Machine Learning
https://doi.org/10.1007/s10994-024-06530-1

Ijuice: integer JUstIfied counterfactual explanations

Alejandro Kuratomi[1] · Ioanna Miliou[1] · Zed Lee[1] · Tony Lindgren[1] · Panagiotis Papapetrou[1]

Received: 8 March 2023 / Revised: 21 December 2023 / Accepted: 14 February 2024

Abstract

Counterfactual explanations modify the feature values of an instance in order to alter its prediction from an undesired to a desired label. As such, they are highly useful for providing trustworthy interpretations of decision-making in domains where complex and opaque machine learning algorithms are utilized. To guarantee their quality and promote user trust, they need to satisfy the *faithfulness* desideratum, when supported by the data distribution. We hereby propose a counterfactual generation algorithm for mixed-feature spaces that prioritizes faithfulness through *k-justification*, a novel counterfactual property introduced in this paper. The proposed algorithm employs a graph representation of the search space and provides counterfactuals by solving an integer program. In addition, the algorithm is classifier-agnostic and is not dependent on the order in which the feature space is explored. In our empirical evaluation, we demonstrate that it guarantees k-justification while showing comparable performance to state-of-the-art methods in *feasibility*, *sparsity*, and *proximity*.

Keywords Machine Learning · Interpretability · Counterfactuals · Justification · Integer Programming · Graph Network

Editors: Dino Ienco, Roberto Interdonato, Pascal Poncelet.

✉ Alejandro Kuratomi
alejandro.kuratomi@dsv.su.se

Ioanna Miliou
ioanna.miliou@dsv.su.se

Zed Lee
zed.lee@dsv.su.se

Tony Lindgren
tony@dsv.su.se

Panagiotis Papapetrou
panagiotis@dsv.su.se

[1] Department of Computer and Systems Sciences, Stockholm University, Borgarfjordsgatan 12, Kista, 16455 Stockholm, Sweden

1 Introduction

Recent advances in machine learning have led to the development of post-hoc, model-agnostic interpretability algorithms. These algorithms are designed to explain the decisions and structure of the machine learning models, aiming to increase the users' understanding of them and promote their trustworthiness (Molnar, 2020; Laugel et al., 2019; Ribeiro et al., 2016). A subset of these interpretability algorithms, known as counterfactual (CF) explanations, attempts to assist the decision-making process by indicating what changes need to be applied to a subset of features so that the original prediction label switches to a desired one (Kuratomi et al., 2022; Wachter et al., 2017; Kyrimi et al., 2020). More concretely, given an instance of interest (IOI) x in a binary classification task and a trained classification model f that assigns x with a class label, a CF explanation provides an answer to the following question: *How should x be transformed to a CF x', so that the predicted label is switched* (Bobek & Nalepa, 2019). CF explanations can be very valuable in different domains. For example, in financial text classification, they may help to understand the sentiment associated with financial asset performance predictions (Yang et al., 2020), or in the image domain, to understand which parts should be blurred or modified in an image to relabel it into another class (Vermeire et al., 2022). Consider, in the medical domain, a complex ML model built on a large medical dataset that accurately predicts the risk of future severe disease occurrence. Given a patient with a predicted high risk of severe disease development, a CF can suggest the shortest action path to prevent this disease (Molnar, 2020; Kyrimi et al., 2020). As the search space for performing this transformation is huge, a variety of CF explanations exist (Molnar, 2020; Rudin, 2019; Lindgren et al., 2019; Tolomei et al., 2017; Laugel et al., 2017; Poyiadzi et al., 2020; Mothilal et al., 2020; Karimi et al., 2020; Pawelczyk et al., 2020) depending on what type of changes and which CF quality measure is prioritized. There is currently no consensus on the best metric to optimize for better CF quality (Molnar, 2020; Rudin, 2019). This paper focuses on generating CFs that satisfy desiderata, such as proximity, feasibility, sparsity, and, most importantly, *faithfulness* so that the CFs are supported by and connected to observations.

Particularly, proximity between the IOI and the CF (Wachter et al., 2017; Tolomei et al., 2017; Lindgren et al., 2019) can be quantified using different distance metrics, e.g., $l2$-norm or $l0$-norm (the latter known as sparsity or minimality Guidotti (2022), which is the number of features that are changed) (Wachter et al., 2017; Byrne, 2019; Miller, 2019; Kuratomi et al., 2022). Another measure is feasibility (Wachter et al., 2017), which indicates whether (1) the values in the CF x' are possible to occur (*plausibility*), (2) only mutable features are modified (*mutability*), and (3) the changes are done in a possible direction only (*directionality*), e.g., age may only increase.

Last but not least, faithfulness (Pawelczyk et al., 2020; Rudin, 2019; Laugel et al., 2019) indicates whether the CF is supported by the data distribution. Faithfulness promotes user trust in decisions based on the CF and may be achieved through CF likelihood (Dandl et al., 2020; Pawelczyk et al., 2020) and/or through *justification* (Laugel et al., 2019; Kuratomi et al., 2022). A given CF is *justified* if it has a *connection* to a correctly classified CF training observation, e. A connection between two instances is defined as the existence of a path between them, such that all the points in the path belong to the same class as the two connected instances, i.e., the CF lies in the same classification region as one or more CF training points. Since counterfactual explanations are able to provide actionable statements for a given instance (which could be a patient with a high risk of developing severe disease, as in the previous example), it is of utmost importance that the ground truth observations support such explanations (Laugel et al., 2019), such that the proposed potential treatments or medical advice embedded in it, adhere to reality as faithfully as possible. Justification is a property that indicates whether the link between the CF explanation obtained and ground truth observations exists or not. It is easier to trust a feasible, justified, and likely CF according to the data, even if it is not the closest CF to its corresponding IOI x. Therefore, the property of justification is important because it increases the faithfulness of the obtained CFs, providing a stronger and more trustworthy CF explanation, as described in Kuratomi et al. (2022). The CF explanation could then be presented in the following manner: *x is predicted to have a negative label, but this label changes to a positive one if x is changed to x', which is justified by the observation e*. Different algorithms have been developed, each aiming to satisfy a different subset of these measures and other measures discussed in this field's state of the art.

The recently proposed JUstIfied Counterfactual Explanations (JUICE) algorithm by Kuratomi et al. (2022) obtains a faithful (justified) CF by *connecting* the generated CF x' to a *single* training observation e in mixed-feature (heterogeneous) spaces. For binary and one-hot encoded categorical features, a connection is defined by performing a set of binary flips, hence traversing the feature space from the CF to a training example or vice-versa. A similar process is followed for ordinal features, where changes are applied sequentially in the order of the ordinal feature. Finally, for continuous features, the values are changed in small steps. If there are several features of the same type, then the permutations of the features that are different are considered. JUICE first starts by navigating from the closest feasible CF training observation, towards the IOI following three sequential steps: (1) search for binary connections, (2) search for ordinal connections, and (3) search for continuous connections. Only feasible feature value changes are considered (changes that hold the properties of plausibility, mutability, and directionality with respect to the IOI). Since JUICE starts from the closest *feasible* nearest neighbor CF, the JUICE CF is feasible.

However, JUICE's CF is only guaranteed to be justified (connected to) one *single* training observation, and is not able to examine whether this justifier instance is representative of the CF class data distribution or not, i.e., it does not differentiate between an outlying or a likely instance. This can be a problem, since justifying the CF with an outlier may imply supporting the explanation to the users with a non-representative observation of the desired class, possibly hindering the explanation's trustworthiness. An experiment showed that the probability of obtaining an outlier instance as justifier through JUICE is almost 10%.

Additionally, the explanation's potential credibility may be greatly increased if instead of outputting a single justifier, which could be an outlier, the algorithm was able to attain a higher number of justifiers, i.e., obtaining justified CFs that are connected to several CF observations instead of just one. Obtaining a CF that is justified by more than one instance would provide more complete explanations of the following form: *x is predicted to have a negative label, but this label changes to a positive one if x is changed to x', which is justified by the set of observations T_k*, where $|T_k| > 1$ (See Table 1). This more complete form provides a higher number of observations that support the region where the CF is located, thereby increasing the trustworthiness of the explanation when compared to having only one justifier. To the best of our knowledge, there is no algorithm for counterfactual explanation generation that focuses on the justification of the counterfactuals from a set of training observations and not just a single justifier, which could be an outlier. We hereby propose such an algorithm, which outputs CF explanations with higher credibility which could increase the utilization of the ML algorithms in different sectors. Our contributions can be summarized as follows:

Table 1 List of variables and definitions

Variable	Definition
A	Adjacency matrix of graph network $\mathcal{G}$
b	Number of binary features in dataset D
$\mathbf{B}(a, b)$	Binary connection between instances a and b
c	Number of one-hot encoded categorical features in dataset D
C_i	Cost parameter of node $n_i \in \mathcal{N}$
CF_T	Closest feasible CF training observation
$\mathbf{C}(a, b)$	Continuous connection between instances a and b
D	Training dataset
$d(a, b)$	Distance function between instances a and b
$\mathbf{dir}$	Feature directionality vector
e	Justifier training observation
$\mathcal{E}$	Set of edges in the graph $\mathcal{G}$
f	Classifier function
$\mathcal{G}$	Graph network
h_u	Number of ordinal values in ordinal feature u
k	Number of potential justifier training observations (defined by the user)
$\mathcal{K}$	Set of potential justifier training observations
$L_1(\cdot), L_2(\cdot)$	Cost functions for problems 1 and 2, respectively
λ	Distance and justification ratio weight parameter
$\mathbf{mut}$	Feature mutability vector
$\mathcal{N}$	Set of nodes in graph $\mathcal{G}$
o	Number of ordinal features in dataset D
$\mathbf{O}(a, b)$	Ordinal connection between instances a and b
p	Number of bins used to discretize the continuous features
$\Psi(a, b)$	Connection (of any type) between instances a and b
$\mathbf{pla}$	Feature plausibility vector
q	Number of training observations in D
r	Number of continuous features in dataset D
s_v	Justifier decision variable for node $n_v \in \mathcal{K}$ in the iJUICE IP
T_k	Set of justifier training observations of x'
w	Total number of features ($w = b + c + r + o$)
x, x'	Instance of interest and its corresponding counterfactual
x_i	CF decision variable for node $n_i \in \mathcal{N}$ in the iJUICE IP
$\mathcal{X}$	Mixed-feature space
y_{ij}	Edge decision variable between nodes $n_i, n_j \in \mathcal{N}$ in the iJUICE IP
$\mathcal{Y}$	Binary class label space

1. We formulate the problem of **k-justification**, a stronger formulation of CF justification, which consists of finding the k closest training instances that can potentially justify a CF instead of a single justifier instance, hence increasing the faithfulness of the CF and avoiding justifications by single outliers.
2. We introduce the **iJUICE** algorithm, shorthand for integer JUstIfied Counterfactual Explanations, a novel algorithm that solves the *k-justification* problem and generates CFs

in mixed-feature spaces that are proximity-optimal, feasible, stable, valid, and at most k-justified by extending JUICE. The main novelty of iJUICE includes k-justification and treating the problem as a graph-based integer program, to guarantee connectivity between the counterfactuals and training observations.

3. We provide an extensive experimental evaluation of iJUICE against 10 model-agnostic competitors on 14 publicly available datasets using different CF evaluation metrics, on which we show that iJUICE improves over JUICE and other state-of-the-art algorithms in terms of proximity while maintaining similar computational times. We additionally perform an **ablation study**, evaluating the impact of different distance measures and weights of k-justification as well as the impact of parameter *k* in proximity and justification ratio performance. We demonstrate the monotonic behavior of proximity and justification ratio, and how *k* may allow for a large amount of justifiers, increasing the trustworthiness of the explanations.
4. Finally, we propose a **mixed-feature space justification verification process** that builds a graph network using a subset of training CF observations, and uses this network to prevent unseen paths to be missed, and hence possibly fail at detecting connection paths to training observations lying in the same classification region.

The rest of the document is divided as follows: Sect. 2 is dedicated to the relation of state-of-the-art counterfactual generation methods that are presented in the literature. Section 3 introduces the notation and concepts required to understand the justification counterfactual generation problem which is ultimately described in Sect. 4. Section 5 introduces the iJUICE method, whilst Sect. 6 presents the evaluation of iJUICE along other state-of-the-art counterfactual generation methods in a set of 14 binary classification datasets. Finally we have Sect. 7 dedicated to the discussion of the usage of the justification property and Sect. 8 presenting the conclusions of the work.

2 Related work

The related work is divided into two subsections. The first subsection describes the different measures used for the evaluation of the CF explanations, and the second subsection describes different methods that contrast with the iJUICE algorithm presented here.

2.1 Evaluation of CF explanations

The evaluation of the CF explanations is an ongoing research topic, and may be performed through different measures related to different CF properties (Guidotti, 2022; Miller, 2019; Sharma et al., 2020; Molnar, 2020; Rudin, 2019; Karimi et al., 2022). Different CF generation algorithms prioritize different counterfactual properties. Several documents exist where these properties, along with the algorithms that prioritize them are reviewed. Guidotti (2022) provides the most recent, extensive overview and benchmark of the CF generation algorithms and their desiderata. Verma et al. (2020) also discuss these CF and algorithm properties, whilst also relating to some of the challenges faced by the current CF generation algorithms. Karimi et al. (2022) focus on the property of recourse and its relation to *actionability* and feasibility of CFs. We leverage the knowledge gathered in these reviews and other documents to provide an overview on the different evaluation measures and the methods prioritizing them in CF explanations.

The property of *validity* indicates whether the obtained CF has another predicted label compared to the original instance of interest (Verma et al., 2020; Wachter et al., 2017; Mothilal et al., 2020; Guidotti, 2022; Karimi et al., 2022, 2020). This property is not guaranteed by all the CF generation algorithms (Guidotti, 2022), meaning that these algorithms could output instances that still present the factual label (Mothilal et al., 2020) and are therefore not valid CFs.

The properties of plausibility, feasibility, actionability and causality are highly connected and discussed throughout the literature, and different researchers have different connotations for these terms. Guidotti indicates that an instance should be plausible if it lies inside the observed range of values (mentioned as *domain-consistency* in Karimi et al. (2022)) and is not considered an outlier with respect to the dataset Guidotti (2022) (mentioned as *density-consistency* in Karimi et al. (2022) and as feasibility in de Oliveira and Martens (2021)). In this sense, the *closeness to the data*, a property mentioned and analyzed by Verma et al. (2020), is highly related to plausibility, since the closer the CF to denser spaces, the better it is in terms of its likelihood with respect to the data. In Russell (2019), *coherence* is related to this closeness to the data, as it is defined as the property of a CF that can be mapped back to the original data. Additionally, the justification property of interest in the present study which was initially proposed by Laugel et al. (2019), is considered an approximation of plausibility (Guidotti, 2022). Guidotti also states that plausibility is sometimes also referred to as feasibility or reliability (Guidotti, 2022). Karimi et al. also defines a third category for plausibility, which is known as *prototype-consistency*, which indicates that a CF is plausible if it is similar to a prototype instance in the dataset (Karimi et al., 2022). However, it is not enough that a CF is plausible. Verma et al. point out that the closeness to the data should be accompanied by a measure of feasibility. In their study, feasibility is defined as the compliance with the causality constraints: if a given CF does not comply with causal relations among features, then the CF is unfeasible (Verma et al., 2020). This is also specified by Guidotti, who explains causality as a prerrequisite to plausibility and actionability (Guidotti, 2022). In Karimi et al. (Karimi et al., 2022) feasibility is defined as actionability, meaning that a CF must only be feasible if the interventions proposed are actually possible, i.e., only actionable features are manipulated, but this does not mean that only actionable features can be changed. In this sense, if a given actionable feature is changed, other non-actionable features may change due to causality among features (Karimi et al., 2022), but these non-actionable features should then be mutable. In Guidotti and Karimi et al., actionability implies mutability (Guidotti, 2022; Karimi et al., 2022).

Another property is known as *diversity*, defined as the difference among a set of CFs (if the algorithm is designed to extract more than one CF) (Guidotti, 2022; Karimi et al., 2020, 2022; Verma et al., 2020; Mothilal et al., 2020). It may be measured in different ways, based on the distance between the CFs or the sparsity among them (Mothilal et al., 2020; Karimi et al., 2020). Sparsity or minimality is simply the number of features changed between the instance of interest and the CF (Guidotti, 2022; Verma et al., 2020; Karimi et al., 2020, 2022).

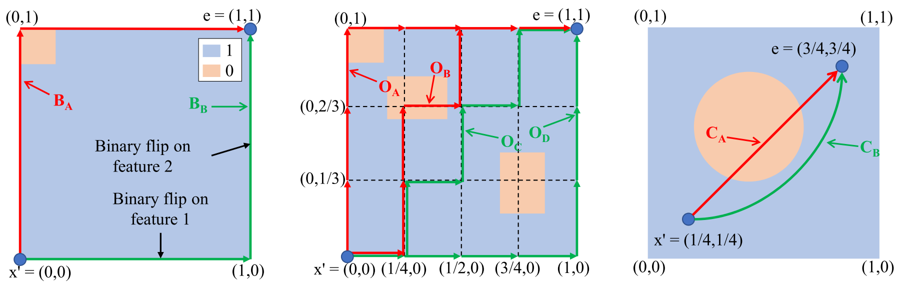
Finally, other properties are described in the literature, such as *discriminative power* (which is not related to discrimination or fairness). Discriminative power is defined as a subjective property shown by CFs that, when read by humans, can easily show why the feature changes proposed would lead to a different predicted class label (Guidotti, 2022). *Stability*, also known as robustness (Guidotti, 2022; Verma et al., 2020; Karimi et al., 2022), is a property of the CF generation algorithm, stating that a stable CF generator should not change its output when run several times for the same instance of interest, or should only change slightly when run for two different but very close instances. *Fairness*, a property associated to both the CF and the CF generation algorithms (Guidotti, 2022) is important for trustworthiness since biased explanations may shed light into biases in the model or hinder its credibility. Guidotti indicates that a fair counterfactual should be indicating similar changes required between a person in a given demographic or sensitive group to reach the desired class label as another belonging to another sensitive group (the concept of *burden* described by Sharma et al. (2020)). Kuratomi et al. (2022) indicate that there are different ways to measure fairness using counterfactuals and propose a way by combining burden with the false negative rate of that specific sensitive group. We now briefly describe some of the algorithms mentioned in the literature that prioritize these properties.

2.2 Counterfactual generation methods

There are different CF generation methods available (Guidotti, 2022). The Nearest Neighbor (NN), Minimum Observable (MO) (Wexler et al., 2019), and Random Forest Tweaking (RT) algorithms (Lindgren et al., 2019) obtain justified CFs at the cost of proximity performance, as they are restricted to searching the CF among existing observations. Both NN and MO search among observations, but NN searches only among training observations, while MO also uses the test data with the predicted label. Similarly, Counterfactual Conditional Heterogeneous Autoencoder (CCHVAE) sacrifices proximity in favor of faithfulness by optimizing for likelihood in an autoencoder latent space with respect to the training data (Pawelczyk et al., 2020). Additionally, just as in the RT method, Actionable Feature Tweaking (FT) must use a random forest classifier to search for CFs (Tolomei et al., 2017).

Furthermore, the Growing Spheres (GS) (Laugel et al., 2017) algorithm expands a sphere centered around the IOI until the decision boundary is met, in the continuous feature space only, which maximizes proximity, while the Diverse Counterfactual Explanation (DiCE) (Mothilal et al., 2020) algorithm is able to output a sparse and diverse set of CFs. The Feasible and Actionable Counterfactual Explanation (FACE) (Poyiadzi et al., 2020) algorithm searches for feasible paths to generate the counterfactual but attains lower proximity performance while also not considering CF faithfulness.

Finally, other algorithms use optimization frameworks to find an optimal CF with respect to an objective function. The Model-Agnostic Counterfactual Explanation (MACE) (Karimi et al., 2020) algorithm uses the Satisfiability Modulo Theory (SMT) to maximize CF proximity or sparsity while preserving feasibility. The Actionable Recourse (AR) (Ustun et al., 2019) algorithm uses an Integer Program (IP) to find the CF by considering feature changes that maximize CF actionability while preserving feasibility, however, it demands a linear classifier. This heavily constraints the applicability of the CF method, since the most accurate ML models present a highly nonlinear behavior (Molnar, 2020; Rudin, 2019). However, as compiled by Guidotti (2022), there are several other optimization-based CF generation algorithms that are constrained on the type of classifier they can explain. Other linear classifier-oriented algorithms are the Diverse Coherent Explanations (DCE) (Russell, 2019), which discretizes continuous features to formulate integer constraints to optimize for diversity and plausibility, and the Distribution-Aware Counterfactual Explanation (DACE) (Kanamori et al., 2020), which is constrained to either linear classifiers or tree ensembles, and considers the Local Outlier Factor (LOF) to output a CF that is close enough to the data distribution (plausible). Among the tree classifier-oriented algorithms are the Counterfactual Explanation for Oblique Decision Trees (CEODT) (Carreira-Perpiñán & Hada, 2021), which provides a distance-optimal CF for tree-based models and can handle oblique function trees and the Optimal Counterfactual ExplAiNer



(a) Binary connection. (b) Ordinal connection. (c) Continuous connection.

Fig. 1 As shown in JUICE: a connection is obtained through the binary path B_B, the ordinal paths O_C or O_D and the continuous path C_B

(OCEAN) (Parmentier & Vidal, 2021), which focuses on actionability and is applied on tree ensembles. These methods share the mathematical programming approach of iJUICE, but they assume a given classifier structure (either linear or tree-based) which makes them not ideal for comparison. We do not constrain the classifiers to have any structure so as to prioritize a higher classifier performance that justified the requirement for explainability algorithms.

Another tree-based counterfactual generator is the LOcal Rule-based Explainer (LORE) (Guidotti et al., 2019; Guidotti, 2022) which is a local surrogate model that builds a tree on synthetic data to provide both feature relevance and counterfactual rules. Moreover, additional methods that output both feature relevance in the form of regression coefficients and counterfactuals (Guidotti, 2022) based on a set of randomly generated points or perturbations are LIME-C and SHAP-C (by Ramon et al. (2020)) and Counterfactual Local Explanations via Regression or CLEAR (by White and Garcez (2019)) which focus on a measure called fidelity for the regression and CF sparsity. Although these methods focus on counterfactual extraction, their main contribution is their capability to deliver both counterfactual explanations and feature relevance-based explanations, which is an aspect iJUICE does not focus on. Therefore, we have chosen to focus on known methods that are similar to iJUICE and focus only on counterfactual generation.

In addition, some solutions employ evolutionary algorithms, like the Multi-Objective Counterfactual (MOC) (Dandl et al., 2020), and the CERTIFAI algorithms (Sharma et al., 2020) to derive diverse and sparse CFs. These methods are also quite interesting, but, due to the randomness associated to the genetic algorithms and their potential non-optimal solutions (obtaining local optima), we believe they would not exactly fit as baselines for the iJUICE algorithm.

Other algorithms that may focus on non-tabular data, like the Search for Evidence Counterfactual (SEDC), initially proposed by Martens and Provost (2014) and later adapted for image counterfactuals by Vermeire et al. (2022), and extended to its multiclass version, SEDC-T, with a given target T class. Furthermore, Guidotti and Ruggieri (2021) propose a novel approach of ensembles for counterfactual generation, with different weak explainers and each focusing on a different measure, among them sparsity, stability, actionability, plausibility and discriminative power. Although interesting, iJUICE currently aims to provide counterfactuals on tabular datasets as a single generation method. Therefore, we have excluded SEDC and its variants and this ensemble method.

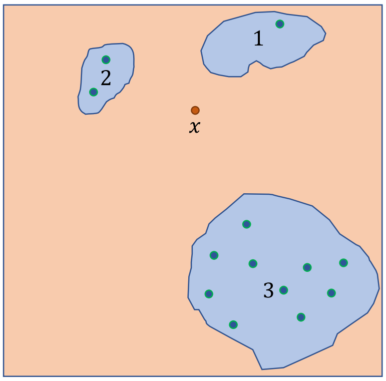


Fig. 2 Point x is surrounded by 3 regions of CF points. The blue region 1 has only 1 justifier. The region 2 presents 2 outlier training observations as justifiers. The region 3 presents a high number of justifiers. A CF found in region 3 has a better support according to the data distribution and could lead to higher explanation trustworthiness and credibility from users

A counterfactual can also be referred to as *recourse*, which, in general, is a term used to define explanations and recommendations in the form of counterfactuals (Verma et al., 2020; Karimi et al., 2022; Ustun et al., 2019; Rawal & Lakkaraju, 2020). Karimi et al. (2022) indicate that recourse explanations are usually sought after by current CF generation algorithms, and that they try to answer the question *what is the set of values that are required to be classified in the desired label?*, while recourse recommendations answer the question *what actions are needed to reach the set of values that are required to be classified in the desired label?*. These two are different and answering the second question is much harder than the first, since it requires the knowledge of causality among features.

Moreover, in the survey by de Oliveira and Martens (2021), algorithms are selected based on their capability to deal with neural network-based classifiers. Finally, the reader is referred to Guidotti (2022) for further details on the different algorithms and their comparison, and to the CARLA framework, which is a benchmarking library that contains several of the counterfactual generation algorithms already implemented, and on which we base ourselves to implement the CCHVAE and FACE algorithms (see Sect. 6.3 and "Appendix A"). None of the described algorithms however guarantee justification as a CF property.

3 Preliminaries

Table 1 summarizes the notation and variables used throughout the rest of the paper. Let D be a dataset used to train a classifier $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ is defined as a *mixed-feature space* of binary, categorical, ordinal, and continuous features, and $\mathcal{Y}$ is a binary class label space. We use $x \in \mathcal{X}$ to refer to an IOI, and x' to refer to a CF to x ($f(x') \neq f(x)$). Finally, let us define $e \in D$, and $f(e) = f(x') \neq f(x)$. e is a counterfactual training observation that may *justify* x'. We say that x' is *justified* by e, when at least a *connection* between x' and e exists. We define the *connection* between any pair of instances α and β, where $\alpha, \beta \in \mathcal{X}$ as a path starting on any of them and reaching the other without crossing any decision boundary of the classifier f, i.e., both instances lie in the same classification region constructed by f.

This definition was initially applied to continuous spaces by Laugel et al. (2019), and Kuratomi et al. extended it to heterogeneous spaces (Kuratomi et al., 2022). In a heterogeneous space, where binary, categorical, ordinal and continuous features exist, the connection between any pair of instances may be verified by checking that *incremental*, *adjacent*

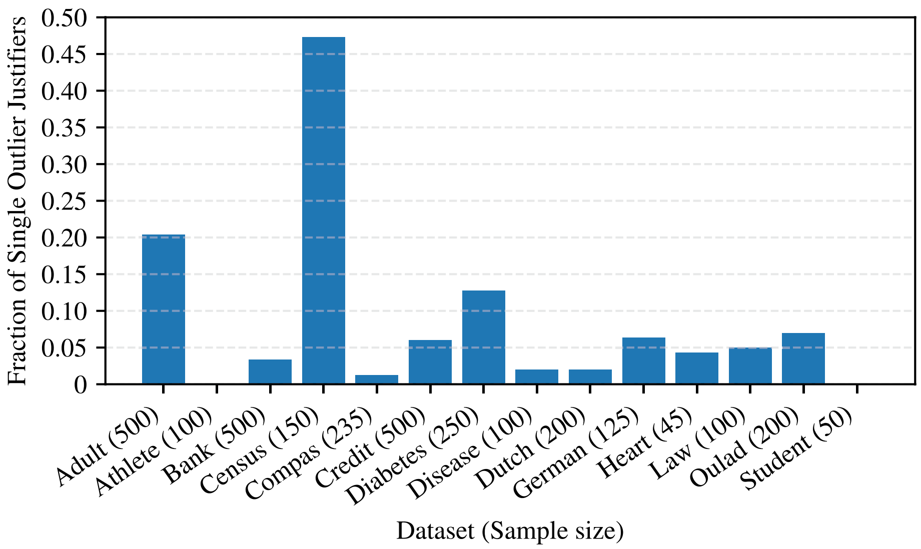


Fig. 3 Fraction of single justifiers that are considered outliers according to the distribution of the counterfactual, desired class training observations

and *feasible* feature changes from any of the instances to the other have always the same class label as that of both of the instances. We defined three kinds of connections for heterogeneous spaces: *binary*, which indicates that two instances are binary-connected if incremental binary feature flips (combinations of the values between the binary features of α and β) lead to a path of the same label between them. Categorical features are included in this kind of connection, since they are one-hot encoded; a similar definition is given for *ordinal* and *continuous* connections. We also indicate that whenever the three connection types exist between two instances, we may say that they are connected in a heterogeneous space (Kuratomi et al., 2022). In the case of a training CF observation e and a CF x', the connection entails justification from e to x'. If there are no other training CF observations connecting to x' in its classification region, this means that x' is only justified by one instance (only e justifies x'). Figure 1 shows how these connections may look like (from Kuratomi et al. (2022)).

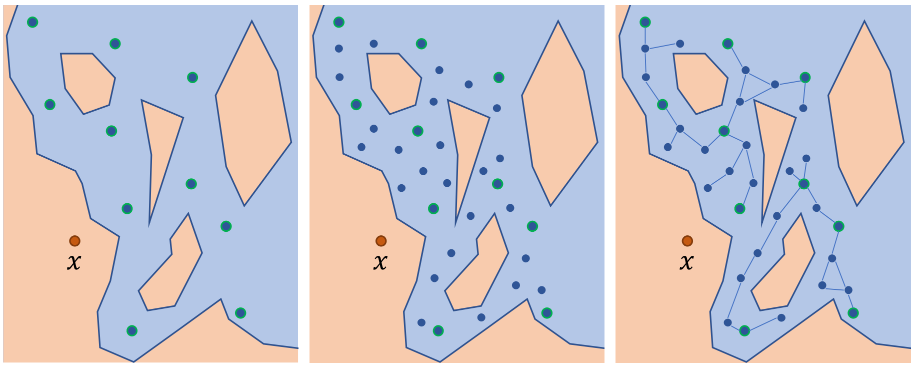
We now introduce the concept of *k-justification* which allows the CF to be at most justified by the k closest training observations. Figure 2 serves to illustrate this concept in a 2-dimensional space. In this figure, point x is located in the light orange class region, surrounded by 3 light blue enumerated regions. Each of the blue regions is located at a different distance from point x and has a different number of supporting instances or blue training observations (surrounded by a green outline). Any of the 3 regions presents at least 1 justifier for any CF x' found inside, so justification may be guaranteed inside of them.

Note however that regions 1 and 2 provide not only few justifiers, but also justifiers that may be considered outliers regarding the distribution of the counterfactual class data points. These two facts may influence the trustworthiness of the explanations if a counterfactual is obtained in region 1 or region 2, where outliers (which may not be representative of the class distribution due to particular characteristics of the instances or measurement errors) have taught the model to create a blue class region. The same situation could arise in real datasets, hence, having a higher number of justifiers may increase the credibility of the CF. If we focus on maximizing the amount of justifiers connected to the CF as part of the objective, we may provide higher trustworthiness. Region 3 has a higher density of blue class instances. Obtaining the CF x' in region 3 should be better for the trustworthiness, credibility and legitimacy of the obtained explanation from the user perspective, since it may be supported by a higher number of points in the dataset.

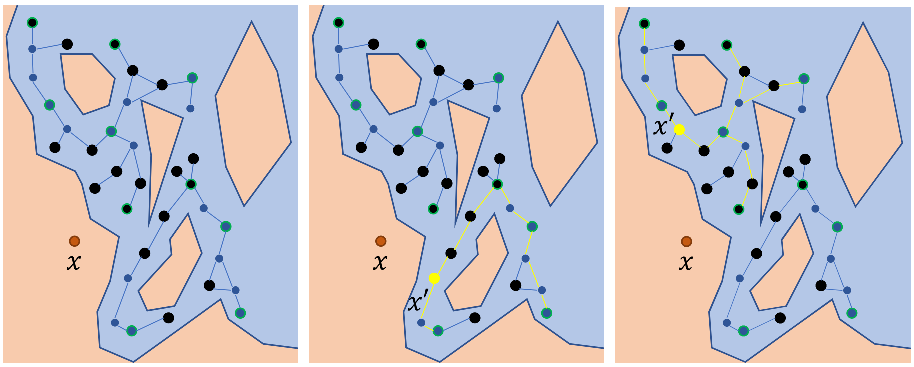
4 Problem description

Problem 1 (Justified Counterfactual Generation) Given a classifier f trained on a dataset D and a cost function $L_1(\cdot)$, a justified counterfactual example $x' \in \mathcal{X}$ to instance x is defined as:

$$x' = \arg\min_{z} \{ L_1(x,z) | f(x) \neq f(z) \wedge \exists\ e \in D : \mathbf{B}(z,e) \wedge \mathbf{O}(z,e) \wedge \mathbf{C}(z,e) \} \tag{1}$$



(a) Step 1: Gather the k-NN CFs. (b) Step 2: Obtain the graph nodes. (c) Step 3: Create the adjacency matrix.



(d) Step 4: Mark the unfeasible points. (e) Step 5 and 6: Solve the IP ($\lambda = 1$). (f) Step 5 and 6: Solve the IP ($\lambda = 0$).

Fig. 4 iJUICE algorithm 2-dimensional example where $\mathbf{B}(x', e)$, $\mathbf{O}(x', e)$ and $\mathbf{C}(x', e)$ denote the existence of a binary, ordinal, and continuous connection, respectively, between x' and e, e being a training CF observation (Kuratomi et al., 2022).

We reformulate Problem 1 in order to consider the concept of *k-justification* and integrate it into the justified counterfactual generation problem, as follows:

Problem 2 (k-justified Counterfactual Generation) Given a classifier f trained on a dataset D, a number $k \in \mathbb{Z}^+$, with $k \le |D|$, and a cost function $L_2(\cdot)$, a justified counterfactual example $x' \in \mathcal{X}$ to instance x, which is justified by at most k close training observations, is defined as:

$$x' = \arg\min_{z} \{L_2(x, z, T_k) | f(x) \neq f(z) \wedge \exists\ T_k \in D : \Psi(z, T_k)\}, \tag{2}$$

where T_k is the set of CF training observations that are connected to (i.e., they justify) x', $1 \le |T_k| \le k$, and k (set by the user) is the maximum number of close CF training observations to x considered to justify x'.

In Eq. 2, $\Psi(a, b)$ is a function that indicates whether points a and b are connected through *any* path in the space, regardless of the feature type or the navigation order of the features. The CF x' is justified by *at least* one and at most k training observations. In general, this strengthens the faithfulness of the potential CFs found inside by avoiding lack of justification or justification by a set of potential outliers.

To illustrate the need for *k-justification*, we measure the probability of obtaining a single-justifier that happens to be an outlier. For this, we perform a small set of experiments on the datasets and classifiers used. We start by estimating the likelihood of each of the training observations in the desired class, and setting a threshold below which we flag them as outliers. Then, we randomly select a set of instances from the undesired class and perform the JUICE algorithm to obtain a CF, which is guaranteed to be justified by a single instance and has then the risk of being justified by an outlier. Finally, we count the times this justifier instance is a training outlier for the set of instances selected. The results are shown in Fig. 3. In only 2 of the 14 datasets there was no justification from an outlier, however, the highest fraction of outlier justification was found in the Census dataset with a total of 47.3% of the sample size. The average fraction of outlier justifications in all datasets is 9.5%, i.e., almost 1 out of 10 CFs obtained could be justified by an outlier, which is a considerable value.

To estimate which of the samples in the desired class are outliers, we implemented a multivariate kernel density estimation, and selected the instances with the lowest 5% log-likelihood according to the density estimation. In order to avoid justifying a CF through a single outlier observation, the iJUICE algorithm, which prioritizes k-justification, is proposed and discussed next.

5 iJUICE: integer JUstIfied counterfactual explanations

We hereby make a brief explanation on mixed-feature or heterogeneous spaces and how to find the connections among instances in it, then proceed to describe the iJUICE algorithm.

5.1 Traversing the mixed-feature space

iJUICE employs a graph network for traversing the mixed-feature space $\mathcal{X}$ and generating candidate CFs. More concretely, let $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$ be the graph network, where each node $n_i \in \mathcal{N}$ corresponds to a point in the mixed-feature space $\mathcal{X}$, and an edge exists between two nodes n_i, n_j when a single feature value changes between them. Let $\mathcal{K}$ be the set of k close CF training observations to the IOI x, with $\mathcal{K} \subseteq \mathcal{N}$. Let x_i be a binary variable that indicates whether any node $n_i \in \mathcal{N} \setminus \mathcal{K}$ is selected as the iJUICE CF. Only a single node n_i may be selected (i.e., $\sum_{n_i \in \mathcal{N} \setminus \mathcal{K}} x_i = 1$). Let s_v be a binary variable that indicates whether any training observation $n_v \in \mathcal{K}$ is chosen as a justifying node for the selected CF by x_i. Several nodes in $\mathcal{K}$ (up to $|\mathcal{K}| = k$ defined by the user) may justify the selected CF.

Moreover, let $y_{ij} \in \mathbb{Z}$ be a variable that indicates the number of times the edge connecting nodes $n_i, n_j \in \mathcal{G}, i \neq j$ is used, and let $C_i \in \mathbb{R}$ be the cost associated to a node $n_i \in \mathcal{N}$, defined as the distance between n_i and IOI x. We also denote the adjacency matrix of $\mathcal{G}$ as A. Finally, a node is feasible with respect to the IOI x if, when selected as CF, the properties of mutability, directionality, and plausibility are satisfied. We use F_i to denote the feasibility of node $n_i \in \mathcal{N}$.

5.2 The iJUICE algorithm

iJUICE comprises six steps, which are outlined in Algorithm 1, while a 2-dimensional example of these steps is depicted in Fig. 4. Next, we describe these steps in more detail.

Algorithm 1 iJUICE

```
input : $x$: IOI, $t, k$: number of nearest CF observations to $x$ (in
        distance and permutations), $f$: classifier, $\vec{mut}$, $\vec{dir}$, $\vec{pla}$:
        features mutability, directionality and plausibility.
output: $CF_{iJUICE}, Justifiers$
1 $\mathcal{K} \leftarrow$ kNN($x, t, k$)
2 $\mathcal{N} \leftarrow$ getNodes($x, \mathcal{K}$)
3 $A \leftarrow$ getAdjacency($\mathcal{N}$)
4 $F \leftarrow$ getFeasibility($x, \mathcal{N}, \vec{mut}, \vec{dir}, \vec{pla}$)
5 $C \leftarrow$ getCost($x, \mathcal{N}$)
6 $CF_{iJUICE}, Justifiers \leftarrow$ solveIP($\mathcal{K}, \mathcal{N}, A, F, C$)
7 return $CF_{iJUICE}, Justifiers$
```

The mutability, directionality and plausibility vectors of the dataset are inputs to the iJUICE algorithm. iJUICE is model-agnostic, but not data-agnotic, i.e., it requires the user input of the properties of mutability of each feature, the directions on which they can change, and the values they can possibly take. This information was manually inserted for each dataset. The details on the properties of mutability and directionality are given in “Appendix B”. The property of plausibility is defined as the values that each feature can take, e.g., for continuous features, from the minimum value to the maximum value observed in the dataset, and for ordinal, the ordinal values that the feature can take. These affect how the graph is built, and therefore the solution.

In step 1, we sort all the training observations by the distance function used and select the set $\mathcal{K}$ of k closest instances with the least feature value permutations with respect to the IOI x (k chosen by the user). We calculate the permutations of the training observations with respect to the IOI x. This number is at most $2^{(b+c)} \cdot p^r \cdot \prod_{l=1}^{o} h_u$ for each instance, where b, c, r and o indicate the number of binary, one-hot encoded categorical, continuous and ordinal features present in the dataset, respectively; p is the number of steps in the discretized continuous features, and h_u is the number of ordinal values in the ordinal feature u. We select the nearest neighbors with the least amount of possible changes with respect to the IOI x, and make the generated graph compact and sparse, which improves the CF actionability and reduces the overall complexity. Figure 4a shows the IOI x and the $k = 10$ selected closest CF training observations to instance x with a green outline.

In step 2, the set $\mathcal{N}$ of nodes of the graph are generated. To do this, the gradient vector $g = x - n_v$ is calculated for every training observation $n_v \in \mathcal{K}$. For each changed feature between x and n_v, all the CF nodes (according to classifier f) corresponding to permutations of the features changed are obtained. Only CF points are considered, which guarantees the validity of the obtained CF. Figure 4b adds the full set of CF nodes in $\mathcal{N}$.

Ideally, for continuous features, iJUICE should check infinitesimal value changes, which is unattainable. As in the original JUICE algorithm, a discretization is performed to obtain a finite number of points. We create $p = 100$ steps in each continuous feature. In contrast to JUICE, which simply splits the difference between x and n_v in 100, iJUICE locates these p steps based on the feature distribution, so that the probability of finding an instance is the same, i.e, the probability of being between two adjacent values is 1% for all step values found. Therefore, if the feature distribution is more dense around a given point, then the space between adjacent values close to that point will be smaller, increasing the number of nodes in the search space as the feature becomes denser, which reduces the probability of unintentionally traversing a decision boundary. Moreover, p is modified to decrease the number of nodes when dealing with larger datasets. The worst case complexity for all the k instances is $\mathcal{O}(k(2^{(b+c)} \cdot p^r \cdot \prod_{l=1}^{o} h_u))$ which is the maximum amount of nodes in the graph (when all features are different between each node in $\mathcal{K}$ and x), defined as $|\mathcal{N}| = k(2^{(b+c)} \cdot p^r \cdot \prod_{l=1}^{o} h_u)$. The adjacency of the filtered CF points in space is defined by the adjacency matrix A.

In step 3, the adjacency matrix A is defined, where an edge exists among two nodes if only one feature is different in the smallest possible value change. Figure 4c shows the edges corresponding to one’s in the adjacency matrix. The definition of the adjacency matrix requires at most $\mathcal{O}(|\mathcal{N}| \cdot \log(|\mathcal{N}|) \cdot w)$ operations, where w is the total number of features.

In step 4, the simultaneous compliance of the properties of mutability, directionality, and plausibility is assessed for each node $n_i \in \mathcal{N}$ with respect to IOI x to calculate its feasibility F_i. F_i is a binary parameter stating the feasibility of node $n_i \in \mathcal{N}$ with respect to the selected node x_i. When non-compliant to any of the three properties, $F_i = 0$, forcing $x_i = 0$. Figure 4d shows the unfeasible instances that are marked as black. Note that a CF training observation in $\mathcal{K}$ may also be blacked out and that a connection may still exist through the unfeasible nodes but they cannot be selected as CFs.

In step 5, the cost associated to each node $C_i, n_i \in \mathcal{N}$ is calculated based on the distance function used. In step 6, we solve problem 2 using an IP formulation. The cost function to be optimized is described in Eq. 3 and is a weighted average between the distance C_i associated with selecting x_i, and the ratio $\frac{|T_k|}{k}$, given x_i. The λ coefficient lies in the [0, 1] range. When $\lambda = 1$, only the distance is relevant, and the constraint in inequality 7 forces the selected CF to have at least one justifier. When $\lambda < 1$ then the formulation will try to increase the number of justifiers s_v so that the second term in Eq. 3 increases and Z decreases. The impact of λ may be observed in the ablation study in Sect. 6.6.2. Given that these parameters are fixed, the solution will always be stable, meaning that for a unique instance the obtained CF will be the same for any iterations done.

$$\min Z = \lambda \sum_{n_i \in \mathcal{N} \setminus \mathcal{K}} C_i x_i - (1-\lambda)\frac{1}{k} \sum_{n_v \in \mathcal{K}} s_v \tag{3}$$

subject to:

$$\sum_{n_i \in N} y_{iv} A_{iv} - \sum_{n_i \in N} y_{vi} A_{vi} = -s_v, \ \forall n_v \in \mathcal{K} \tag{4}$$

$$\sum_{n_i \in N} y_{ij} A_{ij} - \sum_{n_i \in N} y_{ji} A_{ji} = x_j \sum_{n_v \in \mathcal{K}} s_v, \ \forall n_j \in \mathcal{N} \setminus \mathcal{K} \tag{5}$$

$$x_i \leq F_i, \ \forall n_i \in \mathcal{N} \setminus \mathcal{K} \tag{6}$$

$$\sum_{n_v \in \mathcal{K}} s_v \geq 1 \tag{7}$$

$$\sum_{n_i \in \mathcal{N}} x_i = 1 \tag{8}$$

$$x_i \in \{0, 1\} \ \forall n_i \in \mathcal{N} \setminus \mathcal{K} \tag{9}$$

$$s_v \in \{0, 1\} \ \forall n_v \in \mathcal{K} \tag{10}$$

$$y_{ij} \in \mathbb{Z} \ \forall n_i \in \mathcal{N}, n_j \in \mathcal{N} \tag{11}$$

In the IP, constraints 4 and 5 restrict the connections in the graph (they are the multi-source, single-sink flow constraints in a graph network). Constraint 4 is the outflow constraint for each possible justifying training observation: for each of these observations, the in-out balance should be $-s_v$, indicating that if the node $n_v \in \mathcal{K}$ is selected as a justifier ($s_v = 1$), then there must be an edge starting a connection path to the selected CF from n_v. Constraint 5 is the inflow constraint for all $n_j \in \mathcal{N} \setminus \mathcal{K}$. The right-hand side of this equation indicates that the number of edges or paths leading to the chosen CF (selected through x_j) must equal the sum of all the selected justifying instances. Note then that $|T_k| = \sum s_v$.

Constraint 6 forces the selected CF to be feasible. Constraints 7 and 8 enforce at least one justifying instance to the selected CF in the graph network. Constraints 9, 10 and 11 indicate the nature of the decision variables. Note that the variable y_{ij} is an integer, which allows a path in the graph to be traversed more than once.

If the problem is unfeasible for a given IOI x, then iJUICE defaults to the feasible nearest neighbor CF. Figure 4e and f show the optimal solutions when proximity is prioritized ($\lambda = 1$) and when justification is prioritized ($\lambda = 0$), respectively. The selected CF x' is shown as a yellow dot, and the yellow connection paths to different CF training observations are observed. Note that the yellow dot has six justifiers in Fig. 4f, instead of the four shown in Fig. 4e.

5.3 Complexity

An IP formulation is a $NP-complete$ problem (Lenstra, 1983; Kannan & Monma, 1978; Papadimitriou, 1981). It may be solved in pseudopolynomial time given that the number of constraints m is fixed (Papadimitriou, 1981) or that the number of variables n is fixed (Lenstra, 1983). More specifically, the current IP has a fixed $m = 2|\mathcal{N}| + 2$ and a fixed $n = |\mathcal{N}| + |\mathcal{N}|^2$. Most IP formulations are solved through a combination of branch-and-bound and cutting planes algorithms (Basu et al., 2022). Basu et al. (2022) have proved that for an IP with at most s sparsity (all constraints have at most s variables involved), the minimum size of a tree generated by the branch-and-bound algorithm is $2^{\lfloor \frac{n}{2s} \rfloor}$, $s = |\mathcal{N}|$ (constraint 5). Cohen et al. (2021) indicate that the fastest algorithm to solve a linear program, which is the relaxed formulation obtained at each node of the branch-and-bound tree, is expected to have a complexity of $\mathcal{O}(n^{2.5})$. Therefore, for the IP, the number of nodes in the branch-and-bound tree is at least: $2^{\lfloor \frac{|\mathcal{N}|+|\mathcal{N}|^2}{2|\mathcal{N}|} \rfloor} = 2^{\lfloor \frac{|\mathcal{N}|+1}{2} \rfloor}$ and the complexity of step 6 is $\mathcal{O}(2^{\lfloor \frac{|\mathcal{N}|+1}{2} \rfloor}(|\mathcal{N}| + |\mathcal{N}|^2)^{2.5})$. This is the worst-case complexity of the iJUICE algorithm, which is considerably higher than its average complexity due to two considerations:

1. $|\mathcal{N}| = k(2^{(b+c)} \cdot p^r \cdot \prod_{l=1}^{o} h_u)$ is the worst case scenario. Since the gradient vector is sparse $g = x - n_v$ for all $n_v \in \mathcal{K}$, because $\mathcal{K}$ may be selected based on the lowest permutations between x and n_v then the real number of nodes $|\mathcal{N}|$ usually complies with $|\mathcal{N}| \ll k(2^{(b+c)} \cdot p^r \cdot \prod_{l=1}^{o} h_u)$.
2. The solution of the IP is strongly influenced by the sparsity of the constraints (Basu et al., 2022) and the preprocessing stages of the solver package which reduces both constraints and variables based on the sparsity. Since the adjacency matrix A is highly sparse, the practical solution is attained within an acceptable time.

5.4 Verifying k-justification for any given CF

iJUICE guarantees justification and is able to output the justification ratio on the k closest CF training observations. JUICE is able to obtain a justified CF, however, this CF may be far from its justification instance because of the initialization process requiring a feasible instance, i.e., the CF may not be justified by nearby observations to it. For any given CF, we implement an algorithm, shown in Algorithm 2, to measure justification.

Algorithm 2 k-justification verification

```
input  : $x'$: any CF, $k$: number of nearest CF training observations to
         $x'$ (in distance and permutations), $f$: classifier
output: $CF_{iJUICE}, Justifiers$
1 $\mathcal{K} \leftarrow$ kNN$(x, k)$
2 $\mathcal{N}, A \leftarrow$ buildGraph$(x', \mathcal{K})$
3 $T_k \leftarrow \emptyset$
4 for $i \in \mathcal{K}$ do
5     if hasConnection$(\mathcal{N}, A, i, x')$ then
6         $T_k += i$
7 $JustificationRatio \leftarrow \frac{|T_k|}{k}$
8 return $JustificationRatio$
```

The first step is to find the set $\mathcal{K}$ of closest k CF training observations to x'. From this set of instances, in line 2 the graph towards the x' CF is built, obtaining all the permutation nodes among the $\mathcal{K}$ set and x'. Then the connections are verified for each of the nodes in $\mathcal{K}$ through the adjacency matrix A and finally, the justification ratio is calculated. This justification process is an improvement over the mixed-feature verification process previously proposed (Kuratomi et al., 2022) because it is able to consider k closest CF training observations instead of only the closest one. Theoretically, one may set $k = |q|$, in order to consider all the possible justifiers in the dataset, but this considerably increases the complexity. Therefore, we set $k = 10$, which is the same amount of closest CF training observations considered by the initialization of iJUICE. To relax the search for connection with other instances and also reduce the computational complexity, we use 10-bins discretization for the continuous features of the graph network.

6 Empirical evaluation

We compare iJUICE with 10 state-of-the-art counterfactual generation methods, using 14 different binary classification datasets. The datasets and methods are described in Sects. 6.1 and 6.3. The iJUICE implementation is available on GitHub.[1]

6.1 Datasets

We used 14 binary classification datasets to benchmark iJUICE against 10 competitor CF generation methods. The datasets have been selected for their mixed-feature space and their application diversity: three focus on income and professional development, three on financial and credit information, three on the medical area and disease diagnosis, three on student education assessment, one on recidivism, and one on athletes performance. All datasets have been preprocessed and stored in the GitHub repository. These

[1] https://github.com/alku7660/iJUICE.

datasets are preprocessed according to the guidelines proposed by Karimi et al. (2020) and Le Quy et al. (2022). The information on feature feasibility (mutability, directionality) is directly input into the iJUICE algorithm. Further details about the initial amount of instances, specific prediction task, and features of each dataset may be found in the "Appendix B".

6.2 Classifiers

In order to generate CFs for subsets of instances in each of the datasets studied, we first select a classification model and tune its hyperparameters. We perform a grid search on Random Forest (RF) and Multi-Layer Perceptron (MLP) models to obtain the trained classifiers f used on each dataset (based on the F1 score). The details on the grid search hyperparameters, the classification performance, and size of the test subsamples for CF generation are shown in "Appendix C.1".

6.3 Competitors

The selected 10 CF generation competitor algorithms tested have been previously described in Sect. 2, namely NN, MO, FT, RT, GS, FACE, DiCE, MACE, CCHVAE and JUICE. The AR method, which also uses an IP to search for CFs, is not implemented since it requires a linear classifier. For further details about their default configurations and general function, see "Appendix A".

6.4 Parameter setup

There are four parameters to be set in iJUICE:

1. λ: The weight between the distance function and the justification ratio in the IP cost function, Eq. 3. We set this equal to 1, in order to prioritize proximity (as other baselines do) and analyze the impact of λ on the performance of iJUICE (see Sect. 6.6.2).
2. **Distance function**: We use a set of distance measures specified in Sect. 6.5 to define C_i, and study their impact on the performance of iJUICE (see Sect. 6.6.2).
3. **Continuous features discretization**: By default, we use a split of 100 bins for each continuous feature for most of the datasets. The details on the number of continuous features bins used for each feature in each dataset are given in "Appendix B".
4. **Parameter** k: The amount of closest CF training observations to consider for the search of the iJUICE CF. As noted in Sect. 3, specifically in the example illustrated in Fig. 2, if the number k surpasses the number of counterfactual training outliers, and the iJUICE algorithm is set to maximize for the number of justifiers, then iJUICE will eventually output counterfactuals in regions where the most training observations are found, namely, region 3 in Fig. 2. According to this, one may define the parameter k based on the number of counterfactual training outliers, which is not a straightforward problem to tackle. We however define $k = 10$. Additional details on the selection of k may be found in Sect. 6.6.3.

6.5 Performance metrics

As mentioned in Sect. 1, we define in this study feasibility as an overarching property that covers three properties: mutability, directionality and plausibility, the latter satisfied whenever the values of the CF are in the observed or physically possible range of values. Although iJUICE does not optimize for sparsity directly, we measure its capabilities in this property. The performance measures used to evaluate the CFs generated are then listed below:

1. **Proximity**: Measured through the distance between the IOI x and each of the generated CF (the lower the better). We use distance functions that have been used in the literature to address the challenge of evaluating datasets with mixed-feature types:

   (a) **$l1$ & $l0$-norm**: Weighted average of the $l1$-norm for ordinal and continuous features and a simple matching for binary and categorical, based on Sharma et al. (2020):

   $$d(x,x') = \frac{o+r}{w} l1(x,x') + \frac{b+c}{w} match(x,x') \tag{12}$$

   (b) **$l1$, $l0$ & $l\infty$-norm**: Weighted average of the $l1$-norm for ordinal and continuous features, a simple matching for binary and categorical and the $l\infty$-norm, based on Karimi et al. (2020):

   $$d(x,x') = \alpha match(x,x') + \beta l1(x,x') + \gamma l\infty, \tag{13}$$

   We have defined $\alpha = 0.25$, $\beta = 0.25$, and $\gamma = \frac{1}{(\alpha+\beta)w}$, in accordance with the description by Karimi et al., where w is the number of processed features.

   (c) **Max. Percentile Shift**: The maximum percentile change between any of the features j in the dataset, as measured in the training observation distribution, based on Ustun et al. (2019):

   $$d(x,x') = \max_{j \in w} |Q(x_j) - Q(x'_j)| \tag{14}$$

   We take an interest in three distance measures designed for datasets with a mixed-feature space (Sharma et al., 2020; Karimi et al., 2020). The Max. Percentile Shift is able to calculate the distance based on the observed feature distribution regardless of their potentially different scales or nature (Ustun et al., 2019). We also analyze three different commonly used metrics in the literature, namely $l2$-norm, $l1$-norm, and $l\infty$ -norm and their results may be observed in the "Appendix C.2".

2. **Sparsity**: The number of features that are different between the instance of interest and its corresponding CF:

   $$S_{CF} = |\{i | x_i \neq CF_i, i \in \{(1,2,\dots,w)\}\}| \tag{15}$$

3. **Feasibility**: The product of mutability, directionality, and plausibility:

   $$F_{CF} = (mutable_{CF} = 1) \land (direction_{CF} = 1) \land (plausible_{CF} = 1), \tag{16}$$

   $F_{CF} = 1$ if mutability, directionality, and plausibility of the CF are satisfied (as discussed in Kuratomi et al. (2022)).

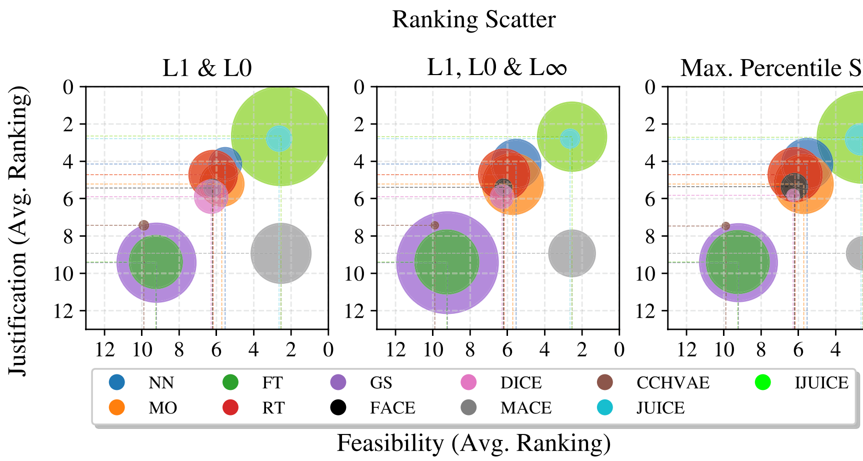


Fig. 5 Each subplot shows the scatter of the methods with respect to a specified distance function. The x-axis represents the feasibility ranking. The y-axis represents the justification ranking. The dashed lines indicate the exact ranking of the center of the dots in both axes. Since the axes are inverted, the higher up a method is, the better the justification ranking, i.e., a relatively higher justification with respect to the other methods. The further to the right a method is, the better the feasibility ranking, i.e., a relatively higher feasibility with respect to the other methods. The size of the circles represents the performance in proximity (or CF distance). The bigger the circle, the better the performance of the method in terms of proximity to the CF. The best method is a big circle located in the upper right corner, whilst the worst method is a small circle located in the lower left corner

4. **Justification ratio**: The ratio of justifier instances in the set $\mathcal{K}$ of possible justifiers:

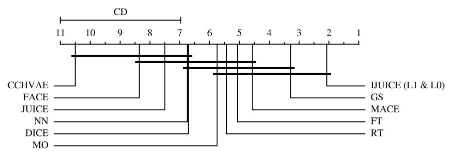
$$J_{CF} = \frac{1}{k} \sum_{n_v \in \mathcal{K}} s_v, \tag{17}$$

6.6 Results

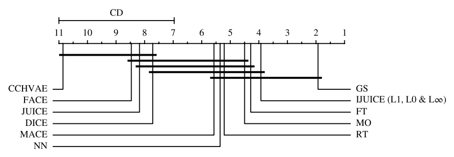
The IP is solved by using an academic license of the Gurobi optimizer package for Python. We compare the performance of iJUICE CFs with respect to CFs generated by other state-of-the-art generation algorithms, including JUICE. In addition, we present the results of the ablation study on the λ weight parameter with regard to the performance of iJUICE in proximity and justification ratio.

6.6.1 CF performance

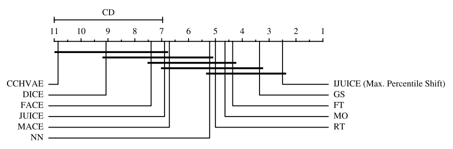
The scatter plot shown in Fig. 5 compares the performance of the CF generation algorithms in terms of average ranking in the measures of feasibility, justification (in the x and y axes), and proximity (through the area of the circles). It is important to note that each of the CF generation algorithms was run in its default configuration. Most of the baseline methods use the $l2$-norm as a default. These configurations are explained more in detail in "Appendix A". Since iJUICE optimizes for any of the mixed-feature spaces distance measures



(a) $l1$ & $l0$-norm distance CD.

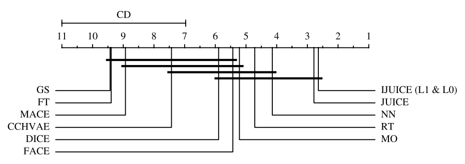


(b) $l1$, $l0$ & $l\infty$-norm distance CD.

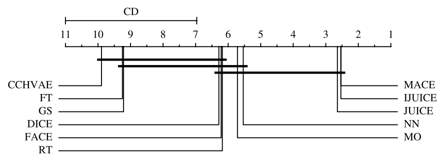


(c) Max. Perc. Shift distance CD.

Fig. 6 Proximity ranking CD diagrams for all CF generation algorithms for $l1$ & $l0$; $l1$, $l0$ & $l\infty$ and Max. Percentile Shift. iJUICE optimizes the corresponding cost function



(a) $l1$ & $l0$-norm justification CD.



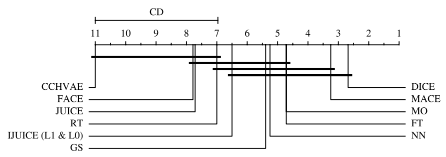
(b) Feasibility CD.

Fig. 7 Justification and feasibility ranking CD diagrams for all CF generation algorithms for $l1$ & $l0$. iJUICE optimizes the corresponding cost function. Feasibility remains equal for all distance measures

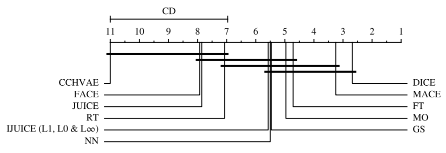
discussed in Sect. 6.5, we use the corresponding distance measure as the cost function for iJUICE in each subplot.

According to Fig. 5, iJUICE performs best in terms of proximity when compared to the other baseline methods (it is the largest circle). Only GS outperforms it in the $l1$, $l0$ & $l\infty$ norm. We also note that JUICE and iJUICE are close in ranking in feasibility and justification, with iJUICE being superior in both and considerably better in terms of proximity. MACE matches JUICE in terms of feasibility due to its feasibility constraints on the CF. JUICE is slightly inferior in feasibility to iJUICE because of its initialization: JUICE selects the closest feasible CF training observation as the starting point. However, if there are no feasible CF training observations, then JUICE defaults to NN. In this case, the CF output, although justified, will be unfeasible. There was an instance in the Athlete dataset which did not have a feasible CF training observation, so an unfeasible but justified observation was selected as CF.

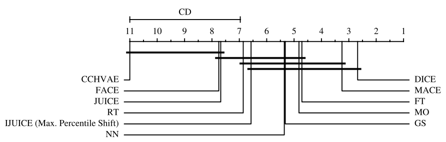
In contrast, iJUICE initializes on the k closest CF training observations without requiring them to be feasible with respect to the IOI x. The selected iJUICE CF is the closest instance to the IOI x that belongs to the graph formed by these k closest CF training observations and their permutations found when navigating in the feature space towards the IOI. If there is no feasible node in the graph, then the problem is unfeasible, and iJUICE defaults to the NN (as in JUICE). However, the risk of finding no feasible node in the



(a) $l1$ & $l0$-norm sparsity CD.



(b) $l1$, $l0$ & $l\infty$-norm sparsity CD.



(c) Max. Perc. Shift sparsity CD.

Fig. 8 Sparsity ranking CD diagrams for all CF generation algorithms for $l1$ & $l0$, $l1$, $l0$ & $l\infty$ norms and Max. Perc. Shift

Table 2 The iJUICE CF changed the marital status, occupation and education level and number of years. Both justifier examples have the same education level and number of education years as the CF

Feat.	IOI x	iJUICE x'	Justifier 1	Justifier 2
Sex	Male	Male	Male	Male
Country	USA	USA	USA	USA
Race	White	White	White	White
Workclass	Private	Private	Local-gov	Private
Marital Status	Single	Married	Single	Married
Occupation	Sales	Prof-specialty	Protection	Adm-cleric
Relationship	Own-child	Own-child	Own-child	Husband
Education Level	HS-grad	Some-college	Some-college	Some-college
Age Group	<25	<25	<25	<25
Education Number	9	10	10	10
Capital Gain	0	0	0	0
Capital Loss	0	0	0	0
Work Hr/week	40	40	40	40

iJUICE graph is lower than in JUICE, since the initialization considers more than one close CF training observation.

In the Nemenyi (Critical Difference - CD) tests in Fig. 6, the statistically different methods are observed for all distance measures applied. iJUICE, GS, NN, MO, FT, and RT are the best and statistically better than JUICE and MACE in the Max. Percentile Shift distance measure. However, the GS method has the lowest combined performance, together with FT, on both feasibility and justification (the lower-left corner circles in Fig. 5). The worst performance in feasibility was attained by the CCHVAE method, which also was the worst in proximity to the IOI (it is the smallest point) due to it not minimizing the distance towards the IOI, but the CF likelihood. Finally, the GS algorithm has the best proximity

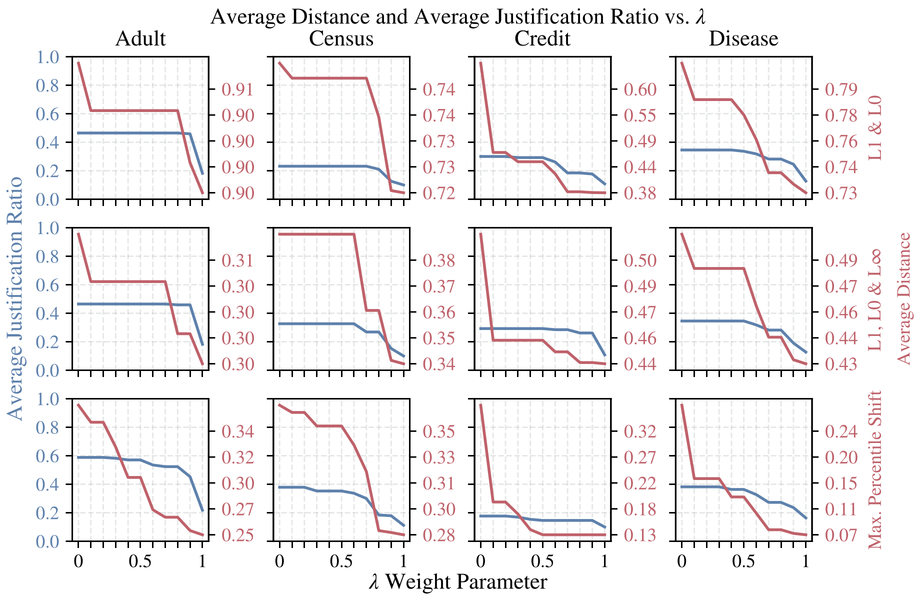


Fig. 9 Impact of λ on iJUICE average proximity and justification ratio performance for 4 datasets

performance in the $l2$, $l1$, and $l\infty$ norms, as shown in the "Appendix C.2". We highlight that the GS algorithm assumes continuous feature spaces and therefore ignores feasible, mixed-feature values.

Moreover, Fig. 7 indicates which methods are statistically different in terms of justification and feasibility. We only present the justification CD diagram corresponding to the $l1$ & $l0$ norm, which is the best ranking for iJUICE, and, since feasibility is guaranteed no matter the distance function used, we simply present the performance of iJUICE for any distance function. Note that there is no significant variation in justification among the different distance functions (All the justification CD diagrams are shown in "Appendix C.3").

In terms of time, iJUICE is on par with respect to methods that also present an optimization framework, like MACE, or to methods optimizing for faithfulness, such as CCH-VAE and JUICE. The GS algorithm is statistically superior to iJUICE in the $l2$, $l1$, and $l\infty$ norms, while statistically not different in the rest. iJUICE is statistically superior to GS in feasibility and justification. Further details on the results are shown in "Appendix C".

In terms of sparsity, Fig. 8 shows the performance of iJUICE when optimizing for distance. iJUICE performs not significantly different from the algorithms tested in these datasets, but not in the best ranking. iJUICE is not the best in sparsity performance due to the distance function used. The distance functions are different from the usual sparsity-associated function, which is the $l0$-norm. However, the cost function could be changed to include either a term on sparsity, or the distance function changed to a pure $l0$-norm which could be calculated as a parameter for the nodes in the graph, in order to improve the sparsity of the CF output. DiCE is the best in sparsity, since it outputs a set of CFs that are sparse in the set of features changed and different among themselves (Mothilal et al., 2020).

Finally, Table 2, shows an example for an IOI x from the Adult dataset, its iJUICE CF through the Max. Percentile Shift distance measure and its justifiers. In this example, the

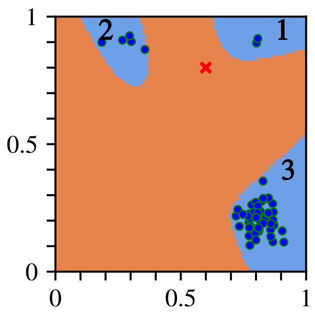


Fig. 10 Point X is surrounded by 3 regions of CF points, created by a classifier. The blue regions 1, 2, and 3 present 2, 5, and 50 justifiers respectively. A CF found in region 3 has a better support according to the data distribution and could lead to higher explanation trustworthiness and credibility from users (Color figure online)

male must increase his education and get college experience and change his occupation to achieve a high-income label.

6.6.2 Ablation study on the λ parameter

We repeat the experiments varying the weight λ in the 0 to 1 range with 0.1 increments for four datasets, namely Adult, Census, Credit, and Disease. The Adult dataset focuses on income prediction and has all types of features. Census has the same task but does not have ordinal features. The Credit dataset focuses on default risk prediction by credit card holders and does not have categorical features. The Disease dataset is a synthetic dataset with all feature types. For each λ we evaluate three distance measures. The results are shown in Fig. 9.

We observe in general that the higher the value of λ the lower the distance (the better the proximity) from the IOI x to the iJUICE CF and the lower the justification ratio. This is in accordance with the cost function of the IP in Eq. 3, where a higher λ parameter indicates a higher weight for the distance function and a lower weight for the justification ratio. In particular, having a $\lambda = 0$ is suboptimal in all cases. To minimize distance while maximizing justification ratio, e.g. for the Adult dataset on the measures of $l1$ & $l0$ and $l1$, $l0$ & $l\infty$, optimal values are $\lambda = 0.8$ and $\lambda = 0.7$, respectively, since these provide the highest justification ratio (above 40%) with a lower distance. This behavior is observed through the Census, Credit, and Disease datasets with other higher-than-zero λ values. Additionally, if proximity is the main priority ($\lambda = 1$) will output the closest CF whilst maintaining at least one justifier instance, which is closer to the IOI x than the JUICE CF.

Finally, the Max. Percentile Shift measure is able to provide the highest justification ratio for the Adult, Census, and Disease datasets, being considerably higher (around 15% higher) than in the $l1$ & $l0$ and $l1$, $l0$ & $l\infty$ measures in the Adult dataset. This behavior is however reversed in the Credit dataset, where Max. Percentile Shift presents the lowest justification ratio. Finally, note that the Max. Percentile Shift is the only measure limited to the [0, 1] range, as does the justification ratio.

6.6.3 Ablation study on the k parameter

Given that real datasets rarely present outlier identifiers (Eiras-Franco et al., 2019), and that high-dimensionality, heterogeneous spaces present an inherent visualization challenge, it may be possible to implement an outlier detection algorithm in order to find the number of outliers in every dataset. An example of this process was done in Sect. 4, when we

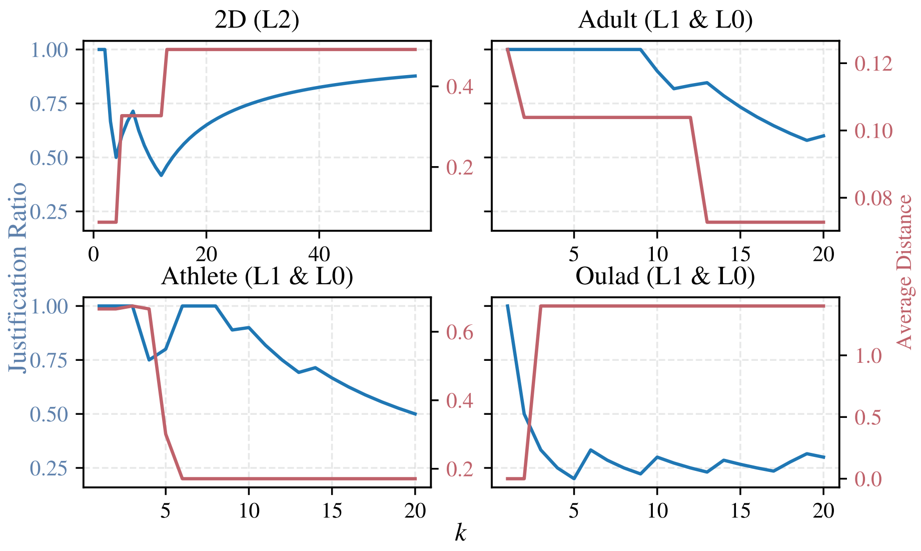


Fig. 11 The red line is the distance from the CF to the IOI. The blue line is the justification ratio of that CF. The steps on the distance function, and the changes on the justification ratio are useful to determine where a large portion of justifiers may be located (Color figure online)

estimated that the set of desired class training instances with the lowest 5% log-likelihood was assumed to be a set of outlier instances.

However, although practical, this method may output an incorrect number of outliers (there is no ground truth to identify outliers). Additionally, there is a large set of possible algorithms for outlier detection (Breunig et al., 2000; Otey et al., 2006; Eiras-Franco et al., 2019) requiring each a defined set of parameters, i.e., we would be transferring the problem of the definition of the parameter k to the definition of the parameters associated to the outlier detection algorithm. An example of this is the LOF algorithm (Breunig et al., 2000), which requires the specification of a distance function and a number of neighbors that is adequate to determine the outliers.

Consequently, we performed instead, a set of experiments to evidence the impact of the parameter k, and provide a reasoning process to select its value. We demonstrate that this parameter is to be defined by the users for the dataset and classifier of interest, according to their justification needs to support the obtained CF. The experiments are performed on a toy dataset emulating the scenario illustrated in Fig. 2, and 3 additional datasets used in the paper (Adult, Athlete, and Oulad). We ran iJUICE iteratively with increasing values for k for a given instance $\mathtt{X}$. We record the proximity of the CF and the justification ratio obtained. For the synthetic 2-dimensional dataset we used a MLP with layers [100, 200, 500, 200, 100] and hyperbolic tangent activations (see Fig. 10). For the publicly available datasets, we used the classifiers referred in Sect. 6.2. We prioritize the justification ratio on both experiments, with $\lambda = 0.01$, meaning 0.99 weight on the justification ratio. The justifier ratio and the average distance are plotted. The results for each dataset are illustrated in Fig. 11.

In these figures, the red line is the distance from the CF to the IOI whilst the blue line is the CF justification ratio. We will first focus on the 2-dimensional dataset (2D) plot (top left corner named). In the range $k \in [1, 2]$, the justifier ratio remains at 100%, since there are 2 justifiers in region 1 (see Fig. 2), the closest region. In the range $k \in [3, 4]$, the justification ratio decreases since having less than 5 potential justifiers will still provide a CF in region 1, while the distance slightly changes. At $k = 5$, the justification ratio increases because at that moment, 2 of the justifiers are in region 1, and 3 are in region 2. Since we are maximizing for justification ratio, the CF found moves from region 1 to region 2 (increasing the average distance), having 3 out of 5 potential justifiers (60%). In the range $k \in [6, 7]$, the justification ratio keeps increasing because there are up to 5 justifiers in the region 2. As k increases in this range, more justifiers are found in region 2, making the distance remain steady until all justifiers found are connected to the CF at $k = 7$, making it 5 out of the 7 potential justifiers (71.4%). In the range $k \in [8, 12]$, the justification ratio decreases again, because there are only 5 justifiers in region 2. That means that the potential justifiers that are from the eighth to the twelfth closest justifiers are located in region 3, but these are not enough to provide higher justification than the five already found in region 2. In the range $k \geq 13$, the CF is finally chosen in region 3. The justification ratio then increases until it reaches 87.7% when it finds 50 justifiers out of 57 (since 7 are in the regions 1 and 2 and 50 are in region 3), leading to a halt in justification ratio increase. Note also that the distance has stabilized from $k = 13$ on, which is expected since the largest, justified counterfactual region is region 3 and the CF found should remain there.

The 2D dataset (see top left plot of Fig. 11) presents an illustrative scenario of the behavior of iJUICE with respect to k, but it is rather simple, since it only has continuous features without any feasibility constraint. Additionally, all the CF training instances in region 1 are closer than all the CF training instances of region 2, and the CF training instances of region 2 are closer than all the CF training instances of region 3. This creates the upward ladder behavior in the distance and the see-saw behavior in the justification ratio. This is also observed in the Oulad dataset (bottom right plot in Fig. 11). However, the iJUICE performance behavior on other real datasets and classifiers is more complex.

In the Adult dataset (top right plot in Fig. 11), in the range $k \in [1, 8]$, specifically from the first to the second value of k, the distance decreases, and the justification ratio remains at 100%. This indicates that the second closest CF observation expands the graph inside the first CF region, bringing in closer points in space to the instance of interest. Additionally, it also shows that this region is relatively large, where the first 8 CF training observations are found. An interesting behavior occurs in the region $k \in [11, 13]$ where the justification ratio increases and the distance decreases. This must occur because the algorithm has found another region where the actual CF training instances were located further than the first 10, but the region has its decision boundary even closer to the instance of interest than the first region does, therefore outputting a closer CF.

In the Athlete dataset (bottom left plot in Fig. 11), the slight change in distance from $k = 2$ to $k = 3$ is due to the sensitivity of the optimizer caused by the small weight $\lambda = 0.01$ of the distance component. In the range $k \in [4, 6]$, the justification ratio increases from 75% to 100% as the distance decreases. This means that the graph that was initially split in two regions, was connected by adding the fourth, fifth and sixth closest training CF instances.

Note that the experiments shown in Fig. 11 are for single, randomly picked instances in the undesired predicted label. These indicate that different values of justification ratio may be achieved with different k, depending on the dataset and classifier trained. In order to avoid having to experiment on every instance on every dataset to select a given k and aggregate the results, we define $k = 10$ for the sake of compactness and simplicity in the empirical evaluation, and allow the users to modify this value as they see fit. In reality where the interest is to explain an individual instance, the designer may do a study on that specific individual to select the most appropriate k value.

7 Discussion

Integrating the property of justification into the generation process of counterfactuals could provide improved trustworthiness to the obtained CFs. However, we are aware that the integration of this property by other established CF generation methods might prove difficult since this property requires finding the connecting paths from training observations towards the selected CFs, which is demanding in heterogeneous spaces. One option is to adapt the iJUICE method to fit as a member of an ensemble of counterfactual generators, as proposed by Guidotti and Ruggieri (2021). Another option is to integrate other methods' generation priorities into the iJUICE framework by using different cost functions considering other measures. For example, a diversity measure that could enhance and output diverse and justified counterfactuals may be of use for several applications, such as in the medical field, where it would be interesting for a doctor or patient to have diverse possible treatments as options, or a likelihood parameter that could be used to obtain highly likely CFs according to the data distribution (which are similar objectives to the DiCE and CCH-VAE methods).

Additionally, it is important to highlight that the verification of the justification property in heterogeneous spaces can be done for any given CF from another generation method with a number of close training observations, making it an evaluation measure for the CF. In order to do this, we developed the verification method, which uses a graph of nodes from the given CF to the closest k CF training observations. Using knowledge of the nature of the features (binary, categorical, ordinal, or continuous), the method is able to assess whether the CF is justified by any of the closest CF training observations and the justification ratio from these.

8 Conclusion

We have hereby developed an IP formulation, called iJUICE, to solve the k-justified counterfactual generation problem, which brings a stronger measure of faithfulness than the single instance justification, which is important for the trustworthiness and actionability of the derived explanations. Additionally, iJUICE performs well also in the measures of feasibility and proximity in heterogeneous spaces. We have empirically demonstrated that iJUICE attains closer, feasible, and k-justified instances under similar run times compared to competitor algorithms while guaranteeing that these instances are also valid CFs and are not changing with iterations of the algorithm, i.e., they are stable. iJUICE may be configured for a set of three different distance functions and convex weights of justification importance. Finally, we have analyzed the behavior of iJUICE in this regard by performing an ablation study on the λ weight parameter, showing the balance between proximity and justification. Even at the highest proximity relevance, iJUICE is able to guarantee at least one justifier for the output CF and may attain a higher justification ratio if desired, where a weight $\lambda > 0$ is always better with respect to proximity performance.

The consideration of other counterfactual properties, such as discriminative power and counterfactual fairness, is also of interest. Additionally, one may further develop the IP into a Mixed-Integer Linear Programming (MILP) formulation, where both continuous features may be considered directly without the need to discretize them, as well as their correlations so that more actionable and feasible changes may be proposed. Furthermore, a natural extension is the consideration of multi-class datasets and several other distance measures to be prioritized inside the objective function.

Appendix A: Methods

This section provides additional details on the configuration of the baseline methods used. These baseline algorithms are adapted and found in the GitHub repository of this paper.

A.1 NN

This method uses the $l2$-norm by default to sort the training observations with respect to the IOI x. It selects the closest training observation x' for which $l(x') \neq f(x)$, where $l(x')$ is the ground truth label of x', and for which $x \neq x'$ ($x = x'$ may occur since duplicates are not removed during the preprocessing).

A.2 MO

Based on Wexler et al. (2019), this method uses the $l2$-norm by default to sort all observations (including testing observations with their corresponding prediction labels) with respect to the IOI x. It picks the closest observation x' for which $f(x') \neq f(x)$ (if x' is a test instance) or for which $l(x') \neq f(x)$ (if x' is a training observation) and for which $x \neq x'$ ($x = x'$ may occur since duplicates are not removed during the preprocessing).

A.3 FT

Based on Tolomei et al. (2017), this method uses a random forest to derive CFs through instance perturbations that may divert the IOI x into a tree path leading to the desired CF label. The perturbation magnitude ϵ at each node in the path is $\epsilon = 0.01$, and the $l2$-norm is used as the default distance measure to assess the cost of attaining each CF. The original source of the implemented algorithm may be observed in this repository.[2]

A.4 RT

Based on Lindgren et al. (2019), this methods uses a random forest to find the CF from the training observations set that co-occurs the most in the different leaves of the random forest trees, prioritizing the frequency of presence-at-leaf in the leaf where the IOI x falls into. The original source of the implemented algorithm may be observed in this repository.[3]

[2] https://github.com/upura/featureTweakPy.

[3] https://github.com/tony-lind/Example-based-tweaking.

A.5 GS

Based on Laugel et al. (2017), this method samples points in the space enclosed between two concentric spheres which have the IOI x at the center. The algorithm stops whenever a CF is sampled (a decision boundary is reached). The method sorts the instances by default using the $l2$-norm and returns the closest CF. The original source of the implemented algorithm may be observed in this repository.[4]

A.6 FACE

Based on Poyiadzi et al. (2020), the method builds a graph based on feasible changes from the IOI x towards the decision boundary. It then prioritizes feasible paths that have a high density in order to maximize the potential actionability of the changes from the IOI towards the CF in the decision boundary. The method uses Dijkstras algorithm to find the shortest path in the graph which is feasible. By default, the method uses the $l2$-norm distance function to assess the distance of the CFs obtained. The algorithm implemented can be found in the CARLA Framework by Pawelczyk et al. (2021).

A.7 DICE

Based on Mothilal et al. (2020), the method is able to obtain a diverse set of CFs, i.e., a set of CFs which portray different changed features to alter the predicted label of the IOI x. The method maximizes tis diversity. In this case, the amount of requested CFs inside the set is one. The original authors implementation can be found and installed via pip in this repository.[5]

A.8 MACE

Based on Karimi et al. (2020), the method uses an optimization model with atomic formulae based on SMT. The SMT allows for feasible CFs which also minimize distance to the IOI x. The method is theoretically capable of optimizing for different distance functions. However, due to issues regarding execution using $l2$-norm, the method is implemented using the $l0$-norm. The original code may be found in this repository.[6]

A.9 CCHVAE

Based on Pawelczyk et al. (2020), the method uses a variational autoencoder to create a low-dimensional latent space where highly likely CFs are attained. The variational autoencoder is, by default, trained with 10 epochs and a batch size of 32. We have implemented 3 hidden layers following a 10-5-10 configuration for the datasets. Once a set of CFs is obtained, they are sorted based on $l1$-norm by default. The original authors'

[4] https://github.com/thibaultlaugel/growingspheres.

[5] https://github.com/interpretml/DiCE.

[6] https://github.com/amirhk/mace.

implementation may be found in this repository,[7] while a more user-friendly implementation is found in the CARLA Framework repository Pawelczyk et al. (2021).

A.10 JUICE

Based on Kuratomi et al. (2022), the method uses an ordered path-searching algorithm to find feasible and connected CF to a single training observation, guaranteeing justification in mixed-feature spaces. The original implementation may be found in this repository.[8]

Appendix B: Datasets

The prediction task for each dataset is described below. If not specified, the dataset may be found at the UCI Machine Learning repository[9]:

1. **Adult**: Income prediction.
2. **Athlete** (synthetic dataset): Olympic medalist prediction.
3. **Bank**: Bank client deposit prediction.
4. **Census**: Income prediction.
5. **Compas**: Recidivism prediction.[10]
6. **Credit**: A default risk prediction dataset.
7. **Diabetes**: Diabetes recovery / readmittance prediction.
8. **Disease** (synthetic dataset): Disease diagnosis prediction.
9. **Dutch**: Person's occupation (high or low-level) prediction.[11]
10. **German**: Credit risk assessment.
11. **Heart**: Heart disease diagnosis.
12. **Law**: Bar exam fail / pass prediction.[12]
13. **Oulad**: Exam fail / pass prediction.[13]
14. **Student**: High / low grade prediction.

The 14 binary classification datasets are further detailed in Tables 3 and 4.

[7] https://github.com/MartinPawel/c-chvae.
[8] https://github.com/alku7660/iJUICE.
[9] https://archive.ics.uci.edu/ml/index.php.
[10] https://www.propublica.org/datastore/dataset/compas-recidivismrisk-score-data-and-analysis.
[11] https://github.com/tailequy/fairness_dataset/tree/main/Dutch_census.
[12] https://github.com/tailequy/fairness_dataset/tree/main/Law_school.
[13] https://analyse.kmi.open.ac.uk/open_dataset.

Table 3 Mutability and directionality for each feature. ✓means the feature is mutable (may be modified), while ✗means the feature is immutable. ⇕ means the feature may increase or decrease in value, while ⇑ means the feature may rise only

Dataset	Binary / Categorical	Ordinal / Continuous	Cont. Feat. Split
Adult 488842 ins. 14 feat	Sex (✗,-) NativeCountry (✗,-) Race (✗,-) WorkClass (✓,⇕) MaritalStatus (✓,⇕) Occupation (✓,⇕) Relationship (✓,⇕)	EducationLevel (✓,⇑) AgeGroup (✗,-) EducationNumber (✓,⇑) CapitalGain (✓,⇕) CapitalLoss (✓,⇕) HoursPerWeek (✓,⇕)	100
Athlete 1000 ins. 6 feat	Sex (✗,-) Diet (✓,⇕) Sport (✓,⇕) TrainingTime (✓,⇕)	Age (✗,-) SleepHours (✓,⇕)	100
Bank 45211 ins. 17 feat	Default (✓,⇕) Housing (✓,⇕) Loan (✓,⇕) Job (✓,⇕) MaritalStatus (✗,-) Education (✓,⇕) Contact (✓,⇕) Poutcome (✓,⇕)	AgeGroup (✗,-) Balance (✓,⇕) Day (✓,⇕) Duration (✓,⇕) Campaign (✓,⇕) Pdays (✓,⇕) Previous (✓,⇕)	100
Census 299285 ins. 41 feat	Sex (✗,-) Race (✗,-) Industry (✓,⇕) Occupation (✓,⇕)	Age (✓,⇑) WageHour (✓,⇕) CapitalGain (✓,⇕) CapitalLoss (✓,⇕) Dividends (✓,⇕) WorkWeeksYear (✓,⇕)	100
Compas 7214 ins. 52 feat	Sex (✗,-) ChargeDegree (✓,⇕) Race (✗,-)	PriorsCount (✓,⇕) AgeGroup (✓,⇑)	100
Credit 30000 ins. 24 feat	isMale (✗,-) isMarried (✗,-) HistoryOverduePaym. (✓,⇕)	TotalOverdueCounts (✓,⇕) TotalMonthsOverdue (✓,⇕) AgeGroup (✓,⇑) EducationLevel (✓,⇑) BillAmountOVL6M[a] (✓,⇕) PaymentAmountOVL6M (✓,⇕) MonthZeroBalanceOVL6M (✓,⇕) MonthLowSpendOVL6M (✓,⇕) MonthHighSpendOVL6M (✓,⇕) RecentBillAmount (✓,⇕) RecentPaymentAmount (✓,⇕)	5
Diabetes 101766 ins. 50 feat	DiabetesMed (✓,⇕) Sex (✗,-) Race (✗,-) A1CResult (✓,⇕) Metformin (✓,⇕) Chlorpropamide (✓,⇕) Glipizide (✓,⇕) Rosiglitazone (✓,⇕) Acarbose (✓,⇕) Miglitol (✓,⇕)	AgeGroup (✓,⇑) TimeInHospital(✓,⇕) NumProcedures (✓,⇕) NumMedications (✓,⇕) NumEmergency (✓,⇕)	100

[a] OVLM: Over Last 6 Months

Table 4 (Continuation) Mutability and directionality for each feature

Dataset	Binary / Categorical	Ordinal / Continuous	Cont. Feat. Split
Disease 1000 ins. 7 feat	Smokes (✓,⇕) Diet (✓,⇕) Stress (✓,⇕)	Weight (✓,⇕) Age (✓,⇑) ExerciseMinutes (✓,⇕) SleepHours (✓,⇕)	100
Dutch 60420 ins. 12 feat	Sex (✗,-) HouseholdPosition (✓,⇕) HouseholdSize (✓,⇕) Country (✗,-) EconomicStatus (✓,⇕) CurEcoActivity (✓,⇕) MaritalStatus (✓,⇕)	EducationLevel (✓,⇑) Age (✓,⇑)	100
German 1000 ins. 20 feat	Sex (✗,-) Single (✓,⇕) Unemployed (✓,⇕) PurposeOfLoan (✓,⇕) InstallmentRate (✓,⇕) Housing (✓,⇕)	Age (✓,⇑) Credit (✓,⇕) LoanDuration (✓,⇕)	5
Heart 303 ins. 7 feat	Sex (✗,-) BloodSugar (✓,⇕) ChestPain (✗,-)	ECG (✗,-) Age (✓,⇑) RestBloodPressure (✓,⇕) Chol (✓,⇕)	100
Law 20798 ins. 12 feat	Sex (✗,-) WorkFullTime (✓,⇕) Race (✗,-) FamilyIncome (✓,⇕) Tier (✓,⇕)	Decile1stYear (✓,⇕) Decile3rdYear (✓,⇕) LSAT(✓,⇕) UndergradGPA (✓,⇕) FirstYearGPA (✓,⇕) CumulativeGPA (✓,⇕)	100
Oulad 32593 ins. 12 feat	Sex (✗,-) Disability (✗,-) Region (✓,⇕) CodeModule (✓,⇕) CodePresentation (✓,⇕) HighestEducation (✓,⇕) IMDBand (✓,⇕)	AgeGroup (✓,⇑) NumPrevAttempts (✓,⇕) StudiedCredits(✓,⇕)	100
Student 395 ins. 33 feat	Sex (✗,-) School (✓,⇕) AgeGroup (✗,-) FamilySize (✓,⇕) ParentStatus (✓,⇕) SchoolSupport (✓,⇕) FamilySupport (✓,⇕) ExtraPaid (✓,⇕) ExtraActivities (✓,⇕) Nursery (✓,⇕) HigherEdu (✓,⇕) Internet (✓,⇕) Romantic (✓,⇕) MotherJob (✓,⇕) FatherJob (✓,⇕) SchoolReason (✓,⇕)	MotherEducation (✓,⇑) FatherEducation (✓,⇑) TravelTime(✓,⇕) ClassFailures (✓,⇕) GoOut (✓,⇕)	100

Appendix C: Results

C.1 Classifier models and test samples

Two classifier models are implemented, namely Random Forest (RF) and Multi-Layer Perceptron (MLP). The best model and configuration for each dataset is selected through a grid search. Table 5 shows the selected models for each dataset with their corresponding test F1 score and test instances used for CF generation.

C.2 $l2$, $l1$ and $l\infty$ norms distance additional results

When running iJUICE with $l2$, $l1$ and $l\infty$ norms as objective, we obtain the results observed in Fig. 12. The distance ranking is observed in Fig. 13. In this case, iJUICE is able to

Table 5 Selected classifiers, test performance, and test instances utilized

Dataset	Best model	Hyperparameters	F1 score	Instances
Adult	RF	$depth_{max} = 10, leaf_{minsize} = 1, split_{minsize} = 1, n = 100$	0.83	17
Athlete	MLP	$act. = tanh, layers = (20, 50, 10), solver = sgd$	0.71	18
Bank	RF	$depth_{max} = 10, leaf_{minsize} = 1, split_{minsize} = 2, n = 200$	0.86	19
Census	RF	$depth_{max} = 10, leaf_{minsize} = 5, split_{minsize} = 10, n = 100$	0.87	19
Compas	MLP	$act. = tanh, layers = (20, 10, 10), solver = adam$	0.68	34
Credit	MLP	$act. = tanh, layers = (50, 1), solver = adam$	0.72	32
Diabetes	MLP	$act. = logistic, layers = (10, 20), solver = sgd$	0.61	24
Disease	MLP	$act. = ReLU, layers = (20, 20, 10), solver = adam$	0.78	37
Dutch	RF	$depth_{max} = 10, leaf_{minsize} = 3, split_{minsize} = 5, n = 50$	0.84	18
German	MLP	$act. = ReLU, layers = (100, 10), solver = sgd$	0.70	43
Heart	RF	$depth_{max} = 5, leaf_{minsize} = 5, split_{minsize} = 5, n = 50$	0.83	17
Law	MLP	$act. = tanh, layers = (50, 1), solver = adam$	0.82	20
Oulad	MLP	$act. = logistic, layers = (100, 10), solver = sgd$	0.67	32
Student	RF	$depth_{max} = 2, leaf_{minsize} = 5, split_{minsize} = 2, n = 200$	0.70	18

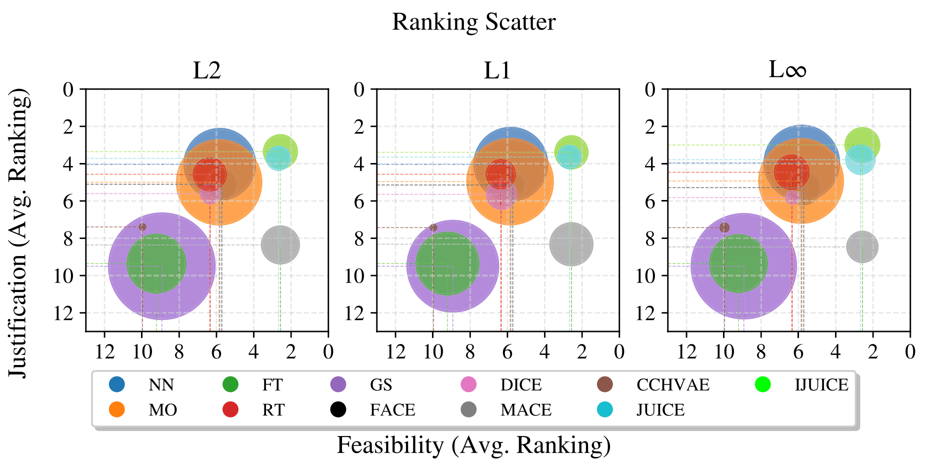
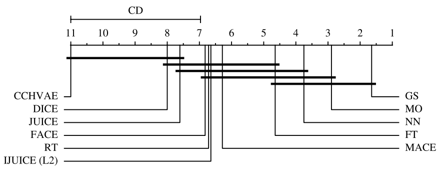
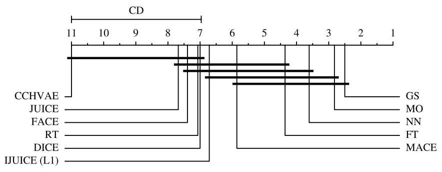


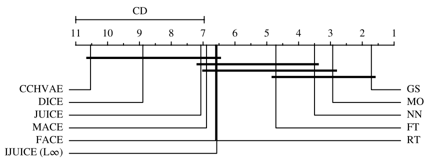
Fig. 12 Each subplot shows the scatter of the methods with respect to a specified distance function. The x-axis on each subplot represents the feasibility ranking. The y-axis represents the justification ranking. The dashed lines indicate the exact ranking of the center of the dots in both axes. The bigger the circle, the better the methods performance in terms of proximity to the CF. The best method is a big circle located on the upper right corner, whilst the worst method is a small circle located in the lower left corner



(a) $l2$-norm distance CD.



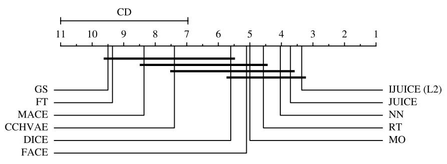
(b) $l1$-norm distance CD.



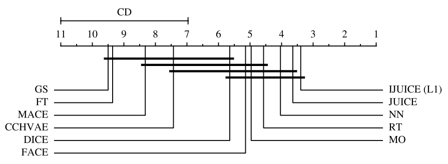
(c) $l\infty$-norm distance CD.

Fig. 13 Distance ranking CD diagrams for all CF generation algorithms. iJUICE optimizes the corresponding cost function

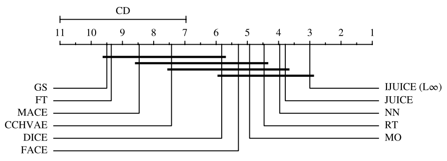
perform statistically better than JUICE in the $l2$ and $l1$-norms, while GS is the only method that is statistically different and better than iJUICE. Note that GS is by this advantage sacrificing the consideration of heterogeneous data and feasibility of the obtained CF points.



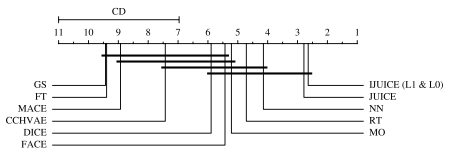
(a) $l2$-norm justification CD.



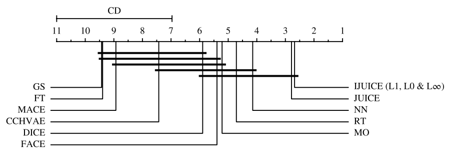
(b) $l1$-norm justification CD.



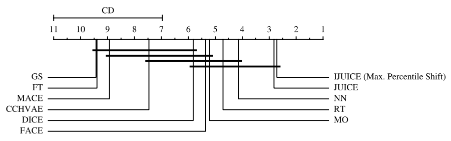
(c) $l\infty$-norm justification CD.



(d) $l1$ & $l0$-norm justification CD.



(e) $l1$, $l0$ & $l\infty$-norm justification CD.



(f) Max. Percentile Shift justification CD.

Fig. 14 Justification CD diagrams for all distance measures used as cost function of iJUICE, while using a $\lambda = 1$ (no weight on the justification ratio)

C.3 Justification

Figure 14 shows the justification ranking for all distance measures considered as cost function by iJUICE. For these plots, $\lambda = 1$, i.e., the weight of the justifier ratio is zero. Even under this condition, iJUICE always presents at least 1 justifying instance, beating the other methods in justification.

C.4 Run times

Run times for all the datasets and CF generation methods are shown in Fig. 15. The ranking of the CF generation algorithms with respect to run time are shown in Fig. 16.

C.5 Sparsity

Sparsity (the number of features that are different among the instance of interest and the CF) for all the datasets and CF generation methods are shown in Fig. 17.

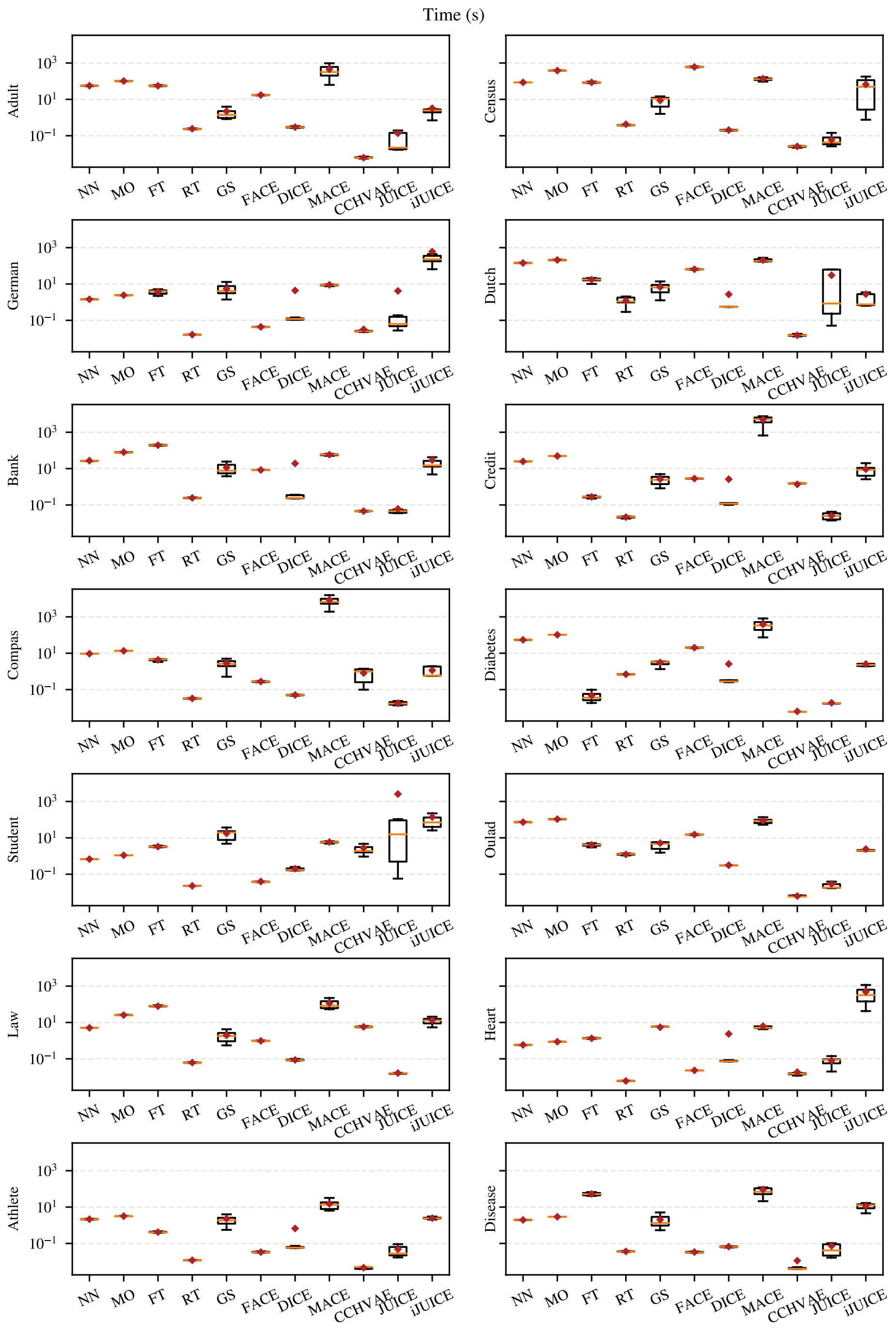
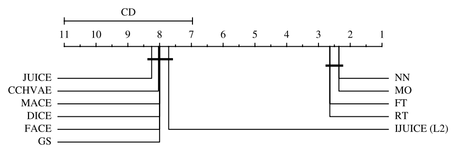
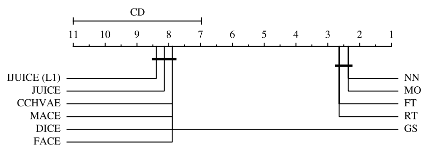


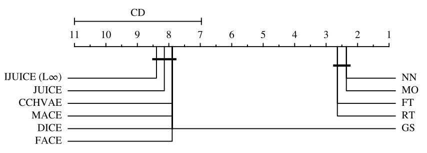
Fig. 15 Average run times (in seconds) per CF generated, for all methods and datasets. iJUICE is optimizing for $l1$, $l0$ distance



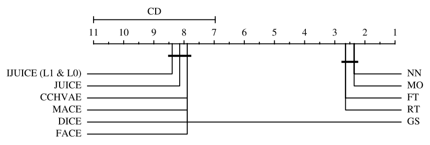
(a) $l2$-norm run time CD.



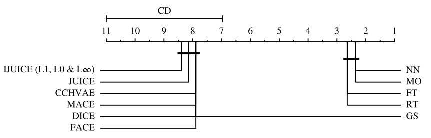
(b) $l1$-norm run time CD.



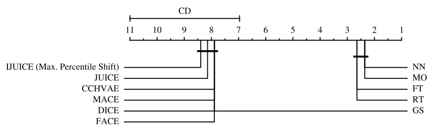
(c) $l\infty$-norm run time CD.



(d) $l1$ & $l0$-norm run time CD.



(e) $l1$, $l0$ & $l\infty$-norm run time CD.



(f) Max. Percentile shift run time CD.

Fig. 16 Run time ranking for all CF generation methods and iJUICE with all distance measures considered as cost functions

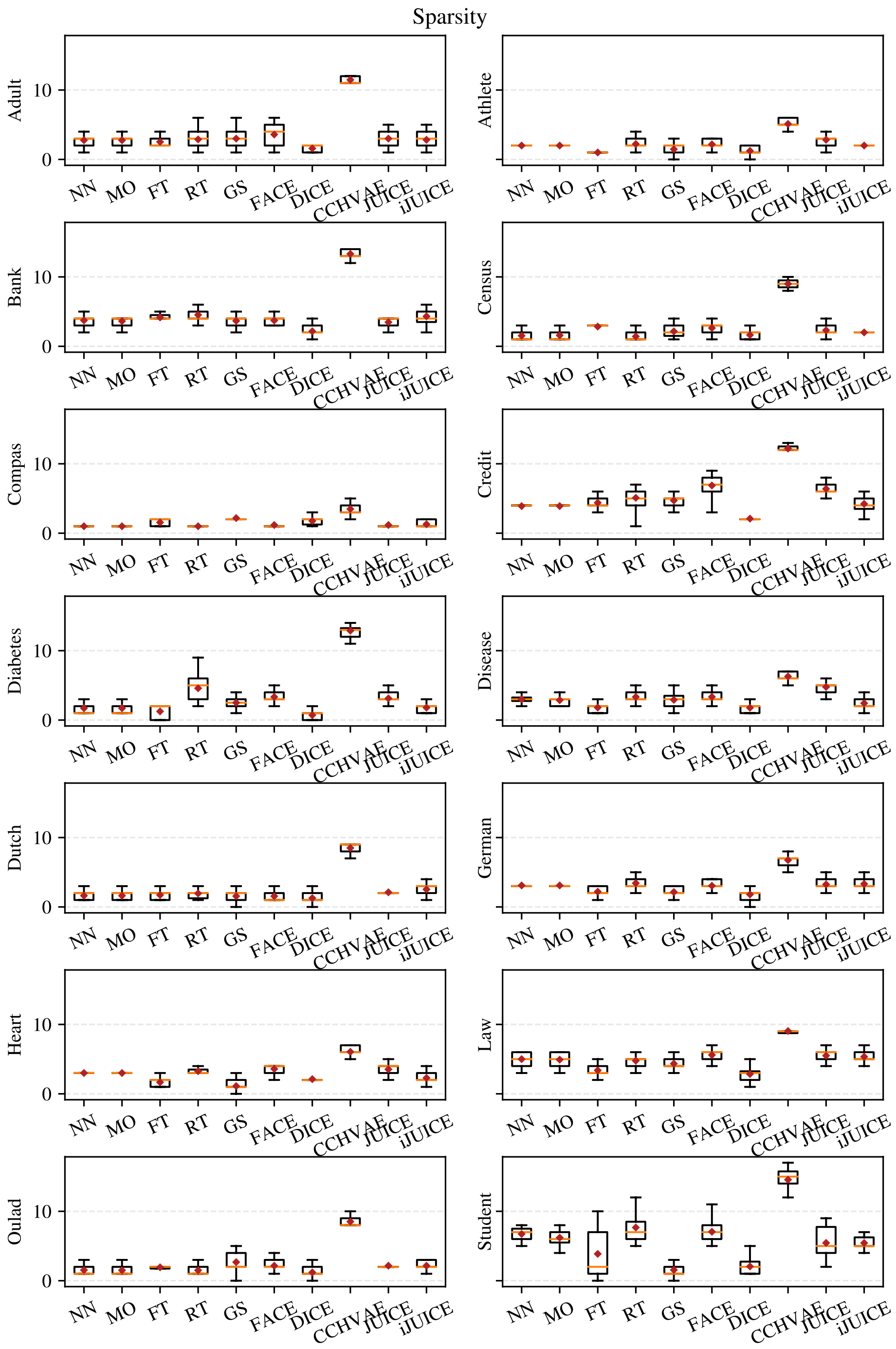


Fig. 17 Average sparsity (number of modified features) per CF generated, for all methods and datasets. iJUICE is optimizing for $l1$, $l0$ distance

Author Contributions All authors contributed to the preparation of the manuscript and the conceptualization. Alejandro Kuratomi initially proposed the approach and developed the solution to the k-justified counterfactual generation problem and the empirical evaluation, as well as the elaboration of the original draft. Dr. Ioanna Miliou greatly contributed to the conceptualization, design of the algorithm and the original draft preparation. Dr. Zed Lee has mainly contributed to the elaboration and design of the algorithm and the draft preparation. Dr. Tony Lindgren contributed both in the conceptualization and draft review and editing. Dr. Panagiotis Papapetrou contributed in the design of the algorithm, supervised the results and analysis, and oversaw the research project in general.

Funding Open access funding provided by Stockholm University. The work of IM and PP has been supported in part by the Digital Futures EXTREMUM project titled*Explainable and Ethical Machine Learning for Knowledge Discovery from Medical Data Sources.*

Data Availability The data used in this research is publicly available in different repositories online. We preprocess it and make it available in our own repository for replicability.

Code Availability The code for the proposed algorithm is made available at the repository (https://github.com/alku7660/iJUICE) together with the mentioned preprocessed datasets.

Declarations

Conflict of interest There are no conflicts of interest by any of the authors.

References

Basu, A., Conforti, M., Di Summa, M., & Jiang, H. (2022). Complexity of branch-and-bound and cutting planes in mixed-integer optimization. *Mathematical Programming* (pp. 1–24).

Bobek, S., & Nalepa, G. J. (2019). Explainability in knowledge discovery from data streams. In *2019 first international conference on societal automation (SA)* (pp. 1–4). IEEE.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on management of data* (pp. 93–104).

Byrne, R. M. (2019). Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning. In *IJCAI* (pp. 6276–6282).

Carreira-Perpiñán, M.Á., & Hada, S.S. (2021). Counterfactual explanations for oblique decision trees: Exact, efficient algorithms. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35, pp. 6903–6911

Cohen, M. B., Lee, Y. T., & Song, Z. (2021). Solving linear programs in the current matrix multiplication time. *Journal of the ACM (JACM), 68*(1), 1–39.

Dandl, S., Molnar, C., Binder, M., & Bischl, B. (2020). Multi-objective counterfactual explanations. In *International conference on parallel problem solving from nature* (pp. 448–469). Springer.

de Oliveira, R. M. B., & Martens, D. (2021). A framework and benchmarking study for counterfactual generating methods on tabular data. *Applied Sciences, 11*(16), 7274.

Eiras-Franco, C., Martinez-Rego, D., Guijarro-Berdinas, B., Alonso-Betanzos, A., & Bahamonde, A. (2019). Large scale anomaly detection in mixed numerical and categorical input spaces. *Information Sciences, 487*, 115–127.

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

Guidotti, R., & Ruggieri, S. (2021). Ensemble of counterfactual explainers. In *Proceedings of discovery science: 24th international conference, DS 2021, Halifax, NS, Canada*, October 11–13, 2021, pp. 358–368. Springer.

Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S., & Turini, F. (2019). Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems, 34*(6), 14–23.

Kanamori, K., Takagi, T., Kobayashi, K., & Arimura, H. (2020). Dace: Distribution-aware counterfactual explanation by mixed-integer linear optimization. In: *IJCAI* (pp. 2855–2862)

Kannan, R., & Monma, C. L. (1978). On the computational complexity of integer programming problems. *Optimization and Operations Research* (pp. 161–172). Chap. 17.

Karimi, A.-H., Barthe, G., Balle, B., & Valera, I. (2020). Model-agnostic counterfactual explanations for consequential decisions. In *International conference on artificial intelligence and statistics* (pp. 895–905). PMLR.

Karimi, A.-H., Barthe, G., Schölkopf, B., & Valera, I. (2022). A survey of algorithmic recourse: Contrastive explanations and consequential recommendations. *ACM Computing Surveys, 55*(5), 1–29.

Kuratomi, A., Miliou, I., Lee, Z., Lindgren, T., & Papapetrou, P. (2022). Juice: Justified counterfactual explanations. In *International conference on discovery science* (pp. 493–508). Springer

Kuratomi, A., Pitoura, E., Papapetrou, P., Lindgren, T., & Tsaparas, P. (2022). Measuring the burden of (un) fairness using counterfactuals. In *Joint European conference on machine learning and knowledge discovery in databases*, (pp. 402–417). Springer

Kyrimi, E., Neves, M. R., McLachlan, S., Neil, M., Marsh, W., & Fenton, N. (2020). Medical idioms for clinical Bayesian network development. *Journal of Biomedical Informatics, 108*, 103495.

Laugel, T., Lesot, M.-J., Marsala, C., & Detyniecki, M. (2019). Issues with post-hoc counterfactual explanations: a discussion. arXiv:1906.04774.

Laugel, T., Lesot, M.-J., Marsala, C., Renard, X., & Detyniecki, M. (2017). Inverse classification for comparison-based interpretability in machine learning. arXiv:1712.08443.

Laugel, T., Lesot, M.-J., Marsala, C., Renard, X., & Detyniecki, M. (2019). Unjustified classification regions and counterfactual explanations in machine learning. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 37–54). Springer.

Le Quy, T., Roy, A., Iosifidis, V., Zhang, W., & Ntoutsi, E. (2022). A survey on datasets for fairness-aware machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 12*(3), 1452.

Lenstra, H. W., Jr. (1983). Integer programming with a fixed number of variables. *Mathematics of Operations Research, 8*(4), 538–548.

Lindgren, T., Papapetrou, P., Samsten, I., & Asker, L. (2019). Example-based feature tweaking using random forests. In *2019 IEEE 20th international conference on information reuse and integration for data science (IRI)* (pp. 53–60). IEEE.

Martens, D., & Provost, F. (2014). Explaining data-driven document classifications. *MIS Quarterly, 38*(1), 73–100.

Miller, T. (2019). Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence, 267*, 1–38.

Molnar, C. (2020). Interpretable machine learning: A guide for making black-box models explainable. https://christophm.github.io/interpretable-ml-book/limo.html

Mothilal, R. K., Sharma, A., & Tan, C. (2020). Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency* (pp. 607–617).

Otey, M. E., Ghoting, A., & Parthasarathy, S. (2006). Fast distributed outlier detection in mixed-attribute data sets. *Data Mining and Knowledge Discovery, 12*, 203–228.

Papadimitriou, C. H. (1981). On the complexity of integer programming. *Journal of the ACM (JACM), 28*(4), 765–768.

Parmentier, A., & Vidal, T. (2021). Optimal counterfactual explanations in tree ensembles. In *International conference on machine learning* (pp. 8422–8431). PMLR.

Pawelczyk, M., Bielawski, S., van den Heuvel, J., Richter, T., & Kasneci, G. (2021). CARLA: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms.

Pawelczyk, M., Broelemann, K., & Kasneci, G. (2020). Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of the web conference 2020* (pp. 3126–3132).

Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., De Bie, T., & Flach, P. (2020). Face: feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM conference on AI, ethics, and society* (pp. 344–350).

Ramon, Y., Martens, D., Provost, F., & Evgeniou, T. (2020). A comparison of instance-level counterfactual explanation algorithms for behavioral and textual data: Sedc, lime-c and shap-c. *Advances in Data Analysis and Classification, 14*, 801–819.

Rawal, K., & Lakkaraju, H. (2020). Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses. *Advances in Neural Information Processing Systems, 33*, 12187–12198.

Ribeiro, M.T., Singh, S., & Guestrin, C. (2016). "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence, 1*(5), 206–215.

Russell, C. (2019). Efficient search for diverse coherent explanations. In *Proceedings of the conference on fairness, accountability, and transparency* (pp. 20–28).

Sharma, S., Henderson, J., & Ghosh, J. (2020). CERTIFAI: Counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. *Proceedings of the AAAI/ACM conference on AI, ethics, and society* (pp. 166–172). https://doi.org/10.1145/3375627.3375812. arXiv: 1905.07857. Accessed 2022-03-05.

Tolomei, G., Silvestri, F., Haines, A., & Lalmas, M. (2017). 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).

Ustun, B., Spangher, A., & Liu, Y. (2019). Actionable recourse in linear classification. In *Proceedings of the conference on fairness, accountability, and transparency* (pp. 10–19)

Verma, S., Dickerson, J., & Hines, K. (2020) Counterfactual explanations for machine learning: A review. arXiv:2010.10596.

Vermeire, T., Brughmans, D., Goethals, S., de Oliveira, R. M. B., & Martens, D. (2022). Explainable image classification with evidence counterfactual. *Pattern Analysis and Applications, 25*(2), 315–335.

Wachter, S., Mittelstadt, B., & Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL and Tech., 31*, 841.

Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viégas, F., & Wilson, J. (2019). The what-if tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics, 26*(1), 56–65.

White, A., & Garcez, A. (2019). Measurable counterfactual local explanations for any classifier. arXiv:1908.03020

Yang, L., Kenny, E. M., Ng, T. L. J., Yang, Y., Smyth, B., & Dong, R. (2020). Generating plausible counterfactual explanations for deep transformers in financial text classification. arXiv:2010.12512.