TABCF: Counterfactual Explanations for Tabular Data Using a Transformer-Based VAE

Emmanouil Panagiotou[*]
Institute of Computer Science, Freie Universität Berlin
DE
emmanouil.panagiotou@fu-berlin.de

Manuel Heurich[*]
Institute of Computer Science, Freie Universität Berlin
DE
manuel.heurich@fu-berlin.de

Tim Landgraf
Institute of Computer Science, Freie Universität Berlin
DE
tim.landgraf@fu-berlin.de

Eirini Ntoutsi
Department of Computer Science, Bundeswehr University Munich
DE
eirini.ntoutsi@unibw.de

Abstract

In the field of Explainable AI (XAI), counterfactual (CF) explanations are one prominent method to interpret a black-box model by suggesting changes to the input that would alter a prediction. In real-world applications, the input is predominantly in tabular form and comprised of mixed data types and complex feature interdependencies. These unique data characteristics are difficult to model, and we empirically show that they lead to bias towards specific feature types when generating CFs. To overcome this issue, we introduce *TABCF*, a CF explanation method that leverages a transformer-based Variational Autoencoder (VAE) tailored for modeling tabular data. Our approach uses transformers to learn a continuous latent space and a novel Gumbel-Softmax detokenizer that enables precise categorical reconstruction while preserving end-to-end differentiability. Extensive quantitative evaluation on five financial datasets demonstrates that TABCF does not exhibit bias toward specific feature types, and outperforms existing methods in producing effective CFs that align with common CF desiderata.

CCS Concepts

• **Computing methodologies** → **Neural networks**; *Learning latent representations.*

Keywords

Machine Learning, Explainable AI, Counterfactual Explanations, Financial Tabular Data

ACM Reference Format:
Emmanouil Panagiotou, Manuel Heurich, Tim Landgraf, and Eirini Ntoutsi. 2024. TABCF: Counterfactual Explanations for Tabular Data Using a Transformer-Based VAE. In *5th ACM International Conference on AI in Finance (ICAIF '24), November 14–17, 2024, Brooklyn, NY, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3677052.3698673

[*]Both authors contributed equally to this research.

1 Introduction

Although Deep Neural Networks (DNN) are highly effective, their complexity makes them difficult to explain, hindering their adoption in crucial fields like healthcare and finance. Explainable AI (XAI) aims to overcome this by making their decisions interpretable [1].

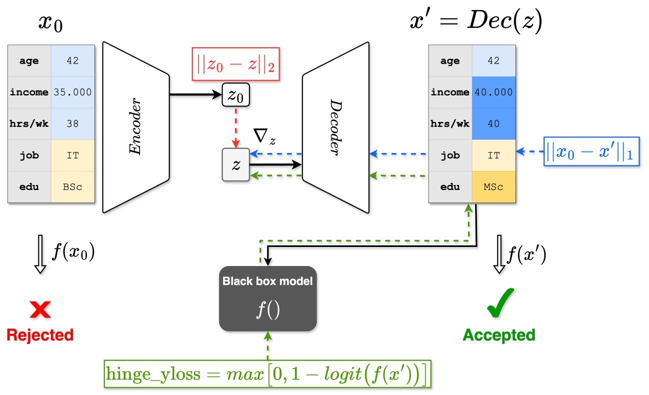


Figure 1: Overview of the counterfactual generation process. The bold arrows indicate data flow, and the dashed arrows indicate backward gradient flow. We iteratively optimize the latent representation z, of the counterfactual x', using three distinct loss terms.

Counterfactual (CF) explanations are one way to gain insight into a black-box model's decision by offering meaningful changes to the input that would result in a favorable outcome. Most initial works find CFs by searching in the input space [22, 28]. More recent studies leverage generative models, such as Variational Autoencoders [18] (VAE). These models capture the underlying structure of the data to generate more realistic and semantically meaningful CFs. However, most of these methods are developed for the vision domain [7, 14], where generative models are particularly effective. In business applications, especially finance, data most often comes in tabular form presenting specific challenges like mixed (numerical and categorical) feature types, inherent imbalances, and complex feature interdependencies. Existing tabular CF methods handle the mixed feature space using simple pre/post-processing [2, 9, 17, 25] or regularization functions [22]. Our experiments show that such data handling leads to *feature-type bias*, for example predominantly changing numerical features when generating CFs.

To address these challenges, we present TABCF, a CF generation method that employs a tranformer-based VAE tailored for tabular data. Our approach maps the mixed input into a unified continuous latent space, leveraging transformers to capture the rich feature interdependencies. Similar to other methods [17, 22, 28] we define CF generation as an optimization problem, and use gradient descent to navigate the latent space. Importantly, our decoder architecture enables precise categorical reconstruction via the Gumbel-Softmax trick [16], while being fully differentiable to ensure optimal gradient flow from the black-box model. Furthermore, we optimize for important CF desiderata [26], such as validity, but also, proximity to the original instance, and feature sparsity, for producing more *actionable* CFs. We compare TABCF's performance against baselines in terms of producing valid and actionable CFs for tabular data, on binary classification problems. Our extensive quantitative evaluation on five financial and census datasets showcases that TABCF is superior in producing effective CF explanations, and does not exhibit feature-type bias. Our contributions include: i) identifying issues in the tabular data handling of existing methods, which impede optimization and result in feature-type bias, and ii) introducing TABCF a novel counterfactual generation method that employs a transformer-based VAE specifically designed for tabular data to address these limitations.

Our paper is structured as follows. We describe all relevant works related to counterfactuals and generative models in Section 2, we introduce our method TABCF in Section 3, defining the transformer-based VAE and the CF generation process in the latent space. In Section 4 we present our experimental evaluation, including datasets, metrics, and baselines, and in Section 5 we present the results. We conclude the paper with a discussion and opportunities for future work in Section 6. We have provided a codebase [1] for reproducing all experiments.

2 Related Work

Recent literature is abundant with studies on CFs that detail numerous desired properties (desiderata), target different data modalities, and employ various methodologies [26] and criteria for evaluation [24]. This section provides an overview of the methods most pertinent to our field of work of tabular CFs.

Counterfactual desiderata. Are desirable properties for effective CF explanations. The most essential property is *validity*, ensuring the proposed changes alter the decision of the black-box model we aim to explain. The vast majority of works simultaneously optimize to find the most minimal changes that lead to the desired outcome [17, 22, 25, 28]. This objective involves the desiderata of *proximity* and *sparsity*, i.e., CFs that are close to the original instance and alter as few features as possible. Finally, some methods find more robust CFs that withstand minor perturbations [3], account for predictive uncertainty [2], causal constraints [17, 23], and plausibility [5].

Counterfactual methods. Wachter et al. [28] first introduced CF explanations as an optimization problem in input space, aimed at changing a model's decision while minimizing the distance to the original instance. This approach inspired various other methods that assume a differentiable black-box model and use gradient descent to find CFs, either in the input space like DiCE [22] or in the latent space of a generative model like REVISE and CCHVAE [17, 25]. Our method falls within the latter category, and therefore we provide a detailed description of these methods in Section 4.3, and use them as baselines in our experiments. Other works assume a model agnostic setting where differentiability is not guaranteed, e.g. in decision trees. Most methods in this scenario use a heuristic search, like genetic algorithms [23], or reinforcement learning to learn a policy for finding CFs [27].

Generative models. Several works rely on generative models to create CFs. While some methods [6] use Generative Adversarial Networks (GANs) [11], the majority [2, 9, 17, 25] prefer Variational Autoencoders (VAEs) [18], for their flexibility and smooth latent representations. However, most generative models are designed for computer vision [8, 14] and are difficult to adapt to the unique characteristics of mixed tabular data [29]. Recently, transformers have emerged as a potential solution, due to their ability to capture complex feature interdependencies [12]. Specifically, the TABSYN [30] method utilizes a transformer-based VAE for synthetic tabular data generation, outperforming existing approaches. To the best of our knowledge, our method TABCF is the first to use a transformer-based autoencoding framework for tabular counterfactual generation.

3 TabCf: Counterfactual explanations for tabular data

In this section, we introduce our method *TABCF*. We start with preliminaries, then we present the architecture of our transformer-based VAE in Section 3.1 and we describe the CF generation process in Section 3.2.

We assume a differentiable black-box classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$ where $\mathcal{X}$ is the input feature space and $\mathcal{Y} = \{0, 1\}$ is the binary class. For an instance classified in the *undesired class* $f(x) = 0$, the goal is to find a CF example that belongs to the *target class*, i.e. $f(x') = 1$. Additionally, we assume a mixed input space of $|N|$ numerical and $|C|$ categorical features $x = [x^{num}, x^{cat}] \in \mathbb{R}^{|N|+|C|}$. We train f by pre-processing the data in the usual fashion so that all numerical features are min-max normalized, and all categorical features are one-hot encoded. Therefore, each row is presented as a k-dimensional vector $x = [x_1^{num}, x_2^{num}, \dots, x_{|N|}^{num}, x_1^{oh}, x_2^{oh}, \dots, x_{|C|}^{oh}]$, with $k = \mathbb{R}^{|N|} + \sum_{i=1}^{|C|} C_i$, where C_i are the discrete domains of each categorical feature.

3.1 Transformer-based VAE for tabular data

As previously mentioned, we build on the architecture used in [30] for synthetic mixed tabular data generation, and adapt it for CF generation. In particular, we employ learnable tokenizers to process the input data and transformers to learn the latent space. To reconstruct precise one-hot samples while maintaining end-to-end differentiability we propose a Gumbel detokenizer. The entire training pipeline of the VAE is presented in Figure 2, and is described in detail hereafter.

[1] github.com/Panagiotou/TABCF

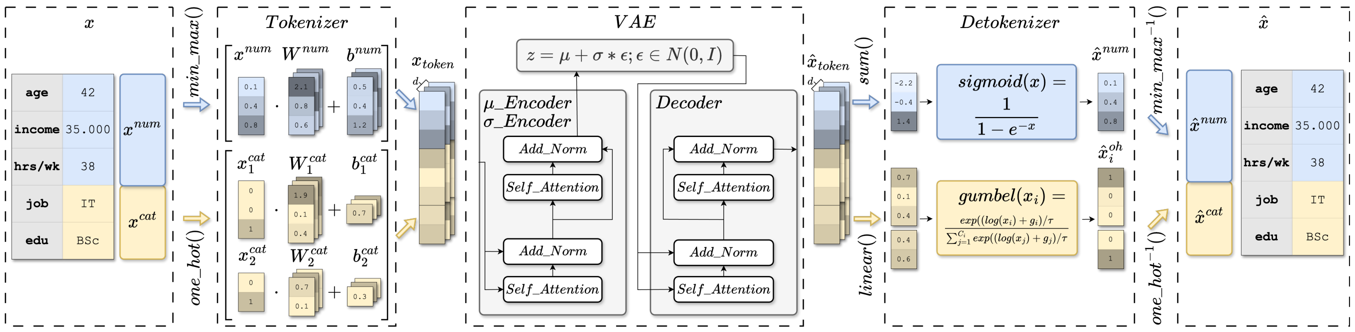


Figure 2: Overview of the Variational Autoencoder training process. Blue indicates the process for numerical, yellow for categorical features. The detokenizer enables a fully differentiable pipeline for categorical features using the Gumbel-Softmax function for sample reconstruction.

Feature Tokenizer. To adapt the transformer architecture for tabular data, [12] proposes a *Feature Tokenizer* as a learnable preprocessing step that converts features into tokens for the subsequent feature-level transformer layers. Specifically, given an input vector x of size k,

$x = [x_1^{num}, x_2^{num}, \dots, x_{|N|}^{num}, x_1^{oh}, x_2^{oh}, \dots, x_{|C|}^{oh}]$

a tokenized vector x_{token} of size $k \times d$ is created

$x_{token} = [t_1^{num}, t_2^{num}, \dots, t_{|N|}^{num}, t_1^{cat}, t_2^{cat}, \dots, t_{|C|}^{cat}]$,

via linear transformation of numerical, and "lookup tables" for each categorical feature. In detail we have,

$t^{num} = x^{num} \cdot \mathbf{W}^{num} + \mathbf{b}^{num}$ and

$t_i^{cat} = x_i^{oh} \cdot \mathbf{W}_i^{cat} + \mathbf{b}_i^{cat}$ for $i \in \{1, \dots, |C|\}$

where each column is a token $t_i^{num}, t_i^{cat} \in \mathbb{R}^{1\times d}$. All weights and biases of the tokenizer, i.e., $\mathbf{W}^{num}, \mathbf{b}^{num} \in \mathbb{R}^{|N|\times d}$ and $\mathbf{W}_i^{cat} \in \mathbb{R}^{C_i \times d}, \mathbf{b}_i^{cat} \in \mathbb{R}^{1\times d}$ are learnable parameters.

The learned column-wise token embeddings are passed to the transformer-based encoder that captures the rich feature interdependencies to output the mean and log variance. The latent vector $z = \mu + \sigma \cdot \epsilon$ is obtained via parameterization [18]. The same inverse procedure is followed for reconstructing the token vector $\hat{x}_{token}$ through the decoder. To get the final output $\hat{x}$ we use our specialized *Gumbel detokenizer*, described hereafter.

Gumbel-Detokenizer. A key requirement for TABCF is a fully differentiable pipeline that allows for optimizing CF samples in the latent space through the back-propagation of gradients. Additionally, it is essential to ensure that the decoded samples adhere to feature-type constraints. For instance, decoded categorical vectors $\hat{x}_i^{cat}$ should be one-hot. Considering this need to reconstruct one-hot data while enabling gradient flow, we introduce a Gumbel detokenizer that uses the Gumbel-softmax [16] trick to generate categorical features. In particular the *gumbel*(.) function,

$$gumbel(x_i) = \frac{exp((log(x_i) + g_i)/\tau}{\sum_{j=1}^{C_i} exp((log(x_j) + g_j)/\tau},$$

where C_i denotes the number of modes per categorical feature, g denotes the probabilistic variable sampled from the Gumbel distribution, and τ denotes the temperature hyperparameter. It is important to note that (for Section 5.2) the *gumbel* function outputs the discretized one-hot samples but uses the *soft samples* for differentiation. After decoding we get the reconstructions,

$$\hat{x}^{num} = \text{sigmoid}(\hat{t}^{num} \cdot \hat{\mathbf{W}}^{num} + \hat{\mathbf{b}}^{num})$$

$$\hat{x}_i^{oh} = \text{gumbel}(\hat{t}_i^{cat} \cdot \hat{\mathbf{W}}_i^{cat} + \hat{\mathbf{b}}_i^{cat}) \text{ for } i \in \{1, \dots, |C|\}$$

This ensures that all input constraints are respected for the reconstructed samples $\hat{x} = [\hat{x}_1^{num}, \hat{x}_2^{num}, \dots, \hat{x}_{|N|}^{num}, \hat{x}_1^{oh}, \hat{x}_2^{oh}, \dots, \hat{x}_{|C|}^{oh}]$, i.e. numerical columns are in the range of $[0, 1]$ and that all categorical vectors follow a one-hot distribution. In conclusion, TABCF enables gradient-based CF generation in the latent space and inherently guarantees tabular feature constraints. In contrast, other gradient-based methods resort to postprocessing techniques or rely on additional regularization losses to maintain categorical constraints [22, 24]. In our experiments, we show that this leads to unwanted feature-type bias.

VAE training. We train the transformer-based VAE using the β-VAE loss, $\mathcal{L} = \|x - \hat{x}\| + \beta \cdot \mathcal{L}_{KL}$ [13], where $\mathcal{L}_{KL}$ denotes the discrete Kullback-Leibler divergence between the latent variable and a standard gaussian. Following [30], we opt for a better reconstruction than a perfectly Gaussian distributed latent space by gradually decreasing $\beta = [\beta_{max}, \beta_{min}]$ during training.

3.2 Counterfactuals in the latent space

After training the transformer-based VAE, we use the latent representations to search for CFs by traversing the latent space via gradient steps. We use a loss term comprised of three components designed for *Validity*, *Proximity*, and *Sparsity* (referring to the desiderata in Section 2).

More specifically, given an input instance x_0, we obtain the initial latent representation $z_0 = Enc(x_0)$ through the encoder. We then initialize the optimization with $z = z_0$ and take gradient steps updating z with $\nabla_z \mathcal{L}_{\text{CF}}$, minimizing the following loss function:

$$\begin{aligned} \mathcal{L}_{\text{CF}}(z) = \; & \text{hinge\_yloss}\Big[f\Big(Dec(z)\Big), y = 1\Big] \\ & + \lambda_{prox\_input} \cdot \|x_0 - Dec(z)\|_1 \\ & + \lambda_{prox\_latent} \cdot \|z_0 - z\|_2 \end{aligned} \quad (1)$$

Validity. The first component computes the difference between the target class ($y = 1$) and the current prediction of the black-box model $f(.)$. The latent vector z being optimized, is first passed through the decoder $Dec(z)$. This reconstructed vector is in the original tabular form, by design of our architecture, and can be directly used to get the prediction of the model $f(Dec(z))$. Following [22], we use the hinge-loss, defined as

$$\text{hinge\_yloss} = max\left[0, 1 - logit\left(f(.)\right)\right]$$

which has two functionalities, i) it heavily penalizes predictions that do not belong to the target class, i.e. when $\mathbb{P}(f(x) = 1) < 0.5$, and ii) it returns a penalty when the target class is achieved, i.e. when $\mathbb{P}(f(x) = 1) \geq 0.5$, proportional to the difference $\mathbb{P}(f(x) = 1) - 0.5$ between the predicted target probability and the decision threshold. The intended effect is pushing instances across the decision boundary to ensure validity, while optimizing for better trade-off solutions (i.e. proximity and sparsity) afterward.

Input proximity and sparsity. The second term, $\|x_0 - Dec(z)\|_1$ measures the L1 distance of the original instance x_0 and the reconstructed sample, where λ_{prox_input} is a weighting hyperparameter. This term serves as a *proximity* loss in the input space. Using the L1 norm (instead of e.g. L2) additionally encourages feature sparsity [31]. It is important to note that although sparsity can be computed using the L0 norm, i.e. count distance, this operation is non-differentiable.

Latent proximity. The last term, $||z_0 - z||_2$ measures the L2 distance to the original latent representation z_0, to encourage proximity in the latent space, where λ_{prox_latent} is a weighting hyperparameter.

Motivated by other works, that either contain the search in a neighborhood of the latent space [25], or minimize the distance in the input space [2, 17], we decide to follow a combined approach ensuring both latent and input proximity. Our ablation study (Section 5.3) empirically demonstrates that this combined method yields superior results in terms of proximity and sparsity. Our optimization process is illustrated in Figure 1.

4 Experimental evaluation

This section presents the datasets, baselines, metrics, and the general setup of our quantitative evaluation.

4.1 Experimental setup

We initialize TABCF by training the VAE for 4.000 epochs, gradually decreasing β from $\beta_{max} = 10^{-3}$ to $\beta_{min} = 10^{-5}$, and use $\tau = 1.0$ for the Gumbel distribution. We choose a maximum of 30.000 training samples for larger datasets to improve comparability. After the VAE is sufficiently trained, we perform Stochastic Gradient Descent (SGD) to find CFs for a maximum of 5.000 steps or until the loss converges and the target class is reached. For loss weighting, we set $\lambda_{prox\_input} = 1$ and $\lambda_{prox\_latent} = 1$, as these hyperparameters yield the best overall results. We discuss this further in our ablation study (Section 5.3). For all baseline competitors we use the implementations of the CARLA framework [24].

4.2 Real world financial datasets

We choose five real-world financial datasets, with a range of different data types, feature counts, and ratios of categorical to numerical features, to ensure a thorough experimental evaluation. In Table 1 we list the characteristics of each dataset, such as the size of the training set used in our experiments, the number of numerical and categorical features, and a description of the binary classification problem.

Table 1: Dataset characteristics.

Dataset	Training Size	#Features (Num/Cat)	Target Class
Lending Club	30.000	8/4	Loan status {**fully paid**, charged off}
Give Me Some Credit	30.000	6/3	SeriousDlqin2yrs {**no**, yes}
Bank Marketing	30.000	7/9	Deposit subscription {**yes**, no}
Credit Default	27.000	14/9	Default payment {**yes**, no}
Adult Census	32.000	4/8	Income {$\geq$ **50K**, $< 50K$}

Lending Club. The Lending Club public dataset [15] comprises detailed information on loans issued by the Lending Club company, including borrower characteristics, loan specifics, and performance indicators such as payment status. It serves as a resource for analyzing the financial performance of peer-to-peer loans with payment status as the target.

Give Me Some Credit. The Give Me Some Credit dataset [25] contains anonymized records of credit users, focusing on features such as their debt-to-income ratio, monthly income, and number of open credit lines. The target variable is a delay in payment for more than 90 days over the last two years.

Bank Marketing. The Bank Marketing dataset [10] contains information from a marketing campaign by a Portuguese bank, including client details, campaign contact information, and the outcome of each contact. It is used for predictive modeling to determine the likelihood of clients subscribing to a term deposit.

Credit Default. The Credit Default dataset [10] includes information on clients' credit card behavior, such as payment history, credit limits, and demographic details. The target is a default in payment.

Adult Census. The Adult dataset [10] contains demographic information and income data from the 1994 U.S. Census. The individual's income level serves as the target. The prediction is based on features like age, occupation, and capital gain.

4.3 Baseline competitors

We compare TABCF to related methods on counterfactual generation that either directly optimize in the input space or employ generative models to first learn latent representations (see Section 2). The descriptions of all methods are listed below.

4.3.1 Baselines operating in input space.

We compare to two state-of-the-art gradient-based methods that do not leverage latent representations.

Wachter [28] first mathematically defined CF explanations, using Stochastic Gradient Descent (SGD) to optimize a loss function. The objective is twofold, optimizing for the label flip of the black-box model, while minimizing the distance to the original sample. However, this method tends to find CFs very close to the decision boundary [26], and clamps categorical features to their original values [24, 28].

DiCE [22] extends the previous approach, taking into account more practical considerations regarding the feature types, such as incorporating a regularization loss for enforcing one-hot representations during optimization (see Section 5.2).

4.3.2 Baselines operating in latent space.

More recent works, similar to our approach, utilize generative autoencoders to learn latent representations of the data, before searching for CFs in that latent space. We have identified two such approaches for VAE-based tabular data CFs.

REVISE [17] employs a VAE of fully connected neural network layers to learn a structured latent space. As with other gradient-based approaches [22, 28], a loss function is optimized to change the predicted class while minimizing the distance to the original instance. This distance is measured in the reconstructed input, similar to our input-proximity loss term. Gradient descent updates the latent representations until the loss converges, after which the final CFs are returned by the decoder.

CCHVAE [25] similarly uses a conditional VAE for representation learning. Unlike earlier methods, it does not optimize a loss; rather CFs are found in the latent space using a model-agnostic search algorithm. Specifically, multiple points are sampled in the latent space with a growing-sphere approach, until some decoded sample matches the CF requirements, i.e., minimal proximity. Although such heuristic-based methods are more efficient, they do not guarantee optimal results and can be more difficult to adapt to new objectives or constraints.

Although the competitors employ various methods to handle categorical data, they all share the common practice of discretization (e.g. rounding) after each optimization step. Our experiments in Section 5.2 demonstrate that this results in feature-type bias.

4.4 Metrics

We evaluate TABCF and all baselines along several metrics to assess the effectiveness of each method. To ensure a fair comparison, each metric is calculated directly in the input space. Additionally, we evaluate all methods on the same test set, selecting $n = 1000$, previously unseen, instances that are not part of the target class, i.e., $X^0_{\text{test}} = \{x^0_i \mid f(x^0_i) = 0,\ i = 1, 2, \dots, n\}$. Each method generates a set of *valid* counterfactuals $CF_{\text{test}} = \{x'_i \mid f(x'_i) = 1\}$. Because CF generation is a difficult non-convex problem [22], it is not guaranteed that a *valid* CF will be found for each instance. Therefore, the number of valid CFs, $n_{val} = |CF_{\text{test}}|$, might be less than the number of test instances, i.e., $n_{val} \leq n$. We define all metrics in detail hereafter.

Validity. The validity score is the most important metric, as it measures the success rate, i.e. the percentage of instances for which optimization successfully switched the decision of the black-box classifier to the target class. More formally,

$$\text{Validity } (\uparrow) = \frac{n_{val}}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(f(x'_i) = 1)$$

Each of the following metrics is calculated only for valid CFs.

Sparsity. The sparsity scores measures the percentage of features changed to achieve a CF. We specifically differentiate between categorical and numerical sparsity,

$$\text{Sparsity Cat } (\downarrow) = \frac{1}{n_{val}}\sum_{i=1}^{n_{val}} \frac{\|x^0_{cat} - x'_{cat}\|_0}{|C|}$$

$$\text{Sparsity Num } (\downarrow) = \frac{1}{n_{val}}\sum_{i=1}^{n_{val}} \frac{\|x^0_{num} - x'_{num}\|_0}{|N|}$$

Where the $L0$ norm $\|x^0 - x'\|_0$, counts the number of features that have different values in x^0 compared to x'. The result is normalized by the total number of features, i.e. $|N|$ for numerical and for $|C|$ categorical.

Proximity. Proximity uses a distance function to measure how close the CF is to the original instance. It is defined only for numerical features since the categorical sparsity metric essentially plays the role of the proximity metric for categorical features.

$$\text{Proximity Num } (\downarrow) = \frac{1}{n_{val}}\sum_{i=1}^{n_{val}} \|x^0_{num} - x'_{num}\|_1$$

The numerical features are standard-normalized, and the L1 norm is used to measure the distance.

4.5 Feature importance and utilization

In our feature utilization experiment, we aim to examine the different features altered by each method during CF generation. The general assumption motivating this study is that features with a positive impact on the model prediction (target class) are good candidates for modification, towards a potential label flip. This is especially true if the positive impact comes with a minimal difference to the original value. Such features in the XAI domain, are referred to as *important* features, and corresponding scores, given a black-box model, can be estimated via *feature importance* methods [19]. We compute feature importance for X^0_{test} for the black-box model to study whether essential features are primarily subject to change for TABCF and the competitors.

For this study, we choose Shapley Additive Explanations (SHAP) [19] as the feature importance baseline. SHAP is a field-tested method for ML-based modeling in the finance domain [21], even further for modeling credit scoring systems [4, 20]. SHAP, originating from cooperative game theory, provides a way to distribute the payoff among players fairly based on their contributions. When applied to neural networks to explain feature importance, SHAP attributes the model's output to its input features by considering all possible feature combinations.

5 Results and Discussion

In this section, we discuss the results of our quantitative evaluation. Then we examine how the processing of tabular data by competitors can cause feature-type bias. Finally, we present the results of our ablation study on the hyperparameters of our loss functions.

5.1 Results for all baselines and datasets

We evaluate all methods on the five tabular datasets and report the average metric values across the test set. Detailed results for each dataset are presented in Table 3. For a comprehensive overview, we further average the results across all datasets to rank the methods, as shown in Table 2.

Table 2: Results averaged over all datasets.

	Validity (%) ↑	**Sparsity** (%) ↓		**Proximity** ↓	**Top 2** ↑
		Cat	**Num**	**Num**	
Wachter	0.92	-	1.0	4.18	1
DiCE	0.93	**0.18**	0.85	2.12	12
REVISE	0.54	0.25	0.98	2.44	3
CCHVAE	0.97	0.29	1.0	**0.43**	12
TabCF (us)	**0.99**	0.27	**0.83**	1.17	**14**

The results in Table 2 demonstrate that our method, TabCF, outperforms the competition. We achieve an almost perfect validity score, finding CFs for 99% of the input instances. Furthermore, in the last column (Top 2), we report the number of times each method ranks first or second best in the per-dataset results of Table 3. Here, TabCF stands out, ranking in the top two, for 14 out of a possible 20 times (70%).

Regarding sparsity and proximity metrics, TabCF changes more categorical features, resulting in 11% worse categorical sparsity on average, compared to the competitors. However, TabCF uses 15% fewer numerical features, while performing minimal changes when they are used. In particular, our performance in numerical proximity ranks second best and is 96% better than competitors on average. We compute these average percentages by comparing the individual performances to TabCF.

Overall, compared to the competitors, TabCF finds valid CFs 99% of the time while making fewer and smaller changes to features on average. Notably, the sparsity metrics reveal a clear bias among all competitors toward using numerical features instead of categorical ones. Specifically, Wachter, REVISE, and CCHVAE alter every single numerical feature when generating counterfactuals, for nearly all test instances (Sparsity Num = 1). Similarly, DiCE utilizes the least categorical features among all methods. We further investigate this observation in Section 5.2.

5.2 Feature utilization

Our observations indicate that competitors exhibit bias towards numerical features when identifying CFs, rather than using categorical features. This bias is problematic because an effective method should use features based solely on their influence on finding CFs, irrespective of their type.

To measure the *importance* of features on the model output, we can use the well-established Shapley explanation method [19] (as detailed in Section 4.5). For example, in Figure 3, we display the impact of various features from the Adult dataset on the output of

Table 3: Results per dataset.

	Validity (%) ↑	**Sparsity** (%) ↓		**Proximity** ↓
		Cat	**Num**	**Num**
		Lending Club		
Wachter	0.95	-	1.0	1.26
DiCE	0.99	0.18	**0.85**	0.83
REVISE	0.88	**0.07**	0.99	0.80
CCHVAE	0.95	0.12	1.0	**0.46**
TabCF	**1.0**	0.27	0.90	0.49
		Give me some credit		
Wachter	0.94	-	1.0	10.15
DiCE	0.98	**0.19**	0.89	3.78
REVISE	0.13	0.44	1.0	7.56
CCHVAE	0.96	0.61	1.0	**0.23**
TabCF	**1.0**	0.61	**0.86**	0.42
		Bank marketing		
Wachter	0.76	-	1.0	3.87
DiCE	0.87	**0.19**	**0.85**	2.49
REVISE	0.51	0.25	0.94	1.86
CCHVAE	0.97	0.23	1.0	**0.46**
TabCF	**1.0**	0.35	0.89	3.69
		Credit default		
Wachter	0.94	-	1.0	4.57
DiCE	0.98	0.31	**0.90**	0.58
REVISE	0.34	0.29	1.0	1.05
CCHVAE	0.99	**0.27**	1.0	**0.40**
TabCF	**1.0**	0.38	0.99	0.58
		Adult		
Wachter	**0.99**	-	1.0	1.06
DiCE	0.85	**0.02**	0.78	2.93
REVISE	0.84	0.19	0.99	0.97
CCHVAE	**0.99**	0.23	1.0	**0.61**
TabCF	0.95	0.33	**0.54**	0.67

the black-box classifier. The visualization reveals that categorical features such as education and occupation positively affect the model's predictions in some cases. Moreover, certain numerical features, like age and hours/week, can have a positive impact even with moderate changes in the feature value (indicated in purple). On the other hand, the capital gain/loss features only show a positive impact when their values are maximal (highlighted in pink). Given that CF methods aim to identify minimal changes with positive outcomes, we expect these methods to favor categorical features, like education and occupation, or numerical features like age and hours/week, when generating CFs for the Adult dataset.

However, we empirically show that this is not the case, by computing the *feature utilization* of each method on the Adult dataset. The resulting histogram in Figure 6 reveals that all competitors primarily use numerical features when generating CFs. DiCE exhibits a predominant imbalance towards numerical features, utilizing them 94% of the time. While the VAE-based methods (REVISE and CCHVAE) perform slightly better, they still exhibit a strong bias towards using the continuous capital gain/loss features. Our method TabCF stands out as the only one achieving a balanced use of both categorical and numerical features. Our assumption is that this tendency is related to how competitors handle categorical features. Specifically, all baseline implementations discretize categorical columns after each optimization step [24]. We illustrate why this processing approach can be problematic, with a real example for DiCE.

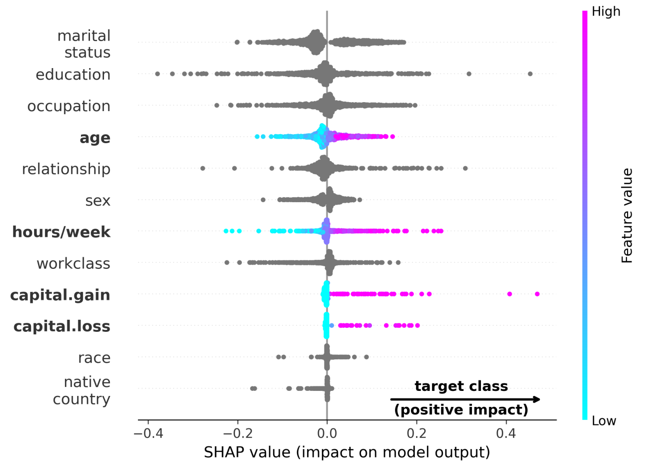


Figure 3: SHAP values for X^0_{test} on the Adult dataset. Colored dots indicate numerical features and grey dots indicate categorical features. The x-axis displays a positive target class impact to the right and a negative impact to the left.

The DiCE method calculates a regularization loss, defined as $\mathcal{L}_{reg} = \| \sum[x^{oh}] - 1\|_2$, for each one-hot encoded vector during optimization. This loss term penalizes vectors x^{oh} that do not sum to 1. Consequently, a gradient update that alters the distribution of x^{oh} results in a positive loss. Additionally, discretization is performed after each optimization step. Figure 4 visualizes the optimization process, showing how the distribution of a one-hot vector $x^{oh} = [c_0, c_1, \ldots, c_6]$ evolves over gradient steps. In two scenarios where a gradient update attempts to change the "hot" value of the feature, the regularization loss (depicted in the right plot) is triggered. Additionally, the gradient updates are insufficient to change the feature value, as the subsequent discretization step always selects the largest value among all c_i to be the "hot" one.

Thus, the regularization loss, together with the discretization approach used by all competitors, introduces feature-type bias towards continuous features. Our method, TABCF avoids this issue by utilizing our Gumbel decoder that produces discretized one-hot reconstructions for querying the black-box model, while using the soft samples for gradient optimization (refer to Section 3.1).

5.3 Ablation losses

The ablation study includes a five-step weight increase in the range [0,1] for λ_{prox_latent} and λ_{prox_input}. Therefore, we conduct 25 runs in total, measuring the metrics of Validity, numerical Proximity, and numerical and categorical Sparsity, on each weight combination, for the Adult dataset.

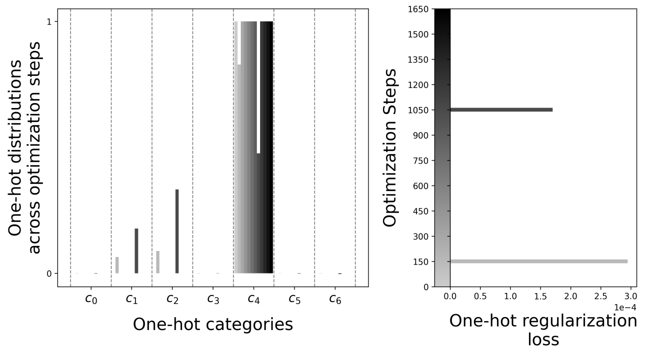
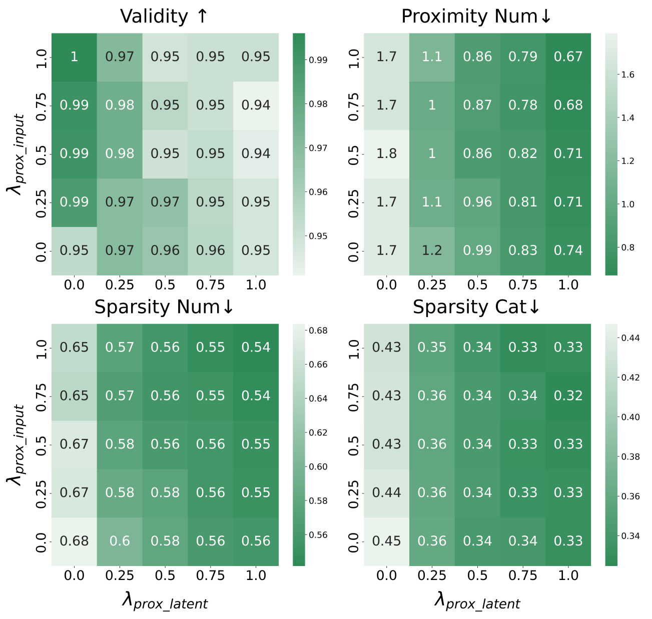


Figure 4: The left plot visualizes a one-hot vector with seven categories (c_i on the x-axis) throughout the optimization process. On the right, the values of the one-hot regularization loss (used by DiCE) are plotted (x-axis) over the number of optimization steps (y-axis).



Validity ↑ (rows λ_{prox_input}, columns λ_{prox_latent})

	0.0	0.25	0.5	0.75	1.0
1.0	1	0.97	0.95	0.95	0.95
0.75	0.99	0.98	0.95	0.95	0.94
0.5	0.99	0.98	0.95	0.95	0.94
0.25	0.99	0.97	0.97	0.95	0.95
0.0	0.95	0.97	0.96	0.96	0.95

Proximity Num↓

	0.0	0.25	0.5	0.75	1.0
1.0	1.7	1.1	0.86	0.79	0.67
0.75	1.7	1	0.87	0.78	0.68
0.5	1.8	1	0.86	0.82	0.71
0.25	1.7	1.1	0.96	0.81	0.71
0.0	1.7	1.2	0.99	0.83	0.74

Sparsity Num↓

	0.0	0.25	0.5	0.75	1.0
1.0	0.65	0.57	0.56	0.55	0.54
0.75	0.65	0.57	0.56	0.55	0.54
0.5	0.67	0.58	0.56	0.56	0.55
0.25	0.67	0.58	0.58	0.56	0.55
0.0	0.68	0.6	0.58	0.56	0.56

Sparsity Cat↓

	0.0	0.25	0.5	0.75	1.0
1.0	0.43	0.35	0.34	0.33	0.33
0.75	0.43	0.36	0.34	0.34	0.32
0.5	0.43	0.36	0.34	0.33	0.33
0.25	0.44	0.36	0.34	0.33	0.33
0.0	0.45	0.36	0.34	0.34	0.33

Figure 5: Ablation study for loss hyperparameters on the Adult dataset, for the Validity, Proximity, and Sparsity metrics. The x-axes show increasing values for λ_{prox_latent}, the y-axes increasing values for λ_{prox_input}. A stronger saturation indicates a better score.

As anticipated, we observe the conflicting nature of the validity desideratum, to proximity and sparsity. This trade-off arises because the validity loss term aims to push instances toward the target class, while the proximity loss terms (in the latent and input space), work to keep the CFs close to the original instance. Hence, sparsity and proximity metrics show better scores (more saturated in the plot) for larger values of the hyperparameters, since the loss terms have more influence. As previously discussed, based on the observations of this experiment, we select $\lambda_{prox\_input} = 1$ and $\lambda_{prox\_latent} = 1$

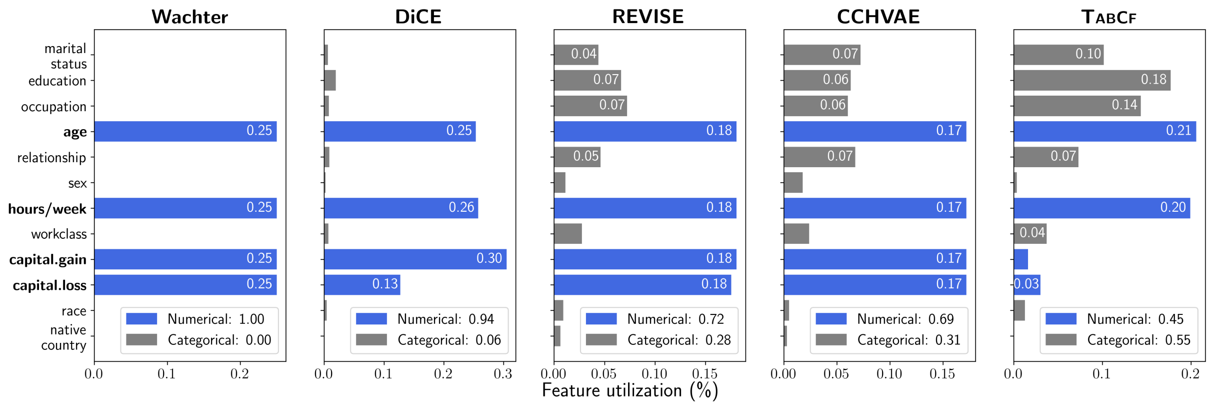


Figure 6: Histogram plot visualizing feature utilization for all methods on the Adult dataset. Competitors exhibit feature-type bias, more frequently using numerical features (blue) than categorical features (grey). In contrast, TABCF employs both feature types with similar frequency.

for weighting both loss terms, in all previous experiments, as these values provided the best overall trade-off.

6 Conclusion and future work

This paper presents *TABCF*, a method that leverages a transformer-based VAE for generating CF explanations for mixed tabular data. Our differentiable Gumbel-Softmax architecture allows precise reconstruction, overcoming feature-type bias present in previous approaches. Additionally, TABCF outperforms competitors in generating valid, proximal, and sparse counterfactuals, thus enhancing the interpretability of black-box models in real-world applications.

In future work, we would like to address user input constraints, such as immutable features or causal relationships between features, which could be achieved by conditioning the latent space. Furthermore, we plan to investigate the effect of distance-preserving Lipschitz-continuous VAEs on proximal counterfactual generation.

Acknowledgments

Our work is funded by the Deutsche Forschungsgemeinsschaft (DFG, German Research Foundation) - SFB1463 - 434502799. Additionally, this work received funds from the European Union under the Horizon Europe MAMMOth project, Grant Agreement ID: 101070285. We further acknowledge the support by the European Union Horizon Europe Project STELAR, Grant Agreement ID: 101070122.

References

[1] Amina Adadi and Mohammed Berrada. 2018. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access* 6 (2018), 52138–52160.
[2] Javier Antorán, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. 2020. Getting a clue: A method for explaining uncertainty estimates. *arXiv preprint arXiv:2006.06848* (2020).
[3] Mohit Bajaj, Lingyang Chu, Zi Yu Xue, Jian Pei, Lanjun Wang, Peter Cho-Ho Lam, and Yong Zhang. 2021. Robust counterfactual explanations on graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 5644–5655.
[4] Niklas Bussmann, Paolo Giudici, Dimitri Marinelli, and Jochen Papenbrock. 2021. Explainable Machine Learning in Credit Risk Management. *Computational Economics* 57 (01 2021). https://doi.org/10.1007/s10614-020-10042-0
[5] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. 2020. Multi-objective counterfactual explanations. In *International conference on parallel problem solving from nature*. Springer, 448–469.
[6] Javier Del Ser, Alejandro Barredo-Arrieta, Natalia Díaz-Rodríguez, Francisco Herrera, Anna Saranti, and Andreas Holzinger. 2024. On generating trustworthy counterfactual explanations. *Information Sciences* 655 (2024), 119898.
[7] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. 2018. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *Advances in neural information processing systems* 31 (2018).
[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
[9] Michael Downs, Jonathan L Chu, Yaniv Yacoby, Finale Doshi-Velez, and Weiwei Pan. 2020. Cruds: Counterfactual recourse using disentangled subspaces. *ICML WHI* 2020 (2020), 1–23.
[10] Andrew Frank. 2010. UCI machine learning repository. *http://archive. ics. uci. edu/ml* (2010).
[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).
[12] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems* 34 (2021), 18932–18943.
[13] Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)* 3 (2017).
[14] Frederik Hvilshøj, Alexandros Iosifidis, and Ira Assent. 2021. ECINN: efficient counterfactuals from invertible neural networks. *arXiv preprint arXiv:2103.13701* (2021).
[15] Julapa Jagtiani and Catharine Lemieux. 2019. The roles of alternative data and machine learning in fintech lending: evidence from the LendingClub consumer platform. *Financial Management* 48, 4 (2019), 1009–1029.
[16] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).
[17] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. 2019. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615* (2019).
[18] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[19] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).
[20] Branka Hadji Misheva, Joerg Osterrieder, Ali Hirsa, Onkar Kulkarni, and Stephen Fung Lin. 2021. Explainable AI in Credit Risk Management. arXiv:2103.00949 [q-fin.RM] https://arxiv.org/abs/2103.00949
[21] Karim El Mokhtari, Ben Peachey Higdon, and Ayşe Başar. 2019. Interpreting financial time series with SHAP values. In *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering* (Toronto,

Ontario, Canada) *(CASCON '19)*. IBM Corp., USA, 166–172.

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

[23] Philip Naumann and Eirini Ntoutsi. 2021. Consequence-aware sequential counterfactual generation. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*. Springer, 682–698.

[24] Martin Pawelczyk, Sascha Bielawski, Johannes van den Heuvel, Tobias Richter, and Gjergji Kasneci. 2021. Carla: a python library to benchmark algorithmic recourse and counterfactual explanation algorithms. *arXiv preprint arXiv:2108.00783* (2021).

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

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

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

[28] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.* 31 (2017), 841.

[29] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional gan. *Advances in neural information processing systems* 32 (2019).

[30] Hengrui Zhang, Jiani Zhang, Balasubramaniam Srinivasan, Zhengyuan Shen, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. 2024. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *ICLR 2024*. https://www.amazon.science/publications/mixed-type-tabular-data-synthesis-with-score-based-diffusion-in-latent-space

[31] Siqiong Zhou, Upala J Islam, Nicholaus Pfeiffer, Imon Banerjee, Bhavika K Patel, and Ashif S Iquebal. 2023. SCGAN: Sparse CounterGAN for counterfactual explanations in breast cancer prediction. *IEEE Transactions on Automation Science and Engineering* (2023).