Counterfactual Explanations for Incomplete Inputs Using Flow-Based Generative Models

Francesco Leofante[1], Daniel Neider[2,3] and Mustafa Yalçıner[2,3,*]

[1]Department of Computing, Imperial College London, UK
[2]TU Dortmund University, August-Schmidt-Straße 1, Dortmund, Germany
[3]Center for Trustworthy Data Science and Security, University Alliance Ruhr, Dortmund, Germany

Abstract

Existing algorithms for the generation of Counterfactual Explanations (CXs) for Machine Learning (ML) models typically operate on fully specified inputs; however, in real-world applications, inputs may contain missing values. Imputation methods can be used to mitigate this issue; however, recent work has shown that imputation can produce invalid CXs, ultimately jeopardizing their explanatory function. To address this challenge, we propose a framework for computing CXs for inputs with missing values using flow-based generative models and Mixed-Integer Linear Programming (MILP). Experiments on four real-world datasets demonstrate that our method is able to generate CXs with higher validity, and lower cost than state-of-the-art methods relying on imputation.

Keywords

Counterfactual Explanations, Incomplete Inputs, Normalizing Flows

1. Introduction

Counterfactual explanations answer a simple question: what should minimally change in the input for a model to predict something else? In interpretability settings, this makes CXs attractive because they offer concrete, human-readable recourse suggestions [1]. Yet most CX methods are designed for a world in which every input feature is known. Real deployments rarely match that ideal. Measurements can be unavailable, records may be partially redacted, and sensors may break. A standard reaction to missing values is to replace them with estimates and then run an off-the-shelf CX method. That pipeline is convenient, but it can be misleading. The counterfactual computed on the imputed instance may not correspond to a successful action on the true underlying state. In other words, the explanation can look valid in the completed feature space while failing in the system where it is actually applied [2, 3].

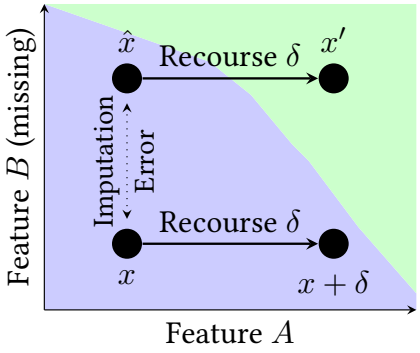

Figure 1: Imputation inaccuracy hinders validity

Why this matters. Consider a user who receives a recommendation after some feature was missing at explanation time. If that recommendation is based on an estimate rather than the real value, then the suggested action may not move the actual instance across the desired decision boundary. This is illustrated in Figure 1, where the green area is the desired classification region, and the blue area is undesired. There, applying the recourse δ that did alter the classification on the estimated value $\hat{x}$, does not alter the

Late-breaking work, Demos and Doctoral Consortium, colocated with the 4th World Conference on eXplainable Artificial Intelligence: July 01–03, 2026, Fortaleza, Brazil
*Corresponding author.
f.leofante@imperial.ac.uk (F. Leofante); daniel.neider@tu-dortmund.de (D. Neider); mustafa.yalciner@tu-dortmund.de (M. Yalçıner)
0000-0001-8245-9429 (F. Leofante); 0000-0001-9276-6342 (D. Neider); 0009-0005-6240-7062 (M. Yalçıner)

classification when applied on the true input x. The resulting explanation is therefore risky: the recourse action may be invalidated when applied to the true but partially hidden instance.

Our approach. We study this problem by building a counterfactual generation procedure that does not treat missingness as an afterthought. Instead, the method reasons about the unknown entries while constructing the counterfactual, using a density model to keep candidate solutions near the data manifold and a MILP encoding to search for explanations that satisfy the desired classification constraint.

In the remainder of the paper, we first place the problem in context, then introduce the technical background needed for our formulation, present the proposed optimization model, and finally evaluate the method on several benchmark datasets against recent baselines.

2. Related Work

Research on counterfactual explanations for incomplete inputs is still sparse. The most direct earlier contribution is the method of Kanamori et al. [2], which constructs counterfactuals that stay valid across several imputations of the missing values. That idea captures an important intuition: if the value is unknown, then the explanation should not hinge on a single potentially inaccurate estimate. A useful point of comparison comes from robustness-oriented CX research. Robust counterfactuals are built to survive certain kinds of uncertainty, such as model retraining, model multiplicity, or noisy application of the recommended action [4]. Although these settings differ from missing inputs, they share the same core challenge: the explanation should remain useful after the environment shifts. This makes robustness methods natural baselines for our study. Outside of the explainability domain, incomplete values are usually handled by imputation, with the choice of strategy depending on the data distribution, evaluation criterion, and downstream task [5]. Related work on feature attribution has also shown that different imputations can substantially alter explanation outputs [6]. Taken together, these studies suggest that missingness can distort not only predictions but also the explanations derived from them.

3. Background

We next introduce the notions used throughout the paper.

Counterfactual explanations. Let $x \in \mathbb{R}^n$ be an input and let $h : \mathbb{R}^n \to \{0, 1\}$ be a binary classifier. A counterfactual explanation is a point $x' \in \mathbb{R}^n$ that changes the classifier output while staying as close as possible to the original input. Given a distance function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$, the cost of x' is commonly measured as $d(x, x')$, often instantiated by an ℓ_p norm [7]. The associated recourse vector is $\delta = x' - x$, which describes the modification that would need to be applied to the input to obtain the counterfactual outcome.

Incomplete observations. Now suppose the true input x is only partially observed through $\tilde{x} \in (\{*\} \cup \mathbb{R})^n$, where $\tilde{x}_i$ is either the observed value x_i or the wildcard $*$. The goal is not merely to find any point that flips the prediction of a completed version of $\tilde{x}$, but to find a counterfactual whose induced action is still effective when applied to the hidden ground-truth instance.

Problem statement. Given a classifier h, a distance metric d, an incomplete observation $\tilde{x}$, the hidden complete input x, and a target label $t \in \{0, 1\}$, we want a counterfactual x' and recourse vector δ such that:

C1 The counterfactual is valid: $h(x') = t$.

C2 The recourse is valid: $h(x + \delta) = t$.

C3 The cost is minimized for: The cost $d(x, x')$ is minimized.

The key difficulty is that only $\tilde{x}$ is available when x' is generated, whereas the actual state x is hidden.

Why C1 and C2 differ. Condition C1 evaluates the explanation point itself, while C2 evaluates the action derived from that point. These are not equivalent under missingness. A counterfactual

can be valid in the imputed space and still fail once the action is transferred to the true instance. Our experiments therefore report both quantities separately. A recent empirical study [3] provides a more detailed discussion of this problem formulation and argues that it captures a realistic scenario encountered in practical applications.

Mixed Integer Linear Programming. A MILP consists of (integer or real-valued) decision variables, linear constraints, and a linear objective function to be minimized or maximized [8]. The general structure of a MILP formulation for integral decision variables x and continuous decision variables y is $\min\{c^T x + d^T y : Ax + By \geq b, (x, y) \in \mathbb{Z}^{n_x} \times \mathbb{R}^{n_y}\}$, where c, d, and b are constant vectors of size n_x, n_y, and m, respectively. The constant matrices A and B are of size $m \times n_x$, and $m \times n_y$, respectively [9].

Given such a MILP formulation, a solver then finds an optimal assignment of the decision variables while ensuring that the constraints are satisfied, if possible. Otherwise, the solver certifies infeasibility [8]. MILP constraints often use absolute values and equalities, which can be rewritten using auxiliary variables and inequalities [10]. We adopt this notation in our formulation for clarity, while relying on standard MILP encodings to handle them.

MILP-encodable Flow Models. A flow model $f : \mathbb{R}^n \to \mathbb{R}^n$ is a neural network that bijectively maps between a simple base distribution characterized by a probability density function p_B and the complex data distribution p_D, which represents the density of the observed data, through a series of invertible transformations $f = f_1 \circ f_2 \circ \ldots \circ f_z$ [11, 12]. Recently, the VeriFlow architecture [13] was proposed as the first flow model that can be seamlessly encoded into MILP. In order to be MILP-encodable, VeriFlow's architecture only contains piecewise linear layer types. We instantiate VeriFlow and use only LU-Layers [14] and fully connected additive coupling layers [15] to implement layers $f_1, \ldots, f_z$.

The training objective of the VeriFlow model is maximizing the log-likelihood on the training data, which corresponds to minimizing the Kullback–Leibler divergence between the base distribution p_B and the data distribution p_D [16]. After training is complete, the flow model can be used for density estimation by picking a point $x \in \mathbb{R}^n$ from the dataset and passing the point through the flow model in *forward* direction $f(x)$ and computing the density using the probability density function p_B.

4. Our Framework

We now present our main procedure to compute CXs under missing values. As a preliminary step, the procedure trains a piecewise linear flow model $f : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ on the training dataset D. The model's domain includes $\mathbb{R}^{n+1}$ because it is also trained on the class label in addition to the n features. We include the class label in order to align the CX with the distribution of the respective target class. Following the VeriFlow framework, we design f such that minimizing $\sum_i^{n+1} f_i(x, c)$ over a free decision variable x and a class label c maximizes the plausibility of x under the empirical data distribution p_D, given c. Additionally, we use a function UDL : $\mathbb{R}^{n+1} \to [0, 1]$ to map $f(x)$ to a density level set, represented as a decimal between 0 and 1. For instance, a level of 0.25 indicates that x is more 'typical' than 75% of the instances in D. Details on the training process are provided in the experiments section.

Using this flow model, our procedure moves on to construct a MILP encoding named PCID (Plausible Counterfactuals for Incomplete Data). The signature $\textsc{pcid}(\tilde{x}, M, h, f, d, p_{\text{val}}, p_{\text{plaus}}, p_{\text{cost}}, p_{\text{faith}}) \to x'$ reflects that it produces a CX x' given all the elements from the problem definition: the incomplete input $\tilde{x}$, the indices of the missing values M, the classifier h, a distance function d, and the trained flow model f. Additionally, PCID expects a set of parameters that balance four (competing) objectives (1) validity using p_{val} (2) plausibility using p_{plaus},(3) cost using p_{cost}, and (4) imputation faithfulness using p_{faith}. We encode the validity as a side constraint rather than an optimization objective in order to ensure high validity independent of the cost it incurs. The last parameter p_{faith} impacts both validity and cost and indicates how much to trust the flow model's estimated value for the missing feature. Intuitively, the parameter p_{faith} indicates whether the input instance $\tilde{x}$ comes from a low-density region of the dataset, where the accuracy of the flow model is expected to be low due to epistemic uncertainty.

$$\text{minimize } p_{\text{cost}} \cdot \sum_{i=1}^{n} |x' - x^{\text{imp}}|_i + p_{\text{faith}} \cdot \sum_{i=1}^{n+1} |\ell_i^{\text{imp}}| + p_{\text{plaus}} \cdot \sum_{i=1}^{n+1} |\ell_i^{\text{cx}}| \tag{1}$$

subject to

$$x'_i = x_i^{\text{imp}} \qquad i \in M \tag{2}$$

$$x_i^{\text{imp}} = \tilde{x}_i \qquad i \in \{1, \ldots, n\}, i \notin M \tag{3}$$

$$\ell_i^{\text{imp}} = f_i(x^{\text{imp}}, 0) \qquad i \in \{1, \ldots, n+1\} \tag{4}$$

$$\ell_i^{\text{cx}} = f_i(x', 1) \qquad i \in \{1, \ldots, n+1\} \tag{5}$$

$$h(x')_1 - h(x')_0 \geq p_{\text{val}} \tag{6}$$

$$x_i^{\text{imp}}, x'_i \in \mathbb{R} \qquad \forall i \in \{1, \ldots, n\}$$

$$\ell_i^{\text{imp}}, \ell_i^{\text{cx}} \in \mathbb{R} \qquad \forall i \in \{1, \ldots, n+1\}$$

Figure 2: Implementation of our encoding $\text{PCID}(\tilde{x}, M, h, f, d, p_{\text{val}}, p_{\text{plaus}}, p_{\text{cost}}, p_{\text{faith}})$

For readability, and without loss of generality, we summarize the main steps of our procedure assuming the target label to be 1 and the class label of $\tilde{x}$ to be 0. However, in practice, these labels can be exchanged depending on users' needs. Given an incomplete point $\tilde{x} \in (\{*\} \cup \mathbb{R})^n$, indices M, classifier h, flow f, and distances d, the main steps of our procedure are as follows:

1. Impute the missing values of $\tilde{x}_m, m \in M$ using a MILP solver by computing $\hat{x}$ such that $\hat{x}_i = x_i$ for $i \notin M$ and $\hat{x}$ is pushed to a high-density region of the data manifold by minimizing $\sum_{i=1}^{n+1} |f(\hat{x}, 0)_i|$.
2. Compute the density level set of the imputed point $\hat{x}$ with $u_{\hat{x}} \leftarrow \text{UDL}(f(\hat{x}, 0))$.
3. Select values for parameters $p_{\text{val}}, p_{\text{plaus}}, p_{\text{cost}}, p_{\text{faith}}$ using $u_{\hat{x}}$ and the inputs $h, f, n, \tilde{x}$.
4. Call $x' \leftarrow \text{PCID}(\tilde{x}, M, h, f, d, p_{\text{val}}, p_{\text{plaus}}, p_{\text{cost}}, p_{\text{faith}})$ and return the counterfactual x^{cx}.

The high-level procedure calls our MILP formulation with preselected parameters based on the dimensionality of the dataset n and the uncertainty around each point $u_{\hat{x}}$. We implement Step 3 as a simple heuristic and therefore keep it generic here. More details will be presented in the experiments section. The details of our MILP formulation $\text{PCID}(\tilde{x}, M, h, f, d, p_{\text{val}}, p_{\text{plaus}}, p_{\text{cost}}, p_{\text{faith}}) \rightarrow x'$ are presented in Figure 2. The optimization goal (1) strikes a balance between three objectives. The first objective minimizes the cost of the CX x^{cx} weighted by p_{cost} while satisfying the side constraints (2) and (6). Equation (2) ensures that the resulting CX retains the same imputed value for the missing feature. This restriction is based on the rationale that a what-if analysis should not depend on potentially uncertain imputed values. Equation (6) ensures that the CX is classified as the target class with a clear marginal distance p_{val} from the decision boundary. A low (high) value for p_{val} leads to lower (higher) cost but increases (decreases) the risk of obtaining a CX that is invalid for the fully specified original instance. The second sum optimizes the faithfulness of the value ascribed to the missing features from $\tilde{x}_m$, weighted by the constant p_{faith}. More precisely, Equation (3) defines a vector x^{imp} that corresponds to the input vector $\tilde{x}$ with the only difference that the wild token $*$ is turned into free variables x_m^{imp} for all $m \in M$. Then, Equation (4) captures the relationship between the point in the data space x^{imp} and its respective density estimate ℓ^{imp} in the latent space as defined by the flow model f. Minimizing the vector ℓ^{imp} as expressed in the objective leads to values x_m^{imp} for $m \in M$ that are most plausible for the instance x^{imp}. The last sum optimizes for the plausibility of the CX x^{cx}, weighted by the constant p_{plaus}. In this context, Equation 5 captures the relationship between the CX x^{cx} and its density estimate ℓ^{cx}, which is minimized in the objective function. In a nutshell, solving the MILP formulation PCID yields assignments to the variable x'_i that balance cost and plausibility, while ensuring validity.

5. Evaluation

Our Implementation. We now describe how the parameters left open in Step 3 of Section 4 are instantiated. The threshold p_{val}, which determines how strongly the counterfactual must cross the classifier's decision boundary, is chosen as a function of the dataset dimensionality n. The motivation is that in high-dimensional settings the classifier h is typically less sensitive to perturbations of a single feature, whereas in lower-dimensional spaces such changes can have a stronger impact. Such single-feature perturbations naturally arise when imputed feature values differ from the unknown true values. To ensure consistent scaling, all parameters $p_{\text{plaus}}, p_{\text{cost}}, p_{\text{val}}$, and p_{faith} are defined relative to the dimension n and constants $C^{\text{plaus}}, C^{\text{cost}}, C^{\text{val}}$, and C^{faith}. These constants were chosen through exploratory tuning, starting from an initial configuration in which all components were weighted equally. Concretely, we use $p_{\text{plaus}} \leftarrow \frac{C^{\text{plaus}}}{n}$, $p_{\text{cost}} \leftarrow \frac{C^{\text{cost}}}{n}$, $p_{\text{val}} \leftarrow \frac{C^{\text{val}}}{n}$, and $p_{\text{faith}} \leftarrow \frac{C^{\text{faith}}}{n \cdot u_{\hat{x}}}$. The parameter p_{faith} additionally depends on the density estimate $u_{\hat{x}}$ obtained in Step 2, which allows the formulation to account for epistemic uncertainty introduced by imputing the missing feature.

Baselines. Our method is compared against all ten CX generation techniques used in the recent empirical evaluation [3]. The baselines are primarily implemented in the RobustX library [17]. The set of evaluated *non-robust* approaches includes: (1) BinaryLinearSearch (BLS) [18]: explores the line segment toward a positively labeled training instance. (2) MCE [7]: formulates CX generation as a MILO problem that enforces boundary crossing. (3) `Wachter` [19]: a gradient-based optimization method for generating counterfactuals. (4) KDTreeNNCE [20]: retrieves nearest neighbors belonging to the target class. (5) ARMIN [2]: a MILO-based approach ensuring validity over multiple imputations. We further consider the following *robust* methods: (1) MCER [21]: designed to remain valid under bounded perturbations of model weights. (2) RNCE [22]: generates counterfactuals that are robust with respect to the training data. (3) PROPLACE [23]: jointly optimizes proximity and plausibility under robustness constraints. (4) STCE [24, 25]: ensures robustness against model updates caused by retraining. (5) APAS [26]: accounts for plausible model variations when generating counterfactuals.

Datasets and preprocessing. We use the same experimental setup as in [3]: We evaluate on four standard tabular benchmarks: WineQuality [27], Diabetes [28], Concrete [29], and Combined Cycle Power Plant [30]. Each dataset contains continuous features only and is scaled to $[0, 1]$ by min–max normalization. Regression datasets are turned into binary tasks with threshold 0.5. We split the data into 80% training and 20% test partitions and train a two-layer ReLU neural network classifier on the training portion. From each test set, we sample batches of 100 instances. For each instance, we delete one feature at random and then employ three imputation strategies: MICE [31], kNN [32], and `Simple` mean imputation. Our method and ARMIN are both designed to handle incomplete inputs and therefore receive the incomplete input without any preceding imputation.

Metrics. We report three primary metrics: (1) *Counterfactual Validity (VCX)*: the fraction of counterfactuals that satisfy the target label; (2) *Recourse Validity (VRC)*: the fraction of recourse actions that flip the classification on the true input; (3) *Cost*: the average ℓ_1 distance between the original input and the counterfactual. We aggregate each method's results over the four datasets and report them in Figures 3 and 4.

Recourse Validity (VRC). Figure 3 shows that our method achieves consistently high recourse validity across datasets and imputation settings. In particular, it outperforms all non-robust baselines and is competitive with the strongest robust methods. This is notable because methods that are effective on fully observed inputs often degrade once recourse is evaluated on the true, incomplete instance. By contrast, our approach is explicitly designed to preserve validity under missingness, which explains its strong performance on this metric. Among the baselines, PROPLACE, STCE, and APAS also obtain high VRC, although, as we will see, at a higher cost.

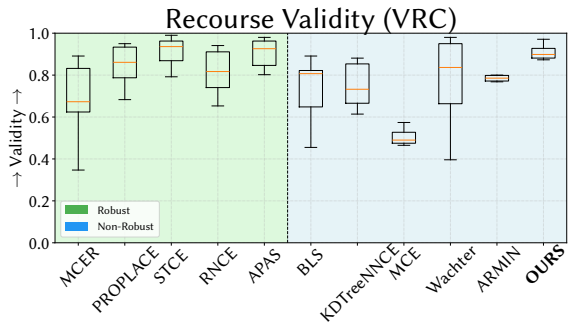

Figure 3: Comparison of validity.

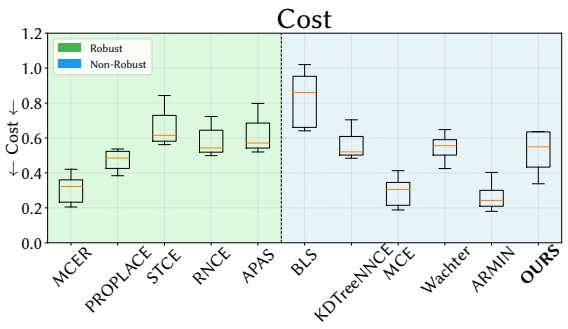

Figure 4: Comparison of cost.

Counterfactual Validity (VCX). Across all evaluated approaches, counterfactual validity is consistently perfect, with the exception of `Wachter`, which attains $84.6\% \pm 14.3\%$. This deviation can be attributed to its reliance on gradient-based optimization, which may converge to suboptimal local minima and thus fail to find a valid counterfactual in some cases. We omit a dedicated visualization for this metric due to its simplicity.

Cost. Figure 4 compares the average cost of the generated counterfactuals. Our method achieves a moderate cost, remaining clearly below the less efficient baselines such as BLS or STCE, while staying in the same range as PROPLACE and RNCE. Although methods such as MCER and ARMIN can sometimes obtain slightly lower costs, they do so at the expense of weaker recourse validity. Overall, the results indicate that our method strikes a strong balance between validity and cost: it produces counterfactuals that are sufficiently close to the original instance, while maintaining high validity on the true underlying input.

Tradeoff Analysis Since our framework also optimizes for the plausibility of counterfactuals, it is natural to ask how this objective affects other key goals. To investigate this, we analyze the trade-off between plausibility and validity in Figure 5. Each point in the figure corresponds to a different plausibility weight used in the MILP formulation and shows the proportion of test instances for which a valid CX was obtained, computed as the number of valid CXs divided by the total number of test instances. The results indicate that increasing the plausibility weight initially improves validity, but only up to a certain threshold, after which validity declines. We conjecture that overly large plausibility weights lead the method to neglect imputation quality, a competing objective crucial for obtaining valid CXs. At the same time, the initial improvement suggests that optimizing plausibility can also support validity.

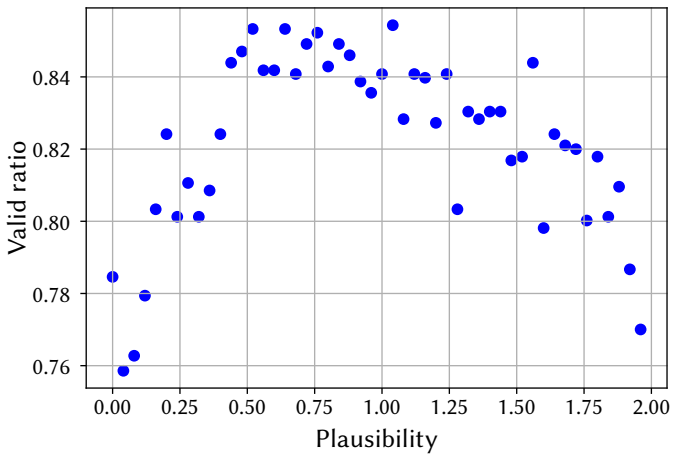

Figure 5: Plausibility vs. validity

Key takeaways. Our method achieves consistently high validity at a slightly lower cost than baselines that achieve similar levels of validity. We also show that our approach to place the computed counterfactual on the data manifold also helps achieve higher validity.

6. Conclusion

We studied counterfactual explanations for inputs with missing values and showed that imputation-based approaches can lead to invalid recourse. To address this issue, we proposed a framework that combines flow-based generative models with a MILP formulation to reason directly about missing features during counterfactual generation. Experiments on four real-world datasets show that our method achieves high recourse validity while maintaining competitive cost compared to existing approaches.

Declaration on Generative AI

During the preparation of this work, the authors used a local instance of GPT 5.2 in order to improve wording, grammar and spelling. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artif. Intell. 267 (2019) 1–38. doi:10.1016/j.artint.2018.07.007.

[2] K. Kanamori, T. Takagi, K. Kobayashi, Y. Ike, Counterfactual explanation with missing values, CoRR abs/2304.14606 (2023). doi:10.48550/ARXIV.2304.14606. arXiv:2304.14606.

[3] F. Leofante, D. Neider, M. Yalçıner, Evaluating counterfactual explanation methods on incomplete inputs, 2026. URL: https://arxiv.org/abs/2604.08004. arXiv:2604.08004.

[4] J. Jiang, F. Leofante, A. Rago, F. Toni, Robust counterfactual explanations in machine learning: A survey, in: Proc. of IJCAI 2024, ijcai.org, 2024, pp. 8086–8094.

[5] M. Thurow, F. Dumpert, B. Ramosaj, M. Pauly, Assessing the multivariate distributional accuracy of common imputation methods, Statistical Journal of the IAOS 40 (2024) 99–108. doi:10.3233/SJI-230015. arXiv:https://doi.org/10.3233/SJI-230015.

[6] T. L. Vo, T. Nguyen, H. L. Hammer, M. A. Riegler, P. Halvorsen, Explainability of machine learning models under missing data, CoRR abs/2407.00411 (2024). doi:10.48550/ARXIV.2407.00411. arXiv:2407.00411.

[7] K. Mohammadi, A. Karimi, G. Barthe, I. Valera, Scaling guarantees for nearest counterfactual explanations, in: AIES '21: AAAI/ACM Conference on AI, Ethics, and Society, Virtual Event, USA, May 19-21, 2021, ACM, 2021, pp. 177–187. doi:10.1145/3461702.3462514.

[8] A. Schrijver, Theory of linear and integer programming, Wiley-Interscience series in discrete mathematics and optimization, Wiley, 1999.

[9] F. Clautiaux, I. Ljubić, Last fifty years of integer linear programming: A focus on recent practical advances, European Journal of Operational Research (2024). doi:https://doi.org/10.1016/j.ejor.2024.11.018.

[10] O. L. Mangasarian, Absolute value programming, Comput. Optim. Appl. 36 (2007) 43–53. doi:10.1007/S10589-006-0395-5.

[11] D. J. Rezende, S. Mohamed, Variational inference with normalizing flows, in: ICML 2015, volume 37, JMLR.org, 2015, pp. 1530–1538.

[12] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, B. Lakshminarayanan, Normalizing flows for probabilistic modeling and inference, J. Mach. Learn. Res. 22 (2021).

[13] F. A. Zaid, D. Neider, M. Yalçiner, Veriflow: Modeling distributions for neural network verification, in: AAAI 2026, AAAI Press, 2026, pp. 28050–28058. doi:10.1609/AAAI.V40I33.40030.

[14] R. Chan, S. Penquitt, H. Gottschalk, Lu-net: Invertible neural networks based on matrix factorization, in: IJCNN 2023, IEEE, 2023, pp. 1–10. doi:10.1109/IJCNN54540.2023.10191440.

[15] L. Dinh, D. Krueger, Y. Bengio, NICE: non-linear independent components estimation, in: ICLR 2015, 2015.

[16] C. M. Bishop, Pattern recognition and machine learning, 5th Edition, Information science and statistics, Springer, 2007.
[17] J. Jiang, L. Marzari, A. Purohit, F. Leofante, Robustx: Robust counterfactual explanations made easy, in: IJCAI, 2025, pp. 11067–11071. doi:10.24963/IJCAI.2025/1264.
[18] F. Leofante, N. Potyka, Promoting counterfactual robustness through diversity, in: AAAI 2024, AAAI Press, 2024, pp. 21322–21330. doi:10.1609/AAAI.V38I19.30127.
[19] S. Wachter, B. Mittelstadt, C. Russell, Counterfactual explanations without opening the black box: Automated decisions and the GDPR, Harv. JLT. (2017).
[20] D. Brughmans, P. Leyman, D. Martens, NICE: an algorithm for nearest instance counterfactual explanations, Data Min. Knowl. Discov. 38 (2024) 2665–2703. doi:10.1007/S10618-023-00930-Y.
[21] J. Jiang, F. Leofante, A. Rago, F. Toni, Formalising the robustness of counterfactual explanations for neural networks, in: Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI Press, 2023, pp. 14901–14909. doi:10.1609/AAAI.V37I12.26740.
[22] J. Jiang, F. Leofante, A. Rago, F. Toni, Interval abstractions for robust counterfactual explanations, Artif. Intell. 336 (2024) 104218.
[23] J. Jiang, J. Lan, F. Leofante, A. Rago, F. Toni, Provably robust and plausible counterfactual explanations for neural networks via robust optimisation, in: Asian Conference on Machine Learning, ACML 2023, 11-14 November 2023, Istanbul, Turkey, Proceedings of Machine Learning Research, PMLR, 2023, pp. 582–597.
[24] S. Dutta, J. Long, S. Mishra, C. Tilli, D. Magazzeni, Robust counterfactual explanations for tree-based ensembles, in: International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA, Proceedings of Machine Learning Research, PMLR, 2022, pp. 5742–5756.
[25] F. Hamman, E. Noorani, S. Mishra, D. Magazzeni, S. Dutta, Robust counterfactual explanations for neural networks with probabilistic guarantees, in: International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA, Proceedings of Machine Learning Research, PMLR, 2023, pp. 12351–12367.
[26] L. Marzari, F. Leofante, F. Cicalese, A. Farinelli, Rigorous probabilistic guarantees for robust counterfactual explanations, in: ECAI 2024, Frontiers in Artificial Intelligence and Applications, IOS Press, 2024, pp. 1059–1066. doi:10.3233/FAIA240597.
[27] C. Paulo, C. A., F. Almeida, M. T., , R. J., Wine Quality, UCI ML Repository, 2009.
[28] N. I. of Diabetes Digestive, K. Diseases, Diabetes dataset, 1988.
[29] I.-C. Yeh, Modeling of strength of high-performance concrete using artificial neural networks, Cement and Concrete Research 28 (1998) 1797–1808.
[30] P. Tuefekci, Prediction of full load electrical power output of a base load operated combined cycle power plant, International Journal of Electrical Power & Energy Systems 60 (2014).
[31] S. van Buuren, K. Groothuis-Oudshoorn, mice: Multivariate imputation by chained equations in r, Journal of Statistical Software 45 (2011) 1–67.
[32] Troyanskaya, Cantor, Sherlock, Brown, Hastie, Tibshirani, Botstein, Altman, Missing value estimation methods for DNA microarrays, Bioinform. (2001).