baselines (DICE, CCHVAE) handle three interrelated features: $\mathrm{open_accounts}$, $\mathrm{total_accounts}$, and $\mathrm{ratio_accounts} = \mathrm{open_accounts}/\mathrm{total_accounts}$. CEs must satisfy the boundary constraints $\mathrm{open_accounts} \geq 0$, $\mathrm{total_accounts} \geq 0$, and $\mathrm{open_accounts} \leq \mathrm{total_accounts}$, defining the shaded feasible region, and the relational constraint $\mathrm{ratio_accounts}$. DICE (Figure 1a) and CCHVAE (Figure 1b) generate many constraint-violating CEs, whereas MARBLE (Figure 1c) satisfies both relational and boundary constraints while remaining competitive on other CE properties (Figure 1d).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

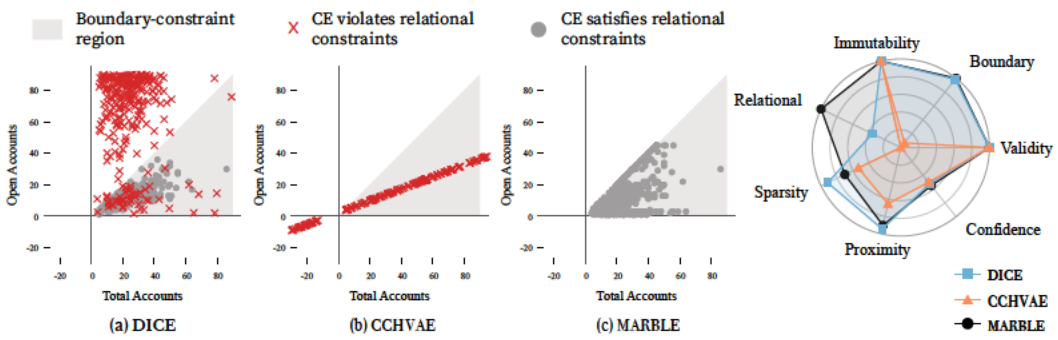


Figure 1: CE generation with (a) DICE, (b) CCHVAE, (c) MARBLE (ours). (d) Comparison on LCLD with STG model: MARBLE satisfies all constraint types – *unlike the two baselines* – while remaining competitive on other CE properties. Metrics are normalized to $[1, 10]$; higher is better.

Contributions. This paper makes the following contributions:

- We show that existing CE methods lack general support for relational constraints, despite handling immutability and boundary constraints.
- We propose MARBLE, the first model-agnostic CE method to *guarantee* satisfaction of all types of relational constraints;
- We demonstrate that MARBLE handles all five relational constraint subtypes, while existing methods struggle particularly with non-linear constraints.

2 Related Work

Realistic tabular instances must satisfy application-domain constraints (Stoian et al., 2024; Bhattacharya et al., 2025). In the CE literature, constraint handling has primarily focused on *immutability* and *boundary constraints*, which restrict individual features through immutability or admissible ranges (Liu et al., 2025). In contrast, constraints coupling multiple features, which we call *relational constraints*, remain largely unaddressed. Studied in adversarial learning for tabular data (Simonetto et al., 2024b), they include: *(i-ii) linear equalities and inequalities*, imposing dependencies, bounds, or orderings across features; *(iii) non-linear equalities and inequalities*, involving products, powers, or divisions; *(iv) boolean constraints*, combining conditions through logical operators; and *(v) safe divisions*, defining ratios when the denominator is zero. These constraints are harder to enforce because they couple features and can induce complex feasible regions.

This limitation is shared by the main families of CE methods, including heuristic-, optimization-, instance-, tree-, stochastic-control-, and generative-based methods (Guidotti, 2024); Appendix A provides a detailed review of these methods. Existing mechanisms can enforce immutability by restricting mutable features (Furman et al., 2026; Mothilal et al., 2020) and admissible ranges through bounded sampling or projection (Liu et al., 2025; Le et al., 2020), but provide no general mechanism for enforcing relational constraints. Closest to relational constraint handling are causal approaches. Mahajan et al. (2020) uses a partial structural causal model to regularize counterfactuals toward causally predicted values, while Verma et al. (2022) restricts feature changes based on causal relations. However, neither provides general relational constraint satisfaction, which may involve algebraic identities, inequalities, logical conditions, or derived-feature consistency rather than causal mechanisms. Moreover, causal models require knowledge that is often unavailable or not uniquely recoverable from observational data (Karimi et al., 2020; Uhler et al., 2013). Moreover, soft causal regularization cannot guarantee constraint satisfaction, while directed structural equations do not naturally express inequalities, boolean conditions, or symmetric identities. Existing methods therefore support only limited hand-specified relations, such as monotonic changes (Verma et al., 2022; Liu et al., 2025). **MARBLE addresses this gap by enforcing relational constraints without requiring a causal model, as validated on four real-world datasets, five neural architectures, and two tabular foundation models (Section 4).**

.

3 CONSTRAINT-AWARE COUNTERFACTUAL EXPLANATIONS

3.1 PROBLEM STATEMENT.

Let f be a classifier over a tabular feature space $\mathcal{X} = \mathcal{X}1 \times \cdots \times \mathcal{X}p$, and $x \in \mathcal{X}$ a query instance with a predicted label $y = f(x)$. We write $P(f(z) = c)$ for the probability assigned to class c at z. Each query has a set of immutable features $I \subseteq \{1, \ldots, p\}$ and relational constraints $\Phi = \{\phi_1, \ldots, \phi_K\}$, each associated with a violation function $v\phi : \mathcal{X} \to \mathbb{R}\geq 0$ such that $v_\phi(x) = 0$ iff x satisfies ϕ.

Given a distance metric d and a reference set $R \subseteq \mathcal{X}$ of observed instances from which new values are drawn, the objective is to find counterfactuals $C(x)$ such that every $C_j \in C(x)$ changes the prediction, preserves all immutable features, and satisfies every constraint in Φ, while maximizing target-class confidence and minimizing sparsity and proximity to x.

3.2 MARBLE

Every candidate CE considered by MARBLE is assembled from values observed in a pool of nearest unlike neighbors (NUNs) from R, defining its discrete search space. Given a user-defined threshold θ, MARBLE proceeds in three stages. First, an ***admission stage*** searches for an initial candidate whose confidence in a target class reaches θ; if no such candidate is found, the query is rejected. Second, a ***repair stage*** attempts to make the candidate CE satisfy all constraints. Third, an ***optimization stage*** seeds an ε-constrained multi-objective evolutionary search with the candidate CE to optimize confidence, sparsity, and proximity subject to the constraints, returning the feasible counterfactuals. Section 3.2.1 defines the search space, and Sections 3.2.2–3.2.4 detail these three stages. A step-by-step description of MARBLE, including the corresponding pseudocode, is provided in Appendix B.

3.2.1 SEARCH SPACE AND CANDIDATE CONSTRUCTION

Let $D \subseteq R$ be the *donor set*: the n NUNs of x in R. For each feature j, the value set $V_j = \{r_j : r \in D\}$ contains an index set of unique values of feature j across D, defining its possible alternatives. A *genome* is a vector $\gamma \in \Gamma = \prod_{j=1}^{p}\{0, 1, \ldots, u_j\}$, where $\gamma_j = 0$ preserves the query value for immutable features ($j \in I$), whereas $\gamma_j = \ell$ ($\ell \geq 1$) replaces mutable features ($j \notin I$) with $V_j(\ell)$.

To turn a genome γ into a CE, MARBLE applies each mutable feature's choice (keeping the query value or using the selected alternative), then applies a *repair operator* to restore relational constraints. Features defined as exact functions of others are recomputed, one-sided inequalities are enforced by clipping, and defined features are recomputed once more if clipping altered their inputs. Repair is best-effort: unresolved violations are left to the optimization. For any genome $\gamma \in \Gamma$, we denote by $C_\gamma \in \mathcal{X}$ the CE obtained after substitution and repair.

3.2.2 ADMISSION STAGE

The goal is to find an initial CE candidate with confidence at least the user-defined threshold θ for a target class $c \neq y$. MARBLE initializes a candidate from each donor $r \in D$ by adopting its values for all mutable features, preserving the query's immutable values, and repairing the result. Each donor is scored by the classifier's confidence in its class $f(r)$ for the resulting candidate, and the highest-scoring donor $r^\star$ defines the target class $c = f(r^\star)$. Starting from this candidate, MARBLE performs *greedy ascent*: it changes one mutable feature at a time to another value in its value set, retaining the change that most increases confidence in c. The candidate is admitted once confidence reaches θ, or the query is rejected when no further improvement is possible. We choose the greedy ascent approach because it relies only on confidence in c, allowing to preserve model-agnosticism, treating the model as a black-box.

3.2.3 REPAIR STAGE

The admitted candidate reaches θ but may still violate relational constraints. Repair considers the same moves as Admission (Section 3.2.2), selecting those that maintain confidence above θ while minimizing constraint violation. It stops when all constraints are satisfied or no further improvement is possible. If successful, the repaired candidate becomes a seed; otherwise, the admitted candidate is retained. Thus, repair improves the starting point when possible but never rejects a query.

3.2.4 Optimization Stage

The admitted genome space is searched with NSGA-III (Deb & Jain, 2014), which maintains solution diversity through reference points generated by the Das-Dennis method (Das & Dennis, 1998) over a 3-objective space with k partitions, under the constrained multi-objective problem:

$$\begin{aligned} \min \quad & o_1(\gamma) = -\left(P(f(C_\gamma)) = c\right) \\ \min \quad & o_2(\gamma) = \tfrac{1}{p}\left|\{\, j : \gamma_j \neq 0 \,\}\right| \\ \min \quad & o_3(\gamma) = d\big(x,\, C_\gamma\big) \\ \text{s.t.} \quad & g_0(\gamma) = \theta - \left(P(f(C_\gamma)) = c\right) \leq 0, \\ & g_k(\gamma) = v_k(C_\gamma) \leq 0, \quad k = 1, \ldots, K. \end{aligned} \tag{1}$$

Where o_1 maximizes confidence in the target class c, o_2 measures the fraction of changed features, and o_3 the Gower distance (Gower, 1971) to the query.

Constraint handling. ε-level comparison (Takahama & Sakai, 2006) ranks genomes by a single infeasibility score, requiring MARBLE to aggregate constraints of potentially different scales. Let $[\cdot]_+ = \max(0, \cdot)$ denote constraint violation. Each constraint i is normalized by its largest violation in the initial population P_0:

$$\sigma_i = \max\Big(1,\ \max_{\gamma \in P_0} [g_i(\gamma)]_+\Big), \tag{2}$$

where the floor at one prevents constraints satisfied throughout P_0 from inflating the normalized violation. The *constraint violation* of a genome is then

$$\mathrm{CV}(\gamma) = \sum_{i=0}^{K} \frac{[g_i(\gamma)]_+}{\sigma_i}, \tag{3}$$

The scales σ_i are computed once and fixed throughout the run, making CV population-independent. A genome is feasible when $\mathrm{CV}(\gamma) \leq \varepsilon(t)$, where $\varepsilon(t)$ decays from ε_0 to zero:

$$\varepsilon(t) = \begin{cases} \varepsilon_0\big(1 - \frac{t}{\tau g}\big)^{\beta} & t < \tau g, \\ 0 & t \geq \tau g, \end{cases} \tag{4}$$

ε_0 is set to the p-th percentile of $\{\mathrm{CV}(\gamma) : \gamma \in P_0\}$, adapting the initial tolerance to population infeasibility. The tolerance reaches zero after a fraction τ of the g generations, with β controlling its decay. It drives both selection steps: binary tournaments favor smaller $[\mathrm{CV} - \varepsilon(t)]_+$, with ties resolved by Pareto dominance and then randomly, while survival prioritizes genomes with $\mathrm{CV} \leq \varepsilon(t)$ and ranks the rest by CV. Thus, early generations explore slightly infeasible regions before progressively converging to feasibility.

Initialization & operators. The initial population contains the admitted candidate(s), strongest-donor candidate, and, if successful, repaired candidate(s). Remaining individuals are sampled feature-wise: each mutable feature retains the query value with probability $\frac{1}{2}$; otherwise, a value is sampled from V_j by donor frequency, favoring frequent values and features shared with the query. Mutation follows the same distribution, while recombination uses two-point crossover.

Output. Throughout the run, every strictly feasible individual ($\mathrm{CV} = 0$) is added to an *elite archive*, which is pruned to its non-dominated front, ensuring that feasible solutions are not lost once evaluated. The returned set $C(x)$ comprises the decoded, strictly feasible, non-dominated solutions from the union of the final population and archive, with objective values S.

Approach correctness. The guarantees of MARBLE follow from its structural components and do not depend on the stochastic evolutionary search. Candidate construction preserves immutable features, archiving retains only constraint-satisfying genomes, and extraction retains only those meeting at least the confidence threshold θ. Thus, MARBLE is *sound* by construction: every returned CE is valid and satisfies all the stated constraints. Meanwhile, like other methods, it is not *complete*: a non-empty output is only guaranteed if the initial candidate is feasible or if the repair stage is able to find a feasible candidate. Empirically, however, MARBLE returns valid, constraint-satisfying CEs for a substantially larger fraction of queries than the evaluated baselines (Section 4, Table 1).

4 EXPERIMENTAL EVALUATION

Datasets and Constraints. We evaluate MARBLE on four TABULARBENCH benchmarks (Simonetto et al., 2024b): i) `URL` (Hannousse & Yahiouche, 2021), distinguishing legitimate from phishing URLs; ii) `LCLD` (George Nathan, 2018), a credit-scoring dataset of loan applications; iii) `WiDS` (Lee et al., 2020), containing ICU admissions for survival prediction; and iv) `CTU` (Chernikova & Oprea, 2022), containing legitimate and botnet traffic. We use TABULARBENCH for its comprehensive domain constraints; Table 7 in Appendix C.1 summarizes each dataset.

Models. We evaluate five deep neural network architectures surveyed by Borisov et al. (2022) and benchmarked in TABULARBENCH: **TABNET** (Arik & Pfister, 2021), using sequential attention for feature selection; **TABTRANSFORMER** (Huang et al., 2020), applying self-attention to categorical feature embeddings; **TORCHRLN** (Shavitt & Segal, 2018), learning regularization coefficients to induce weight sparsity; **STG** (Yamada et al., 2020), using stochastic gates for feature selection; and **VIME** (Yoon et al., 2020), a self-supervised framework based on mask estimation and feature reconstruction. Further details are provided in Appendix C.1.

Metrics. We assess CE quality using five criteria evaluated over a stratified subsample of 1000 queries from the test set, preserving the class distribution. *Validity* (**Val**) measures the proportion of queries for which at least one CE flips the predicted class, while *Sparsity* (**Spa**) measures the fraction of modified features. *Proximity* (**Prox**) is evaluated using ℓ_1, measuring the sum of feature deviations; lower values indicate closer CEs. *Confidence* (**Conf**) is the predicted probability of the CE target class. Finally, *constraint satisfaction* (**CS**) measures the fraction of CEs satisfying all domain constraints, with separate metrics for *immutability* (**IC**), *boundary* (**BC**), and relational (**RC**) constraints. Further details are provided in Appendix C.2; IC, BC, RC reported in the Appendices.

Baselines MARBLE is compared with five state-of-the-art CE methods: NNCE (Wexler et al., 2019), DICE (Mothilal et al., 2020), C-CHVAE (Pawelczyk et al., 2020), COUNTERGAN (Nemirovsky et al., 2022), and FASTAR (Verma et al., 2022). These baselines were selected for their public implementations, complementary CE generation families, and established use in the literature (see Table 4). We also considered MOC, CERTIFAI, and CFM-FW. We excluded MOC due to its R-only implementation and CERTIFAI due to unresolved errors in its public implementation. CFM-FW, the closest existing method in terms of relational-constraint support, could not be evaluated because no public implementation is available. Hyperparameters for MARBLE and all baselines are provided in Appendix C.3.

Table 1: Benchmark results (mean across models; standard deviation as subscript, $x_{\pm s}$). Methods producing no valid CE score 0 for validity and constraint satisfaction (CS), with CS requiring all constraints. Best and second-best results are **bold** and underlined.

Method	URL		WiDS		CTU-13		LCLD	
	Val↑	CS↑	Val↑	CS↑	Val↑	CS↑	Val↑	CS↑
DICE	$\underline{0.98_{\pm 0.05}}$	$0.79_{\pm 0.08}$	$\mathbf{1.00}_{\pm 0.00}$	$0.54_{\pm 0.13}$	$\underline{0.85_{\pm 0.16}}$	$\underline{0.04_{\pm 0.04}}$	$\underline{0.92_{\pm 0.17}}$	$\underline{0.34_{\pm 0.07}}$
NNCE	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$\underline{0.99_{\pm 0.00}}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
C-CHVAE	$\mathbf{1.00}_{\pm 0.00}$	$0.00_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$0.91_{\pm 0.08}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.81_{\pm 0.17}$	$0.00_{\pm 0.00}$
COUNTERGAN	$0.85_{\pm 0.12}$	$0.48_{\pm 0.12}$	$0.87_{\pm 0.05}$	$0.00_{\pm 0.00}$	$0.10_{\pm 0.17}$	$0.00_{\pm 0.00}$	$0.08_{\pm 0.06}$	$0.00_{\pm 0.00}$
FASTAR	$0.75_{\pm 0.16}$	$0.65_{\pm 0.31}$	$0.07_{\pm 0.14}$	$0.04_{\pm 0.08}$	$0.40_{\pm 0.40}$	$0.00_{\pm 0.00}$	$0.44_{\pm 0.33}$	$0.00_{\pm 0.00}$
MARBLE	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.91}_{\pm 0.16}$	$\mathbf{0.91}_{\pm 0.16}$	$\mathbf{0.93}_{\pm 0.17}$	$\mathbf{0.93}_{\pm 0.17}$

5 RESULTS

5.1 HOW EFFECTIVE IS MARBLE?

Validity & constraint satisfaction. Table 1 compares MARBLE with state-of-the-art baselines in terms of valid, constraint-satisfying CEs. MARBLE achieves at least 91% across all datasets, outperforming the next-best method by 87 and 59 percentage points on `CTU` and `LCLD`, respectively.

Table 2: Quality metrics benchmark. Each cell is the mean over models with the across-model standard deviation as a subscript ($x_{\pm s}$); Quality metrics are computed over flipped instances only. "–" means no CEs were generated.

Dataset	Method	Val↑	Conf↑	Spars↓	Prox↓
URL	DICE	$0.98_{\pm 0.05}$	$0.80_{\pm 0.13}$	$0.03_{\pm 0.00}$	$1.22_{\pm 0.20}$
	NNCE	$\mathbf{1.00}_{\pm 0.00}$	$0.88_{\pm 0.07}$	$0.36_{\pm 0.00}$	$4.67_{\pm 0.06}$
	C-CHVAE	$\mathbf{1.00}_{\pm 0.00}$	$0.65_{\pm 0.04}$	$1.00_{\pm 0.00}$	$4.74_{\pm 0.09}$
	COUNTERGAN	$0.85_{\pm 0.12}$	$0.91_{\pm 0.05}$	$0.70_{\pm 0.02}$	$3.59_{\pm 1.15}$
	FASTAR	$0.75_{\pm 0.16}$	$0.59_{\pm 0.07}$	$\mathbf{0.02}_{\pm 0.00}$	$\mathbf{0.38}_{\pm 0.14}$
	MARBLE	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.92}_{\pm 0.03}$	$0.08_{\pm 0.02}$	$1.20_{\pm 0.62}$
WiDS	DICE	$\mathbf{1.00}_{\pm 0.00}$	$0.61_{\pm 0.08}$	$0.06_{\pm 0.03}$	$3.21_{\pm 1.51}$
	NNCE	$\mathbf{1.00}_{\pm 0.00}$	$0.77_{\pm 0.08}$	$0.78_{\pm 0.00}$	$15.98_{\pm 0.35}$
	C-CHVAE	$\mathbf{1.00}_{\pm 0.00}$	$0.62_{\pm 0.07}$	$0.96_{\pm 0.00}$	$17.21_{\pm 0.95}$
	COUNTERGAN	$0.87_{\pm 0.05}$	$0.78_{\pm 0.07}$	$0.85_{\pm 0.01}$	$12.89_{\pm 1.13}$
	FASTAR	$0.07_{\pm 0.14}$	$0.51_{\pm 0.00}$	$\mathbf{0.01}_{\pm 0.00}$	$\mathbf{0.59}_{\pm 0.17}$
	MARBLE	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.92}_{\pm 0.06}$	$0.17_{\pm 0.02}$	$5.34_{\pm 1.44}$
CTU-13	DICE	$0.85_{\pm 0.16}$	$0.78_{\pm 0.21}$	$0.01_{\pm 0.02}$	$5.09_{\pm 5.42}$
	NNCE	$0.00_{\pm 0.00}$	–	–	–
	C-CHVAE	$0.00_{\pm 0.00}$	–	–	–
	COUNTERGAN	$0.10_{\pm 0.17}$	$\mathbf{0.97}_{\pm 0.04}$	$0.22_{\pm 0.01}$	$3.55_{\pm 2.97}$
	FASTAR	$0.40_{\pm 0.40}$	$0.71_{\pm 0.24}$	$\mathbf{0.00}_{\pm 0.01}$	$\mathbf{0.44}_{\pm 0.38}$
	MARBLE	$\mathbf{0.91}_{\pm 0.16}$	$0.87_{\pm 0.10}$	$0.02_{\pm 0.01}$	$3.32_{\pm 1.37}$
LCLD	DICE	$0.92_{\pm 0.17}$	$\mathbf{0.76}_{\pm 0.17}$	$0.06_{\pm 0.01}$	$0.99_{\pm 0.32}$
	NNCE	$0.01_{\pm 0.00}$	$0.62_{\pm 0.05}$	$0.43_{\pm 0.00}$	$2.37_{\pm 0.36}$
	C-CHVAE	$0.81_{\pm 0.17}$	$0.51_{\pm 0.01}$	$0.45_{\pm 0.01}$	$4.57_{\pm 3.20}$
	COUNTERGAN	$0.08_{\pm 0.06}$	$0.54_{\pm 0.03}$	$0.41_{\pm 0.00}$	$0.43_{\pm 0.19}$
	FASTAR	$0.44_{\pm 0.33}$	$0.55_{\pm 0.04}$	$\mathbf{0.04}_{\pm 0.00}$	$\mathbf{0.19}_{\pm 0.18}$
	MARBLE	$\mathbf{0.93}_{\pm 0.17}$	$0.75_{\pm 0.12}$	$0.22_{\pm 0.05}$	$1.01_{\pm 0.99}$

On these two datasets, NNCE, C-CHVAE, COUNTERGAN, and FASTAR fail to generate any valid, constraint-satisfying CE. In contrast, the simpler linear constraints of URL and WiDS are well handled by the baselines, with NNCE reaching 100% and 99%, respectively.

CE properties. Focusing on the valid CE, we analyze their validity, sparsity, and confidence in Table 2. MARBLE matches or exceeds the highest validity (Val) across all four datasets, reaching perfect validity (1.00) on URL and WiDS, and leading on CTU-13 (0.91) and LCLD (0.93). It ranks first or second in confidence (Conf), achieving 0.92 on both URL and WiDS, 0.87 on CTU-13, and 0.75 on LCLD. It consistently places third in sparsity (Spars), with values of 0.08, 0.17, 0.02, and 0.22 on those same datasets. For proximity (Prox), MARBLE ranks second on URL (1.20) and CTU-13 (3.32), and third on WiDS (5.34), though it falls to fourth on LCLD (1.01), narrowly behind DICE (0.99). Thus, enforcing all constraints does not prevent MARBLE from remaining competitive in the other CE quality metrics. Full benchmark results are provided in Appendix D.

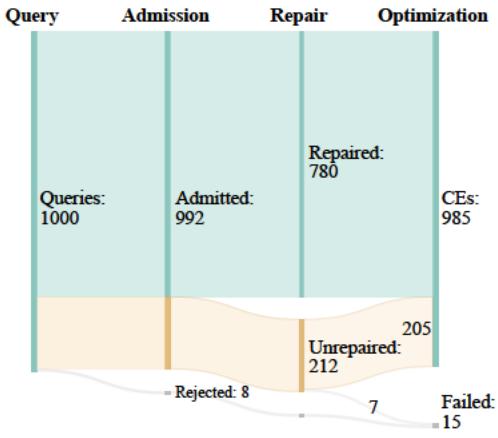
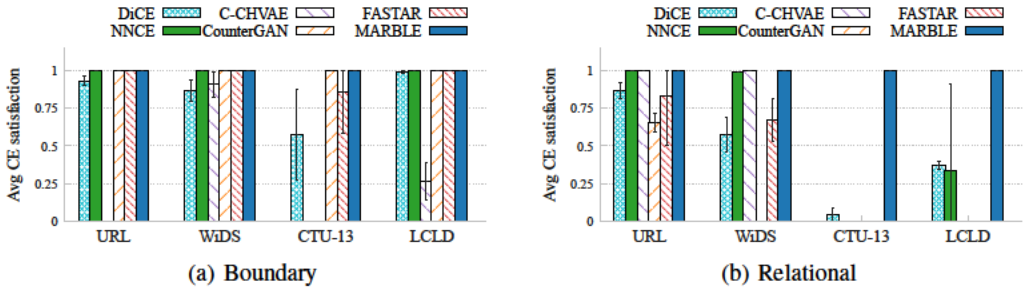


Figure 2: Impact of different stages on the TabTransformer model (CTU dataset).

Stage-by-stage performance. Figure 2 shows the number of samples that are valid and satisfy all constraints after each stage of MARBLE on CTU, the only dataset that exercises all stages. Results for all datasets are provided in Appendix E. Out of 1000 queries (Figure 2), stage 1 (Admission) finds 992 candidate CEs that flip the prediction, yet none satisfy the relational constraints. Stage 2



(a) Boundary (b) Relational

Figure 3: Constraint satisfaction by family: (a) *boundary* and (b) *relational*, per dataset (bars) and averaged across models (error bars: std). The constraints are only measured for valid CE.

(Repair) repairs 780 of them, and stage 3 (Optimization) recovers 205 more to bring the total to 985 valid constrained CEs. The benefit of this multi-stage approach lies in the efficiency of stages 1 and 2: they are sufficient for simpler datasets (for instance, stage 1 alone is enough for URL, WiDS and LCLD – Tables 15, 16 and 18), while challenging datasets like CTU require stage 2 and stage 3. (Table 17).

Stage ablation. We perform a subtractive stage ablation on CTU with STG, the most challenging architecture for CE generation (Section 5.3). Removing either Repair or Admission reduces validity from 1.00 to 0.50: without Repair, infeasible candidates rely solely on the GA to recover constraint satisfaction; without Admission, the optimizer lacks a suitable initial candidate for half the queries. Removing Optimization has a smaller effect, reducing validity to 0.84, as Admission and Repair alone suffice for most, but not all, queries.

5.2 Which constraints are the hardest to enforce?

Constraint satisfaction MARBLE achieves 100% satisfaction of boundary and relational constraints across all settings (Figure 3). The other baselines demonstrate inconsistent performances across the datasets and constraint types: boundary constraints (BC) are better satisfied than relational (RC) across most settings, with the exception of CTU, where DICE and FASTAR drop to 57% and 86% boundary satisfaction respectively, while COUNTERGAN remains the only baseline achieving 100% boundary satisfaction across all datasets. The performance of the baselines is worse for relational constraints: their CEs achieve under 38% and 5% relational constraint satisfaction on LCLD and CTU respectively. On URL and WiDS, C-CHVAE and NNCE achieve at least 99% relational constraint satisfaction, but the remaining methods achieve less than 87% and at most 67% relational constraint satisfaction on URL and WiDS respectively.

This result reflects how baselines enforce constraints: *immutability* is typically handled through masking and *boundary* constraints through clipping, so both are largely satisfied. The few boundary violations arise from missing CEs or numerical noise in integer-valued features (e.g., rounding C-CHVAE outputs on LCLD increases satisfaction to approximately 95–96%). In contrast, relational constraints couple multiple features through equalities, inequalities, or logical combinations (Section 2) and cannot generally be enforced independently. Lacking mechanisms for joint constraint satisfaction, all baselines achieve low relational satisfaction with large standard deviations across models (Figure 3), whereas MARBLE consistently achieves 100% by enforcing feasibility by design.

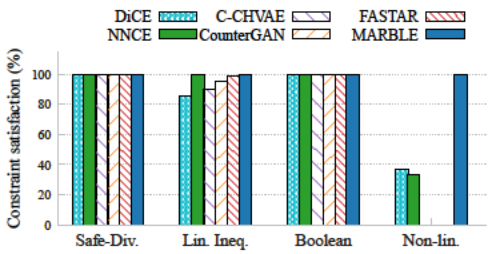


Figure 4: Relational constraint satisfaction per subcategory on LCLD, averaged across models. Baselines largely preserve unchanged constraints but struggle with non-linear relations .

Figure 4 details relational constraint satisfaction on LCLD, which covers four of the five constraint subcategories (safe-div., lin. ineq., boolean, and non-lin.); results for all datasets are provided in Appendix D. Non-linear constraints are the most challenging: C-CHVAE, FASTAR and COUNTERGAN fail to satisfy any, while NNCE and DICE satisfy only $\approx 35\%$; in contrast, MARBLE achieves full satisfaction.

Table 3: Benchmark results by models. Each cell is the mean over datasets with the across-dataset standard deviation as a subscript ($x_{\pm s}$). A dataset on which no valid counterfactual is produced scores 0 for both validity and constraint satisfaction. CS is the binary intersection of all constraint conditions. Best method is in **bold** and the second best is underlined.

Method	VIME		TorchRLN		TabTr		TabNet		STG	
	Val↑	CS↑	Val↑	CS↑	Val↑	CS↑	Val↑	CS↑	Val↑	CS↑
DICE	$0.97_{\pm 0.05}$	$0.37_{\pm 0.26}$	$0.96_{\pm 0.07}$	$0.46_{\pm 0.32}$	$0.94_{\pm 0.11}$	$0.42_{\pm 0.33}$	$\mathbf{1.00}_{\pm 0.00}$	$0.50_{\pm 0.34}$	$0.81_{\pm 0.22}$	$0.38_{\pm 0.37}$
NNCE	$0.50_{\pm 0.58}$	$0.50_{\pm 0.58}$	$0.50_{\pm 0.58}$	$0.50_{\pm 0.58}$	$0.50_{\pm 0.58}$	$0.50_{\pm 0.58}$	$0.50_{\pm 0.58}$	$0.50_{\pm 0.58}$	$0.50_{\pm 0.58}$	$0.50_{\pm 0.58}$
C-CHVAE	$0.70_{\pm 0.47}$	$0.24_{\pm 0.48}$	$0.68_{\pm 0.47}$	$0.23_{\pm 0.46}$	$0.75_{\pm 0.50}$	$0.19_{\pm 0.38}$	$0.73_{\pm 0.49}$	$0.24_{\pm 0.48}$	$0.65_{\pm 0.47}$	$0.24_{\pm 0.48}$
COUNTERGAN	$0.47_{\pm 0.49}$	$0.11_{\pm 0.22}$	$0.47_{\pm 0.42}$	$0.12_{\pm 0.23}$	$0.48_{\pm 0.66}$	$0.27_{\pm 0.38}$	$0.55_{\pm 0.20}$	$0.16_{\pm 0.22}$	$0.45_{\pm 0.51}$	$0.16_{\pm 0.31}$
FASTAR	$0.24_{\pm 0.29}$	$0.04_{\pm 0.07}$	$0.73_{\pm 0.30}$	$0.25_{\pm 0.38}$	$0.35_{\pm 0.43}$	$0.24_{\pm 0.48}$	$0.49_{\pm 0.40}$	$0.18_{\pm 0.35}$	$0.24_{\pm 0.26}$	$0.16_{\pm 0.30}$
MARBLE	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.99}_{\pm 0.01}$	$\mathbf{0.99}_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.01}$	$\mathbf{1.00}_{\pm 0.01}$	$0.99_{\pm 0.01}$	$\mathbf{0.99}_{\pm 0.01}$	$\mathbf{0.81}_{\pm 0.22}$	$\mathbf{0.81}_{\pm 0.22}$

5.3 Which Models Are Best for Constrained Counterfactual Explanations?

Table 3 shows that TorchRLN is the easiest model to flip predictions on, with an average validity across methods of 0.72, followed by TabNet (0.66), VIME (0.65), and TabTr (0.63). MARBLE achieves near-perfect validity (≥ 0.99) on all four, and DiCE also performs strongly (0.94–1.00). For constraint satisfaction, the picture splits depending on methods. Among baselines, TorchRLN is again the most favorable (average CS of 0.31 excluding MARBLE), followed by TabNet (0.30) and TabTr (0.30), while VIME is the hardest (0.25); still, no baseline exceeds 0.50 CS on any model, showing that existing methods struggle to satisfy constraints regardless of the model. MARBLE achieves 0.99–1.00 CS on four out of five models, making model choice nearly irrelevant when constraints are handled explicitly. The exception is STG, which is consistently the hardest model across both metrics, ranking last in average validity (0.58) and average CS (0.29). Even MARBLE drops to 0.81 in both Val and CS on STG, a degradation not observed on any other model.

6 Discussion and Extended Results

6.1 Generalization to Foundation Models

To evaluate adaptability to emerging architectures, we extend our benchmark to Tabular Foundation Models (TFMs) such as TABICL and TABPFN (see Tables 19 and 20 in Appendix F). Using TFMs to generate counterfactuals with baseline approaches improves their ability to satisfy relational constraint. For instance, NNCE's constraint satisfaction on the LCLD dataset rises from 0.00 on standard models to 0.67 on TABICL. DICE continues to yield poor CS (0.33–0.35) on complex datasets, whereas NNCE manages to achieve near-perfect validity scores. In contrast, MARBLE still resolves these competing objectives when used with TFMs, consistently achieving near-perfect validity and constraint adherence (CS ≈ 1.00) while minimizing overall distance metrics, thereby confirming its transferability to foundation architectures.

6.2 Ablations & Sensitivity Analyses

Impact of the Elite search optimization: Table 23 in Appendix G shows that bare optimization loses all valid solutions in roughly half of CTU-13 runs due to ε-relaxed selection (Val = 0.49); the seed fallback recovers most failures (Val = 0.99), and the archive closes the remaining gap (Val = 1.00) while also improving solution quality by enabling extraction across all generations rather than only the final population (e.g., sparsity $0.182 \rightarrow 0.105$ on WiDS).

Impact of constraint optimization: Table 24 in Appendix G shows that explicit constraint handling is essential: removing it causes large CR drops on WiDS ($1.00 \rightarrow 0.43$), CTU ($1.00 \rightarrow 0.20$),

and LCLD ($1.00 \rightarrow 0.30$), confirming that in-distribution generation alone is insufficient to produce CEs satisfying relational constraints. URL is the only dataset unaffected (CR stays at 1.00), because the constraints were never binding.

Multi-objective optimization: Table 26 in Appendix G shows that confidence is the most critical objective, with its removal causing severe Conf degradation across all datasets (e.g., $0.989 \rightarrow 0.591$ on URL, $0.936 \rightarrow 0.534$ on WiDS), while dropping sparsity increases the number of modified features ($0.105 \rightarrow 0.237$ on WiDS, $0.152 \rightarrow 0.308$ on LCLD) and dropping proximity indirectly worsens sparsity on WiDS.

Robustness to hyperparameter variations: Appendix H shows that MARBLE is robust to the distance metric choice and converges at population size=100–200 and maximum number of generations=50–100, with small gains beyond these values. Higher θ improves validity on harder datasets at the cost of larger, denser edits, while larger donor pools also improve validity at significant runtime cost (e.g., CTU: $\sim$3 min to $\sim$4.5 hours when unbounded); both are parameters worth tuning rather than maximizing.

Isolating the repair operator: Appendix I shows that applying MARBLE's projection repair post-hoc to baseline methods can improve constraint satisfaction in favorable cases (e.g., C-CHVAE on URL: CS $0.00 \rightarrow 0.97$ with Val $1.00 \rightarrow 0.98$), but in most cases it degrades validity, sometimes severely (DICE on CTU: Val $0.85 \rightarrow 0.17$; FASTAR on CTU: Val $0.40 \rightarrow 0.01$). The repair operator is necessary but not sufficient: MARBLE's simultaneous validity and constraint satisfaction relies on constraints being enforced throughout generation rather than applied as a post-hoc correction.

6.3 Limitations and Future Work

Gray-box setting. While MARBLE does not require access to the target model's internals or gradients (model-agnostic), the constraint optimization still requires knowledge of the domain constraints to enforce and access to a reference set (same distribution as training data) to draw values from. In practice, CEs are used for algorithmic recourse and model auditing where the constraints of the field are known, making MARBLE a viable solution for real-world applications. We discuss the applications of MARBLE in Appendix A.2.

Pareto optimization. MARBLE constraint optimization objective is at odds with the proximity objective. While MARBLE does not achieve the closest CE, we demonstrate in Appendix Tables 31-32 that our solution achieves the best pareto front by maximizing the hyper-volume (HV) of the objectives.

Hyper-parameter tuning. MARBLE achieves optimal performance with different hyper-parameters for each setting (model and dataset). The results of the paper are on an independent test set never used for hyper-parameter optimization. Designing universal constrained CE could be a follow-up work but is currently out of our scope.

7 Conclusion

This work identifies relational constraint satisfaction as a missing requirement in CE generation. Our results show that existing CE methods can perform well under independently enforceable constraints while failing when feasibility depends on relationships among features. This distinction matters whenever CEs are expected to represent coherent, actionable domain states rather than merely cross a model's decision boundary.

Beyond demonstrating that relational constraints can be enforced, our results highlight the importance of MARBLE's multi-stage design. Admission efficiently solves simpler cases, Repair restores feasibility when independently selected feature values violate relational constraints, and Optimization handles remaining difficult cases while refining CE quality. The stage-wise analysis and ablation show that candidate generation, feasibility restoration, and quality optimization play complementary roles: simpler datasets can be handled without invoking the full pipeline, whereas more constrained settings require their combination.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks that require disclosure: implementing parts of the method, specifically the constraint-repair and admission stages of MARBLE; assisting with the writing of the method section (Section 3). We have not used generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating or proving mathematical claims, proposing or refining hypotheses, interpreting results, or writing proofs. Translation and survey design are not applicable to this work.

Additionally, we used generative AI tools for the following tasks with recommended disclosure: creating and editing software code for the benchmark evaluation scripts, the ablation study pipeline, and the sensitivity analysis pipeline; assisting with the creation of scripts that extract raw data from generated experimental artifacts; and brainstorming the visual design of Figure 3.

We have reviewed all AI-assisted work. All LLM-generated code was verified and tested for correctness by the authors through unit tests and manual inspection, and documentation was written to ensure each component behaves as intended and is well-understood by reviewers and practitioners. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

While our method generates CE for tabular data, we recognize the importance of responsible and ethical use. It is conceivable that individuals could misuse CE to game ML models, for instance by identifying the minimal feature changes needed to obtain a favorable outcome through fraudulent means, such as falsifying financial records to secure credit approval. Additionally, deploying instance-based CE methods introduces privacy risks: CE grounded in training instances can expose sensitive information through explanation linkage attacks Goethals et al. (2023). Nevertheless, CE also offer significant societal benefits, including actionable recourse for affected individuals (Karimi et al., 2020), support for regulatory compliance such as GDPR, and greater transparency in high-stakes decision-making domains such as healthcare (Bhattacharya et al., 2025).

REPRODUCIBILITY STATEMENT

We take several steps to ensure reproducibility. Full pseudocode for MARBLE and the GREEDYASCENT subroutine is given in Appendix B (Algorithms 1 and 2). All experiments use publicly available datasets and pre-trained models from the TABULARBENCH framework (Appendix C.1). The evaluation protocol and metric definitions are detailed in Appendix C.2, and the exact hyperparameter configurations for MARBLE and every baseline method are reported in Appendix C.3. All runs use a fixed random seed to eliminate stochastic variation. The sensitivity analysis (Appendix H) and ablation studies (Appendix G) follow a paired-trajectory design sharing seeds and parameters to isolate individual effects. Source code implementing MARBLE, the constraint specifications, and all evaluation scripts is included as supplementary material and will be publicly released under the MIT License.

REFERENCES

Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 6679–6687, 2021.

Aditya Bhattacharya, Tim Vanherwegen, and Katrien Verbert. "show me how": Benefits and challenges of agent-augmented counterfactual explanations for non-expert users. In *Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '25, pp. 174–184, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713132. doi: 10.1145/3699682.3728321. URL https://doi.org/10.1145/3699682.3728321.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems*, 35(6):7499–7519, 2022.

Alesia Chernikova and Alina Oprea. Fence: Feasible evasion attacks on neural networks in constrained environments. *ACM Transactions on Privacy and Security*, 25(4):1–34, 2022.

Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. In Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann (eds.), *Parallel Problem Solving from Nature – PPSN XVI*, pp. 448–469. Springer International Publishing, 2020. ISBN 978-3-030-58112-1.

Indraneel Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998. doi: 10.1137/S1052623496307510.

Raphael Mazzine Barbosa de Oliveira, Kenneth Sörensen, and David Martens. A model-agnostic and data-independent tabu search algorithm to generate counterfactuals for tabular, image, and text data. *European Journal of Operational Research*, 317(2):286–302, 2024. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2023.08.031. URL https://www.sciencedirect.com/science/article/pii/S0377221723006598.

Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014. doi: 10.1109/TEVC.2013.2281535.

Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the Missing: Towards Contrastive Explanations with Pertinent Negatives. *Advances in neural information processing systems*, 31, 2018.

Amit Dhurandhar, Tejaswini Pedapati, Avinash Balakrishnan, Pin-Yu Chen, Karthikeyan Shanmugam, and Ruchir Puri. Model Agnostic Contrastive Explanations for Structured Data, 2019. URL http://arxiv.org/abs/1906.00117.

Oleksii Furman, Ulvi Movsum-zada, Patryk Marszałek, Maciej Zieba, and Marek Śmieja. Dicoflex: Model-agnostic diverse counterfactuals with flexible control. *Advances in Neural Information Processing Systems*, 38:71592–71623, 2026.

George Nathan. All Lending Club loan data. https://www.kaggle.com/datasets/wordsforthewise/lending-club, 2018. Consulted on August 7th 2026.

Sofie Goethals, Kenneth Sörensen, and David Martens. The privacy issue of counterfactual explanations: explanation linkage attacks. *ACM Transactions on Intelligent Systems and Technology*, 14(5):1–24, 2023.

J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4): 857–871, 1971. ISSN 0006341X, 15410420. URL http://www.jstor.org/stable/2528823.

Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking: R. guidotti. *Data mining and knowledge discovery*, 38(5):2770–2824, 2024.

Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, Salvatore Ruggieri, and Franco Turini. Factual and Counterfactual Explanations for Black Box Decision Making. 34 (6):14–23, 2019. ISSN 1941-1294. doi: 10.1109/MIS.2019.2957223. URL https://ieeexplore.ieee.org/document/8920138.

Hangzhi Guo, Thanh H. Nguyen, and Amulya Yadav. CounterNet: End-to-End Training of Prediction Aware Counterfactual Explanations. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 577–589. Association for Computing Machinery, 2023. ISBN 979-8-4007-0103-0. doi: 10.1145/3580305.3599290. URL https://dl.acm.org/doi/10.1145/3580305.3599290.

Abdelhakim Hannousse and Salima Yahiouche. Towards benchmark datasets for machine learning based website phishing detection: An experimental study. *Engineering Applications of Artificial Intelligence*, 104:104347, 2021.

Md Golam Moula Mehedi Hasan and Douglas Talbert. Data augmentation using counterfactuals: Proximity vs diversity. In *The International FLAIRS Conference Proceedings*, volume 35, 2022.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Hiroki Arimura. Dace: Distribution-aware counterfactual explanation by mixed-integer linear optimization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pp. 2855–2862, 2020.

Amir-Hossein Karimi, Julius von Kügelgen, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Thai Le, Suhang Wang, and Dongwon Lee. Grace: Generating concise and informative contrastive sample to explain neural network model's prediction. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 238–248, 2020.

Meredith Lee, Jesse Raffa, Marzyeh Ghassemi, Tom Pollard, Sharada Kalanidhi, Omar Badawi, Karen Matthys, and Leo Anthony Celi. WiDS (Women in Data Science) Datathon 2020: ICU Mortality Prediction. *PhysioNet*, January 2020. doi: 10.13026/vc0e-th79. URL https://doi.org/10.13026/vc0e-th79. Version 1.0.0.

Jinping Liu, Xiaoqiang Wu, Shiyi Liu, and Subo Gong. Model-agnostic counterfactual explanation: A feature weights-based comprehensive causal multi-objective counterfactual framework. 266: 126063, 2025. ISSN 0957-4174. doi: 10.1016/j.eswa.2024.126063. URL https://www.sciencedirect.com/science/article/pii/S0957417424029300.

Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers, 2020. URL http://arxiv.org/abs/1912.03277.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. 267:1–38, 2019. ISSN 0004-3702. doi: 10.1016/j.artint.2018.07.007. URL https://www.sciencedirect.com/science/article/pii/S0004370218305988.

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.

Daniel Nemirovsky, Nicolas Thiebaut, Ye Xu, and Abhishek Gupta. CounteRGAN: Generating counterfactuals for real-time recourse and interpretability using residual GANs. In *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, pp. 1488–1497. PMLR, 2022. URL https://proceedings.mlr.press/v180/nemirovsky22a.html.

Emmanouil Panagiotou, Manuel Heurich, Tim Landgraf, and Eirini Ntoutsi. Tabcf: Counterfactual explanations for tabular data using a transformer-based vae. In *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF '24, pp. 274–282, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400710810. doi: 10.1145/3677052.3698673. URL https://doi.org/10.1145/3677052.3698673.

Alan Gabriel Paredes Cetina, Kaouther Benguessoum, Raoni Lourenco, and Sylvain Kubler. Counterfactual explainable ai (xai) method for deep learning-based multivariate time series classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 17393–17400, 2026.

Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of the web conference 2020*, pp. 3126–3132, 2020.

Annabelle Redelmeier, Martin Jullum, Kjersti Aas, and Anders Løland. MCCE: Monte Carlo sampling of valid and realistic counterfactual explanations for tabular data. 38(4):1830–1861, 2024. ISSN 1573-756X. doi: 10.1007/s10618-024-01017-y. URL https://doi.org/10.1007/s10618-024-01017-y.

Lisa Schut, Oscar Key, Rory Mc Grath, Luca Costabello, Bogdan Sacaleanu, Yarin Gal, et al. Generating interpretable counterfactual explanations by implicit minimisation of epistemic and aleatoric uncertainties. In *International Conference on Artificial Intelligence and Statistics*, pp. 1756–1764. PMLR, 2021.

Shubham Sharma, Jette Henderson, and Joydeep Ghosh. CERTIFAI: A Common Framework to Provide Explanations and Analyse the Fairness and Robustness of Black-box Models. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 166–172. ACM, 2020. ISBN 978-1-4503-7110-0. doi: 10.1145/3375627.3375812. URL https://dl.acm.org/doi/10.1145/3375627.3375812.

Ira Shavitt and Eran Segal. Regularization learning networks: deep learning for tabular datasets. *Advances in neural information processing systems*, 31, 2018.

Thibault Simonetto, Salah GHAMIZI, and Maxime Cordy. Constrained adaptive attack: Effective adversarial attack against deep neural networks for tabular data. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=ZtTWKr51yH.

Thibault Simonetto, Salah Ghamizi, and Maxime Cordy. Tabularbench: Benchmarking adversarial robustness for tabular deep learning in real-world use-cases. *Advances in neural information processing systems*, 37:78394–78430, 2024b.

Francesco Spinnato, Riccardo Guidotti, Mirco Nanni, Daniele Maccagnola, Giulia Paciello, and Antonio Bencini Farina. Explaining crash predictions on multivariate time series data. In Poncelet Pascal and Dino Ienco (eds.), *Discovery Science*, pp. 556–566. Springer Nature Switzerland, 2022. ISBN 978-3-031-18840-4.

Mihaela CÄ Stoian, Salijona Dyrmishi, Maxime Cordy, Thomas Lukasiewicz, and Eleonora Giunchiglia. How realistic is your synthetic data? constraining deep generative models for tabular data. *arXiv preprint arXiv:2402.04823*, 2024.

T. Takahama and S. Sakai. Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. In *2006 IEEE International Conference on Evolutionary Computation*, pp. 1–8, 2006. doi: 10.1109/CEC.2006.1688283.

Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 465–474, 2017.

Asterios Tsiourvas, Wei Sun, and Georgia Perakis. Manifold-Aligned Counterfactual Explanations for Neural Networks. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, pp. 3763–3771. PMLR, 2024. URL https://proceedings.mlr.press/v238/tsiourvas24a.html.

Caroline Uhler, Garvesh Raskutti, Peter Bühlmann, and Bin Yu. Geometry of the faithfulness assumption in causal inference. *The Annals of Statistics*, 41(2), Apr 2013. doi: 10.1214/12-aos1080.

Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *Proceedings of the conference on fairness, accountability, and transparency*, pp. 10–19, 2019.

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.

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(12):1–42, 2024.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *SSRN Electronic Journal*, 2017. ISSN 1556-5068. doi: 10.2139/ssrn.3063289.

James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1):56–65, 2019.

Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *International conference on machine learning*, pp. 10648–10659. PMLR, 2020.

Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela Van der Schaar. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in neural information processing systems*, 33:11033–11043, 2020.

Roozbeh Yousefzadeh and Dianne P O'Leary. Debugging trained machine learning models using flip points. In *ICLR 2019 Debugging Machine Learning Models Workshop*, 2019.

A Extended Related Work

This appendix extends the related work by organizing tabular CE methods into six generation families and their supported constraints (Section A.1), and discussing CE quality axes and their application-dependent importance (Section A.2). Table 4 provides an overview of relevant state-of-the-art CE methods, including their family (Section A.1), code availability, venue/year, and support for immutability, boundary, and relational constraints.

Table 4: State-of-the-art CE methods, including their family (Section A.1), code availability, venue/year, and support for immutability, boundary, and relational constraints. Rows highlighted in gray indicate the baselines included in our benchmark (Section 5). CFM-FW, the closest baseline, is not included in our benchmark because its implementation is not publicly available; we contacted the authors to request access and, if possible, include it during rebuttal.

Method	Family	Code	Venue & Year	Imm.	Bound.	Relational Constraints				
						Lin. Eq.	Lin. Ineq.	Non-lin.	Boolean	Safe-Div.
WACH	Optim.	✓	(SSRN, 2017)							
CEM	Optim.	✓	(NeurIPS, 2018)	✓						
MACEM	Optim.	×	(arXiv, –)		✓					
DiCE	Optim.	✓	(FAccT, 2020)	✓	✓					
DACE	Optim.	✓	(IJCAI, 2020)	✓	✓					
ReLU-CFx	Optim.	×	(AISTATS, 2024)							
MOC	Heu.	✓	(PPSN, 2020)	✓	✓					
CERTIFAI	Heu.	✓	(AIES, 2020)	✓	✓					
CFM-FW	Heu.	×	(ESWA, 2025)	✓	✓	✓	✓			
CF-Now	Heu.	✓	(EJOR, 2024)							
CounterNet	Heu.	✓	(KDD, 2023)	✓						
NNCE	Inst.	✓	(TVCG, 2019)	✓						
MCCE	Inst.	✓	(DAMI, 2024)	✓						
LORE	Dec.-T	✓	(IS, 2019)							
Feature-Tweaking	Dec.-T	✓	(KDD, 2017)	✓						
FastAR	Stoch.	✓	(AAAI, 2022)	✓	✓	✓				
CCHVAE	Gen.	✓	(WWW, 2020)	✓	✓					
CounterGAN	Gen.	✓	(UAI, 2022)	✓	✓					
TABCF	Gen.	✓	(ICAIF, 2024)							
Marble	Heu.	✓	(–, –)	✓	✓	✓	✓	✓	✓	✓

A.1 Families of CE Methods

Generating CEs for tabular data requires balancing desirable properties such as confidence, proximity, and sparsity with domain constraints. Following Guidotti (2024), tabular CE methods can be organized into four families, with two additional families introduced since.

Optimization-based methods formulate CE generation as minimizing a differentiable objective over desired properties. Wachter et al. (2017) balances validity and proximity; CEM (Dhurandhar et al., 2018) adds elastic-net regularization and autoencoder reconstruction to promote manifold consistency, while its model-agnostic successor MACEM (Dhurandhar et al., 2019) approximates gradients from class-probability queries and handles numerical and categorical features. DiCE (Mothilal et al., 2020) additionally optimizes diversity and supports immutable features. DACE (Kanamori et al., 2020) uses mixed-integer linear optimization with Mahalanobis- and local-outlier-factor-based costs to favor CEs close to the empirical distribution. More recently, ReLU-CFx (Tsiourvas et al., 2024) formulates CE search over ReLU networks as a MIP, encoding the non-linear local outlier factor through mixed-integer constraints and decomposing the search over *live polytopes*; it requires white-box access to the network.

Heuristic-search-based methods define a cost function and make local, model-agnostic choices to reduce it, without a differentiable model. MOC frames generation as a multi-objective problem over confidence, proximity and sparsity, and plausibility (Dandl et al., 2020). Similarly, CERTIFAI uses a genetic algorithm that can incorporate feasible ranges and immutable features (Sharma et al., 2020). Recent work extends this line. CFM-FW runs NSGA-II over a five-term objective, predicted-output loss, proximity, sparsity, local outlier factor for manifold adherence, and a causality term, initializing its population with a KD-tree; the constraints it enforces are immutability, feasible ranges,

and monotonic (increasing or decreasing) directions, and reuses the resulting Pareto set to attribute feature importance (Liu et al., 2025). CF-Now emphasizes generality and speed: it is model- and data-agnostic, needs no training data; first finds a valid CE and then improves its proximity through tabu search, judging realism by mean absolute deviation and Mahalanobis distance (de Oliveira et al., 2024). Departing from the post-hoc paradigm, CounterNet instead trains the classifier and an amortized CE generator end-to-end so that explanations align with the model's decision boundary and are produced in a single forward pass rather than a per-instance search (Guo et al., 2023).

Instance-based methods return CE by selecting the most similar examples from a dataset. NNCE retrieves the nearest unlike neighbour (NUN), which gives both validity and in-distribution plausibility (Wexler et al., 2019). MCCE (Redelmeier et al., 2024) factorizes the joint distribution of the mutable features, conditioned on the immutable features and the target class, into univariate conditionals, each estimated by a CART regression tree, then Monte-Carlo samples a synthetic set per query and returns the low-cost valid samples as CE, scoring them by Gower distance and L_0 sparsity; relying on trees rather than gradients makes it model-agnostic .

Decision-tree-based methods fit an interpretable tree that approximates the black-box model and read CE off its structure. LORE (Guidotti et al., 2019) generates synthetic neighbors using a genetic algorithm, ensuring some share the same black-box prediction as the input instance while others do not. A decision tree is then trained on these neighbors and used to extract factual rules – *the path the instance follows to reach its prediction* – and a set of counterfactual rules with a different classification. While specific to tree-based ensemble models for binary classification, Feature-Tweaking (Tolomei et al., 2017) uses the internal structure of the ensemble to find the changes needed to move an instance from one class to another, over a set of actionable features. The goal is to find a counterfactual that most trees classify as the target class, by limiting how much each feature can change to meet the decision conditions in each tree.

Stochastic-control-based methods frame generation as an Markov decision process and learn a policy that minimizes the expected cost of reaching a target state. FastAR produces sequential CE and integrates causality through an SCM, so the recourse respects the order and cost of real-world actions rather than a single one-shot edit (Verma et al., 2022).

Generative methods use deep generative models to synthesize CE from a learned data distribution, targeting plausibility directly. CounteRGAN trains a residual GAN to produce realistic counterfactual samples (Nemirovsky et al., 2022), and C-CHVAE searches the latent space of a conditional variational autoencoder so that generated instances stay on the data manifold (Pawelczyk et al., 2020). TABCF (Panagiotou et al., 2024) uses a transformer-based variational autoencoder to generate CEs, implementing a Gumbel-Softmax architecture that eliminates a feature bias toward using only continuous features; however, this approach does not account for any constraints.

A.2 Desiderata of Counterfactuals

A CE is judged along several axes. *Validity* asks whether the target label is achieved, and *constraint satisfaction* asks whether the instance is admissible under the domain's constraints, whether immutability, boundary, or relational constraints. The two are distinct: an instance can flip the prediction yet violate one or more constraints (Simonetto et al., 2024b). *Confidence* measures how far beyond the decision threshold the target class is attained, so that the flip is robust rather than marginal; objectives that reward only crossing the boundary tend to leave the prediction barely inside the target region, which motivates loss functions that reward comfortably exceeding the threshold (Mothilal et al., 2020; Paredes Cetina et al., 2026). *Proximity* is the distance between the query and the CE under a chosen distance metric, kept small so the CE stays close to the original instance (Guidotti, 2024). *Sparsity* is the number of features that change, kept small so that the CE is easy to read (Verma et al., 2024); when the two conflict, sparsity is usually preferred (Wachter et al., 2017), favoring fewer features changed by larger amounts over many changes slightly.

However, the relative importance of these axes depends on the intended use of the CE, as illustrated in Table 5. Validity is a requirement across all scenarios, whereas the importance of the other properties varies. When a CE is intended to be acted upon, as in algorithmic recourse (Ustun et al., 2019) or clinical decision support (Bhattacharya et al., 2025) (see Table 5), proximity and sparsity are important because the prescribed changes should be limited and actionable. Constraint satisfaction is also critical: immutable and relational constraints ensure that the proposed changes are

Table 5: Importance of CE properties across application scenarios.

Scenario	Validity	Spars.	Prox.	Conf.	Immut.	Bound.	Rel. Const.
Algorithmic recourse	**Hard Req.**	High	High	Low	**Hard Req.**	High	**Hard Req.**
Clinical decision support	**Hard Req.**	High	High	High	High	**Hard Req.**	**Hard Req.**
Fairness / bias auditing	**Hard Req.**	Low	Low	High	Disabled*	Low	Low
Model debugging	**Hard Req.**	Low	Low	Low	Low	Low	Low
Business what-if	**Hard Req.**	Low	Low	Low	Low	High	**Hard Req.**
Synthetic data generation	**Hard Req.**	Low	Low	Low	Low	High	High

feasible, while clinical settings additionally require high confidence and adherence to physiological boundaries. Conversely, when CEs are used to interrogate the model, as in fairness or bias auditing (Sharma et al., 2020) or model debugging (Yousefzadeh & O'Leary, 2019), minimality is secondary to what the CE reveals about model behavior, reducing the importance of proximity and sparsity. In fairness auditing, for instance, immutability can be disabled for the sensitive attribute under investigation, allowing the CE to vary that attribute and reveal whether it affects the prediction. Model debugging by contrast, can benefit from low-confidence CE that sit close to the decision boundary, since these surface edge cases and unstable regions of the model. Finally, when CE populate a space of feasible alternatives rather than explain a single decision, as in business what-if analysis (Spinnato et al., 2022) or synthetic data generation (Hasan & Talbert, 2022), closeness to the query is irrelevant and proximity and sparsity are low, yet feasibility remains essential, since boundary and relational constraints must hold for the generated instances to represent coherent states of the domain. **Across these scenarios, relational constraints are particularly important whenever CEs are expected to represent feasible real-world instances and may be relaxed when CEs instead serve primarily as probes of model behavior, highlighting their role in determining whether generated alternatives are meaningful within the application domain.**

As reported in Table 4, existing CE methods mostly support immutable and boundary constraint types through a few mechanisms. Immutability is enforced by restricting the search to a subset of mutable features (Mothilal et al., 2020; Sharma et al., 2020). Boundary constraints are imposed by sampling values only from allowed ranges (Mothilal et al., 2020) or by projecting generated candidates back onto those ranges after perturbation (Le et al., 2020), and monotonicity is handled by constraining the direction of admissible changes (Verma et al., 2022; Liu et al., 2025). Beyond these hard restrictions, the realism of CE is pursued through proxies for closeness to the data distribution: penalizing the reconstruction error of an autoencoder (Dhurandhar et al., 2018), adding either the distance to the k nearest observed points as an objective (Dandl et al., 2020) or the local outlier factor (Liu et al., 2025), or minimizing the classifier's epistemic uncertainty (Schut et al., 2021); instance-based and generative methods pursue the same goal by selecting observed points from a reference dataset (Wexler et al., 2019) or sampling from a learned distribution (Nemirovsky et al., 2022). None of these mechanisms enforce user-specified relationships between multiple features. An example of an overlooked source of relational constraints is feature engineering. In deployed pipelines, an end user can typically alter only a subset of the model's inputs, while the remaining features are derived internally by domain-specific processing, as in credit-scoring systems where scores are computed from user-given values before reaching the model (Simonetto et al., 2024b). Such derived features cannot be perturbed freely and must stay consistent with the inputs from which they are computed. **Related constraint-handling mechanisms (including relation constraints) have been studied for adversarial examples on tabular data (Simonetto et al., 2024a). However, CE generation has a different objective: beyond changing the model prediction while satisfying domain constraints, it must optimize explanation-specific properties such as confidence and sparsity.**

B Further Details on MARBLE

This appendix provides additional technical detail on the MARBLE algorithm. Algorithm 1 gives the full pseudocode for the generation procedure, covering candidate selection, admission, candidate repair, and the call to ϵ-NSGA-III. Algorithm 2 describes the general-purpose GreedyAscent subroutine used in both the admission and repair steps.

Algorithm 1 Algorithm of MARBLE

Input: Classifier f; query x; reference set R; immutable set I; constraints Φ with per-constraint violation functions $\{v_\phi\}_{\phi\in\Phi}$; threshold θ; distance d; candidate count n; population size z; generations g; ϵ-percentile p; ϵ-cutoff τ; ϵ-decay β

if $\exists\, \phi \in \Phi_I : v_\phi(x) > 0$ **then**
  return FAIL
end if
$D \leftarrow$ the n instances of $\{r \in R : f(r) \neq f(x)\}$ nearest to x under d
if $D = \emptyset$ **then**
  return FAIL
end if
$r^\star \leftarrow \arg\max_{r\in D} P\big(f(C_{\mathbf{1}_r}) = f(r)\big)$ {*strongest donor*}
$\gamma^* \leftarrow$ GREEDYASCENT with substitution moves and confidence merit from $\mathbf{1}_{r^\star}$ {*admission, Section 3.2.2*}
if $\gamma^* = \perp$ **then**
  return FAIL
end if
$\Sigma \leftarrow \{\gamma^*, \mathbf{1}_{r^\star}\}$ {seeds}
if $\sum_k v_k(C_{\gamma^*}) > 0$ **then**
  $\gamma' \leftarrow$ GREEDYASCENT with substitution moves and lexicographic merit from γ^* {*repair, Section 3.2.3*}
  if $\gamma' \neq \perp$ **then**
    $\Sigma \leftarrow \{\gamma'\} \cup \Sigma$
  end if
end if
$C(x), S \leftarrow \varepsilon$-NSGA-III$(f, c, \Sigma, \theta, \Phi, z, g)$ {*optimization, Section 3.2.4*}
if $C(x) = \emptyset$ **then**
  return FAIL
end if
return $C(x)$

Algorithm 2 GREEDYASCENT$(\gamma, \mathcal{N}, \mu, \prec, A)$

Input: Initial genome γ; move set $\mathcal{N}$; merit μ with strict order $\prec$ (lower is better); acceptance predicate A
Output: An accepted genome, or $\perp$

loop
  if $A(\mu(\gamma))$ **then**
    return γ
  end if
  $\gamma' \leftarrow \arg\min_{\prec}\{\, \mu(\eta) : \eta \in \mathcal{N}(\gamma)\,\}$
  if $\mu(\gamma') \prec \mu(\gamma)$ **then**
    $\gamma \leftarrow \gamma'$
  else
    return $\perp$
  end if
end loop

C MORE ON EXPERIMENTS

This appendix provides supplementary material for the experimental evaluation. Section C.1 briefly describes the TABULARBENCH framework used to access datasets, constraints, and pre-trained models. Section C.2 formally defines the six quality criteria and ten metrics used to assess CE quality. Section C.3 reports the full hyperparameter configuration for MARBLE and all baseline methods.

C.1 TABULARBENCH

TABULARBENCH is a comprehensive open-source framework and benchmark designed to evaluate the adversarial robustness of tabular deep learning classification models (Simonetto et al., 2024b). It helps researchers and developers assess how vulnerable tabular models are to malicious data manipulations. Although its primary domain is adversarial robustness, its implementation exposes a Python API for accessing datasets along with their constraints, as well as models pre-trained on these datasets. For instance, TABULARBENCH supports five architectures:

- **TABNET** (Arik & Pfister, 2021), which employs a sequential, multi-step architecture that utilizes instance-wise attention for sparse feature selection at each decision step, subsequently aggregating these intermediate subnetwork representations to formulate the final prediction;
- **TABTRANSFORMER** (Huang et al., 2020), which leverages self-attention mechanisms to map categorical variables into interpretable contextual embeddings, significantly enhancing model robustness against noisy inputs;
- **TORCHRLN** (Shavitt & Segal, 2018) optimizes a customized loss through an efficient hyperparameter tuning scheme. By jointly learning continuous regularization coefficients for the network weights, TorchRLN effectively mitigates model sensitivity and induces a highly sparse architecture;
- **STG** (Yamada et al., 2020) integrates differentiable feature selection directly into the optimization pipeline by applying a probabilistic continuous relaxation of the ℓ_0 penalty to the input features.
- **VIME** (Yoon et al., 2020) addresses tabular representation learning by deploying deep encoder-predictor networks optimized via combined self-supervised and semi-supervised objectives.

Table 6: The model architectures of TABULARBENCH.

Model	Hyperparameters
TABTRANSFORMER	$hidden_dim$, n_layers, $learning_rate$, $norm$, $\theta_{\text{TABTRANSFORMER}}$
TABNET	n_d, n_steps, γ, cat_emb_dim, $n_independent$, n_shared, $momentum$, $mask_type$
TORCHRLN	$hidden_dim$, $depth$, $heads$, $weight_decay$, $learning_rate$, $dropout$
STG	$hidden_dims$, $learning_rate$, lam
VIME	p_m, α, K, β

TABULARBENCH also provides a set of datasets with domain constraints. These datasets are summarized in Table 7.

Table 7: Properties of the evaluated TABULARBENCH datasets, including feature dimensionality, attribute types** (with min–max values for numeric features corresponding to boundary constraints), number of immutable features ($|I|$), and number of relational constraints ($|\Phi|$) by category.

Dataset	Features	Categorical	Numeric**	Immutable ($\|I\|$)	Relational Constraints ($\|\Phi\|$)					
					Lin. Eq.	Lin. Ineq.	Non-lin.	Boolean	Safe-Div.	Total
URL	63	0	63	0	0	8	0	6	0	14
LCLD	28	5	23	14	0	2	5	1	1	9
WiDS	108	9	99	0	0	31	0	0	0	31
CTU	757	0	757	325	2	324	0	0	34	360

C.2 METRICS

We evaluate over a test set $\mathcal{T} \subseteq \mathcal{X}$ of query instances. For each query $x \in \mathcal{T}$ with predicted label $y = f(x)$, a method returns a counterfactual set $C(x)$, and we write $c \neq y$ for the target class of a counterfactual $C_j \in C(x)$. Feature k of a point z is denoted $z^{(k)}$, $k \in \{1, \ldots, p\}$. CE quality is assessed using five criteria and ten metrics, as follows:

1. *Validity **(VAL)***: Proportion of queries for which at least one counterfactual flipping the predicted class is found (higher is better):
$$\frac{1}{|\mathcal{T}|}\sum_{x\in\mathcal{T}}\mathbf{1}_{\left[\exists\, C_j\in C(x):\, f(C_j)\neq y\right]}.$$

2. *Sparsity **(SPA)***: Fraction of the p features altered between a counterfactual and its query, averaged over all counterfactuals and queries (lower is better, i.e. fewer changes):
$$\frac{1}{|\mathcal{T}|}\sum_{x\in\mathcal{T}}\frac{1}{|C(x)|}\sum_{C_j\in C(x)}\frac{1}{p}\sum_{k=1}^{p}\mathbf{1}_{\left[C_j^{(k)}\neq x^{(k)}\right]}.$$

3. *Proximity*: Distance $d(x, C_j)$ between a counterfactual and its query, reported under three choices of d. Numeric features are normalized by their observed range r_k so that heterogeneous scales are comparable, and categorical features contribute a mismatch indicator; we write
$$\delta_k(x, C_j) = \begin{cases} \dfrac{|C_j^{(k)} - x^{(k)}|}{r_k}, & k \text{ numeric}, \\ \mathbf{1}_{\left[C_j^{(k)}\neq x^{(k)}\right]}, & k \text{ categorical}. \end{cases}$$
We report (i) ℓ_1, the sum of per-feature deviations; (ii) ℓ_2, which penalizes larger deviations more heavily; and (iii) Gower (G), the mean per-feature deviation over the mixed numeric/categorical space. Lower values indicate closer, more actionable counterfactuals:
$$\ell_1 = \sum_{k=1}^{p}\delta_k, \qquad \ell_2 = \sqrt{\sum_{k=1}^{p}\delta_k^2}, \qquad \text{Gower} = \frac{1}{p}\sum_{k=1}^{p}\delta_k,$$
each averaged over $C_j \in C(x)$ and over queries as above.

4. *Counterfactual Confidence **(CONF)***: Mean confidence $P(f(C_j) = c)$ that the model assigns to the target class, averaged over all queries. Higher values indicate that counterfactuals land well inside the target decision region rather than marginally past the threshold θ:
$$\frac{1}{|\mathcal{T}|}\sum_{x\in\mathcal{T}}\frac{1}{|C(x)|}\sum_{C_j\in C(x)}P\big(f(C_j)=c\big).$$

5. *Constraint Satisfaction **(CS)***: Fraction of counterfactuals that respect the query's domain constraints. We report three families and their conjunction. **Immutability (IC)** checks that every immutable feature is preserved, $\mathbf{1}_{[\,\forall k\in I:\, C_j^{(k)}=x^{(k)}\,]}$; **relationship (CR)** checks that every relational constraint is satisfied, $\mathbf{1}_{[\,\forall\phi\in\Phi:\, v_\phi(C_j)=0\,]}$; and **boundary (BC)** checks that each feature stays within its admissible domain $\mathcal{X}_k$. The aggregate **CS** requires all three to hold simultaneously. For a family with predicate $\mathrm{sat}(\cdot)$, the score is (higher is better):
$$\frac{1}{|\mathcal{T}|}\sum_{x\in\mathcal{T}}\frac{1}{|C(x)|}\sum_{C_j\in C(x)}\mathbf{1}_{\left[\mathrm{sat}(C_j)\right]}.$$

Validity gating. Some methods return candidates that do not actually flip the predicted class. To prevent such non-flipping outputs from inflating quality scores, we compute all metrics (SPA, proximity, CONF, and CS) exclusively over valid counterfactuals (those with $f(C_j) \neq y$). Likewise, validity itself counts only instances for which at least one prediction-flipping candidate exists. This ensures that a method cannot appear sparse or proximal by returning cheap edits that never cross the decision boundary.

C.3 Parameter values

Table 8 reports the configuration used by each method in the benchmark. COUNTERGAN was excluded from the TabTransformer and TabNet combinations on `LCLD` and `WiDS`: both datasets contain categorical features, and embedding-based architectures cannot backpropagate classifier gradients through integer categorical lookups.

Table 8: Hyperparameter configuration for marble and all baseline methods on the benchmark.

Method	Parameter	Value
MARBLE	Confidence threshold θ	0.51
	Constraint handling	ϵ-level
	{Confidence, sparsity, proximity} objectives	all enabled
	Population size z	200
	Generation budget g	200
	Early stopping	enabled
	Donor-pool size n	100: 10,000 (`CTU`) ; 300,000 (`LCLD`)
DICE	Sampling strategy	random
	Counterfactuals per instance	1
NNCE	Counterfactuals per instance	1
COUNTERGAN	Strategy	`countergan`
	Discriminator steps	2
	Generator steps	4
	Training iterations	2,000
CCHVAE	Latent dimension	2
	Intermediate dimension	5
	Categorical latent dimension	3
	Learning rate	0.001
	Training epochs	80
	Batch size	100
	Counterfactuals per instance	1
	Search samples	1,000
	Max search iterations	500
	Step size / ℓ-norm	0.5 / 2
FASTAR	PPO timesteps	500,000
	Coefficient of the manifold-distance penalty	0.1

C.4 Reproducibility

Hardware. All experiments specifically for MARBLE were conducted on a MacBook Pro equipped with an Apple M4 Max chip (14 cores) and 36 GB of unified memory, running macOS 26.5. No GPU acceleration was used; all computations were performed on the CPU.

Sampling. For each (dataset, model) pair, we draw a stratified sample of 1,000 test instances with equal representation per predicted class; this *benchmark* split forms the basis of all reported results. The *ablation* split is a smaller stratified draw of 100 instances taken from within the benchmark split, so that ablation studies, which are substantially more expensive per instance, evaluate on a strict subset of the reported rows rather than on disjoint ones. A separate set of 100 *tuning* instances is drawn from the test rows that remain after excluding the benchmark split, ensuring the hyperparameter sweep never sees the rows the paper reports on. All three draws use independent random number generator streams with per-class seeds, so that a change to one class's quota cannot shift another class's selection. The donor pool from which MARBLE assembles CE features is drawn from the training set: candidates are ranked by NUN proximity, and the top-ranked instances form the pool available to the evolutionary search.

D FULL BENCHMARK RESULTS

This section reports the full benchmark results broken down per model. Figure 5 shows the breakdown of constraint satisfaction of the immutability family. Tables 9–13 present results for STG, TabNet, TabTransformer, TorchRLN, and VIME respectively, each evaluated across all four datasets. Table 14 further disaggregates relational constraint satisfaction by subcategory, averaged over models. All metrics are computed on valid CEs only. Noteworthy, COUNTERGAN is excluded from the TabNet and TabTransformer columns on `LCLD` and `WiDS`: the method backpropagates the generator loss through the classifier, but embedding-based architectures route categorical features through discrete integer lookups that break the gradient path.

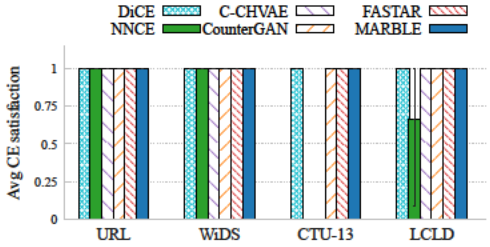


Figure 5: Constraint satisfaction of *immutable* family per dataset (bars) and average across models (error bars: std). The constraints are only measured on valid CE.

Table 9: Benchmark results of model STG. – means the method did not generate results.

Dataset	Method	Val↑	Conf↑	Spars↓	Prox$_{\ell_1}$↓	Prox$_{\ell_2}$↓	Prox$_{G}$↓	CS↑	IC↑	BC↑	RC↑
URL	DICE	1.00	0.664	0.035	1.478	1.031	0.023	0.86	1.00	0.91	0.95
	NNCE	1.00	0.829	0.354	4.570	1.728	0.073	1.00	1.00	1.00	1.00
	C-CHVAE	1.00	0.617	1.000	4.655	1.308	0.074	0.00	1.00	0.00	1.00
	COUNTERGAN	0.96	0.906	0.713	4.646	1.232	0.074	0.68	1.00	1.00	0.68
	FASTAR	0.61	0.516	0.016	0.544	0.544	0.009	1.00	1.00	1.00	1.00
	MARBLE	1.00	0.890	0.089	1.698	0.966	0.027	1.00	1.00	1.00	1.00
WiDS	DICE	1.00	0.530	0.083	4.669	1.640	0.043	0.45	1.00	0.80	0.50
	NNCE	1.00	0.674	0.779	16.020	2.911	0.148	0.99	1.00	1.00	0.99
	C-CHVAE	1.00	0.568	0.956	16.738	2.825	0.155	0.96	1.00	0.96	1.00
	COUNTERGAN	0.82	0.708	0.839	12.701	1.819	0.118	0.00	1.00	1.00	0.00
	FASTAR	0.01	0.506	0.009	0.715	0.715	0.007	0.77	1.00	1.00	0.77
	MARBLE	1.00	0.825	0.136	6.379	2.137	0.059	1.00	1.00	1.00	1.00
CTU-13	DICE	0.61	0.551	0.017	6.187	1.909	0.008	0.00	1.00	0.32	0.00
	NNCE	0.00	–	–	–	–	–	–	–	–	–
	C-CHVAE	0.00	–	–	–	–	–	–	–	–	–
	COUNTERGAN	0.00	–	–	–	–	–	–	–	–	–
	FASTAR	0.22	0.509	0.001	0.938	0.938	0.001	0.00	1.00	1.00	0.00
	MARBLE	0.62	0.748	0.011	2.481	1.382	0.003	1.00	1.00	1.00	1.00
LCLD	DICE	0.61	0.522	0.083	1.554	1.040	0.055	0.36	1.00	0.98	0.36
	NNCE	0.00	–	–	–	–	–	–	–	–	–
	C-CHVAE	0.58	0.502	0.467	9.728	4.597	0.347	0.00	1.00	0.07	0.00
	COUNTERGAN	0.01	0.505	0.419	0.644	0.263	0.023	0.00	1.00	1.00	0.00
	FASTAR	0.13	0.501	0.036	0.486	0.486	0.017	0.00	1.00	1.00	0.00
	MARBLE	0.62	0.550	0.297	2.722	1.347	0.097	1.00	1.00	1.00	1.00

Table 10: Benchmark results of model TabNet. – means the method did not generate results; × means the method is incompatible with the dataset.

Dataset	Method	Val↑	Conf↑	Spars↓	Prox$_{\ell_1}$↓	Prox$_{\ell_2}$↓	Prox$_{G}$↓	CS↑	IC↑	BC↑	RC↑
URL	DICE	1.00	0.813	0.028	1.051	0.831	0.017	0.80	1.00	0.94	0.86
	NNCE	1.00	0.887	0.362	4.675	1.747	0.074	1.00	1.00	1.00	1.00
	C-CHVAE	1.00	0.632	1.000	4.628	1.301	0.073	0.00	1.00	0.00	1.00
	COUNTERGAN	0.69	0.833	0.704	1.715	0.358	0.027	0.66	1.00	1.00	0.66
	FASTAR	0.72	0.651	0.019	0.234	0.225	0.004	0.98	1.00	1.00	0.98
	MARBLE	1.00	0.936	0.076	0.895	0.538	0.014	1.00	1.00	1.00	1.00
WiDS	DICE	1.00	0.728	0.016	0.724	0.586	0.007	0.74	1.00	0.97	0.77
	NNCE	1.00	0.708	0.774	15.497	2.816	0.143	0.99	1.00	1.00	0.99
	C-CHVAE	1.00	0.547	0.955	16.163	2.767	0.150	0.96	1.00	0.96	1.00
	COUNTERGAN	×	×	×	×	×	×	×	×	×	×
	FASTAR	0.00	–	–	–	–	–	–	–	–	–
	MARBLE	1.00	0.958	0.176	2.972	0.983	0.028	1.00	1.00	1.00	1.00
CTU-13	DICE	1.00	0.957	0.002	0.794	0.680	0.001	0.07	1.00	0.86	0.07
	NNCE	0.00	–	–	–	–	–	–	–	–	–
	C-CHVAE	0.00	–	–	–	–	–	–	–	–	–
	COUNTERGAN	0.41	1.000	0.218	2.641	0.282	0.003	0.00	1.00	1.00	0.00
	FASTAR	0.90	0.987	0.001	0.011	0.011	0.000	0.00	1.00	0.44	0.00
	MARBLE	0.98	0.965	0.012	1.608	0.927	0.002	1.00	1.00	1.00	1.00
LCLD	DICE	1.00	0.627	0.058	0.804	0.665	0.029	0.40	1.00	0.99	0.40
	NNCE	0.03	0.521	0.679	6.133	2.200	0.219	0.00	0.00	1.00	0.00
	C-CHVAE	0.94	0.508	0.454	5.418	2.564	0.194	0.00	1.00	0.30	0.00
	COUNTERGAN	×	×	×	×	×	×	×	×	×	×
	FASTAR	0.36	0.581	0.036	0.048	0.048	0.002	0.00	1.00	1.00	0.00
	MARBLE	1.00	0.764	0.214	0.688	0.429	0.025	1.00	1.00	1.00	1.00

Table 11: Benchmark results of model TabTransformer. – means the method did not generate results; × means the method is incompatible with the dataset.

Dataset	Method	Val↑	Conf↑	Spars↓	Prox$_{\ell_1}$↓	Prox$_{\ell_2}$↓	Prox$_{G}$↓	CS↑	IC↑	BC↑	RC↑
URL	DICE	1.00	0.917	0.027	1.061	0.847	0.017	0.77	1.00	0.95	0.81
	NNCE	1.00	0.952	0.363	4.721	1.747	0.075	1.00	1.00	1.00	1.00
	C-CHVAE	1.00	0.707	1.000	4.797	1.329	0.076	0.00	1.00	0.00	1.00
	COUNTERGAN	0.95	0.962	0.683	3.603	1.079	0.057	0.59	1.00	1.00	0.59
	FASTAR	0.97	0.643	0.017	0.279	0.269	0.004	1.00	1.00	1.00	1.00
	MARBLE	1.00	0.950	0.072	0.728	0.430	0.012	1.00	1.00	1.00	1.00
WiDS	DICE	1.00	0.644	0.051	3.051	1.385	0.028	0.56	1.00	0.92	0.59
	NNCE	1.00	0.863	0.778	15.877	2.874	0.147	0.99	1.00	1.00	0.99
	C-CHVAE	1.00	0.732	0.956	18.542	2.960	0.172	0.77	1.00	0.77	1.00
	COUNTERGAN	×	×	×	×	×	×	×	×	×	×
	FASTAR	0.00	–	–	–	–	–	–	–	–	–
	MARBLE	1.00	0.978	0.187	5.495	1.831	0.051	1.00	1.00	1.00	1.00
CTU-13	DICE	0.77	0.550	0.040	14.027	2.762	0.019	0.00	1.00	0.20	0.00
	NNCE	0.00	–	–	–	–	–	–	–	–	–
	C-CHVAE	0.00	–	–	–	–	–	–	–	–	–
	COUNTERGAN	0.01	0.930	0.222	6.874	1.228	0.009	0.00	1.00	1.00	0.00
	FASTAR	0.12	0.504	0.001	0.401	0.401	0.001	0.00	1.00	1.00	0.00
	MARBLE	0.98	0.774	0.017	4.696	1.772	0.006	1.00	1.00	1.00	1.00
LCLD	DICE	1.00	0.861	0.057	0.862	0.716	0.031	0.36	1.00	0.98	0.36
	NNCE	0.00	–	–	–	–	–	–	–	–	–
	C-CHVAE	1.00	0.530	0.448	3.595	1.701	0.128	0.00	1.00	0.27	0.00
	COUNTERGAN	×	×	×	×	×	×	×	×	×	×
	FASTAR	0.33	0.545	0.036	0.164	0.164	0.006	0.00	1.00	1.00	0.00
	MARBLE	1.00	0.868	0.216	0.898	0.652	0.032	1.00	1.00	1.00	1.00

Table 12: Benchmark results of model TorchRLN. – means the method did not generate results.

Dataset	Method	Val↑	Conf↑	Spars↓	Prox$_{\ell_1}$↓	Prox$_{\ell_2}$↓	Prox$_{G}$↓	CS↑	IC↑	BC↑	RC↑
URL	DICE	1.00	0.936	0.029	1.127	0.864	0.018	0.86	1.00	0.97	0.89
	NNCE	1.00	0.939	0.360	4.672	1.736	0.074	1.00	1.00	1.00	1.00
	C-CHVAE	1.00	0.693	1.000	4.763	1.333	0.076	0.00	1.00	0.00	1.00
	COUNTERGAN	0.79	0.938	0.724	3.593	1.019	0.057	0.74	1.00	1.00	0.74
	FASTAR	0.86	0.615	0.016	0.333	0.333	0.005	0.94	1.00	1.00	0.94
	MARBLE	1.00	0.945	0.070	0.672	0.421	0.011	1.00	1.00	1.00	1.00
WiDS	DICE	1.00	0.615	0.059	3.527	1.501	0.033	0.50	1.00	0.83	0.56
	NNCE	1.00	0.831	0.779	16.072	2.901	0.149	1.00	1.00	1.00	1.00
	C-CHVAE	1.00	0.627	0.956	16.802	2.824	0.156	0.91	1.00	0.91	1.00
	COUNTERGAN	0.88	0.855	0.847	11.861	1.748	0.110	0.00	1.00	1.00	0.00
	FASTAR	0.31	0.509	0.009	0.468	0.468	0.004	0.57	1.00	1.00	0.57
	MARBLE	1.00	0.956	0.180	6.618	2.142	0.061	1.00	1.00	1.00	1.00
CTU-13	DICE	0.86	0.976	0.003	1.305	0.896	0.002	0.10	1.00	0.84	0.10
	NNCE	0.00	–	–	–	–	–	–	–	–	–
	C-CHVAE	0.00	–	–	–	–	–	–	–	–	–
	COUNTERGAN	0.09	0.987	0.231	1.150	0.115	0.002	0.00	1.00	1.00	0.00
	FASTAR	0.76	0.827	0.001	0.411	0.411	0.001	0.00	1.00	1.00	0.00
	MARBLE	0.97	0.943	0.024	4.699	1.567	0.006	1.00	1.00	1.00	1.00
LCLD	DICE	1.00	0.918	0.057	0.836	0.696	0.030	0.39	1.00	1.00	0.39
	NNCE	0.01	0.590	0.429	2.727	1.428	0.097	0.00	1.00	1.00	0.00
	C-CHVAE	0.74	0.518	0.448	1.899	0.969	0.068	0.00	1.00	0.41	0.00
	COUNTERGAN	0.12	0.569	0.412	0.275	0.115	0.010	0.00	1.00	1.00	0.00
	FASTAR	1.00	0.594	0.036	0.070	0.070	0.002	0.00	1.00	1.00	0.00
	MARBLE	1.00	0.809	0.171	0.228	0.169	0.008	1.00	1.00	1.00	1.00

Table 13: Benchmark results of model VIME. – means the method did not generate results.

Dataset	Method	Val↑	Conf↑	Spars↓	Prox$_{\ell_1}$↓	Prox$_{\ell_2}$↓	Prox$_{G}$↓	CS↑	IC↑	BC↑	RC↑
URL	DICE	0.90	0.660	0.034	1.382	0.994	0.022	0.75	1.00	0.91	0.82
	NNCE	1.00	0.793	0.358	4.698	1.754	0.075	1.00	1.00	1.00	1.00
	C-CHVAE	1.00	0.626	1.000	4.839	1.336	0.077	0.00	1.00	0.00	1.00
	COUNTERGAN	0.87	0.900	0.696	4.386	1.151	0.070	0.60	1.00	1.00	0.60
	FASTAR	0.59	0.516	0.016	0.508	0.508	0.008	0.24	1.00	1.00	0.24
	MARBLE	1.00	0.876	0.106	2.028	1.030	0.032	1.00	1.00	1.00	1.00
WiDS	DICE	1.00	0.546	0.071	4.070	1.568	0.038	0.43	1.00	0.83	0.47
	NNCE	1.00	0.755	0.780	16.455	2.946	0.152	0.99	1.00	1.00	0.99
	C-CHVAE	1.00	0.626	0.956	17.817	2.901	0.165	0.96	1.00	0.96	1.00
	COUNTERGAN	0.92	0.768	0.849	14.102	2.165	0.131	0.00	1.00	1.00	0.00
	FASTAR	0.00	–	–	–	–	–	–	–	–	–
	MARBLE	1.00	0.889	0.150	5.255	1.816	0.049	1.00	1.00	1.00	1.00
CTU-13	DICE	0.99	0.853	0.008	3.140	1.390	0.004	0.04	1.00	0.65	0.04
	NNCE	0.00	–	–	–	–	–	–	–	–	–
	C-CHVAE	0.00	–	–	–	–	–	–	–	–	–
	COUNTERGAN	0.00	–	–	–	–	–	–	–	–	–
	FASTAR	0.00	–	–	–	–	–	–	–	–	–
	MARBLE	0.99	0.903	0.013	3.119	1.445	0.004	1.00	1.00	1.00	1.00
LCLD	DICE	1.00	0.866	0.057	0.876	0.729	0.031	0.35	1.00	0.99	0.35
	NNCE	0.02	0.642	0.429	2.009	1.112	0.072	1.00	1.00	1.00	1.00
	C-CHVAE	0.78	0.507	0.452	2.194	1.099	0.078	0.00	1.00	0.29	0.00
	COUNTERGAN	0.11	0.549	0.412	0.364	0.162	0.013	0.00	1.00	1.00	0.00
	FASTAR	0.36	0.530	0.036	0.178	0.178	0.006	0.00	1.00	1.00	0.00
	MARBLE	1.00	0.777	0.195	0.532	0.411	0.019	1.00	1.00	1.00	1.00

E STAGE PERFORMANCE

Tables 15–18 report the stage flow for each dataset across all five models. For URL, WiDS, and LCLD (except STG), the pipeline is effectively resolved at Stage 1: all 1000 instances are admitted and already satisfy all constraints, leaving repair unnecessary, while optimization still improves

Table 14: Per-subcategory relational-constraint satisfaction (%) on valid counterfactuals, averaged over contributing models. A valid counterfactual scores 1 for a subcategory only if it satisfies *every* constraint of that subcategory; models with no valid counterfactual are excluded. '-' marks a subcategory the dataset does not define. n/a means no counterfactuals was generated.

Dataset	**Method**	Lin. Eq.	Lin. Ineq.	Non-lin.	Boolean	Safe-Div.
URL	DICE	-	86.6	-	100.0	-
	NNCE	-	100.0	-	100.0	-
	C-CHVAE	-	100.0	-	99.7	-
	COUNTERGAN	-	65.3	-	100.0	-
	FASTAR	-	83.1	-	100.0	-
	MARBLE	-	100.0	-	100.0	-
WiDS	DICE	-	57.9	-	-	-
	NNCE	-	99.4	-	-	-
	C-CHVAE	-	100.0	-	-	-
	COUNTERGAN	-	0.2	-	-	-
	FASTAR	-	67.0	-	-	-
	MARBLE	-	100.0	-	-	-
CTU-13	DICE	43.9	12.6	-	-	95.8
	NNCE	n/a	n/a	-	-	n/a
	C-CHVAE	n/a	n/a	-	-	n/a
	COUNTERGAN	0.0	0.0	-	-	3.0
	FASTAR	30.7	6.6	-	-	100.0
	MARBLE	100.0	100.0	-	-	100.0
LCLD	DICE	-	85.3	37.2	100.0	100.0
	NNCE	-	100.0	33.3	100.0	100.0
	C-CHVAE	-	90.2	0.0	100.0	100.0
	COUNTERGAN	-	95.7	0.0	100.0	100.0
	FASTAR	-	98.8	0.0	100.0	99.5
	MARBLE	-	100.0	100.0	100.0	100.0

confidence, sparsity, and proximity. LCLD with STG is the only exception, with 372 instances rejected at admission and 13 requiring repair, resulting in 625 solved instances. CTU is the only dataset exercising all three stages across every model. Admission is near-universal for four models (992–1000 admitted), while STG rejects 380 queries. No admitted candidate satisfies all constraints, making repair necessary in every case. Repair success ranges from 44.9% (TabNet, 449/999) to 78.6% (TabTrans, 780/992), while Stage 3 recovers many unrepaired instances. For example, VIME repairs 614/1000 candidates but ultimately solves 992, with optimization contributing 378 additional valid constrained CEs. Overall, final coverage on CTU ranges from 61.9% (STG) to 99.2% (VIME).

Table 15: Pipeline stage flow for URL across models.

Stage	STG	TabNet	TabTrans	VIME	TorchRLN
Queries	1000	1000	1000	1000	1000
Admitted	1000	1000	1000	1000	1000
Rejected	0	0	0	0	0
Constraint-satisfied	1000	1000	1000	1000	1000
Need repair	0	0	0	0	0
Repaired	0	0	0	0	0
Unrepaired	0	0	0	0	0
Solved	1000	1000	1000	1000	1000
Unresolved	0	0	0	0	0
Failed	0	0	0	0	0

Table 16: Pipeline stage flow for WiDS across models.

Stage	STG	TabNet	TabTrans	VIME	TorchRLN
Queries	1000	1000	1000	1000	1000
Admitted	1000	1000	1000	1000	1000
Rejected	0	0	0	0	0
Constraint-satisfied	1000	1000	1000	1000	1000
Need repair	0	0	0	0	0
Repaired	0	0	0	0	0
Unrepaired	0	0	0	0	0
Solved	1000	1000	1000	1000	1000
Unresolved	0	0	0	0	0
Failed	0	0	0	0	0

Table 17: Pipeline stage flow for CTU across models.

Stage	STG	TabNet	TabTrans	VIME	TorchRLN
Queries	1000	1000	1000	1000	1000
Admitted	620	1000	992	1000	992
Rejected	380	0	8	0	8
Constraint-satisfied	0	1	0	0	2
Need repair	620	999	992	1000	990
Repaired	398	449	780	614	727
Unrepaired	222	550	212	386	263
Solved	619	979	985	992	973
Unresolved	1	21	7	8	19
Failed	381	21	15	8	27

Table 18: Pipeline stage flow for LCLD across models.

Stage	STG	TabNet	TabTrans	VIME	TorchRLN
Queries	1000	1000	1000	1000	1000
Admitted	628	1000	1000	1000	1000
Rejected	372	0	0	0	0
Constraint-satisfied	615	1000	1000	1000	1000
Need repair	13	0	0	0	0
Repaired	10	0	0	0	0
Unrepaired	3	0	0	0	0
Solved	625	1000	1000	1000	1000
Unresolved	3	0	0	0	0
Failed	375	0	0	0	0

F Benchmark Results on Tabular Foundation Models

The benchmark results, evaluated across multiple replicate seeds (1, 2, 3, 4, and 42), indicate that MARBLE consistently achieves high validity and constraint satisfaction across the Tabular Foundation Model (TFM) architectures (Tables 19-20). The NNCE baseline exhibits comparable performance on the URL and WIDS datasets but demonstrates a reduction in constraint satisfaction on the LCLD dataset, yielding a score of 0.67. A similar trend is observed with DICE, which adheres to URL constraints effectively but exhibits degraded CS performance on the WIDS and LCLD datasets (0.35 and 0.33, respectively).

Comparing the performance of these counterfactual baselines on TFMs against standard tabular architectures reveals that NNCE benefits from the foundational models. Specifically, using NNCE

with the TABICL model improves its constraint satisfaction on the LCLD dataset from 0.00 to 0.67, indicating that TFMs can partially mitigate the baseline's limitations in feature constraint adherence.

Table 19: Benchmark results for the TABICL foundation model. The reported metrics are averaged across multiple replicate seeds, where each seed controls the redrawing of both the explained instances and the foundation model's in-context training set.

Dataset	Method	Val↑	Conf↑	Spars↓	Prox$_{\ell_1}$↓	Prox$_{\ell_2}$↓	Prox$_{G}$↓	CS↑
URL	DICE	$0.98_{\pm 0.00}$	$0.629_{\pm 0.002}$	$\mathbf{0.042}_{\pm 0.001}$	$1.465_{\pm 0.016}$	$0.930_{\pm 0.006}$	$0.023_{\pm 0.000}$	$0.82_{\pm 0.01}$
	NNCE	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.996}_{\pm 0.000}$	$0.365_{\pm 0.002}$	$4.709_{\pm 0.040}$	$1.748_{\pm 0.008}$	$0.075_{\pm 0.001}$	$\mathbf{1.00}_{\pm 0.00}$
	MARBLE	$\mathbf{1.00}_{\pm 0.00}$	$0.896_{\pm 0.000}$	$0.085_{\pm 0.001}$	$\mathbf{0.823}_{\pm 0.006}$	$\mathbf{0.499}_{\pm 0.002}$	$\mathbf{0.013}_{\pm 0.000}$	$\mathbf{1.00}_{\pm 0.00}$
WIDS	DICE	$0.93_{\pm 0.02}$	$0.535_{\pm 0.003}$	$\mathbf{0.085}_{\pm 0.028}$	$\mathbf{3.891}_{\pm 1.203}$	$\mathbf{1.337}_{\pm 0.178}$	$\mathbf{0.036}_{\pm 0.011}$	$0.35_{\pm 0.05}$
	NNCE	$\mathbf{1.00}_{\pm 0.00}$	$0.879_{\pm 0.053}$	$0.781_{\pm 0.001}$	$17.230_{\pm 0.114}$	$2.991_{\pm 0.015}$	$0.160_{\pm 0.001}$	$0.99_{\pm 0.00}$
	MARBLE	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.907}_{\pm 0.009}$	$0.195_{\pm 0.005}$	$5.655_{\pm 0.112}$	$1.470_{\pm 0.019}$	$0.052_{\pm 0.001}$	$\mathbf{1.00}_{\pm 0.00}$
LCLD	DICE	$0.98_{\pm 0.02}$	$0.555_{\pm 0.014}$	$\mathbf{0.057}_{\pm 0.002}$	$0.789_{\pm 0.030}$	$0.654_{\pm 0.023}$	$0.028_{\pm 0.001}$	$0.33_{\pm 0.02}$
	NNCE	$\mathbf{1.00}_{\pm 0.00}$	$0.585_{\pm 0.056}$	$0.426_{\pm 0.006}$	$1.705_{\pm 0.119}$	$1.004_{\pm 0.124}$	$0.061_{\pm 0.004}$	$0.67_{\pm 0.47}$
	MARBLE	$0.99_{\pm 0.00}$	$\mathbf{0.676}_{\pm 0.010}$	$0.224_{\pm 0.010}$	$\mathbf{0.571}_{\pm 0.067}$	$\mathbf{0.342}_{\pm 0.042}$	$\mathbf{0.020}_{\pm 0.002}$	$\mathbf{0.99}_{\pm 0.00}$

Table 20: Benchmark results for the TABPFN foundation model. The reported metrics are averaged across multiple replicate seeds, where each seed controls the redrawing of both the explained instances and the foundation model's in-context training set.

Dataset	Method	Val↑	Conf↑	Spars↓	Prox$_{\ell_1}$↓	Prox$_{\ell_2}$↓	Prox$_{G}$↓	CS↑
URL	DICE	$\mathbf{1.00}_{\pm 0.00}$	$0.645_{\pm 0.004}$	$\mathbf{0.048}_{\pm 0.002}$	$1.716_{\pm 0.085}$	$0.999_{\pm 0.022}$	$0.027_{\pm 0.001}$	$0.80_{\pm 0.01}$
	NNCE	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.971}_{\pm 0.001}$	$0.365_{\pm 0.002}$	$4.700_{\pm 0.049}$	$1.747_{\pm 0.009}$	$0.075_{\pm 0.001}$	$\mathbf{1.00}_{\pm 0.00}$
	MARBLE	$\mathbf{1.00}_{\pm 0.00}$	$0.938_{\pm 0.001}$	$0.116_{\pm 0.000}$	$\mathbf{1.355}_{\pm 0.008}$	$\mathbf{0.696}_{\pm 0.003}$	$\mathbf{0.022}_{\pm 0.000}$	$\mathbf{1.00}_{\pm 0.00}$
WIDS	DICE	$0.99_{\pm 0.00}$	$0.552_{\pm 0.004}$	$\mathbf{0.064}_{\pm 0.008}$	$\mathbf{2.999}_{\pm 0.347}$	$\mathbf{1.212}_{\pm 0.079}$	$\mathbf{0.028}_{\pm 0.003}$	$0.36_{\pm 0.04}$
	NNCE	$\mathbf{1.00}_{\pm 0.00}$	$0.698_{\pm 0.007}$	$0.792_{\pm 0.002}$	$19.567_{\pm 0.338}$	$3.245_{\pm 0.037}$	$0.181_{\pm 0.003}$	$0.98_{\pm 0.01}$
	MARBLE	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.969}_{\pm 0.008}$	$0.259_{\pm 0.010}$	$7.556_{\pm 0.552}$	$1.792_{\pm 0.091}$	$0.070_{\pm 0.005}$	$\mathbf{1.00}_{\pm 0.00}$
LCLD	DICE	$0.99_{\pm 0.01}$	$0.556_{\pm 0.022}$	$\mathbf{0.060}_{\pm 0.002}$	$0.793_{\pm 0.046}$	$0.644_{\pm 0.030}$	$0.028_{\pm 0.002}$	$0.35_{\pm 0.10}$
	NNCE	$\mathbf{1.00}_{\pm 0.00}$	$0.584_{\pm 0.000}$	$0.429_{\pm 0.000}$	$1.639_{\pm 0.000}$	$0.867_{\pm 0.000}$	$0.059_{\pm 0.000}$	$0.00_{\pm 0.00}$
	MARBLE	$\mathbf{1.00}_{\pm 0.00}$	$\mathbf{0.678}_{\pm 0.010}$	$0.221_{\pm 0.016}$	$\mathbf{0.607}_{\pm 0.095}$	$\mathbf{0.364}_{\pm 0.059}$	$\mathbf{0.022}_{\pm 0.003}$	$\mathbf{1.00}_{\pm 0.00}$

G Ablation Studies

We conduct three ablation studies to validate MARBLE's architectural design and property guarantees. To isolate the impact of each component, we use a strict paired-trajectory design: each ablated variant shares the exact random seed and parameters of the full baseline, ensuring performance differences are solely attributable to the ablated mechanism. In Table 21 we show the fixed marble configuration shared by every ablation arm. Studies were performed on STG as it is the hardest model to generate CEs for (See 5.3). We evaluate on a stratified subsample of 100 queries drawn from the test set, with none overlapping the benchmark split.

- **Elite Archive.** We isolate the elite archive's contribution through three arms built up from bare optimization: the first extracts CEs from the final population only; the second adds the validated-seed fallback; the third is the full method, adding the elite archive. Because ϵ-selection can disregard constraint-satisfying individuals after they are found, the bare arm exposes how often the final population loses all feasible solutions, the seed-fallback arm measures how many losses the pre-search witness recovers, and the full method shows the archive makes the fallback unnecessary by retaining every feasible individual ever evaluated. This validates the non-emptiness guarantee: the archive ensures a run returns the best feasible solution found, rather than one lost to selection pressure.
- **Relational Constraints Formulation.** We run MARBLE without giving the relational constraints Φ for the search, rendering both the repair operator ρ and the inequality bounds g_k useless. We evaluate the resulting counterfactuals post-hoc against the true constraints to analyze whether the feasibility MARBLE attains is actively enforced by these mechanisms, or would instead be inherited from the reference-data distribution, in which case explicit constraint handling would be unnecessary.

- **Objective Space Dimensionality.** Under a drop-one design, we disable each objective: confidence, sparsity, and proximity in turn, re-scoring every resulting solution in the full three-objective space. This isolates each objective's contribution to the solution quality; for proximity in particular, it tests whether MARBLE's NUN sampling already bounds distance enough to make an explicit proximity objective unnecessary and whether the extra objective dilutes selection pressure on confidence and sparsity.

Table 21: Fixed MARBLE configuration for all ablation studies.

Parameter	**Sym.**	**Value**	**Parameter**	**Sym.**	**Value**
Confidence threshold	θ	0.51	Crossover rate	p_c	0.9
Donor-pool size	n	100,000	Mutation rate	p_m	0.9
NUN candidate pool	—	100,000	Early stopping	—	enabled
Population size	z	200	ϵ-cutoff fraction	—	0.7
Generation budget	g	200	ϵ-decay exponent	—	5.0
Proximity distance	d	Gower	ϵ-percentile	—	75

Elite archive ablation. We build up from bare optimization to the full method in three arms (Table 22). On URL, WiDS and LCLD the validity differences are negligible, but CTU exposes the archive's role. Bare optimization solves only 49 of 100 instances: ϵ-relaxed selection discards the constraint-satisfying seed in roughly half of all GA runs, leaving the final population with no valid solution (Table 23). The seed fallback recovers 50 of those 51 failures by returning the pre-search witness, raising validity to 0.99. The elite archive closes the remaining gap at 1.00 by retaining every constraint-satisfying individual the search ever evaluated. The archive also improves solution quality. On WiDS, validity is 1.00 in all three arms, yet the full method yields sparser counterfactuals (sparsity 0.105 vs. 0.182) because extraction selects the best individual across all generations, not just those surviving in the final population. In both cases the archive delivers what the non-emptiness property asserts: the best constraint-satisfying solution the search ever found, whether or not the final population retained it.

Table 22: Elite-archive ablation: building up from bare optimization to the full method. Metrics are computed over valid instances only.

Dataset	**Arm**	Val↑	Conf↑	Spars↓	Prox$_G$ ↓	CS↑
URL	only optimization	**1.00**	0.987	0.0463	**0.0318**	**1.00**
	+ seed fallback	**1.00**	0.987	0.0463	**0.0318**	**1.00**
	MARBLE (+ elite archive)	**1.00**	**0.989**	**0.0457**	0.0332	**1.00**
WiDS	only optimization	**1.00**	0.932	0.1817	0.0828	**1.00**
	+ seed fallback	**1.00**	0.932	0.1817	0.0828	**1.00**
	MARBLE (+ elite archive)	**1.00**	**0.936**	**0.1049**	**0.0726**	**1.00**
CTU	only optimization	0.49	0.920	0.0108	0.0053	**1.00**
	+ seed fallback	0.99	0.719	0.0444	**0.0187**	**1.00**
	MARBLE (+ elite archive)	**1.00**	**0.729**	**0.0421**	0.0189	**1.00**
LCLD	only optimization	0.46	0.525	0.1879	0.0817	**1.00**
	+ seed fallback	**0.47**	**0.526**	0.1611	0.0690	**1.00**
	MARBLE (+ elite archive)	**0.47**	**0.526**	**0.1520**	**0.0677**	**1.00**

Table 23: Elite-archive inner-mechanism stages (STG/CTU-13, per 100 explained instances). Each row adds one safety net to the bare optimization. *Final-pop feasible*: a feasible individual survives in the final population; *Archive feasible*: the elite archive holds a feasible individual; *Archive rescue*: final-pop infeasible ∧ archive feasible; *Seed fallback*: empty extractions rescued by the pre-GA candidate ("—" = not present).

Arm	Final-pop feasible	Archive feasible	Archive rescue	Seed fallback	Validity
only optimization	49	0	0	—	0.49
+ seed fallback	49	0	0	50	0.99
MARBLE (+ elite archive)	50	100	**50**	0	1.00

Relational constraint ablation. Our second ablation asks whether MARBLE manages to respect the relational constraints even when it is not given them. Table 24 show that in URL, removing the constraint set changes essentially nothing: MARBLE reaches the same counterfactuals either way (CR stays at 1.00) because it flips the label by editing features outside the constraint set, so the constraints were never binding. The other three datasets tell a different story, in every case dominated by a collapse in relational-constraint satisfaction. In WiDS, dropping the constraints sharply lowers CR ($1.00 \rightarrow 0.43$) without quality benefit in return: sparsity and proximity are slightly worse ($0.105 \rightarrow 0.106$, $0.0726 \rightarrow 0.0740$) and validity is unchanged. LCLD shows a validity-for-feasibility trade-off: a large drop in CR ($1.00 \rightarrow 0.30$) buys sparser counterfactuals ($0.152 \rightarrow 0.097$) and a slight validity gain ($0.47 \rightarrow 0.54$); the gain is modest because the immutable-feature constraints remain active and keep limiting reachability. In CTU, removing constraints also degrades feasibility significantly (CR $1.00 \rightarrow 0.20$); the unconstrained arm edits fewer features (sparsity $0.042 \rightarrow 0.033$) but moves farther from the original input ($Prox_G$ $0.189 \rightarrow 0.203$). Validity remains 1.00 throughout. In short, on these models the relational constraints cost almost no reachability, only LCLD gains validity without them, yet they are what keep the returned counterfactuals relationally feasible. In Table 25 we provide a breakdown of the relational constraint satisfaction per subcategory, and it exhibits a similar behavior like the baselines in the benchmark (See Table 14), where boolean and safe-division constraints were respected while linear equations, linear inequalities and non-linear inequalities suffered the biggest degradation. This shows that relying simply in generating in-distribution CE is not sufficient to generate fully feasible CEs.

Table 24: Relational-constraint ablation: with dataset constraints vs. an empty constraint set. Metrics are computed over valid instances only.

Dataset	**Arm**	Val↑	Conf↑	Spars↓	Prox$_G$ ↓	CS↑	CR↑	BC↑	IC↑
URL	MARBLE	**1.00**	**0.989**	**0.0457**	0.0332	**1.00**	**1.00**	**1.00**	**1.00**
	no constraints	**1.00**	**0.989**	0.0459	**0.0330**	**1.00**	**1.00**	**1.00**	**1.00**
WiDS	MARBLE	**1.00**	0.936	**0.1049**	**0.0726**	**1.00**	**1.00**	**1.00**	**1.00**
	no constraints	**1.00**	**0.939**	0.1057	0.0740	0.43	0.43	**1.00**	**1.00**
CTU	MARBLE	**1.00**	0.729	0.0421	**0.0189**	**1.00**	**1.00**	**1.00**	**1.00**
	no constraints	**1.00**	**0.809**	**0.0327**	0.0203	0.20	0.20	**1.00**	**1.00**
LCLD	MARBLE	0.47	0.526	0.1520	**0.0677**	**1.00**	**1.00**	**1.00**	**1.00**
	no constraints	**0.54**	**0.527**	**0.0966**	0.0678	0.30	0.30	**1.00**	**1.00**

Objective-space ablation. Our findings in Table 26 show that the three objectives in MARBLE are complementary as no ablated arm dominates the full configuration across all metrics simultaneously. Removing confidence causes the most severe degradation, with Conf dropping from 0.989 to 0.591 on URL, 0.936 to 0.534 on WiDS, and 0.729 to 0.543 on CTU; the effect is small on LCLD (0.526 to 0.517) because the CE already lie near the decision boundary, leaving little margin for further improvement. While the confidence-ablated arm achieves the best sparsity and proximity on several datasets, it does so at a severe confidence cost. Removing sparsity consistently worsens Spars, most notably on WiDS (0.1049 to 0.2369) and LCLD (0.1520 to 0.3075), confirming that without explicit

Table 25: Per-subcategory relational-constraint satisfaction (%) on valid counterfactuals for the relational-constraint ablation (STG). A counterfactual satisfies a subcategory only if it satisfies all constraints within it. '-' indicates that the dataset does not define the subcategory.

Dataset	Arm	Lin. Eq.	Lin. Ineq.	Non-lin.	Boolean	Safe-Div.
URL	no constraints	-	100.0	-	100.0	-
WiDS	no constraints	-	43.0	-	-	-
CTU	no constraints	34.0	22.0	-	-	99.0
LCLD	no constraints	-	53.7	29.6	100.0	100.0

Table 26: Objective-space ablation. Metrics are computed over valid instances only.

Dataset	Arm	Val↑	Conf↑	Spars↓	Prox_G ↓	CS↑
URL	MARBLE	**1.00**	0.989	0.0457	0.0332	**1.00**
	no proximity	**1.00**	**0.991**	0.0471	0.0369	**1.00**
	no confidence	**1.00**	0.591	**0.0283**	**0.0101**	**1.00**
	no sparsity	**1.00**	0.917	0.0990	0.0300	**1.00**
WiDS	MARBLE	**1.00**	0.936	0.1049	0.0726	**1.00**
	no proximity	**1.00**	**0.949**	0.1710	0.1022	**1.00**
	no confidence	**1.00**	0.534	**0.0887**	**0.0208**	**1.00**
	no sparsity	**1.00**	0.903	0.2369	0.0782	**1.00**
CTU	MARBLE	**1.00**	0.729	**0.0421**	0.0189	**1.00**
	no proximity	**1.00**	**0.731**	**0.0421**	0.0192	**1.00**
	no confidence	0.99	0.543	0.0436	**0.0171**	**1.00**
	no sparsity	0.98	0.623	0.0456	0.0177	**1.00**
LCLD	MARBLE	**0.47**	0.526	0.1520	**0.0677**	**1.00**
	no proximity	**0.47**	0.526	**0.1512**	0.0703	**1.00**
	no confidence	**0.47**	0.517	0.2878	0.0767	**1.00**
	no sparsity	**0.47**	**0.539**	0.3075	0.0799	**1.00**

pressure the optimizer modifies additional features. The proximity objective plays a subtler but still meaningful role. For example, its removal on WiDS increases both Prox_G (0.0726 to 0.1022) and Spars (0.1049 to 0.1710), suggesting that proximity pressure indirectly regularises sparsity. By penalizing the overall distance to the original instance, the optimizer is discouraged from spreading small perturbations across many features, which in turn keeps sparsity low. This coupling is not uniform across datasets; on URL and CTU the sparsity increase is marginal, showing that the sparsity objective alone is enough in those cases. Still, the WiDS result shows that the objectives can reinforce each other, further supporting the full three-objective formulation. Finally, for all valid CEs, all constraint-satisfaction metrics (CS, CR, BC, IC) remain at 1.00 across every arm and dataset, confirming that feasibility is enforced by the constraint-handling mechanism independently of the objective formulation.

H Sensitivity Analysis

To verify that MARBLE is robust to hyperparameter variations and operational extremes, we evaluate its sensitivity across four domains. In Table 27 we show the default configuration used for MARBLE across different runs, as well as the sweep values used per experiment. We perform all experiments using the same structure as in Section G (STG and same subsample).

- **Operational Robustness (θ and Distance).** Sweeping the target confidence threshold ($\theta \in [0.51, 0.99]$) demonstrates that unlike gradient-based baselines, MARBLE maintains stable feasibility at high-confidence extremes by exploring deeper donor combinations. Finally, replacing Gower distance with L_1 or L_2 norms degrades categorical stability, empirically validating Gower as the optimal metric for mixed tabular schemas.

- **Donor Pool Size.** Evaluating the multi-donor fallback tier confirms an explicit trade-off: expanding the `donor_pool_size` increases the combinatorial search space, predictably improving the probability of finding a θ-crossing assignment (reachability) at the marginal cost of localized plausibility.
- **Evolutionary Compute Scaling (z, g,).** Increasing population size (z) and generation budget (g) monotonically improves Pareto hypervolume up to diminishing returns without destabilizing constraint satisfaction.

Table 27: Fixed marble configuration for the sensitivity analysis and sweep range.

Parameter	**Sym.**	**Default**	**Sweep range**
Confidence threshold	θ	0.51	$\{0.51, 0.6, 0.7, 0.8, 0.9, 0.99\}$
Donor-pool size	n	100000	$\{1, 10, 50, 100, 500, 1000, 5000, 10000, 50000, \infty\}$
Proximity distance	d	Gower	{Gower, Euclidean, Manhattan}
Population size	z	200	$\{10, 25, 50, 100, 200, 400\}$
Generation budget	g	200	$\{10, 25, 50, 100, 200, 400\}$
Held fixed throughout: {confidence, sparsity, proximity} objectives all enabled; early stopping enabled; crossover rate $= 0.9$; mutation rate $= 0.9$ NUN candidate pool $= 100000$; ϵ-cutoff fraction $= 0.7$; ϵ-decay exponent $= 5.0$; ϵ-percentile $= 75$; seed $= 1$.			

Table 28: Sensitivity to proximity distance metric. Metrics are computed over valid instances only

Dataset	**Distance**	Val↑	Conf↑	Spars↓	Prox$_{\ell_1}$ ↓	Prox$_{\ell_2}$ ↓	Prox$_G$ ↓	CS↑
URL	Gower	**1.00**	0.987	0.0449	**1.982**	**1.232**	**0.0315**	**1.00**
	Euclidean	**1.00**	**0.989**	0.0467	2.252	1.370	0.0357	**1.00**
	Manhattan	**1.00**	0.988	**0.0444**	2.195	1.362	0.0348	**1.00**
WiDS	Gower	**1.00**	0.933	**0.1102**	**7.651**	**2.575**	**0.0708**	**1.00**
	Euclidean	**1.00**	0.944	0.1556	10.470	3.025	0.0969	**1.00**
	Manhattan	**1.00**	**0.947**	0.1519	10.543	3.052	0.0976	**1.00**
CTU	Gower	**1.00**	0.728	**0.0425**	**14.214**	**3.265**	**0.0188**	**1.00**
	Euclidean	**1.00**	0.731	0.0432	14.506	3.328	0.0192	**1.00**
	Manhattan	**1.00**	**0.733**	0.0428	14.336	3.303	0.0189	**1.00**
LCLD	Gower	0.47	**0.528**	0.1543	**1.945**	**1.129**	**0.0695**	**1.00**
	Euclidean	0.47	0.527	**0.1520**	1.947	1.143	0.0695	**1.00**
	Manhattan	**0.51**	0.526	0.1625	2.103	1.176	0.0751	**1.00**

The distance d enters MARBLE both as the proximity objective $o_3 = d(x, \delta(\gamma))$ and, through the NUN donor ranking, as the retrieval metric; sweeping it over Gower, Euclidean and Manhattan (see Table 28) shows MARBLE is robust to the choice. Validity, confidence and all four constraint families are essentially unchanged across the three metrics: URL, WiDS and CTU are fully covered, and constraint satisfaction metric stays at 1.00 apart from the LCLD boundary term (≈ 0.98), with confidence moving by at most 0.014 (on WiDS). The only validity difference is on LCLD, where Manhattan reaches 0.51 against 0.47 for the other two metrics. The one systematic effect is on proximity, where Gower yields the lowest distance under all three families (ℓ_1, ℓ_2 and Gower) on every dataset, including the two it does not optimize, so it returns the most proximal counterfactuals without affecting any other property.

Increasing θ from 0.51 to 0.99 (see Table 29) consistently raises confidence, but the search produces larger and denser edits to achieve it, while constraint satisfaction and validity stay at 1.00 in every case where a CE is found. On URL, edits grow modestly (sparsity from 0.0449 to 0.0565, Gower proximity from 0.0315 to 0.0383). WiDS shows a similar but smaller change (sparsity $0.1102 \rightarrow 0.1105$, Gower proximity $0.0708 \rightarrow 0.0743$ at $\theta = 0.9$). The more significant effect is on validity. URL remains fully valid at all thresholds. WiDS holds at 1.00 up to $\theta = 0.9$ then drops to 0 at $\theta = 0.99$. CTU stays at 1.00 through $\theta = 0.8$, drops to 0.74 at $\theta = 0.9$, and reaches 0 at $\theta = 0.99$. LCLD is the most sensitive: validity starts at only 0.47 at $\theta = 0.51$, falls to 0.13 at $\theta = 0.6$, and reaches 0 for any $\theta \geq 0.7$. Under the fixed search budget (population 200, 200 generations), an aggressive confidence target can therefore make the constrained search completely infeasible.

Table 29: Sensitivity to the target-class confidence threshold θ. Metrics are computed over valid instances only.

Dataset	θ	Val↑	Conf↑	Spars↓	Prox$_G$ ↓	CS↑
URL	0.51	**1.00**	0.987	**0.0449**	0.0315	**1.00**
	0.6	**1.00**	0.987	0.0452	**0.0313**	**1.00**
	0.7	**1.00**	0.988	0.0451	0.0323	**1.00**
	0.8	**1.00**	0.988	0.0451	0.0322	**1.00**
	0.9	**1.00**	0.988	**0.0449**	0.0318	**1.00**
	0.99	**1.00**	**0.993**	0.0565	0.0383	**1.00**
WiDS	0.51	**1.00**	0.933	0.1102	**0.0708**	**1.00**
	0.6	**1.00**	0.935	0.1122	0.0717	**1.00**
	0.7	**1.00**	0.937	**0.1091**	0.0724	**1.00**
	0.8	**1.00**	0.938	0.1133	0.0727	**1.00**
	0.9	**1.00**	**0.941**	0.1105	0.0743	**1.00**
	0.99	0.00	–	–	–	–
CTU	0.51	**1.00**	0.728	**0.0425**	**0.0188**	**1.00**
	0.6	**1.00**	0.775	0.0450	0.0210	**1.00**
	0.7	**1.00**	0.823	0.0484	0.0239	**1.00**
	0.8	**1.00**	0.870	0.0552	0.0291	**1.00**
	0.9	0.74	**0.927**	0.0559	0.0282	**1.00**
	0.99	0.00	–	–	–	–
LCLD	0.51	**0.47**	0.528	**0.1543**	**0.0695**	**1.00**
	0.6	0.13	**0.605**	0.2637	0.1200	**1.00**
	0.7	0.00	–	–	–	–
	0.8	0.00	–	–	–	–
	0.9	0.00	–	–	–	–
	0.99	0.00	–	–	–	–

Varying the donor-pool size n, i.e., the number of NUN instances from which candidate values are drawn, from 1 to unbounded (see Table 30) mainly affects validity, while constraint satisfaction remains at 1.00. On URL and WiDS, results are stable: validity remains 1.00, confidence is nearly constant ($\approx$ 0.987 and 0.93–0.94, respectively), and sparsity and Gower proximity vary only slightly. In CTU, validity remains 0.50 up to $n = 1000$, rises to 0.85 at $n = 5000$, peaks at 0.96 at $n = 10000$, then falls to 0.90 at $n = 50000$ and 0.60 when unbounded. The additional validity comes from harder instances, with lower confidence (0.736) and larger edits (Gower 0.019 at $n = 10000$). This non-monotonic behaviour shows that larger pools do not necessarily yield better results, making n worth tuning. On LCLD, validity increases from 0.45 at $n = 1$ to 0.64, plateauing at $n = 5000$, as more donor values enable additional reachable flips. Runtime also increases with n: negligibly on URL and WiDS, but on CTU from about three minutes for small pools to 30 minutes at $n = 10000$ and 4.5 hours when unbounded, despite lower validity.

Table 31 reports sensitivity to the NSGA-III population size z, and adds two front-level indicators: the dominated hypervolume (HV), normalised to $[0, 1]$, which measures how much of the objective space the returned Pareto front covers, and the mean front cardinality $|C(x)|$, the average number of CEs returned per instance. Results point to z=100–200 as a practical sweet spot. HV gains beyond this range are small on three of four datasets (URL: 0.956 at z=100 vs. 0.959 at z=400; CTU: 0.692 vs. 0.696; LCLD: 0.481 vs. 0.487), while only WiDS continues to gain noticeably (0.856 $\rightarrow$ 0.905). Sparsity decreases monotonically with z, confidence shows only minor gains beyond z=100, and Gower proximity can worsen at larger populations (Prox$_G$ on URL rises from 0.0310 to 0.0336 at z=400), indicating that very large populations shift search effort toward front diversity at the cost of proximity. A notable exception is CTU: at z=10 validity is 0.90, increasing to 1.00 starting at $z\geq$50, showing that harder instances require a minimum population to reliably find feasible CEs. Finally, $|C(x)|$ grows roughly proportionally to z, meaning larger populations directly give the user more alternative CEs, which may itself justify a moderately large z even after HV gains become negligible.

Table 30: Sensitivity to the donor pool size n. Metrics are computed over valid instances only.

Dataset	n	Val↑	Conf↑	Spars↓	Prox$_G$ ↓	CS↑	Time(s)↓
URL	1	**1.00**	0.987	0.0465	0.0343	**1.00**	30
	10	**1.00**	0.988	0.0479	0.0332	**1.00**	34
	50	**1.00**	0.987	0.0457	0.0325	**1.00**	34
	100	**1.00**	0.987	0.0451	0.0326	**1.00**	34
	500	**1.00**	0.987	0.0446	0.0323	**1.00**	38
	1000	**1.00**	**0.989**	**0.0432**	0.0334	**1.00**	36
	5000	**1.00**	0.987	0.0449	**0.0315**	**1.00**	38
	10000	**1.00**	0.987	0.0449	**0.0315**	**1.00**	39
	50000	**1.00**	0.987	0.0449	**0.0315**	**1.00**	39
	All	**1.00**	0.987	0.0449	**0.0315**	**1.00**	38
WiDS	1	**1.00**	0.931	**0.1056**	0.0727	**1.00**	62
	10	**1.00**	0.939	0.1173	0.0762	**1.00**	66
	50	**1.00**	**0.941**	0.1184	0.0756	**1.00**	68
	100	**1.00**	0.939	0.1156	0.0741	**1.00**	71
	500	**1.00**	0.937	0.1121	0.0744	**1.00**	72
	1000	**1.00**	0.938	0.1124	0.0733	**1.00**	74
	5000	**1.00**	0.936	0.1093	0.0722	**1.00**	80
	10000	**1.00**	0.934	0.1092	0.0710	**1.00**	85
	50000	**1.00**	0.933	0.1102	**0.0708**	**1.00**	88
	All	**1.00**	0.933	0.1102	**0.0708**	**1.00**	89
CTU	1	0.50	0.906	0.0101	**0.0055**	**1.00**	173
	10	0.50	**0.944**	0.0094	0.0066	**1.00**	303
	50	0.50	0.941	0.0095	0.0062	**1.00**	365
	100	0.50	0.942	**0.0078**	0.0062	**1.00**	370
	500	0.50	0.939	0.0081	0.0060	**1.00**	416
	1000	0.50	0.936	0.0080	0.0058	**1.00**	397
	5000	0.85	0.764	0.0375	0.0166	**1.00**	1464
	10000	**0.96**	0.736	0.0430	0.0190	**1.00**	1903
	50000	0.90	0.751	0.0403	0.0179	**1.00**	5257
	All	0.60	0.864	0.0208	0.0110	**1.00**	16143
LCLD	1	0.45	0.532	**0.1063**	**0.0500**	**1.00**	86
	10	0.56	0.555	0.1429	0.0710	**1.00**	108
	50	0.57	**0.557**	0.1460	0.0738	**1.00**	115
	100	0.60	0.554	0.1577	0.0757	**1.00**	119
	500	0.61	0.544	0.1452	0.0702	**1.00**	130
	1000	0.62	0.543	0.1452	0.0711	**1.00**	141
	5000	**0.64**	0.535	0.1417	0.0706	**1.00**	162
	10000	**0.64**	0.535	0.1456	0.0736	**1.00**	167
	50000	**0.64**	0.535	0.1496	0.0749	**1.00**	208
	All	**0.64**	0.534	0.1479	0.0769	**1.00**	306

We report the sensitivity to the generation budget g on Table 32. Our findings show that MARBLE converges faster along this axis than along the population size. On most datasets, g=50 is enough. HV stabilizes by g=50 on URL (0.957) and by g=100 on WiDS (0.884) and LCLD (0.486), with no further gains at g=400; on CTU, HV peaks early at g=25 (0.709) and slightly regresses thereafter, suggesting that additional generations do not help once the feasible region has been found. The metric trends are consistent: sparsity and proximity improve from g=10 to g=50 but remain essentially flat beyond that point across all four datasets. Unlike the population-size sweep, increasing g does not monotonically grow the front cardinality $|C(x)|$: it peaks at intermediate budgets (g=50–100) and then decreases, e.g., from 826 to 633 on LCLD and from 682 to 366 on CTU. This happens because longer evolutionary runs allow dominated solutions to be pruned from the archive, yielding a more compact but better-quality front. Together with the population-size results, these findings suggest that z controls the breadth of the search while g controls its depth, and that moderate values of both (z=100–200, g=50–100) are sufficient for MARBLE to produce well-converged Pareto fronts.

Table 31: Sensitivity to the NSGA-III population size z. Metrics are computed over valid instances only.

Dataset	z	Val↑	Conf↑	Spars↓	$\text{Prox}_G \downarrow$	CS↑	HV↑	$\lvert C(x)\rvert$
URL	10	**1.00**	0.984	0.1111	0.0366	**1.00**	0.885	50
	25	**1.00**	0.985	0.0684	0.0316	**1.00**	0.933	97
	50	**1.00**	0.986	0.0517	0.0315	**1.00**	0.950	148
	100	**1.00**	0.986	0.0456	**0.0310**	**1.00**	0.956	227
	200	**1.00**	0.987	**0.0449**	0.0315	**1.00**	0.957	328
	400	**1.00**	**0.989**	0.0451	0.0336	**1.00**	**0.959**	457
WiDS	10	**1.00**	0.932	0.3648	0.1114	**1.00**	0.611	92
	25	**1.00**	0.931	0.2631	0.0913	**1.00**	0.715	141
	50	**1.00**	0.931	0.1955	0.0805	**1.00**	0.791	205
	100	**1.00**	0.934	0.1402	0.0742	**1.00**	0.856	296
	200	**1.00**	0.933	0.1102	**0.0708**	**1.00**	0.885	396
	400	**1.00**	**0.938**	**0.0951**	0.0711	**1.00**	**0.905**	577
CTU-13	10	0.90	0.692	0.0627	0.0208	**1.00**	0.639	4
	25	0.98	0.719	0.0478	0.0191	**1.00**	0.681	38
	50	**1.00**	0.722	0.0440	**0.0187**	**1.00**	0.688	107
	100	**1.00**	0.726	0.0431	0.0188	**1.00**	0.692	199
	200	**1.00**	0.728	0.0425	0.0188	**1.00**	0.695	373
	400	**1.00**	**0.728**	**0.0421**	0.0188	**1.00**	**0.696**	639
LCLD	10	**0.47**	**0.529**	0.2470	0.0777	**1.00**	0.418	100
	25	**0.47**	0.528	0.2021	0.0713	**1.00**	0.452	212
	50	**0.47**	0.528	0.1679	0.0698	**1.00**	0.475	300
	100	**0.47**	0.526	0.1565	**0.0678**	**1.00**	0.481	472
	200	**0.47**	0.528	0.1543	0.0695	**1.00**	**0.488**	772
	400	**0.47**	0.527	**0.1535**	0.0685	**1.00**	0.487	1164

I TRANSFERABILITY OF THE REPAIR PROJECTION TO BASELINE METHODS

In this section, we aimed to determine whether MARBLE's constraint satisfaction comes primarily from its repair projection or from the interplay of its other mechanisms, we applied the repair operator as a post-hoc step to the CEs generated by the five baselines and re-evaluate them. Table 33 reports the results. We report the same results as in Table 1 where invalid CEs are assigned 0 CS.

In a few favorable cases the repair operator alone is remarkably effective. On the URL dataset, C-CHVAE jumps from CS $= 0.00$ to 0.97 while validity drops only marginally ($1.00 \rightarrow 0.98$), and COUNTERGAN improves from 0.48 to 0.76 with no validity loss at all. On LCLD, FASTAR gains meaningful constraint satisfaction ($0.00 \rightarrow 0.33$) at a modest validity cost ($0.44 \rightarrow 0.37$).

However, in the majority cases, the repair destabilizes the validity. Adjusting feature values to satisfy relational constraints shifts CEs away from the model's decision boundary, causing predictions to revert to the original class. DICE on WiDS drops from perfect validity (1.00) to 0.81; C-CHVAE on LCLD collapses from 0.81 to 0.24. The effect is most severe on CTU-13, where DICE falls from 0.85 to 0.17 and FASTAR from 0.40 to 0.01, causing CS to remain near zero.

These results confirm that the projection repair operator is a necessary but not sufficient component of MARBLE's constraint compliance. Applying it in isolation improves CS on some dataset/method pairs, yet it cannot recover validity once features have been perturbed. In contrast, MARBLE achieves simultaneous validity and constraint satisfaction because constraints are enforced throughout generation rather than applied as a post-hoc correction.

Table 32: Sensitivity to the NSGA-III generation budget g. Metrics are computed over valid instances only.

Dataset	g	Val↑	Conf↑	Spars↓	$\text{Prox}_G \downarrow$	CS↑	HV↑	$\lvert C(x)\rvert$
URL	10	**1.00**	0.985	0.0619	**0.0310**	**1.00**	0.935	121
	25	**1.00**	0.986	**0.0446**	**0.0310**	**1.00**	0.956	296
	50	**1.00**	0.988	0.0449	0.0321	**1.00**	**0.957**	339
	100	**1.00**	0.987	0.0451	0.0317	**1.00**	**0.957**	335
	200	**1.00**	0.987	0.0449	0.0315	**1.00**	**0.957**	328
	400	**1.00**	**0.988**	**0.0446**	0.0320	**1.00**	**0.957**	314
WiDS	10	**1.00**	0.922	0.2568	0.0952	**1.00**	0.724	95
	25	**1.00**	**0.935**	0.1640	0.0801	**1.00**	0.820	201
	50	**1.00**	0.934	0.1113	0.0716	**1.00**	0.883	375
	100	**1.00**	0.934	0.1107	0.0713	**1.00**	0.884	393
	200	**1.00**	0.933	**0.1102**	**0.0708**	**1.00**	**0.885**	396
	400	**1.00**	0.934	0.1139	0.0721	**1.00**	0.883	383
CTU-13	10	**1.00**	0.726	0.0477	0.0195	**1.00**	0.695	182
	25	**1.00**	**0.740**	0.0426	0.0198	**1.00**	**0.709**	393
	50	**1.00**	0.729	**0.0421**	0.0189	**1.00**	0.697	682
	100	**1.00**	0.728	0.0425	0.0189	**1.00**	0.695	486
	200	**1.00**	0.728	0.0425	**0.0188**	**1.00**	0.695	373
	400	**1.00**	0.728	0.0426	**0.0188**	**1.00**	0.695	366
LCLD	10	**0.47**	**0.531**	0.2188	0.0804	**1.00**	0.447	152
	25	**0.47**	0.529	0.1786	0.0713	**1.00**	0.475	496
	50	**0.47**	0.527	0.1596	0.0685	**1.00**	0.485	826
	100	**0.47**	0.527	0.1543	0.0692	**1.00**	0.486	834
	200	**0.47**	0.528	0.1543	0.0695	**1.00**	**0.488**	772
	400	**0.47**	0.527	**0.1505**	**0.0680**	**1.00**	0.487	633

Table 33: Effect of post-hoc projection repair on baseline constraint satisfaction. Primed columns ($'$) show scores after applying MARBLE's projection repair operator to the baseline's counterfactuals. Each cell is the mean over models ($x_{\pm s}$). A method that produces no valid counterfactual scores 0 for both validity and constraint satisfaction.

Dataset	**Method**	Val↑	Val↑′	CS↑	CS↑′
URL	DICE	$0.98_{\pm 0.05}$	$0.91_{\pm 0.08}$	$0.79_{\pm 0.08}$	$0.85_{\pm 0.11}$
	NNCE	$1.00_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$	$1.00_{\pm 0.00}$	$\mathbf{1.00}_{\pm 0.00}$
	C-CHVAE	$1.00_{\pm 0.00}$	$0.98_{\pm 0.02}$	$0.00_{\pm 0.00}$	$0.97_{\pm 0.03}$
	COUNTERGAN	$0.85_{\pm 0.12}$	$0.86_{\pm 0.11}$	$0.48_{\pm 0.12}$	$0.76_{\pm 0.18}$
	FASTAR	$0.75_{\pm 0.16}$	$0.66_{\pm 0.32}$	$0.65_{\pm 0.31}$	$0.63_{\pm 0.36}$
	MARBLE	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$
WiDS	DICE	$1.00_{\pm 0.00}$	$0.81_{\pm 0.08}$	$0.54_{\pm 0.13}$	$0.66_{\pm 0.13}$
	NNCE	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$0.99_{\pm 0.00}$	$1.00_{\pm 0.00}$
	C-CHVAE	$1.00_{\pm 0.00}$	$0.97_{\pm 0.04}$	$0.91_{\pm 0.08}$	$0.95_{\pm 0.07}$
	COUNTERGAN	$0.87_{\pm 0.05}$	$0.52_{\pm 0.48}$	$0.00_{\pm 0.00}$	$0.46_{\pm 0.42}$
	FASTAR	$0.07_{\pm 0.14}$	$0.07_{\pm 0.14}$	$0.04_{\pm 0.08}$	$0.07_{\pm 0.14}$
	MARBLE	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$	$1.00_{\pm 0.00}$
CTU-13	DICE	$0.85_{\pm 0.16}$	$0.17_{\pm 0.14}$	$0.04_{\pm 0.04}$	$0.02_{\pm 0.02}$
	NNCE	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	C-CHVAE	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	COUNTERGAN	$0.10_{\pm 0.17}$	$0.09_{\pm 0.17}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	FASTAR	$0.40_{\pm 0.40}$	$0.01_{\pm 0.01}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	MARBLE	$0.91_{\pm 0.16}$	$0.91_{\pm 0.16}$	$0.91_{\pm 0.16}$	$0.91_{\pm 0.16}$
LCLD	DICE	$0.92_{\pm 0.17}$	$0.67_{\pm 0.16}$	$0.34_{\pm 0.07}$	$0.49_{\pm 0.14}$
	NNCE	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
	C-CHVAE	$0.81_{\pm 0.17}$	$0.24_{\pm 0.11}$	$0.00_{\pm 0.00}$	$0.08_{\pm 0.06}$
	COUNTERGAN	$0.08_{\pm 0.06}$	$0.09_{\pm 0.11}$	$0.00_{\pm 0.00}$	$0.01_{\pm 0.01}$
	FASTAR	$0.44_{\pm 0.33}$	$0.37_{\pm 0.36}$	$0.00_{\pm 0.00}$	$0.33_{\pm 0.39}$
	MARBLE	$0.93_{\pm 0.17}$	$0.93_{\pm 0.17}$	$0.93_{\pm 0.17}$	$0.93_{\pm 0.17}$