applied sciences

MDPI

Article

A Framework and Benchmarking Study for Counterfactual Generating Methods on Tabular Data

Raphael Mazzine Barbosa de Oliveira * and David Martens

Department of Engineering Management, University of Antwerp, Prinsstraat 13, 2000 Antwerpen, Belgium; david.martens@uantwerpen.be

* Correspondence: Raphael.MazzineBarbosaDeOliveira@uantwerpen.be

Abstract: Counterfactual explanations are viewed as an effective way to explain machine learning predictions. This interest is reflected by a relatively young literature with already dozens of algorithms aiming to generate such explanations. These algorithms are focused on finding how features can be modified to change the output classification. However, this rather general objective can be achieved in different ways, which brings about the need for a methodology to test and benchmark these algorithms. The contributions of this work are manifold: First, a large benchmarking study of 10 algorithmic approaches on 22 tabular datasets is performed, using nine relevant evaluation metrics; second, the introduction of a novel, first of its kind, framework to test counterfactual generation algorithms; third, a set of objective metrics to evaluate and compare counterfactual results; and, finally, insight from the benchmarking results that indicate which approaches obtain the best performance on what type of dataset. This benchmarking study and framework can help practitioners in determining which technique and building blocks most suit their context, and can help researchers in the design and evaluation of current and future counterfactual generation algorithms. Our findings show that, overall, there's no single best algorithm to generate counterfactual explanations as the performance highly depends on properties related to the dataset, model, score, and factual point specificities.

Keywords: artificial explainable intelligence; counterfactual explanations; artificial neuronal networks; black box models; algorithm benchmarking

Citation: de Oliveira, R.M.B.; Martens, D. A Framework and Benchmarking Study for Counterfactual Generating Methods on Tabular Data. *Appl. Sci.* **2021**, *11*, 7274. https://doi.org/10.3390/app11167274

Academic Editor: Jose Antonio Iglesias Martinez

Received: 9 July 2021
Accepted: 3 August 2021
Published: 7 August 2021

1. Introduction

Machine learning algorithms are becoming ever more common in our daily lives [1]. One of the reasons for this widespread application is the high prediction accuracy those methods can achieve. However, this gain in predictive accuracy comes at a cost of increased complexity of the prediction models and in a resulting lack of interpretability [2]. The inability to explain why certain predictions are made can have a drastic impact on the adoption of automated decision-making in society, as people are often reluctant to use such complex models, even if they are known to improve the predictive performance [3–9]. Furthermore, these models may hide unfair biases that discriminate against sensitive groups [10–13]. These problems can be even more critical when models are constantly created and updated, as often observed in real-time applications [14].

To solve this deficiency, multiple strategies are proposed in a research domain commonly referred to as 'eXplainable AI' (XAI) [15], aimed at unveiling the high complexity of the models obtained through machine learning methodologies as deep neural networks [16,17], ensemble methods [18,19], and support vector machines [20]. They also have vast application in various fields, including finance [21,22], medicine [23,24], and self-driving cars [25,26].

The XAI methodologies are diverse, and can roughly be categorized according to the model that provides the explanation, the type of data, the scope of explanation, and general objective [15]. First, in terms of model type, explainability can be derived from intrinsically comprehensible models [27,28], and common examples are decision trees and

linear regression. However, complex models ironically require additional post-hoc methodologies to obtain explanations. These can be designed for specific types of models [18,29] or be model agnostic [27,30]. In terms of data type, explanations naturally vary across different variations of data. For example, images often use a group of pixels to identify image patterns or features [31], while the explanations for tabular data decisions are mostly described in feature value combinations [32]. Analyzing the scope of the explanations [15]: these can be global, trying to explain how the model works over the complete dataset, or local, where a single point prediction is explained. Finally, the explanation objectives can be diverse [33], depending on the objective of the application (model improvement, bias detection, decision justification, and others) and role of the user receiving the explanation (manager, data scientist, end user) [34]. One particular explanation methodology of interest is the counterfactual explanation [34,35], which generates local explanations, but has diverse implementation settings as they can be model specific, model agnostic, work with different types of data, and serve several objectives.

Counterfactual explanations have a long history, even beyond that of XAI, with roots in both psychology and philosophy [35–37]. According to Lipton [38] counterfactual explanations answer the question *"Why P rather than Q?"*. The outcomes P and Q are termed fact and foil, respectively, where the former is the event that occurred (in this article referred to as factual), and the foil is the alternate case that did not occur (called counterfactual here). Several studies indicate the advantages of counterfactual explanations [13,39], as they can make algorithmic decisions more understandable for humans [35]. Additionally, the recent legal requirements for the use of machine learning in decision-making process, such as those asked in GDPR, can also be fulfilled with counterfactual explanations [40]. Counterfactual explanations additionally have key advantages over feature importance methodologies (like LIME [41] or SHAP [2]): although the weight of each feature may relate to its importance in the model's prediction score, it may not explain the model decision, as those methodologies do not track the influence that features have on changing the output class [42].

All these potential benefits caused the development of a continuously increasing number of counterfactual explanation generation methodologies over the last few years [43,44], which seemingly started with the work of Martens and Provost in 2014 [34]. In recent reviews of those methodologies [44,45], it was found that they have a wide variation in implementation aspects, like optimization strategy, model and data access, and input data compatibility. This variation, most of the time, is implemented as trade-offs for desired objectives in the counterfactual generation.

The reviews found important patterns among all methodologies. For example, most of the methods use heuristic rules and/or gradient optimization techniques as search strategy, are designed to work with differentiable models (such as neural networks), and are focused on tabular data. However, for the other aspects, we find less clear patterns, especially because they try to focus on specific settings (for example, generating counterfactuals without the need to access the model or data) or try to have different goals (for example, generating diverse counterfactuals). This variation and complexity in the generation of counterfactuals may be one reason why no benchmarking study with different algorithmic approaches has been conducted in the literature yet [44]. However, such study is of fundamental importance to explore which algorithmic implementations are preferred in specific data settings and to indicate the research directions for improved counterfactual learning.

The development of a benchmarking study relies primarily in the overall experimental setup to run different counterfactual explanation generators and the test conditions. To compare different counterfactual explanations, clear and well-defined metrics are also to be provided. Some works [33] give an overall framework to systematically analyze explainable approaches; however, these evaluations are mostly "high level", without an experimental comparison, and do not provide means to effectively compare the outputs of counterfactual explanation generators.

This article aims to address this research gap and contributes in several ways in the design and implementation of benchmarks of counterfactual explanations' algorithms:

1. A benchmarking study of 10 counterfactual generation algorithms, on 22 datasets, assessing the algorithmic and explanation quality on nine relevant metrics.
2. Analysis and discussion on which algorithmic approach performs well for what dataset type. This insight can guide practitioners in choosing a suitable method.
3. A benchmark framework implementation to test different counterfactual generation algorithms. The framework is open-source, allows for easily adding new algorithms, and is intended to be a gold standard to evaluate and compare counterfactual generating algorithms.

All the points described above represent a novel approach to test and evaluate counterfactual generation algorithms. The outcome of this paper is not just a comparative analysis of the tested algorithms, but the benchmark framework can also integrate new approaches and help in their development. Additionally, the theoretical definition and scores detailed in this paper may help with establishing procedures to evaluate the quality of counterfactual explanations.

2. Benchmark Components

2.1. Counterfactual Explanation Generation Methods

Our choice of counterfactual generating algorithms is motivated by the following prerequisites: an open-source implementation, the use of novel theoretical and algorithmic strategies, and compatibility (in the code implementation) with artificial neural network models and tabular data. This resulted in 10 different algorithmic approaches to generate counterfactual explanations. The techniques can be categorized according to (1) the search strategy, (2) required access to the training data and (3) model internals, (4) whether the method is agnostic to a prediction model or only works for a specific model, (5) whether it implements additional optimization terms, and (6) whether it allows for additional custom terms, such as enforcing one-hot encoding, as summarized in Table 1.

Each technique follows a certain search strategy. The two most common approaches are the use of convex optimization methodologies or custom heuristic rules. Convex optimization routines will include calculation of gradients, and hence require a differentiable scoring function. For a neural network, this requires having access to the weights [46–49] or using numerical estimation of gradients [47]. In the case of heuristic rules, different procedures have been proposed that usually do not require accessing model inner settings and, therefore, have better model compatibility. Note that, as the method ALIBIC has two different implementations, both are included in this study. One uses the internal neural network weights to calculate gradients (we call this implementation ALIBIC), while the other will estimate the gradients without using the internal weights, which we refer to as ALIBICNOGRAD.

Some algorithms require having access to the training data and/or the model internals. The training data requirement is generally aimed at improving counterfactual explanation quality, since the data can serve as supporting evidence in the counterfactual generation process. Such usage is implemented by ALIBIC, DiCE, GrowingSpheres, LORE, MACE, SEDC, and SynAS. Similarly, algorithms like ALIBIC (although it is not mandatory), CADEX, DiCE, MACE, ML-Explain, and SynAS require access to the inner model settings, such as the already cited neural network weights, to allow the use of specific optimization techniques.

Several algorithms take advantage of additional optimization terms to improve some aspects of the counterfactual explanation quality. When convex optimization techniques are used, additional loss terms can include functions that measure sparsity (the number of changes needed to generate a counterfactual), proximity (how close the counterfactual is from the factual case), feasibility (statistical similarity of the counterfactual point to the data manifold), and diversity (generate several different counterfactual explanations); these terms are discussed in more detail in Section 3.2.

For sparsity and improvement of the distance from the counterfactual to the factual point, ALIBIC and ML-Explain use L1 and L2 norms to calculate an elastic net loss term. In addition, to improve both the proximity to the factual point and feasibility of changes, ALIBIC and DiCE use distance metrics, such Euclidean distance, with some modifications and/or additional terms, where ALIBIC uses autoencoders trained with factual data and DiCE by using a distance metric that considers each feature variation. Finally, diversity is enhanced in DiCE by a function that considers the distance between each counterfactual explanation generated.

Notwithstanding, these additional optimization terms are also used in algorithms that do not use convex optimization. For example, GrowingSpheres adds terms to control both sparsity (L0 norm) and proximity (L2 norm) of the solution, LORE adds a distance function (L2 norm) to control proximity and uses a genetic algorithm to improve feasibility, MACE adds additional terms to control proximity, sparsity, and diversity in the form of requirements of the satisfiability solver (SS) and uses feature values and ranges (as observed in the dataset) to enhance feasibility, while SEDC improves the feasibility of its results by calculating (and using) the mean of each feature.

Finally, some algorithms allow for including additional custom constraints to control feature changes. The constraints intend to avoid unwanted counterfactuals. One-hot encoding (OHE) is a very common example of a constraint that is included in several algorithms (ALIBIC, DiCE, MACE, SynAS, and LORE). This constraint requires that only one of the dummy variables corresponding to a single nominal variable be assigned the numerical value of one. For example, a blood type categorical feature does not allow for being both O type and A type at the same time. Another common constraint to counterfactual results is the inclusion of weights that penalize changes in specified features (CADEX, DiCE, MACE, and SynAS), and ranges (ALIBIC, DiCE, MACE, and SynAS) that can constrain features to vary between specific chosen values.

Table 1. Overall counterfactual explanation generation algorithm characteristics. Black filled circles indicate the feature is present, while white filled circles represent that the feature is not present. **CO**: Convex Optimization. **HE**: Heuristic. **SS**: Satisfiability Solver. **REQ.**: Required. **NRE.**: Not Required. **OPT**: Optional. **Spar.**: Sparsity. **Prox.**: Proximity. **Feas.**: Feasibility. **Diver.**: Diversity. **W**: Weights. **R**: Range. **OHE**: One-Hot Encoding.

Algorithm	General Characteristics				Additional Optimization Terms				Custom Constraints		
	Find Method	Data Access	Model Access	Model Comp.	Spar.	Prox.	Feas.	Diver.	W	R	OHE
ALIBIC [47]	CO	REQ.	OPT.	Agnostic	●	●	●	○	○	●	●
CADEX [50]	CO	NRE.	REQ.	Specific	○	○	○	○	●	●	●
DiCE [46]	CO	OPT.	REQ.	Specific	○	●	●	●	●	●	●
Growing Spheres [51]	HE	OPT. [1]	NRE.	Agnostic	●	●	○	○	○	○	○
LORE [52]	HE	REQ.	NRE.	Agnostic	○	●	●	○	○	○	●
MACE [53]	SS	REQ.	REQ.	Specific	●	●	●	●	●	●	●
ML-Explain [48]	CO	NRE.	REQ.	Specific	●	●	○	○	○	○	○
SEDC [42]	HE	REQ. [2]	NRE.	Agnostic	○	○	●	○	○	○	○
SynAS [49]	CO	REQ.	REQ.	Specific	○	○	○	○	●	●	●

[1] In the original paper, it theoretically has a step to generate points, hence not depending on data; however, in the implementation example, it uses the training and test data. In our benchmarking study, we provide training data. [2] The SEDC implementation does not ask, directly, for the dataset access; however, it requires the mean for each feature.

2.2. Datasets

The datasets are chosen from the public and well-known data repository (UCI Machine Learning Repository). Although all datasets are used to generate counterfactuals for all algorithms, some dataset characteristics may lead to a preference of one counterfactual explanation generation algorithm rather than another. For example, a dataset with only categorical features may not lead to good results by an algorithm that assumes all features are continuous. Similarly, a dataset with a large number of rows may give advantage to

algorithms that use the data points as evidence to generate counterfactuals. Therefore, datasets of various data types (both categorical, numerical, and mixed variables), application domain, number of features, ratio of feature types, class balance, and kinds of data constraints are selected (see Table 2). Several of these datasets are widely used in counterfactual explanation studies like Adult [46,52–58], Statlog-German Credit [46,49,50,52], Breast Cancer Wisconsin (BCW) [47,55,56,59,60], and Wine [60–62]. Note that the number of datasets included in the papers where these algorithms have been proposed range from 1 to only 4 [46], which further motivates the need for a large-scale benchmarking study.

2.3. Models

The artificial neural network models are built using TensorFlow [63] and the Keras package (Official website: https://keras.io/, accessed on 9 July 2021) in Python. The model architecture is chosen to optimize (validation) model performance. A single layer neural network is used as the model to be explained, with a variable size of neurons in the hidden layer. As the literature shows that there is no single best rule of thumb to follow to determine the optimal number of hidden neurons [64,65], we use a grid-search methodology to define it, looking for values up to twice the number of input vector size plus one [66,67]. The grid also includes 4/5, 3/5, 2/5, and 1/5 of the maximum value. Other hyperparameters, such as the learning rate and the number of epochs, are also chosen using a grid-search. All other hyperparameters are set to the default values from the TensorFlow/Keras package. The model with the best AUC on the validation set was selected for each dataset. The full model specifications can be found in the Appendix A.

Table 2. Dataset Characteristics-Table with summarized information about each dataset used in the benchmarking study. Black filled circles indicate that the feature is present, while white filled circles represent that the feature is not present.

	Dataset Information			**Feature Information**			**Realistic Constraints**	
Dataset	**Data Type**	**Area**	**N. Rows**	**CAT.**	**NUM.**	**CLAS.**	**Univariate**	**Multivariate**
Soybean (small) [68]	COMP.	CAT.	47	35	0	4	●	●
Lymphography [68,69]	COMP.	CAT.	148	18	0	4	●	●
Hayes–Roth [68]	SOCI.	CAT.	160	5	0	3	●	●
Balance Scale [68]	SOCI.	CAT.	625	4	0	3	●	●
Tic-Tac-Toe [1] [68]	COMP.	CAT.	958	9	0	2	●	●
Car Evaulation [68]	N/A	CAT.	1728	6	0	4	●	●
Chess [2] [68]	GAME	CAT.	3196	36	0	2	●	●
Nursery [68]	COMP.	CAT.	12,960	8	0	5	●	●
Iris [68]	LIFE	NUM.	150	0	4	3	●	○
Wine [68]	PHYS.	NUM.	178	0	13	3	●	○
Ecoli [68]	LIFE	NUM.	336	0	7	8	●	○
CMSC [3] [68,70]	PHYS.	NUM.	540	0	18	2	●	○
BCW [4] [68]	LIFE	NUM.	569	0	10	2	●	●
PBC [5] [68]	COMP.	NUM.	5473	0	10	5	●	●
ISOLET [68]	COMP.	NUM.	7797	0	617	26	●	○
MAGIC GT [6] [68]	PHYS.	NUM.	19,020	0	10	2	●	○
SDD [7] [68]	COMP.	NUM.	58,509	0	49	11	●	○
Statlog-GC [68]	BUSI.	MIX.	511	19	19	2	●	●
Student Perf. [68,71]	SOCI.	MIX.	649	27	3	2	●	●
Internet Adv. [68]	COMP.	MIX.	690	9	6	2	●	●
Default of CCC [68,72]	BUSI.	MIX.	30,000	9	14	2	●	●
Adult [68]	SOCI.	MIX.	48,842	9	5	2	●	●

[1] Tic-Tac-Toe Endgame. [2] Chess (King–Rook vs. King). [3] Climate Model Simulation Crashes. [4] Breast Cancer Wisconsin. [5] Page Blocks Classification. [6] MAGIC Gamma Telescope. [7] Sensorless Drive Diagnosis.

2.4. Realistic Counterfactuals

If the counterfactual generation is left unconstrained, solutions might depict situations that will simply be deemed unacceptable to the end user. We briefly touch upon this problem by defining a subspace called realistic space.

Firstly, it is important to highlight the difference between valid and realistic counterfactuals. Let us start with what a valid counterfactual means. In this article, valid counterfactuals are those that flipped the original factual prediction category. For example, consider the situation where someone is denied a credit card. A valid counterfactual would simply be an alternate situation which changes the classification result, from not approved to approved, while an invalid counterfactual is also an alternate situation returned by the counterfactual generator (with some feature changes), but that was not able to flip the original factual prediction. One could also argue that an invalid counterfactual is simply not a counterfactual. On the other hand, realistic counterfactuals are defined by how realistic the counterfactual is in the input space, and is detailed below.

Counterfactuals correspond to datapoints while the difference (changes) between the factual and counterfactual is the explanation. It is, however, possible that these datapoints are simply unrealistic, as some features may have been modified to values that are impossible in reality (Figure 1 illustrates this situation). One simple example is the one-hot key encoded feature: categorical/nominal features that are typically transformed into several binary features. This transformation implies some rules: the dummy variables must only have one "active" variable, and they all must be binary. Re-iterating the previously mentioned example: one cannot have both type O and type A blood type, or be old and young, or married and divorced. Although the prediction model can easily provide an output for a datapoint where both dummy features are set to one (potentially even leading to a counterfactual), it is not a realistic result, without a genuine representation of what this feature means to represent.

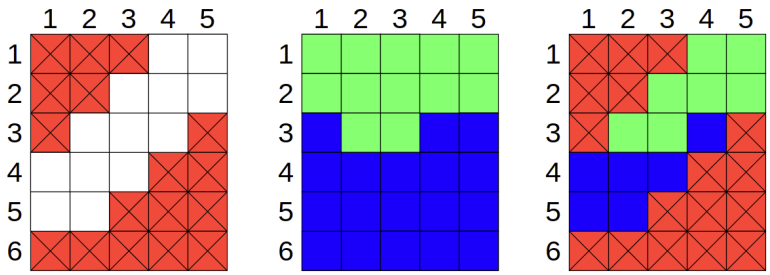

Figure 1. (**Left**) representation of the realistic space, where the white squares are the realistic states for a dataset and the red crossed squares are the unrealistic entries. (**Middle**) representation of model's output for all values, as models are just mathematical formulations, any input will lead to an output (even the unrealistic dataset points), the green and blue squares represent two different model output classifications. (**Right**) superposition of the two first, where it shows the realistic values to a classification.

The same issue occurs with other kinds of features. In the case of numeric features, there might be an unrealistic set or range of feature values. Age, for example, must be larger than zero, as negative ages are not realistic entries. This shows that, although the model can accept a wide variation of numerical inputs, only a subset of them may correspond to a realistic point.

The generation of new points that follow the inner rules that define a realistic datapoint is a challenge in itself, having its own methodologies that aim to achieve it [73]. This constraint has deep implications in the generation of counterfactual explanations, as, if left unchecked, the counterfactual generation can lead to such unrealistic data points. It is important to remember that this problem is distinct from the task of generating actionable

counterfactuals [44], which is much more complex [74]. While previously we required that the counterfactual point needs to be realistic, the actionability constraint requires that the difference between the initial data instance and the counterfactual is achievable. For example, if the change implies age decreasing 5 years, the explanation is not actionable, as the change is impossible to perform; however, it can be realistic if the targeted age is larger than zero.

The requirement to generate points that are realistic highlights the need to define a particular input space where each of the features (and their combination) are conceptually genuine in relation. We call this subspace the realistic space, and will formalize this next.

Consider a set of m features $\{a_1, a_2, ..., a_m\}$. The realistic space (**R**) is defined as a subspace $\mathbf{R} \subseteq \mathbb{R}^m$, which represents all realistic features' combinations. A dataset (**D**) is (or should be, depending on potential data quality issues such as a negative age) a subset of the realistic space ($\mathbf{D} \subseteq \mathbf{R}$). If we get a point (**x**) in the dataset ($\mathbf{x} \in \mathbf{D}$) with class y, and a model M which transforms the point into a specific class ($\hat{y}$), we can define a counterfactual as a close-boundary point (**c**) where the model outputs a different class: $M(\mathbf{c}) = \hat{y}, y \neq \hat{y}$. The realistic space can span the entire real space ($\mathbf{R} = \mathbb{R}^m$) when there are no constraints and the features are free to vary. In most real world application settings, however, some constraints will be imposed to keep the counterfactuals realistic.

We, therefore, argue that, for real-life applications, especially when involving end-users requiring explanations that make sense, a counterfactual point should belong to the realistic space ($\mathbf{c} \in \mathbf{R}$). This preference is additionally implied by the already mentioned fact that an unrealistic counterfactual is not supported by any data observation (as all training points are, expectedly, inside the realistic space).

To have a better understanding of how the realistic space can be defined by constraints, we can categorize these into two types of constraints: univariate and multivariate. Univariate constraints define the requirements a single feature must follow (for example, age larger than zero), while multivariate constraints describe how a set of variables, dependent on each other, must change (for example, the one-hot encoding rule).

This categorization is additionally motivated by the fact that many algorithms treat these requirements differently. For example, ALIBIC, DiCE, MACE, and SynAS can manually restrict the range of certain features, while ALIBIC and DiCE also use, respectively, autoencoders and statistical approaches to generate counterfactuals that are more consistent with the data manifold (thereby, hopefully, respecting both univariate and multivariate constraints). It is also worth noting that, in some situations, we can have both kinds of constraints, like with one-hot encoding: to ensure binary values (univariate) and single activation (multivariate).

3. Experiments

3.1. Experimental Design

The experimental design is summarized in Figure 2, starting with the raw data processing, normalization by using the mean (μ_a) and variance (σ_a^2) of each continuous feature ($\mathbf{a}_{norm} = (\mathbf{a} - \mu_a)/\sigma_a^2$) and one-hot encoding of categorical features. Some counterfactual generating algorithms are not tailored towards multiclass datasets—as they, for example, do not explicitly allow for say: why class1 and not class2? However, rather: why class1 and not other than class1? Therefore, the target variable is made binary for multiclass datasets, considering the majority class versus all others. A training/validation/test split is made (60/20/20), followed by model training with definitions discussed in Section 2.3, factual data selection (the data instances to explain) and, finally, the counterfactual generation. The same models and data are used for all counterfactual explanation generation algorithms. This is an important configuration as different data processing or models can cause unreliable comparison results. Even simple modifications as scaling can lead to different distances between points which can lead to misleading results. The same can happen when working with different models, where different classification boundaries can lead to misleading results when comparing distinct counterfactual generators. The factual

data consist of up to 100 random points for each dataset and class. The selection primarily relies on data from the test set; however, as the test set of some small datasets has less than 100 instances, additional data are obtained from the validation (and potentially even the training set). For the counterfactual generation, we produce one counterfactual for each factual row. Although some algorithms allow for generating multiple counterfactuals for a single point, we always select the first generated counterfactual and consider it as being the best counterfactual explanation the algorithm can produce. All steps can be verified in the provided repository. Moreover, this benchmarking framework can accept other algorithms as it is intended to be an open benchmarking tool for counterfactual explanation algorithms.

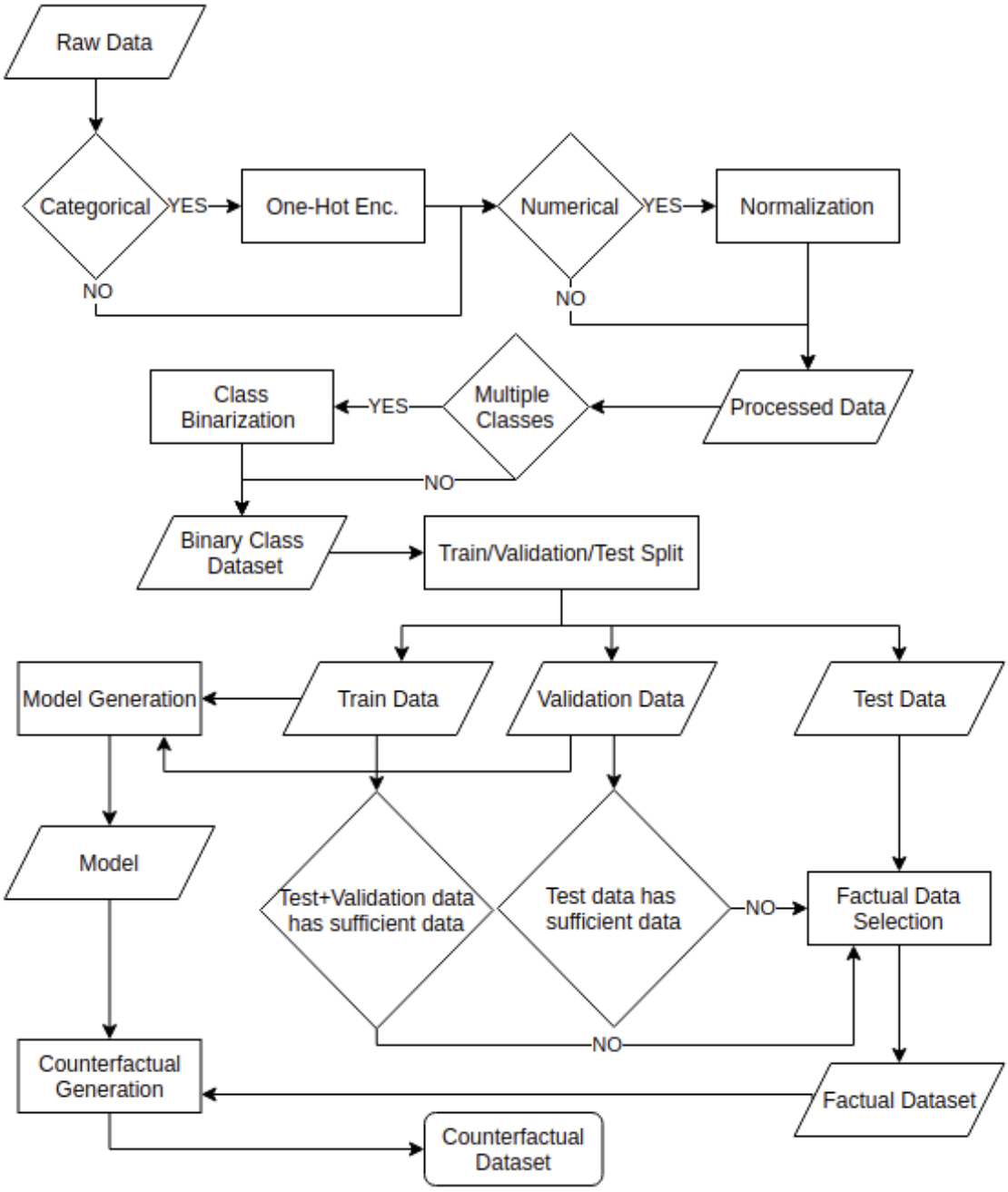

Figure 2. Flowchart detailing the benchmarking process from taking the raw dataset to the counterfactual generation.

3.2. Metrics

A crucial component of this benchmarking study is defining the metrics to evaluate the explanations and the generating process.

Coverage is simple, but arguably one of the most important metrics, which verifies if the provided solution actually flipped the factual class, so, in other words, whether a counterfactual explanation has been found.

Sparsity is a common metric used in counterfactual explanation studies as it's desirable to have short explanations, i.e., counterfactuals that differ in as few features as possible [75]. L2 is the Euclidean distance from the factual point to the counterfactual point; it is a largely used metric for counterfactuals as, by definition, a counterfactual should be as close as possible to the factual point.

Two additional distance metrics, MADD and MD, help to understand how the algorithms generate results with respect to the dataset. The MADD distance (MADD) uses the L1 distance between the factual and counterfactual, divided by the Median Absolute Deviation (MAD). This metric is reported [40,46] as being a valuable metric since it can work with different ranges across features, as it considers the variation of each changed feature. The Mahalanobis Distance (MD) [76], commonly used to find multivariate outliers, can take the correlation between features into account [62,77]. Together, these three distance metrics measure different aspects of the solutions that might be important for specific user defined requirements. For example, if the user is only concerned about having counterfactuals that lie as close as possible to the factual instance, with only continuous features, the Euclidean distance is an important metric to evaluate. However, if the distance is important, but different types and ranges of features are to be considered (such as those found in mixed datasets), the MADD score can be more suited. Finally, if the counterfactual should follow correlations between features found in data, the MD distance can be of special importance.

Stability is a score that verifies how reliable an algorithm is in generating consistent results across different runs using the same model and input data. This is calculated by comparing the results from two runs using the same parameters. A high stability means the algorithm, given the same input data and model, generates identical counterfactual results, which is of course a desirable characteristic for explainable approaches [33]. Randomization components in the search strategy can have a detrimental effect on this metric.

Realistic scores (univariate and multivariate) are metrics that tell if the algorithm generated a counterfactual respecting the constraints defined by the dataset (consequently showing if the counterfactual is inside the realistic space). This means that, if it equals one, it fulfills all constraints as defined by the dataset problem. The next subsection reports on how these constraints are defined.

Finally, the last metric is the time consumed to generate a counterfactual, measured in seconds, using the hardware and software setup described in the support resource (Appendix B). A full review of all metrics can be found in Table 3, where the formula for each individual calculation is given.

This benchmarking study does not evaluate actionability, which is a characteristic that allows a counterfactual to provide guidance on how a factual point can change the classification result by taking feasible steps. This is justified by our main focus, evaluating counterfactual explanations, which does not need to be actionable. Furthermore, we consider realistic counterfactuals more fundamental than actionable counterfactuals since the former is a prerequisite for the latter. Moreover, defining such constraints is considered a complex property [74] because it includes time and effort considerations, while most tabular datasets provide just a static picture of feature states. Therefore, it is not possible to infer (only having the static data) how each feature can vary over time. One solution for this problem is to design custom rules (see Table 1) that constrain how each feature can vary. However, such modeling requires laborious efforts and can be error-prone, which may discourage the use for real world applications. Finally, imposing the actionability constraints itself is then again arguably a 'simple' boolean requirement: does the algorithm

allow for imposing actionability constraints or not, for example: age can only increase (which hence relates to our realistic constraints). This further motivates why we excluded this aspect from the benchmarking study.

Table 3. Metrics used in the study with their respective descriptions.

Metric	Formula	Description
Coverage	$H(M(\mathbf{x}), M(\mathbf{c})), H = \begin{cases} 0, if\ y = \hat{y} \\ 1, if\ y \neq \hat{y} \end{cases}$	Verify if the counterfactual class ($\hat{y}$) has, indeed, a different class from the factual point class (y) using the same model (M)
Sparsity	$\frac{1}{m}\sum_{p=1}^{m} I(x_p, c_p), I = \begin{cases} 1, if\ x_p = c_p \\ 0, if\ x_p \neq c_p \end{cases}$	Share of features in the counterfactual array (c_p) that are the same as the features in the factual array (x_p) for all m features.
Stability	$I(\mathbf{c_1}, \mathbf{c_2}), I = \begin{cases} 1, if \sum_{p=1}^{m}(c_{p,1} - c_{p,2}) = 0 \\ 0, if \sum_{p=1}^{m}(c_{p,1} - c_{p,2}) \neq 0 \end{cases}$	Returns 1 if the same counterfactual is returned in two different runs ($\mathbf{c_1}$ and $\mathbf{c_2}$) with same model and input data, returns 0 otherwise.
L2	$\sqrt{\sum_{p=1}^{m}(x_p - c_p)^2}$	Euclidean (L2) distance between the factual point and the counterfactual point.
RUC	$\begin{cases} 1, if\ c_p \in \mathbf{R} \\ 0, if\ c_p \notin \mathbf{R} \end{cases}$	Univariate Constraint (RUC), score is 1 if features are in the realistic space ($\mathbf{R}$), 0 if not.
RMC	$\begin{cases} 1, if\ \{c_1, c_2, ..., c_m\} \in \mathbf{R} \\ 0, if\ \{c_1, c_2, \ldots, c_m\} \notin \mathbf{R} \end{cases}$	Multivariate Constraint (RMC), score is 1 if set of features are in realistic space, 0 if not.
MADD	$\begin{cases} \frac{1}{m_{num}}\sum_{p=1}^{m_{num}} \frac{\lvert x_p - c_p \rvert}{MAD_p}, if\ x_p\ is\ numerical \\ \frac{1}{m_{cat}}\sum_{p=1}^{m_{cat}} (I(x_p, c_p)), if\ x_p\ is\ categorical \end{cases}$	MAD Distance (MADD), L1 distance between the factual point and the counterfactual point divided by the feature's MAD, which is the Mean Absolute Deviation for the feature.
MD	$\sqrt{(\mathbf{x} - \mathbf{u}) \cdot \mathbf{C}^{-1} \cdot (\mathbf{x} - \mathbf{u})}$	Mahalanobis Distance (MD) of the counterfactual point and the problem's dataset. Where $\mathbf{u}$ is the vector with the mean values of features and $\mathbf{C^{-1}}$ is the inverse covariance matrix of the feature.
CT	$t_{final} - t_{init}$	Counterfactual generation Time (CT), time to generate a counterfactual in seconds.

3.3. Dataset Realistic Constraints

For each dataset, one or more realistic constraints are defined to evaluate how the counterfactual explanation generation algorithms behave.

For categorical features with more than two possible values, the one-hot key encoding constraint is present, which imposes two rules: The values must be 0.0 or 1.0 (univariate constraint) and it must have just one "activated" entry (multivariate constraint). Categorical features that only have two values (binary features) only have the univariate constraint to check if values are realistic. For numerical features, we apply a naive univariate constraint which checks if the counterfactual feature is inside the minimum and maximum values of the feature. Therefore, values outside this range are considered unrealistic in our analysis, since those higher (or lower) values do not have supporting evidence in the dataset. The one-hot encoding features are applied for all categorical-only and mixed datasets, while the continuous feature range constraints are used for all numerical-only and mixed datasets.

Additionally, we impose three extra sets of general constraints mentioned before (all multivariate constraints) that relate to specific feature associations found in datasets. The first set is a custom constraint related to the Internet Adv. dataset, which has three features that correspond to *height*, *width* and *ratio* (*width*/*height*); therefore, the constraint verifies if the relationship between features ($ratio = width/height$) is maintained. The second set of custom constraints was created for the PBC dataset, which checks the integrity of five features (*area*, *eccen*, *p_black*, *p_and*, *mean_tr*) that are derived from nonlinear association of five features (*height*, *length*, *area*, *blackpix*, *wb_trans*). The last set of constraints refers to the BCW dataset, which verifies that one feature (*area*) relates to another feature

(*radius*), following the equation $area = \pi(radius)^2$. A table with all information about the constraints can be found in the Appendix A.

3.4. Algorithm Comparison

The benchmarking framework performance ranking follows a similar statistical analysis as proposed by Demsar [78] to compare the performance of different models in multiple datasets. For each factual point and metric evaluated, we calculate the ranking of every algorithm as r_i^j, where j is the jth algorithm tested, and i is the ith factual row tested. Thus, the best algorithm will receive ranking, one, the second best ranking, two, and so on. When different algorithms have the same performance, an average ranking is assigned $\bar{r}_i^{j_1,j_L} = \frac{1}{L}\sum_{j=j_1}^{j_L} r_j^i$ to the L algorithms that obtained the same result. The average ranking ($\bar{R}_j$) of algorithm j over all (Q) factuals is then obtained per dataset: $\bar{R}_j = \frac{1}{Q}\sum_{i=1}^{Q} r_i^j$. The non-parametric Friedman test [79,80] is used to evaluate the hypothesis that all datasets perform the same. If the null hypothesis (all tests perform the same) is rejected, we proceed with a posthoc Nemenyi test [81] to check which model(s) performed better than others.

4. Results

The score ranking results of each algorithm in different dataset groupings (total, numerical, categorical, mixed) are presented in Tables 4 and 5. The ranking analysis shows some discrepancies with numerical results; for example, an algorithm can have a better ranking in sparsity while not having the best numerical mean sparsity. The reason for this apparent paradox is the difference in methodology to calculate the rankings and the numerical results. When calculating the algorithm's numerical results, for counterfactuals that do not need to be realistic, we only consider points that are valid results, while for realistic counterfactuals we only select points that, besides being valid points, are inside the realistic space. On the other hand, for the ranking calculation, we consider all results, including invalid and non realistic results (which are classified as being the worst ranking possible). This difference in calculation leads to a situation that is illustrated in Figure 3, where a better ranking algorithm can have a worse average result if compared to another algorithm.

In Table 4, we evaluate all counterfactual results, regardless of whether they are realistic or not. Considering all datasets, we find that CADEX (CA) achieves the best coverage, finding counterfactuals for 3549 of the 3925 (90%) tested factual instances, which is statistically significantly better than all other algorithms. Analyzing sparsity, we have a statistical tie between two algorithms, SYNAS (SY) and SEDC (SE), where the latter has a slightly better ranking with an average sparsity of 0.93 while the former has a sparsity of 0.87. This means that, for SYNAS, on average, it must change 7% of features, while, for SEDC, it must change 13% of features. For distance metrics, GrowingSpheres (GS) presents statistically better results for all rankings: for the L2 metric, it has as mean distance 0.75, which is about 5.7 times shorter than the second best ranking (CA with a mean L2 of 4.3). For MADD, a similar situation happens, where GS is about 2.7 times shorter than the second best ranking result, ALIBICNOGRAD (ALN). MD distance follows the two other distances trend, where GS has a mean distance value of 0.67, while CA (the second best ranking) has 4.35 (about 6.5 times larger).

We find different rankings if we consider specific data subsets: for categorical datasets, for instance, we do not have CA as being statistically best in coverage anymore, as it is able to find 1309 counterfactuals for a total of 1327 factual instances (about 98.6%) while the best coverage is achieved by GS, which finds counterfactuals for 1322 of them (about 99.6%). However, the better numerical result for GS is not sufficient to lead to a significant difference as compared to CA, as both are statistically tied. We also have differences in sparsity, where there is a statistical draw between three algorithms: LORE (LO), GS, and SY. The numerically best in sparsity (LO) has a mean sparsity of 0.9, while GS and SY

obtain values 0.59 and 0.92, respectively. The reader may notice that SY has actually the best numerical result in sparsity, a discrepancy that was already explained earlier in this section and relates to the share of realistic counterfactuals generated. For distance metrics, GS remains the statistically best for two metrics: L2, which is the best having an L2 mean of 0.48, while the second best (LO) has 1.51 (about 3 times larger), and MD with a mean equal to 0.22, which is almost 3.5 times better than the second best (LO).

For numerical datasets, CA is statistically the best in coverage, producing 1443 counterfactuals for a total of 1598 factual instances tested (about 90.3%), which is significantly more than the second best algorithm (GS) that found 201 less (coverage of about 77.7%). For sparsity, SE is the solo statistical best algorithm with an average of 0.77; however, the second place (SY) has a best numerical average (0.88), but it finds 272 less counterfactual results than SE (hence explaining the lower ranking). For distance metrics, we have a less clear advantage for GS, which is among the best of two metrics, L2 and MD, but it is not the best for MAD. For L2, the two best algorithms are CA, with a mean L2 of 2.17, and GS, whose average is 1.15. In this case, CA has a slightly better ranking and worse numerical mean result for two reasons—first due to the higher number of realistic counterfactuals and second due to the larger variation in results. Regarding the CA variation in L2, we see this by the larger standard deviation (equal to 2.62), while GS is much more stable (0.93). When evaluating MAD, now we see ALIBIC (AL) and ALN as being the statistically best performing algorithms, where the latter (mean MADD equal to 5.7×10^5 is slightly better than the former (mean MADD, which is 5.9×10^5. For MD, we have the same situation that happens to L2, with CA having a slightly better ranking and statistically tied with GS, but CA has a worse mean MD (5.49) if compared to GS (1.37).

Table 4. Average rankings for dataset groupings. White bold ranking numbers highlight the best numerical result for each score and the gray cells show the algorithms that, statistically, can be considered the best. The results are rounded to one decimal place; then, in some cases, the best ranking can have the same numerical results as other, worse, scores. **AL**: ALIBIC, **ALN**: ALIBICNOGRAD, **CA**: CADEX, **DI**: DiCE, **GS**: Growing Spheres, **LO**: LORE, **MA**: MACE, **ML**: ML-Explain, **SE**: SEDC, **SY**: SynAS.

		AL	ALN	CA	DI	GS	LO	MA	ML	SE	SY
NUM.	**coverage**	4.6	4.6	**3.8**	5.2	4.5	7	6.6	8	4.9	5.8
	sparsity	4.3	4.3	5.6	6.4	4.7	6.5	7.1	8.2	**3.6**	4.3
	L2	3.8	3.7	**2.9**	6.5	**2.9**	7.7	7.5	8	5.8	6.3
	MADD	**3.9**	**3.8**	4.6	6.2	4.4	7.5	6.9	8	4.5	5.3
	MD	3.8	3.8	**2.8**	6.5	**3.1**	7.6	7.6	8.1	5.8	6.1
CAT.	**coverage**	6.8	6.4	**3.3**	6.9	**3.3**	5	5.5	6.1	6.4	5.3
	sparsity	6.6	6	4.3	6.8	4.3	**4**	6	7.3	5.4	**4.3**
	L2	7	6.5	5.3	7.2	**1**	5	6.7	5.4	5.7	5.3
	MADD	6.8	6.2	**3.8**	6.9	**3.6**	4.1	5.5	7.1	6.2	4.7
	MD	7	6.4	5.2	7.1	**1**	5	6.7	5.3	5.8	5.3
MIX.	**coverage**	6.6	4.3	4.3	**3.8**	4.2	8.2	8.3	7.1	**3.9**	4.4
	sparsity	6.5	3.6	5.9	4.5	5.3	8.2	8.3	7.5	**2.6**	**2.6**
	L2	6.6	3.9	6.1	4.6	**2.2**	8.2	8.3	6.8	3.5	4.7
	MADD	7.2	3.8	5.8	3.9	3.6	8	8	7.4	**3.1**	4.1
	MD	6.6	3.9	6.1	4.4	**2.2**	8.2	8.3	6.8	3.6	5
TOTAL	**coverage**	5.9	5.1	**3.8**	5.4	4	6.6	6.6	7.1	5.2	5.3
	sparsity	5.6	4.7	5.3	6.1	4.7	6.1	7	7.7	**4**	**3.9**
	L2	5.6	4.7	4.5	6.2	**2.1**	6.9	7.4	6.8	5.2	5.5
	MADD	5.7	4.6	4.6	5.8	**3.9**	6.5	6.7	7.6	4.7	4.8
	MD	5.6	4.7	4.4	6.2	**2.2**	6.9	7.5	6.8	5.2	5.5

Table 5. Average rankings for dataset groupings, but only considering results that respect the realistic constraint rules. **AL**: ALIBIC, **ALN**: ALIBICNOGRAD, **CA**: CADEX, **DI**: DiCE, **GS**: Growing Spheres, **LO**: LORE, **MA**: MACE, **ML**: ML-Explain, **SE**: SEDC, **SY**: SynAS.

		AL	ALN	CA	DI	GS	LO	MA	ML	SE	SY
NUM.	**coverage**	4.9	4.9	5.2	4.8	4.9	6.7	6.2	7.2	4.2	6
	sparsity	4.8	4.8	6.1	5.5	5	6.5	6.5	7.2	3.3	5.3
	L2	4.3	4.3	5	5.6	4.2	7	6.6	7.2	4.7	6.2
	MAD	4.4	4.3	5.7	5.6	4.8	6.9	6.4	7.1	4.1	5.8
	MD	4.4	4.4	4.9	5.6	4.3	6.9	6.6	7.2	4.7	6.1
CAT.	**coverage**	5.8	6	2.3	6.2	7.2	4	4.8	7.2	7.2	4.3
	sparsity	5.7	5.9	3.3	6.1	7.2	3.4	5.3	7.2	7.2	3.7
	L2	5.7	5.9	3.3	6.1	7.2	3.4	5.3	7.2	7.2	3.7
	MAD	6	6.2	2.9	6.3	7	3.6	5	7	7	4.2
	MD	5.7	5.9	3.2	6.1	7.2	3.4	5.4	7.2	7.2	3.7
MIX.	**coverage**	5.9	5.5	3	5.6	6	6	6	6	5.5	5.5
	sparsity	5.9	5.5	3.2	5.6	6	5.9	6	6	5.4	5.4
	L2	5.9	5.4	3.2	5.6	6	5.9	6	6	5.4	5.5
	MAD	6	5.4	3.2	5.5	6	6	6	6	5.4	5.5
	MD	5.9	5.4	3.1	5.6	6	5.9	6	6	5.4	5.5
TOTAL	**coverage**	5.5	5.4	3.7	5.5	6	5.6	5.7	6.9	5.6	5.3
	sparsity	5.4	5.3	4.4	5.7	6	5.3	6	6.9	5.2	4.8
	L2	5.2	5.1	3.9	5.8	5.7	5.5	6	6.9	5.7	5.2
	MAD	5.3	5.2	4.1	5.8	5.8	5.6	5.8	6.8	5.4	5.2
	MD	5.2	5.2	3.9	5.8	5.7	5.5	6	6.9	5.7	5.1

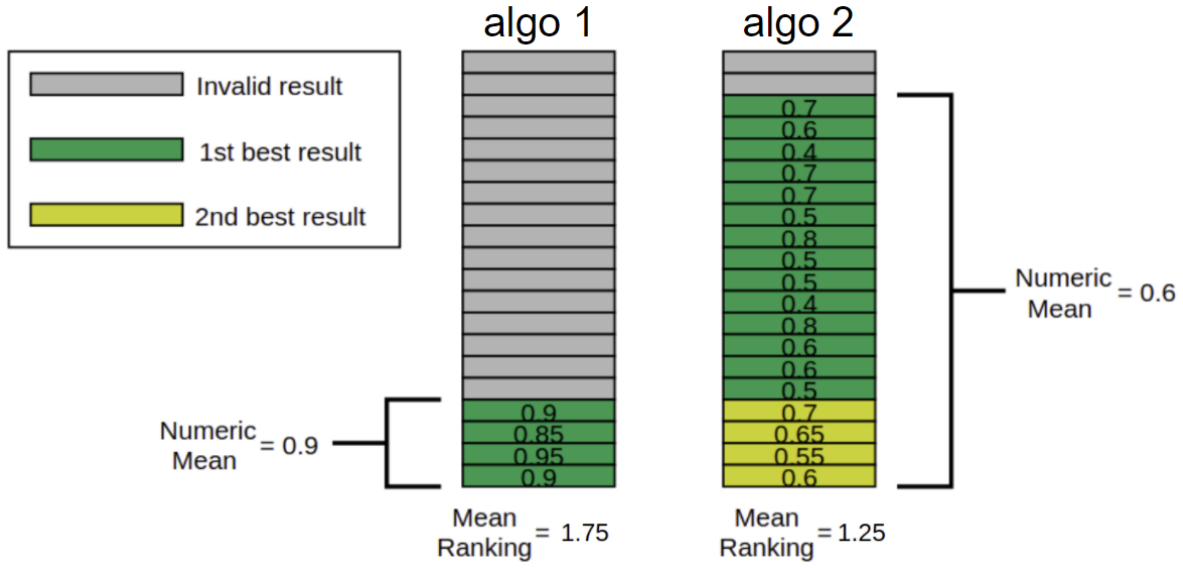

Figure 3. Example of how numeric and ranking results can have discrepancies. When calculating numerical statistical properties from results, like mean sparsity, we only consider valid (or valid and realistic) results. In the figure, this means the gray cells' (invalid results) values are not considered in the calculation of numerical scores. Thus, analyzing algorithm 1 (left), we have a mean value of 0.9 which is better, since we are considering sparsity, than algorithm 2 (on the right), which has a mean of 0.6. On the other hand, to calculate rankings, we consider all results, where invalid results are considered the worst situation possible. Then, for the algorithm 1, we only have four cells classified as first and 14 as second best and two ties—while algorithm 2 has 14 cells considered as first, 4 as second best, and two ties. Therefore, the mean ranking for algorithm 1 is 1.75, which is worse than algorithm 2 (1.25), even though the left side has a better numeric mean.

In the case of mixed datasets, CA is no longer among the best algorithms, where we verify a statistical tie between DiCE (DI) and SE. Evaluating DI, it is able to generate 896 counterfactuals from a total of 1000 (89.6%), while SE generates 863 (86.3%). For sparsity, a similar situation that occurs when evaluating all datasets happens, where SE and SYNAS are considered statistically the best, but, in this case, SE has a slightly better ranking score. In numeric terms, SE has a mean sparsity of 0.95, which is slightly worse than SY (0.98), but SEDC's improved coverage (89.6% versus 76.7%) compensates this difference in ranking numbers. For distance metrics, GS performs best for L2 and MD, while SE has best results for MAD. Then, evaluating L2, we see that GS has a mean distance of 0.59, which is more than three times lower than the second best performing algorithm, SE, which has an average L2 of 1.85. The same happens to MD, where GS has a mean value of 0.30 while SE (the second best) is about 4.8 times longer (average of 1.43). However, for MADD, we have very different results from what we verified previously with other dataset types, where SE is the statistically best performing algorithm, being superior to the second best performing algorithm (GS) 44.5% of times, inferior in 26.8% cases, and performing the same in 28.7% of factual instances.

The previous results referred to a situation where counterfactuals are simply those that could flip the classification value (valid), no matter if the counterfactual is realistic or not. Then, in Table 5, we add this additional requirement, where a counterfactual is only considered if it follows all realistic constraints the dataset has, i.e., it is valid and realistic.

Analyzing the overall results for all datasets, we already see major changes as compared to the situation where realistic counterfactuals are not required. The only similarity is that CA still performs best in terms of coverage. However, the generated number of counterfactuals dropped significantly as it only is able to produce these for 2576 of the 3925 factual instances (65% of total), 27% lower than before. The impact in coverage is even larger for the other algorithms, where, for GS, it generated 77% fewer counterfactuals, SE 56%, ALN 52%, SY 45%, DI 48%, AL 39%, MACE (MA) 23%, and LO 17%. This has a direct impact on other scores as discrepancies between the mean numerical values and ranking scores are more salient. Evaluating sparsity, for instance, the best ranking is now obtained by CA where the mean sparsity is 0.44, but, if we compare to the second best ranking result (ALN), we have a mean sparsity equal to 0.65. This situation continues for distance metrics, where, for all distances, CA has the best ranking, for L2, its mean distance is 4.54, while the second best (ALN) achieved 1.5, a value about three times better. Comparing MAD, CA has a mean distance of 4.8×10^6 and the second best (SY) has a much better score of 2.1×10^4. For MD distance, the average distance for CA is 2.89 while the second best, SY, is 0.92. Therefore, it is clear that, although CA has the best overall rankings for all metrics, this is mostly because of its superior capacity of generating counterfactuals inside the realistic space.

Considering the categorical datasets, CA remains the best or among the best. For coverage, it has a high performance, generating 1309 counterfactuals from a total of 1327; this is the same value it obtained in the analysis not considering realistic constraints. For sparsity, we have a statistical tie between CA and LO, where the former has a slightly better ranking and mean sparsity of 0.6 and the latter has a mean sparsity of 0.9. In distance metrics, CA is the solo best for MADD, where it has a mean of 0.40 while the second best (LO) achieves a value of 0.10, which is four times better. For L2, we have a statistical draw between CA that is slightly better in ranking and has the L2 mean equal to 3.25, and LO, which has a lower mean of 1.51. Similarly, for MD, we have a statistical tie between CA and LO, where the first has a slightly lower ranking and a mean MD of 1.7 while LO has 0.78, which is also a lower value. Therefore, evaluating all metrics, except coverage, we still see the impact of CA's superior capability to generate counterfactuals in obtaining a better ranking despite inferior numerical results.

Interestingly, analyzing numerical datasets, we see a major shift in ranking results as compared to the overall results. Now, CA is not among the best for any score, even for coverage. For this dataset type, coverage was superior for SE, where it produced

981 counterfactuals for a total of 1598 factual instances (61%). SE also is the best performing algorithm for sparsity, with a mean sparsity of 0.76, which is the third best numerical value, just behind SY (0.89) and LO (0.86). For distance metrics, there is no single best algorithm, in L2 distance for example, GS, ALN, and AL perform statistically the same with mean L2 distances of 1.14, 1.27, and 1.26, respectively. For MADD score, SE, ALN, and AL are the statistically best performing in the ranking analysis where their respective mean MADD are 3.6×10^5, 1.6×10^5, and 2.2×10^5. AL and ALN are still among the best in MD distance together with GS, where their mean MD distances are 1.2, 1.27, and 1.14, respectively. Therefore, for distance analysis, it is worth observing that AL and ALN were always among the best performing algorithms.

Finally, for mixed datasets, we see a trend similar to that of the overall analysis. CA performs statistically best for all scores. In coverage, for instance, it obtains the best result by far, generating counterfactual results for 597 factual cases (59.7% of total). In comparison, the second best performing algorithm (SE) could only generate 114 counterfactuals (11.4% of total). Similar to what happened in previous cases, such difference in coverage will lead to discrepancies in the ranking and numerical result values. For sparsity, CA is statistically the best with a mean sparsity of 0.57, and the second best (SE) has a much better mean sparsity of 0.97. In distance metrics, CA performs as statistically best, where, in L2, it has a mean distance of 11.25, but the second best, SE, has a lower mean of 2.19, which is about five times better. For MADD, CA has a mean distance of 1.9×10^7, which is much higher than the second best performing algorithm (ALN) that has 2.6×10^6 as a mean MADD score. For MD distance, a similar scenario happens where CA, the best in ranking, has a mean of 7.23 while the second best, SE, has a lower mean of 2.24.

Figure 4 shows how long each algorithm takes to generate a counterfactual depending on the number of features in numerical and categorical datasets. We find that, for numerical datasets, the algorithms ALN, GS, MA, and SY tend to take more time to generate results for datasets with a higher number of features. However, LO seems to have a more unstable behavior so it might depend on other factors as the used data generation methodology (genetic algorithms) and/or factors depending on the algorithmic implementation. Regarding the stability in time to generate counterfactuals, ML-Explain (ML) has the best standard deviation of just 0.74, closely followed by CA (1.19), AL (1.99), SY (2.74), SE (4.36), DI (4.76), GS (8.66), ALN (13.10), MA (150.69), and LO (227.89). Therefore, especially for algorithms like LO and MA, the counterfactual generation does not only depend on general dataset characteristics but also on the specific factual point to be explained. We find, for instance, that class imbalance can be an important characteristic that defines the counterfactual generation time, as can be seen when applying the LO algorithm to the SDD dataset (where the majority class has nine times more data than the minority): the generation of a counterfactual from majority to minority (mean 66 seconds) is 4.5 times lower than generating from minority to majority (mean 298 seconds).

In categorical datasets, AL and ALN are among the most sensitive to the number of features (see Figure 4, right chart). Although MA has higher generation times for an increased number of features, it has unexpectedly low values for the dataset with a large number of categorical features (Soybean small). As reported previously, this variation can have roots in the algorithmic implementation and specific characteristics of the dataset and each factual instance. This claim is supported by the fact that MA had the highest standard deviation for this kind of dataset, where ML has again the lowest value (0.49), followed by SE (0.51), SY (0.66), CA (0.66), LO (2.30), DI (3.71), GS (6.63), ALN (15.08), AL (15.18), and MA (215.55). It is worth observing that the generation time variation for LO is much lower for categorical features (about 99 times lower), hence being more evidence that data characteristics can highly influence the counterfactual generation time in specific algorithmic implementations.

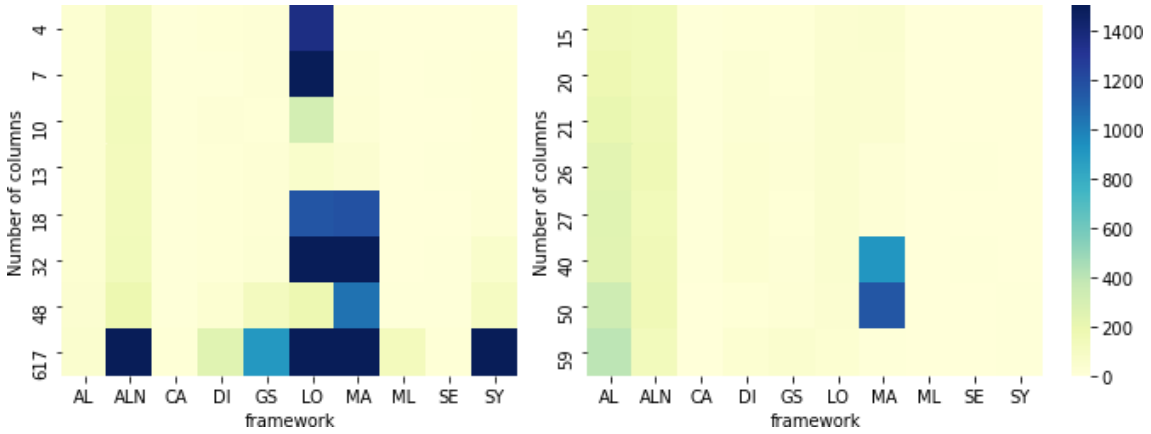

Figure 4. Heatmap with time (depending the number of columns) used by each algorithm to generate a counterfactual in datasets with only numerical features (**left**) and only categorical features (**right**). Color scale indicates the counterfactual generation time in seconds.

This benchmarking study also evaluates the algorithm stability (Figure 5), which is the generation of the same results by giving the same feature inputs over different runs. We observe that, for most algorithms—namely CA, DI, LO, ML, SE, and SY—maximum stability was achieved, meaning it always returns the same counterfactual for a given input. However, algorithms MA, GS, AL, and ALN have problems with returning the same results. For MA, it has full stability for all mixed and categorical datasets, and it also has maximum stability for all numerical datasets except for BCW, where MA could not return the same result for just one factual example. Since the instability for MA is just for one factual case over all others, it still can be considered a high stability algorithm. However, the situation is different for other algorithms. GS, for example, returns the same result for all numerical and categorical datasets, yet provided 32 different results for mixed datasets, making the overall stability of the algorithm be equal to 99%. The most unstable algorithms are AL and ALN, where the former could generate the same results for only 52% of categorical, 92% of numerical, and 24% of mixed datasets. ALN has very low reproducible results for numerical datasets, where only 18% could be generated in different runs, and it also gets low (but better than AL) stability for categorical (68%) and mixed (30%) datasets. For this analysis, it is important to highlight that AL, ALN, GS, and MA do not have the option to generate reproducible results such as random seeds.

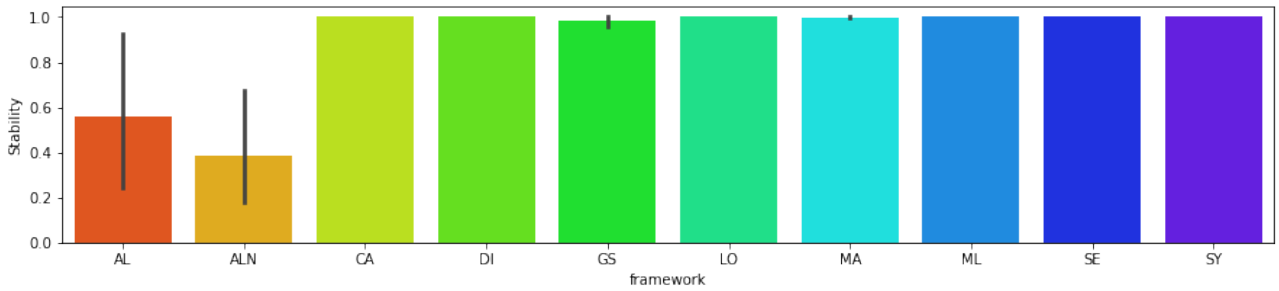

Figure 5. Mean stability of each algorithm for all datasets. The gray lines show the minimum and maximum value reached analyzing it for each dataset type (numerical, categorical, mixed). The algorithms CA, DI, LO, ML, SE, and SY have maximum stability for all datasets. While AL has minimum stability for mixed datasets and maximum stability for numerical, ALN has minimum stability for numerical datasets and maximum stability for categorical datasets, GS has minimum stability for mixed datasets and maximum stability for numerical and categorical datasets, and MA has minimum stability for numerical datasets and maximum stability for categorical and mixed datasets.

5. Discussion

One of the biggest challenges faced by most algorithms is to generate a counterfactual for every factual case. CA is the best overall performing algorithm in coverage; still, it could not generate counterfactuals for all factual instances. Although the algorithms' theoretical background may explain such low performance, we cannot exclude issues regarding the algorithmic implementation and fine-tuning of available parameters (since we always use default settings). As discussed earlier, this has a direct impact on the ranking analysis because not generating a counterfactual is considered the worst scenario case. Therefore, several algorithms actually have better scores (in sparsity and distance metrics), but they obtain a low ranking because of their low coverage. Having a higher coverage, then, could rearrange this ranking and, most importantly, gives more trust in the algorithm, since very low coverage like we observe for ML can discourage users from adopting such implementations.

We can see in Figure 6 that the coverage also depends on the dataset type, as some algorithms can perform better for a specific kind of data but not for others. This reinforces the idea that each algorithm can have preferred data characteristics that make it more efficient as compared to another algorithm. This becomes even more complex if we consider other scores, where the ranking analysis also shows that algorithms can have a better performance for some scores in specific data settings. For example, if we consider counterfactuals that are not realistic, GS is among the best algorithms for MADD in categorical datasets, but the same does not happen if we consider numerical or mixed datasets. However, this does not happen with L2 or MD distance, where GS is always among the best.

The situation becomes even more critical if we consider the percentage of results that followed realistic constraints. Figure 7 shows most algorithms are far from following such constraints, where the overall best is CA, which could generate counterfactuals for 66% of factual instances, while no other algorithm could achieve more than 33%. This low performance is particularly worse for mixed datasets, where only CA could generate a fair number of counterfactuals (60%), while others did not achieve more than 11%. The direct impact of these results is that most algorithms (excluding CA for categorical datasets) struggle with the generation of counterfactuals inside the realistic space, and practitioners may be careful when using them for this purpose.

When analyzing the coverage drop for numerical datasets, SE stands out as the best algorithm, which has a small reduction in performance (about 8% less) as compared to the non-constrained case. This happens because this algorithm highly relies on dataset characteristics, where using the mean to replace feature values takes advantage of the nature of such kind of feature where a normal distribution is frequent.

It is also important to note the impact of a very common constraint found in tabular data, the one-hot encoding process, where algorithms that do not implement any theoretical and/or algorithmic approach to handle it (GS, ML, SE) perform poorly in categorical and mixed dataset types, where such a constraint is present. This is especially noticeable for GS where it has the best ranking scores (for distance metrics) and good coverage for counterfactuals without realistic constraints, but it has a high performance drop when such constraints are applied. This can be explained by the fact that GS (and also SE and ML) treats all feature variables as numerical continuous (then, possibly breaking the univariate constraint of being binary), and they do not keep track of the set of features each one-hot encoded category has, breaking the multivariate constraint of having just one encoded feature activated (equal to 1).

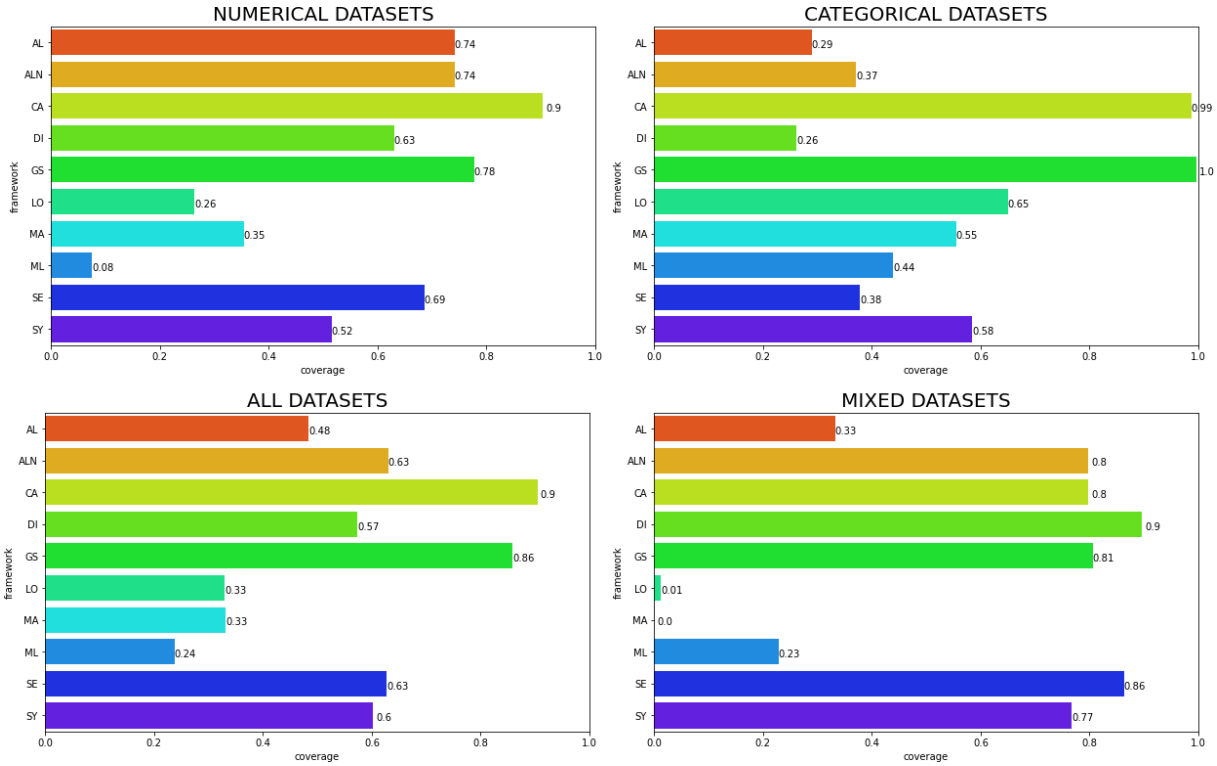

Figure 6. Proportion of factual examples that generates counterfactual explanations. Each chart represents a dataset type; the bottom left chart analyzes all datasets used in the study.

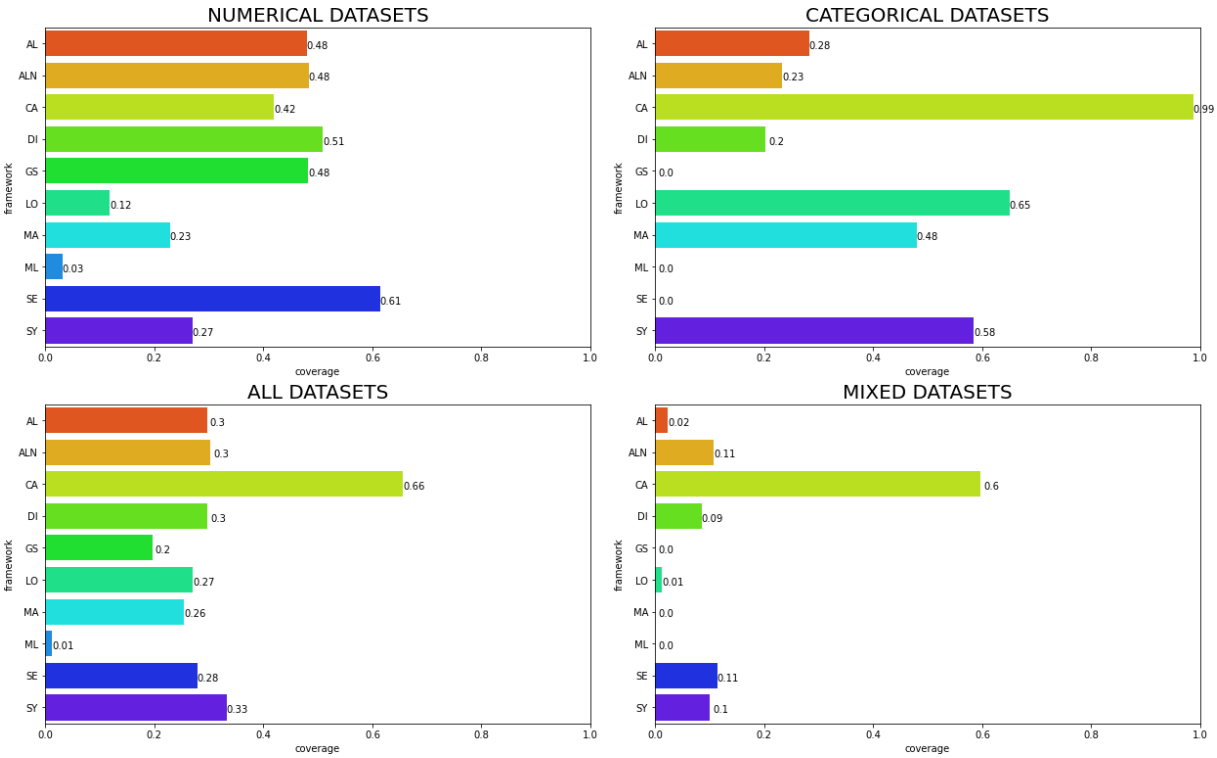

Figure 7. Coverage of algorithms that follows all realistic constraints. Each chart represents a data type, while the bottom left chart analyzes all datasets used in the study.

Additionally, in the realistic counterfactual generation, if we analyze the results for numerical datasets, we can see a slightly better performance of algorithms that used the dataset in their generation process (SE, DI, GS, AL, ALN), although this is not a clear trend as some algorithms that also used the dataset (LO, MA and SY) performed worse than CA, which does not require the dataset in its generation process. In this case, it is understandable that counterfactual characteristics may be more compatible with inner dataset constraints when using features as supporting evidence, although there are real world application cases where providing the dataset is not an option (i.e., the dataset is private or not accessible). In those cases, algorithms like CA have a clear advantage.

As previously mentioned, algorithms have different performances for different scores in distinct input data characteristics. Such high complexity means that, for specific data settings, some algorithms can outperform others that are considered the overall best. For example, CA is considered to have the best overall coverage in realistic counterfactuals, but, for a specific subset (numerical datasets), SE is actually the best in coverage and, additionally, although SE is the best in coverage for numerical datasets, it is not for L2 and MD distances. Therefore, with the objective to improve our insights on which specific situations an algorithm performs better in than the others, we use all counterfactual results to generate decision trees that highlight the best counterfactual generating approaches. These decision trees use as input features several characteristics of the model, dataset, and factual point, and it has as a target variable the best performing algorithm grouped in relevant labels such as type of search methodology, dataset access, and model internal access.The decision tree in Figure 8 is one example of such analysis where the referred input features were used to classify which algorithm's search strategy best performed in L2 distance for categorical data. Additional details about the decision tree settings can be found in the Appendix C.

In this analysis, we find some interesting patterns, confirming that selecting the best algorithmic approach can depend on several parameters that includes characteristics from the model (like number of neurons, AUC score), dataset (number of categorical or numerical columns), and/or the factual point (prediction score, balance share of factual point class).

Considering the counterfactual explanations that do not need to be realistic, we find for categorical datasets that, although GS, LO, and SY are the best ranked, if we consider a large number of categorical columns (more or equal to 45) and models that have a large number of neurons (bigger than 10.5), SE is actually the best algorithm by far (in sparsity) if the factual point class is highly unbalanced (share of the factual class larger than 0.8). It is important to notice that this observation involved characteristics from the dataset (number of categorical columns), model (number of neurons) and factual point (factual point class share). This highlights that, although there are overall best algorithms, specific settings can achieve better performance with other implementations.

Bearing that in mind, it is proposed that such decision trees can be used to suggest to end-users which algorithm they should use for their specific problem. Such implementation can output results like the one of Figure 9, where a certain performance value for each algorithm in every metric is provided. The user can then select the one that best suits their requirements. Notwithstanding, the decision tree suggestions do not reduce the importance of the overall ranking, which may be better suited both to end-users that are not sure of which particular setting they will work with and to research that wants to compare the performance of their algorithms.

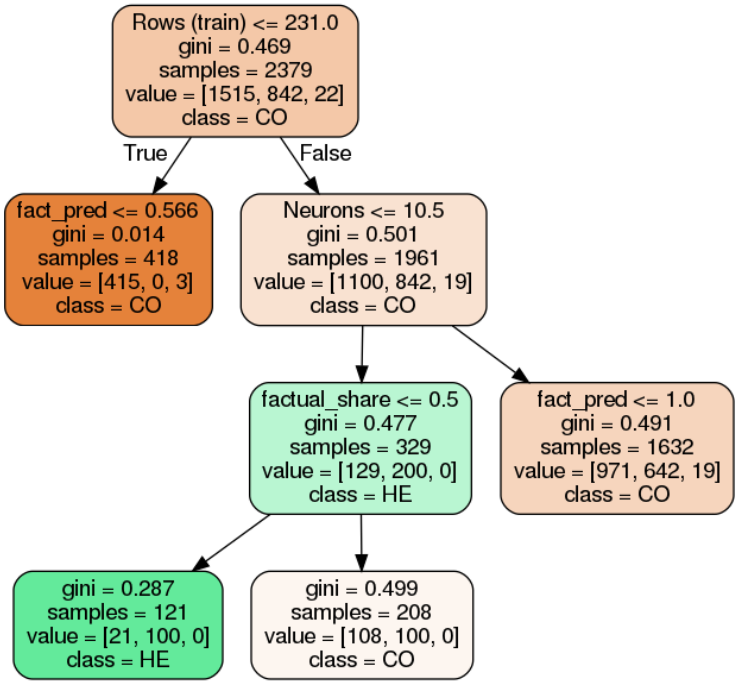

Figure 8. Decision tree to decide which algorithm implementation pattern performs best in realistic counterfactual generation using categorical data. In this case, this decision tree evaluates the best performing in L2 distance. On top of each node, the splitting condition is shown, which includes the following information: Rows (train)—number of rows in the training set, fact_pred-factual instance prediction score, Neurons—the number of neurons of the NN model, factual_share-the datset percentage of the factual point class. In second on the node, there is the Gini coefficient and, below it, the number of total samples analyzed. As the penultimate, we see the values for each of the three classes: first, it is CO, which represents convex optimization based methods (AL, ALN, CA, DI, ML, SY), in second, there is HE, which represents the heuristic based methods (GS, LO, SE) and, finally, the SS based method (MA). Last in the node, there is an indication of which class is the majority, hence which type performs best.

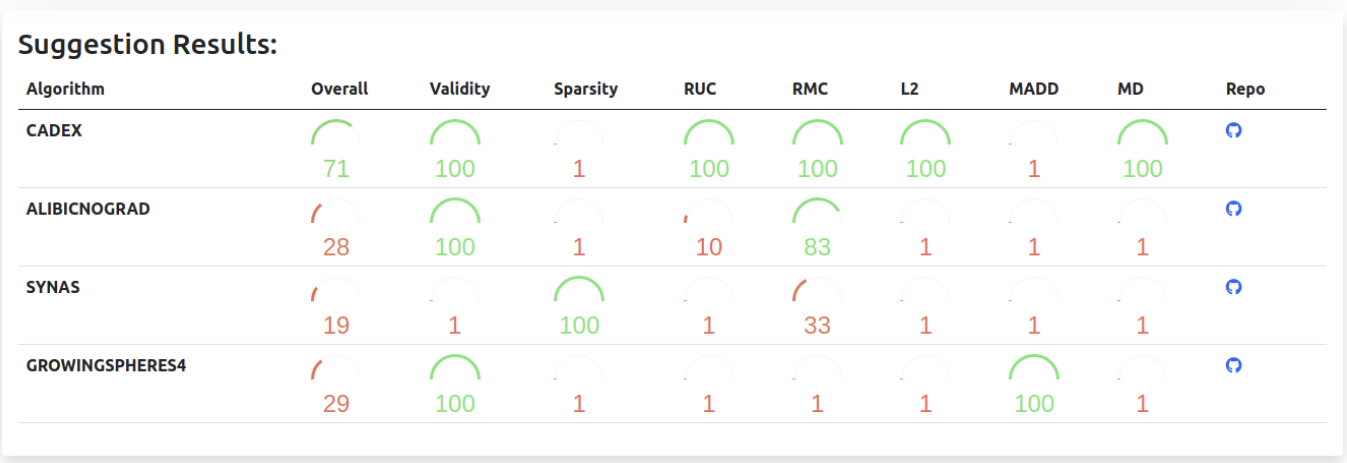

Figure 9. Proposed user interface to the suggestion system which tells the best performing algorithms for a specific dataset setting. The scores are calculated by using decision trees and reflect how well (compared to the others) the algorithm performed.

This kind of analysis using decision trees can be expanded to evaluate how implementation choices, such as search strategy, model access, and data usage affect scores in different types of datasets. For example, evaluating realistic counterfactuals and categorical

datasets, we obtain the decision tree presented in Figure 8, where the first split node is related to the number of instances the dataset has, and it totally separates heuristic rules from convex optimization methods (and, to a minor extent, SS). This can be related to how the heuristic rules work, where the dataset usage is not just to enhance loss term parameters like in convex optimization, but they actively rely on evidence from the dataset to generate counterfactuals. Therefore, not having many instances can lead to substandard performance. Although Figure 8 just shows the decision tree for L2 score, the same behavior is observed for MAD, MD, sparsity, and coverage scores.

Another important observation obtained from the higher performance achieved by CA, especially on realistic counterfactuals, is the fact it does not have any additional loss term applied (as we only used the default settings) and no access to the dataset. We envision that there is room for improvements in generating counterfactuals for all kinds of data and those that follow realistic constraints, and the already mentioned theoretical and/or algorithmic implementation may be further improved to achieve such performance. We also highlight that the characteristic of being a realistic counterfactual is highly desirable since results lacking this property only provide a mathematical justification rather than a logical reasoning for the factual instance classification. Therefore, as most algorithms underperform on this condition, this should be a goal to pursue.

Two additional properties were evaluated by this study, stability and time, in order to generate the counterfactual. Although they are not directly associated with the counterfactual quality, they may be important for real-world applications where such properties can have an important weight. For stability, in some circumstances, it might be a required characteristic [33], as the user may not want to receive different results in different runs (for the same input). Although it can be argued that such instability brings more diversity to explanations, this should not be done in an uncontrolled way, hence the use of strategies like providing a random seed setting could be used to circumvent this. This is notably critical for AL and ALN, which presented very low stability as compared to other algorithms.

In terms of time requirements, some users may have limited time to give an answer (real-time applications) or it can also have resource limitations, where large processing times may lead to an incompatibility with large scale applications. Additionally, it must be noted that we did not use very large datasets, which are becoming common in real-world scenarios. Therefore, the time to generate a counterfactual must be a common concern, and those algorithms that had poor performance (like AL, ALN, MA, LO, GS and SY) may face challenges in adoption for those specific cases.

Taking all the previous observations into account, we conclude, today, that there is no single 'silver bullet' for counterfactual generation. End-users may evaluate their specific requirements based on metrics like the ones we presented in this article to select which algorithmic implementation will mostly correspond to their expectation. To help in this task, Table 6 provides general guidance on when different algorithms seem to be best suited. We want to reiterate that these are overall findings, and more specific settings in data, model, and factual point may actually find other optimal algorithms.

We propose also that researchers may use this benchmarking study to compare novel approaches and test modifications of existing ones. Aiming to unveil the intrinsic complexity of several parameters and scores, the use of decision trees to suggest algorithms can also be considered a solution to the current heterogeneity in performance.

Finally, it is worth remembering again that these results represent the theories and (most importantly) algorithmic implementations in a specific time; then, both theory and algorithms can be updated, leading to different results that we presented here. However, our intention is also to give a starting point for the counterfactual explanation generators benchmarking, proposing a process pipeline, required theoretical foundations, scoring metrics, and possible questions that will lead to the refinement of the area and possible broader adoption.

Table 6. General properties, and context in which each counterfactual generation algorithm seems to be the most suited. This table includes general instructions and, as explained in the text, may vary depending on the model, dataset, and factual point properties.

Algorithm	Properties
ALIBIC	• Model specific: Differentiable and using TensorFlow/Keras • Low distance metrics (L2, MADD and MD) for realistic numerical counterfactuals
ALIBICNOGRAD	• Model agnostic • Similar performance as ALIBIC
CADEX	• Model specific: Differentiable and using TensorFlow/Keras • High coverage for realistic and non-realistic counterfactuals • Low counterfactual generation time • Low variation in counterfactual generation time for different datasets • High stability
DiCE	• Model specific: Differentiable and using TensorFlow/Keras or Pytorch (newest version includes model agnostic methodologies) • High coverage for mixed features datasets in non-realistic settings • High stability
GROWINGSPHERES	• Model agnostic • High coverage when non-realistic counterfactuals are needed • Medium coverage for numerical datasets that need to be realistic • Low distance metrics (L2, MADD and MD) for non-realistic counterfactuals • Low distance metrics (L2, MADD and MD) for realistic counterfactuals in numerical datasets • High stability
LORE	• Model agnostic • Good coverage for categorical datasets in realistic settings • Low distance metrics (L2, MADD and MD) for categorical datasets that need to be realistic • High sparsity for categorical datasets in realistic and not-realistic settings • High stability
MACE	• Model specific: Must be a Decision Tree, Multilayer Perceptron, Random Forest or Logistic Regression • High stability
MLEXPLAIN	• Model specific: Differentiable and using PyTorch • Low counterfactual generation time • High stability
SEDC	• Model agnostic • High sparsity for non-realistic counterfactuals • High coverage for realistic numerical datasets • High sparsity for realistic numerical datasets • Low MADD distance score for numerical datasets • Low counterfactual generation time • High stability
SYNAS	• Model specific: Differentiable and using TensorFlow/Keras • High sparsity for non-realistic counterfactuals • Low counterfactual generation time • High stability

6. Conclusions

Benchmarking studies have fundamental value in scientific research, for example, in the case of image classification, the ImageNet Challenge [82] allowed us to test and compare novel methodologies that lead to advancements in the area. In the case of Counterfactual Explanations, a very recent field in XAI, our benchmarking framework aims to be beneficial in several forms. First, it allows those who are building new theoretical and/or algorithmic approaches to test their implementations more quickly and correct possible bugs or flaws in theory. Second, it provides a common ground to make an extensive analysis towards other implementations, removing any possible difference on data treatment, model de-

sign, and scoring metric that might benefit one implementation over another. Third, the benchmarking study can reveal which algorithms perform best in specific conditions and, ultimately, can lead to finding which specific implementation strategies work for each case. Furthermore, as a side effect of how the benchmarking framework was built, the standardization of the counterfactual algorithm implementation, in terms of required variables and operation steps, can also benefit both researchers and end-users since this often has a wide variation among the algorithms, making their understanding and application more difficult. Similarly, the use of objective scores presented in this work can also create a pattern both in nomenclature and evaluation that are often diverse in this area. From a future perspective, besides the obvious addition of new algorithmic implementations, the refinement of constraints definition made by domain experts will have a positive impact on the results of this benchmarking study as it will enhance the value of realistic constraints (RUC, RMC).In addition, the consideration of multiple counterfactuals and how to evaluate them can contribute to specific cases where variation of explanations are needed. In the same way, despite the importance of artificial neural networks, the focus of this benchmark, the addition of new model types (in particular, black box models) will give a broader compatibility to algorithms and reveal new insights on them.

Author Contributions: Conceptualization, D.M. and R.M.B.d.O.; methodology, R.M.B.d.O.; software, R.M.B.d.O.; validation, D.M., R.M.B.d.O.; formal analysis, R.M.B.d.O.; investigation, R.M.B.d.O.; resources, D.M.; data curation, R.M.B.d.O.; writing—original draft preparation, R.M.B.d.O.; writing—review and editing, D.M.; visualization, R.M.B.d.O.; supervision, D.M.; project administration, D.M.; funding acquisition, D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Flanders AI Impulse Program ("Onderzoeksprogramma Artificiele Intelligentie (AI) Vlaanderen").

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The benchmark algorithm and additional data from results can be found in https://github.com/ADMAntwerp/CounterfactualBenchmark (accessed on 4 August 2021).

Acknowledgments: The authors sincerely thank Google Cloud for providing resources to run the benchmarks on their cloud platform.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Model Specification

A general overview of models is presented in Section 2.3. Below, the full details for each dataset are presented. Parameters that are not explicitly indicated must be considered the default for the package. For all models, the following configuration is always the same:

- Number of hidden layers: 1
- Number of output classes: 2
- Activation function (hidden layer): ReLU
- Activation function (output layer): Softmax
- Optimizer: RMSprop
- Loss function: Categorical Crossentropy

Table A1. Overall data for each dataset and model. fac0 and fac1 are the factual datasets with original class 0 and 1, respectively. The % Maj. is the share of rows of the majority class. Col is the number of columns, Neu is the selected model number of neurons in the single hidden layer, Epc is the number of epochs used in training, and LR is the learning rate.

	Rows								AUC			Accuracy				
Dataset	**Total**	**Fac0**	**Fac1**	**% Maj.**	**Col**	**Neu**	**Ep**	**LR**	**Train**	**Valid**	**Test**	**Train**	**Valid**	**Test**	**Fac0**	**Fac1**
Adult	32,561	100	100	0.76	107	215	50	0.0001	0.92	0.91	0.91	0.87	0.86	0.85	0.72	0.87
BCW	198	41	100	0.76	32	65	100	0.001	1	0.85	0.77	0.99	0.88	0.74	0.88	0.9
BalanceScale	625	100	100	0.54	20	16	50	0.01	1	1	1	1	0.99	0.98	0.98	0.99
CMSC	540	37	100	0.91	18	22	100	0.001	1	0.98	0.79	1	0.95	0.94	0.97	0.95
CarEvaluation	1728	100	100	0.70	21	17	50	0.01	1	1	1	1	1	0.99	0.98	1
Chess	28,056	100	100	0.90	40	64	500	0.01	1	1	1	1	1	1	1	1
DefaultOfCCC	30,000	100	100	0.78	90	72	50	0.0001	0.79	0.77	0.79	0.83	0.82	0.83	0.68	0.84
Ecoli	336	100	100	0.57	7	15	50	0.001	0.99	1	1	0.98	0.94	0.97	0.95	0.96
HayesRoth	132	96	36	0.61	15	31	500	0.0001	0.95	0.96	0.94	0.87	0.86	0.88	0.83	0.97
ISOLET	7797	100	100	0.96	617	247	500	0.001	1	1	1	1	1	1	1	1
InternetAdv	2359	100	100	0.84	1558	1246	50	0.0001	1	0.98	0.99	0.99	0.97	0.97	0.96	0.96
Iris	150	97	53	0.67	4	3	50	0.01	1	1	1	1	0.93	0.97	1	0.94
Lymphography	148	69	79	0.55	50	40	100	0.001	1	0.92	0.94	1	0.81	0.86	0.91	0.95
MagicGT	19,020	100	100	0.65	10	12	500	0.001	0.94	0.92	0.93	0.88	0.86	0.87	0.88	0.89
Nursery	12,960	100	100	0.67	26	10	50	0.01	1	1	1	1	1	1	1	1
PBC	5473	100	100	0.90	10	12	100	0.01	1	0.99	0.99	0.98	0.97	0.98	0.91	0.99
SDD	58,509	100	100	0.91	48	19	500	0.001	1	1	1	1	1	1	1	1
SoybeanSmall	47	30	17	0.64	59	23	50	0.01	1	1	1	1	1	1	1	1
StatlogGC	1000	100	100	0.70	59	119	100	0.0001	0.91	0.83	0.79	0.85	0.75	0.76	0.63	0.85
StudentPerf	649	100	100	0.54	43	17	100	0.0001	0.82	0.88	0.71	0.75	0.76	0.66	0.74	0.66
TicTacToe	958	100	100	0.65	27	11	500	0.01	1	1	1	1	0.98	0.99	0.98	1
Wine	178	100	70	0.60	13	5	50	0.01	1	1	1	1	0.97	1	0.99	1

Table A2. Description of constraint types for each dataset. For univariate constraints, Feature Range sets a continuous range where a variable can be changed, Feature Binary allows some features to assume only 0.0 or 1.0. For multivariate constraints, the Second Order describe one or more features that have a second order degree relationship between them, and OHE describes the activation rule of one-hot encoding.

Dataset	**Univariate Constraint**	**Multivariate Constraint**
Balance Scale	Feature Binary	OHE
Car Evaluation	Feature Binary	OHE
Hayes–Roth	Feature Binary	OHE
Chess	Feature Binary	OHE
Lymphography	Feature Binary	OHE
Nursery	Feature Binary	OHE
Soybean (small)	Feature Binary	OHE
Tic-Tac-Toe	Feature Binary	OHE
BCW	Feature Range	Second Order
Ecoli	Feature Range	
Iris	Feature Range	
ISOLET	Feature Range	
SDD	Feature Range	
PBC	Feature Range	Second Order
CMSC	Feature Range	
MAGIC GT	Feature Range	
Wine	Feature Range	

Table A2. *Cont.*

Dataset	Univariate Constraint	Multivariate Constraint
Default of CCC	Feature Range, Feature Binary	OHE
Student Perf.	Feature Range, Feature Binary	OHE
Adult	Feature Range, Feature Binary	OHE
Internet Adv.	Feature Range, Feature Binary	OHE, Second Order
Statlog–GC	Feature Range, Feature Binary	OHE

Appendix B. Hardware Used for Experiments

For the experiments, we were granted resources by Google to use their cloud computing environment. We used (for all datasets, except InternetAdv) N1 machine type, with 52 cores in a CPU platform based on Intel Skylake or later, with 195 GB of memory, 400 GB of SSD storage, and OS Ubuntu 18.04. For InternetAdv, N2D machine type was used, with 48 cores of AMD Rome or later CPUs, with 384 GB of memory, 400 GB of SSD storage, and OS Ubuntu 18.04.

Appendix C. Decision Tree Analysis

The decisions trees were produced using the Python package Scikit-learn [83] using a fixed random state equal to 0 and maximum depth of 3; all other parameters were set as default. The used data has eight features that are related to model, dataset, and factual point characteristics:

- Model features:
  - (1) **neurons**—Number of neurons the model has
  - (2) **auc_test**—AUC score of the model in the test set
- Dataset features:
  - (3) **rows_train**—Number of rows in the train dataset.
  - (4) **column_numerical**—Number of numerical columns in the dataset
  - (5) **columns_categorical**—Number of categorical columns in the dataset
- Factual Point features:
  - (6) **misclassified**—Tells if the original factual class was misclassified (1 if yes, 0 if not) by the model.
  - (7) **factual_prediction**—Prediction probability for the factual class, ranging from (0.5 to 1.0).
  - (8) **factual_share**—Percentage of rows of the factual class in the dataset.

As target feature, we selected the best performing algorithm for the specific factual data point. In the case where more than one algorithm tie as best, we created additional rows having the same input features but with different targets representing the best algorithms.

For this analysis, we do not focus on getting highy accurate decision trees, we focus on the Gini coefficient which low values (close to 0) showing a preference for a specific algorithm, while numbers close to 1 show that there is no preferable algorithm for the split conditions.

As there are several (more than 200) decision trees generated by this benchmark, you can find all of them in the official CF benchmark repository link.

References

1. Lee, I.; Shin, Y.J. Machine learning for enterprises: Applications, algorithm selection, and challenges. *Bus. Horizons* **2020**, *63*, 157–170. [CrossRef]
2. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. In Proceedings of the Advances in Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 4765–4774.
3. Martens, D.; Baesens, B.; Van Gestel, T.; Vanthienen, J. Comprehensible credit scoring models using rule extraction from support vector machines. *Eur. J. Oper. Res.* **2007**, *183*, 1466–1476. [CrossRef]

4. Kayande, U.; De Bruyn, A.; Lilien, G.L.; Rangaswamy, A.; Van Bruggen, G.H. How incorporating feedback mechanisms in a DSS affects DSS evaluations. *Inf. Syst. Res.* **2009**, *20*, 527–546. [CrossRef]
5. Umanath, N.S.; Vessey, I. Multiattribute data presentation and human judgment: A cognitive fit perspective. *Decis. Sci.* **1994**, *25*, 795–824. [CrossRef]
6. Limayem, M.; DeSanctis, G. Providing decisional guidance for multicriteria decision-making in groups. *Inf. Syst. Res.* **2000**, *11*, 386–401. [CrossRef]
7. Lilien, G.L.; Rangaswamy, A.; Van Bruggen, G.H.; Starke, K. DSS effectiveness in marketing resource allocation decisions: Reality vs. perception. *Inf. Syst. Res.* **2004**, *15*, 216–235. [CrossRef]
8. Arnold, V.; Clark, N.; Collier, P.A.; Leech, S.A.; Sutton, S.G. The differential use and effect of knowledge-based system explanations in novice and expert judgment decisions. *Mis Q.* **2006**, 30, 79–97. [CrossRef]
9. Angelov, P.P.; Gu, X. Toward anthropomorphic machine learning. *Computer* **2018**, *51*, 18–27. [CrossRef]
10. Verma, S.; Rubin, J. Fairness Definitions Explained. In *Proceedings of the International Workshop on Software Fairness*; FairWare '18; Association for Computing Machinery: New York, NY, USA, 2018; pp. 1–7. [CrossRef]
11. Dunkelau, J.; Leuschel, M. Fairness-Aware Machine Learning. Available online: https://www3.hhu.de/stups/downloads/pdf/fairness-survey.pdf (accessed on 4 August 2021).
12. Soares, E.; Angelov, P. Fair-by-design explainable models for prediction of recidivism. *arXiv* **2019**, arXiv:1910.02043.
13. Dodge, J.; Liao, Q.V.; Zhang, Y.; Bellamy, R.K.; Dugan, C. Explaining models: An empirical study of how explanations impact fairness judgment. In Proceedings of the 24th International Conference on Intelligent User Interfaces (ACM UIU 2019), Los Angeles, CA, USA, 16–20 March 2019; pp. 275–285.
14. Škrjanc, I.; Iglesias, J.A.; Sanchis, A.; Leite, D.; Lughofer, E.; Gomide, F. Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: A survey. *Inf. Sci.* **2019**, *490*, 344–368. [CrossRef]
15. Linardatos, P.; Papastefanopoulos, V.; Kotsiantis, S. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy* **2020**, *23*, 18. [CrossRef] [PubMed]
16. Samek, W.; Wiegand, T.; Müller, K.R. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv* **2017**, arXiv:1708.08296.
17. Gu, X.; Angelov, P.P. Highly interpretable hierarchical deep rule-based classifier. *Appl. Soft Comput.* **2020**, *92*, 106310. [CrossRef]
18. Hatwell, J.; Gaber, M.M.; Azad, R. gbt-hips: Explaining the classifications of gradient boosted tree ensembles. *Appl. Sci.* **2021**, *11*, 2511. [CrossRef]
19. Petkovic, D.; Altman, R.; Wong, M.; Vigil, A. Improving the explainability of Random Forest classifier–user centered approach. In Proceedings of the Pacific Symposium on Biocomputing 2018 (PBS 2018), Big Island, HI, USA, 3–7 January 2018.
20. Barbella, D.; Benzaid, S.; Christensen, J.M.; Jackson, B.; Qin, X.V.; Musicant, D.R. *Understanding Support Vector Machine Classifications via a Recommender System-Like Approach*; DMIN: Las Vegas, NV, USA, 2009; pp. 305–311.
21. Kute, D.V.; Pradhan, B.; Shukla, N.; Alamri, A. Deep learning and explainable artificial intelligence techniques applied for detecting money laundering—A critical review. *IEEE Access* **2021**, 9, 82300–82317. [CrossRef]
22. Demajo, L.M.; Vella, V.; Dingli, A. Explainable ai for interpretable credit scoring. *arXiv* **2020**, arXiv:2012.03749.
23. Porto, R.; Molina, J.M.; Berlanga, A.; Patricio, M.A. Minimum Relevant Features to Obtain Explainable Systems for Predicting Cardiovascular Disease Using the Statlog Data Set. *Appl. Sci.* **2021**, *11*, 1285. [CrossRef]
24. Gulum, M.A.; Trombley, C.M.; Kantardzic, M. A Review of Explainable Deep Learning Cancer Detection Models in Medical Imaging. *Appl. Sci.* **2021**, *11*, 4573. [CrossRef]
25. Soares, E.; Angelov, P.; Filev, D.; Costa, B.; Castro, M.; Nageshrao, S. Explainable density-based approach for self-driving actions classification. In Proceedings of the 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019; pp. 469–474.
26. Lorente, M.P.S.; Lopez, E.M.; Florez, L.A.; Espino, A.L.; Martínez, J.A.I.; de Miguel, A.S. Explaining Deep Learning-Based Driver Models. *Appl. Sci.* **2021**, *11*, 3321. [CrossRef]
27. Das, A.; Rad, P. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv* **2020**, arXiv:2006.11371.
28. Vilone, G.; Longo, L. Explainable artificial intelligence: A systematic review. *arXiv* **2020**, arXiv:2006.00093.
29. Bajaj, M.; Chu, L.; Xue, Z.Y.; Pei, J.; Wang, L.; Lam, P.C.H.; Zhang, Y. Robust Counterfactual Explanations on Graph Neural Networks. *arXiv* **2021**, arXiv:2107.04086.
30. Dindorf, C.; Teufl, W.; Taetz, B.; Bleser, G.; Fröhlich, M. Interpretability of input representations for gait classification in patients after total hip arthroplasty. *Sensors* **2020**, *20*, 4385. [CrossRef] [PubMed]
31. Vermeire, T.; Martens, D. Explainable image classification with evidence counterfactual. *arXiv* **2020**, arXiv:2004.07511.
32. Ramon, Y.; Martens, D.; Evgeniou, T.; Provost, F. A comparison of instance-level counterfactual explanation algorithms for behavioral and textual data: SEDC, LIME-C and SHAP-C. *Adv. Data Anal. Classif.* **2020**. [CrossRef]
33. Sokol, K.; Flach, P. Explainability fact sheets: A framework for systematic assessment of explainable approaches. In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT* '20), Barcelona, Spain, 27–30 January 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 56–67.
34. Martens, D.; Provost, F. Explaining data-driven document classifications. *Mis Q.* **2014**, *38*, 73–100. [CrossRef]

35. Byrne, R.M.J. Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), Macao, China, 10–16 August 2019; International Joint Conferences on Artificial Intelligence Organization: CA, USA, 2019; pp. 6276–6282. Available online: https://www.researchgate.net/profile/Ken-Kobayashi-4/publication/344589981_DACE_Distribution-Aware_Counterfactual_Explanation_by_Mixed-Integer_Linear_Optimization/links/5f827659a6fdccfd7b57d084/DACE-Distribution-Aware-Counterfactual-Explanation-by-Mixed-Integer-Linear-Optimization.pdf (accessed on 7 August 2021). [CrossRef]
36. Menzies, P.; Beebee, H. Counterfactual Theories of Causation. In *The Stanford Encyclopedia of Philosophy*, Winter 2020 ed.; Zalta, E.N., Ed.; Metaphysics Research Lab, Stanford University: Stanford, CA, USA, 2020.
37. Kahneman, D.; Miller, D.T. Norm theory: Comparing reality to its alternatives. *Psychol. Rev.* **1986**, *93*, 136. [CrossRef]
38. Lipton, P. Contrastive Explanation. *R. Inst. Philos. Suppl.* **1990**, *27*, 247–266. [CrossRef]
39. Binns, R.; Van Kleek, M.; Veale, M.; Lyngs, U.; Zhao, J.; Shadbolt, N. 'It's Reducing a Human Being to a Percentage' Perceptions of Justice in Algorithmic Decisions. In Proceedings of the 2018 Chi Conference on Human Factors in Computing Systems, Montréal, QC, Canada, 21–26 April 2018; pp. 1–14.
40. Wachter, S.; Mittelstadt, B.; Russell, C. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harv. JL & Tech.* **2017**, *31*, 841.
41. Ribeiro, M.T.; Singh, S.; Guestrin, C. "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, San Francisco, CA, USA, 13–17 August 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 1135–1144.
42. Fernández-Loría, C.; Provost, F.; Han, X. Explaining Data-Driven Decisions Made by AI Systems: The Counterfactual Approach. *arXiv* **2020**, arXiv:2001.07417.
43. Keane, M.T.; Kenny, E.M.; Delaney, E.; Smyth, B. If Only We Had Better Counterfactual Explanations: Five Key Deficits to Rectify in the Evaluation of Counterfactual XAI Techniques. *arXiv* **2021**, arXiv:2103.01035.
44. Karimi, A.H.; Barthe, G.; Schölkopf, B.; Valera, I. A survey of algorithmic recourse: Definitions, formulations, solutions, and prospects. *arXiv* **2020**, arXiv:2010.04050.
45. Verma, S.; Dickerson, J.; Hines, K. Counterfactual Explanations for Machine Learning: A Review. *arXiv* **2020**, arXiv:2010.10596.
46. Mothilal, R.K.; Sharma, A.; Tan, C. Explaining machine learning classifiers through diverse counterfactual explanations. In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT* '20), Barcelona, Spain, 27–30 January 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 607–617.
47. Looveren, A.V.; Klaise, J. Interpretable Counterfactual Explanations Guided by Prototypes. *arXiv* **2019**, arXiv:1907.02584.
48. Afonichkin, I. Explaining Machine Learning Models by Generating Counterfactuals. Available online: https://aaltodoc.aalto.fi/handle/123456789/39894 (accessed on 4 August 2021).
49. Ramakrishnan, G.; Lee, Y.C.; Albarghouthi, A. Synthesizing Action Sequences for Modifying Model Decisions. *arXiv* **2019**, arXiv:1910.00057.
50. Moore, J.; Hammerla, N.; Watkins, C. Explaining Deep Learning Models with Constrained Adversarial Examples. *arXiv* **2019**, arXiv:1906.10671.
51. Laugel, T.; Lesot, M.J.; Marsala, C.; Renard, X.; Detyniecki, M. Inverse Classification for Comparison-based Interpretability in Machine Learning. *arXiv* **2017**, arXiv:1712.08443.
52. Guidotti, R.; Monreale, A.; Giannotti, F.; Pedreschi, D.; Ruggieri, S.; Turini, F. Factual and Counterfactual Explanations for Black Box Decision Making. *IEEE Intell. Syst.* **2019**, *34*, 14–23. [CrossRef]
53. Karimi, A.; Barthe, G.; Balle, B.; Valera, I. Model-Agnostic Counterfactual Explanations for Consequential Decisions. In Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS 2020), Online, 26–28 August 2020; Volume 108, pp. 895–905
54. Sharma, S.; Henderson, J.; Ghosh, J. Certifai: Counterfactual explanations for robustness, transparency, interpretability, and fairness of artificial intelligence models. *arXiv* **2019**, arXiv:1905.07857.
55. White, A.; Garcez, A.d. Measurable counterfactual local explanations for any classifier. *arXiv* **2019**, arXiv:1908.03020.
56. Yousefzadeh, R. Interpreting Machine Learning Models and Application of Homotopy Methods. Ph.D. Thesis, University of Maryland, College Park, MA, USA, 2019
57. Chapman-Rounds, M.; Schulz, M.A.; Pazos, E.; Georgatzis, K. EMAP: Explanation by Minimal Adversarial Perturbation. *arXiv* **2019**, arXiv:1912.00872.
58. Mahajan, D.; Tan, C.; Sharma, A. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv* **2019**, arXiv:1912.03277.
59. Artelt, A.; Hammer, B. Efficient computation of counterfactual explanations of LVQ models. *arXiv* **2019**, arXiv:1908.00735.
60. Artelt, A.; Hammer, B. Convex Density Constraints for Computing Plausible Counterfactual Explanations. *arXiv* **2020**, arXiv:2002.04862.
61. Rathi, S. Generating counterfactual and contrastive explanations using SHAP. *arXiv* **2019**, arXiv:1906.09293.
62. Lucic, A.; Oosterhuis, H.; Haned, H.; de Rijke, M. FOCUS: Flexible Optimizable Counterfactual Explanations for Tree Ensembles *arXiv* **2019**, arXiv:1911.12199.

63. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Software. Available online: tensorflow.org (accessed on 4 August 2021).
64. Sheela, K.G.; Deepa, S.N. Review on methods to fix number of hidden neurons in neural networks. *Math. Probl. Eng.* **2013**, *2013*. [CrossRef]
65. Vujicic, T.; Matijevic, T.; Ljucovic, J.; Balota, A.; Sevarac, Z. Comparative analysis of methods for determining number of hidden neurons in artificial neural network. In Proceedings of the Central European Conference on Information and Intelligent Systems (CECIIS 2016), Varaždin, Croatia, 21–23 September 2016; University of Zagreb, Faculty of Organization and Informatics Varaždin: Varaždin, Crotatia, 2016; pp. 219–223.
66. Chen, C.S.; Lin, J.M.; Lee, C.T. Neural network for WGDOP approximation and mobile location. *Math. Probl. Eng.* **2013**, *2013*, 369694:1–369694:11. [CrossRef]
67. Wilson, R.L. Business implementation issues for neural networks. *J. Comput. Inf. Syst.* **1992**, *32*, 15–19.
68. Dua, D.; Graff, C. UCI Machine Learning Repository. Available online: http://archive.ics.uci.edu/ml (accessed on 4 August 2021).
69. Zwitter, M.; Soklic, M. Lymphography Data Set. Available online: https://archive.ics.uci.edu/ml/datasets/Lymphography (accessed on 4 August 2021).
70. Lucas, D.D.; Klein, R.; Tannahill, J.; Ivanova, D.; Brandon, S.; Domyancic, D.; Zhang, Y. Failure analysis of parameter-induced simulation crashes in climate models. *Geosci. Model Dev. Discuss.* **2013**, *6*, 585–623. [CrossRef]
71. Cortez, P.; Silva, A.M.G. Using Data Mining to Predict Secondary School Student Performance. Available online: http://www3.dsi.uminho.pt/pcortez/student.pdf (accessed on 4 August 2021).
72. Yeh, I.C.; Lien, C.H. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Syst. Appl.* **2009**, *36*, 2473–2480. [CrossRef]
73. Nazabal, A.; Olmos, P.M.; Ghahramani, Z.; Valera, I. Handling incomplete heterogeneous data using vaes. *Pattern Recognit.* **2020**, *107*, 107501:1–107501:11. [CrossRef]
74. Karimi, A.H.; Schölkopf, B.; Valera, I. Algorithmic Recourse: From Counterfactual Explanations to Interventions. *arXiv* **2020**, arXiv:2002.06278.
75. Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.* **2019**, *267*, 1–38. [CrossRef]
76. Mahalanobis, P.C. *On the Generalized Distance in Statistics*; National Institute of Science of India, Park Street: Calcutta, India, 1936.
77. Kanamori, K.; Takagi, T.; Kobayashi, K.; Arimura, H. DACE: Distribution-Aware Counterfactual Explanation by Mixed-Integer Linear Optimization. In Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, Yokohama, Japan, 11–17 July 2020; International Joint Conferences on Artificial Intelligence Organization: CA, USA, 2020; pp. 2855–2862. Available online: https://www.ijcai.org/proceedings/2019/0876.pdf (accessed on 7 August 2021).
78. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30.
79. Friedman, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J. Am. Stat. Assoc.* **1937**, *32*, 675–701. [CrossRef]
80. Friedman, M. A comparison of alternative tests of significance for the problem of m rankings. *Ann. Math. Stat.* **1940**, *11*, 86–92. [CrossRef]
81. Nemenyi, P. *Distribution-Free Multiple Comparisons*; Princeton University: Princeton, NJ, USA, 1963.
82. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis. IJCV* **2015**, *115*, 211–252. [CrossRef]
83. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.