Multi-Objective Counterfactual Explanations

Susanne Dandl(✉), Christoph Molnar, Martin Binder, and Bernd Bischl

Department of Statistics, LMU Munich, Ludwigstr. 33, 80539 Munich, Germany
susanne.dandl@stat.uni-muenchen.de

Abstract. Counterfactual explanations are one of the most popular methods to make predictions of black box machine learning models interpretable by providing explanations in the form of 'what-if scenarios'. Most current approaches optimize a collapsed, weighted sum of multiple objectives, which are naturally difficult to balance a-priori. We propose the Multi-Objective Counterfactuals (MOC) method, which translates the counterfactual search into a multi-objective optimization problem. Our approach not only returns a diverse set of counterfactuals with different trade-offs between the proposed objectives, but also maintains diversity in feature space. This enables a more detailed post-hoc analysis to facilitate better understanding and also more options for actionable user responses to change the predicted outcome. Our approach is also model-agnostic and works for numerical and categorical input features. We show the usefulness of MOC in concrete cases and compare our approach with state-of-the-art methods for counterfactual explanations.

Keywords: Interpretability · Interpretable machine learning · Counterfactual explanations · Multi-objective optimization · NSGA-II

1 Introduction

Interpretable machine learning methods have become very important in recent years to explain the behavior of black box machine learning (ML) models. A useful method for explaining *single* predictions of a model are counterfactual explanations. ML credit risk prediction is a common motivation for counterfactuals. For people whose credit applications have been rejected, it is valuable to know why they have not been accepted, either to understand the decision making process or to assess their actionable options to change the outcome. Counterfactuals provide these explanations in the form of "if these features had different values, your credit application would have been accepted". For such explanations to be plausible, they should only suggest small changes in a few features.

This work has been partially supported by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A and by the Bavarian State Ministry of Science and the Arts in the framework of the Centre Digitisation.Bavaria (ZD.B). The authors of this work take full responsibility for its content.

T. Bäck et al. (Eds.): PPSN 2020, LNCS 12269, pp. 448–469, 2020.
https://doi.org/10.1007/978-3-030-58112-1_31

Therefore, counterfactuals can be defined as close neighbors of an actual data point, but their predictions have to be sufficiently close to a (usually quite different) desired outcome. Counterfactuals explain why a certain outcome was not reached, can offer potential reasons to object against an unfair outcome and give guidance on how the desired prediction could be reached in the future [35]. Note that counterfactuals are also valuable for predictive modelers on a more technical level to investigate the pointwise robustness and the pointwise bias of their model.

2 Related Work

Counterfactuals are closely related to adversarial perturbations. These have the aim to deceive ML models instead of making the models interpretable [30]. Attribution methods such as Local Interpretable Model-agnostic Explanations (LIME) [27] and Shapley Values [22] explain a prediction by determining how much each feature contributed to it. Counterfactual explanations differ from feature attributions since they generate data points with a different, desired prediction instead of attributing a prediction to the features.

Counterfactual methods can be model-agnostic or model-specific. The latter usually exploit the internal structure of the underlying ML model, such as the trained weights of a neural network, while the former are based on general principles which work for arbitrary ML models - often by only assuming access to the prediction function of an already fitted model. Several model-agnostic counterfactual methods have been proposed [8,11,16,18,25,29,37]. Apart from Grath et al. [11], these approaches are limited to classification. Unlike the other methods, the method of Poyiadzi et al. [25] can obtain plausible counterfactuals by constructing feasible paths between data points with opposite predictions.

A model-specific approach was proposed by Wachter et al. [35], who also introduced and formalized the concept of counterfactuals in predictive modeling. Like many model-specific methods [15,20,24,28,33] their approach is limited to differentiable models. The approach of Tolomei et al. [32] generates explanations for tree-based ensemble binary classifiers. As with [35] and [20], it only returns a single counterfactual per run.

3 Contributions

In this paper, we introduce Multi-Objective Counterfactuals (MOC), which to the best of our knowledge is the first method to formalize the counterfactual search as a multi-objective optimization problem. We argue that the mathematical problem behind the search for counterfactuals should be naturally addressed as multi-objective. Most of the above methods optimize a collapsed, weighted sum of multiple objectives to find counterfactuals, which are naturally difficult to balance a-priori. They carry the risk of arbitrarily reducing the solution set to a single candidate without the option to discuss inherent trade-offs – which should be especially relevant for model interpretation that is by design very hard to precisely capture in a (single) mathematical formulation.

Compared to Wachter et al. [35], we use a distance metric for mixed feature spaces and two additional objectives: one that measures the number of feature changes to obtain sparse and therefore more interpretable counterfactuals, and one that measures the closeness to the nearest observed data points for more plausible counterfactuals. MOC returns a Pareto set of counterfactuals that represents different trade-offs between our proposed objectives, and which are constructed to be diverse in feature space. This seems preferable because changes to different features can lead to a desired counterfactual prediction[1] and it is more likely that some counterfactuals meet the (hidden) preferences of a user. A single counterfactual might even suggest a strategy that is interpretable but not actionable (e.g., 'reduce your number of pregnancies') or counterproductive in more general contexts (e.g., 'increase your age to reduce the risk of diabetes'). In addition, if multiple otherwise quite different counterfactuals suggest changes to the same feature, the user may have more confidence that the feature is an important lever to achieve the desired outcome. We refer the reader to Appendix A for two concrete examples illustrating the above.

Compared to other counterfactual methods, MOC is model-agnostic and handles classification, regression and mixed feature spaces, which furthermore increases its practical usefulness in general applications. Together with [16], our paper also includes one of the first benchmark studies that compares multiple counterfactual methods on multiple, heterogeneous datasets.

4 Methodology

[35] loosely define counterfactuals as:

> "You were denied a loan because your annual income was £30,000. If your income had been £45,000, you would have been offered a loan. Here the statement of decision is followed by a counterfactual, or statement of how the world would have to be different for a desirable outcome to occur. Multiple counterfactuals are possible, as multiple desirable outcomes can exist, and there may be several ways to achieve any of these outcomes."

We now formalize this statement by stating four objectives, which a counterfactual should adhere to. In the subsequent section we provide detailed definitions of these objectives and tie them together as a multi-objective optimization problem in order to generate a diverse set of different trade-off solutions.

4.1 Multi-Objective Counterfactuals

Definition 1 (Counterfactual Explanation). *Let $\hat{f} : \mathcal{X} \rightarrow \mathbb{R}$ be a prediction function, $\mathcal{X}$ the feature space and $Y' \subset \mathbb{R}$ a set of desired outcomes. The latter*

[1] Rashomon effect [5].

can either be a single value or an interval of values. We define a counterfactual explanation $\mathbf{x}'$ *for an observation* $\mathbf{x}^*$ *as a data point fulfilling the following: (1) its prediction* $f(\mathbf{x}')$ *is close to the desired outcome set* Y'*, (2) it is close to* $\mathbf{x}^*$ *in the* $\mathcal{X}$ *space, (3) it differs from* $\mathbf{x}^*$ *only in a few features, and (4) it is a* plausible *data point according to the probability distribution* $\mathbb{P}_{\mathcal{X}}$*. For classification models, we assume that* $\hat{f}$ *returns the probability for a user-selected class and* Y' *has to be the desired probability (range).*

This can be translated into a multi-objective minimization task:

$$\min_{\mathbf{x}} \mathbf{o}(\mathbf{x}) := \min_{\mathbf{x}} \left(o_1(\hat{f}(\mathbf{x}), Y'),\, o_2(\mathbf{x}, \mathbf{x}^*), o_3(\mathbf{x}, \mathbf{x}^*), o_4(\mathbf{x}, \mathbf{X}^{obs})\right), \tag{1}$$

with $\mathbf{o} : \mathcal{X} \to \mathbb{R}^4$ and $\mathbf{X}^{obs}$ as the observed (i.e. training) data. The first component o_1 quantifies the distance between $\hat{f}(\mathbf{x})$ and Y'. We define it as:[2]

$$o_1(\hat{f}(\mathbf{x}), Y') = \begin{cases} 0 & \text{if } \hat{f}(\mathbf{x}) \in Y' \\ \inf\limits_{y' \in Y'} |\hat{f}(\mathbf{x}) - y'| & \text{else} \end{cases}.$$

The second component o_2 quantifies the distance between $\mathbf{x}^*$ and $\mathbf{x}$ using the Gower distance to account for mixed features [10]:

$$o_2(\mathbf{x}, \mathbf{x}^*) = \frac{1}{p} \sum_{j=1}^{p} \delta_G(x_j, x_j^*) \in [0, 1]$$

with p being the number of features. The value of δ_G depends on the feature type:

$$\delta_G(x_j, x_j^*) = \begin{cases} \frac{1}{\widehat{R}_j} |x_j - x_j^*| & \text{if } x_j \text{ is numerical} \\ \mathbb{I}_{x_j \neq x_j^*} & \text{if } x_j \text{ is categorical} \end{cases}$$

with $\widehat{R}_j$ as the value range of feature j, extracted from the observed dataset.

Since the Gower distance does not take into account how many features have been changed, we introduce objective o_3, which counts the number of changed features using the L_0 norm:

$$o_3(\mathbf{x}, \mathbf{x}^*) = ||\mathbf{x} - \mathbf{x}^*||_0 = \sum_{j=1}^{p} \mathbb{I}_{x_j \neq x_j^*}.$$

The fourth objective o_4 measures the weighted average Gower distance between $\mathbf{x}$ and the k nearest observed data points $\mathbf{x}^{[1]}, ..., \mathbf{x}^{[k]} \in \mathbf{X}^{obs}$ as an empirical approximation of how likely $\mathbf{x}$ originates from the distribution of $\mathcal{X}$:

$$o_4(\mathbf{x}, \mathbf{X}^{obs}) = \sum_{i=1}^{k} w^{[i]} \frac{1}{p} \sum_{j=1}^{p} \delta_G(x_j, x_j^{[i]}) \in [0, 1] \text{ where } \sum_{i=1}^{k} w^{[i]} = 1.$$

[2] We chose the L_1 norm over the L_2 norm for a natural interpretation. Its non-differentiability is negligible for evolutionary optimization.

Throughout this paper, we set k to 1. Further procedures to increase the plausibility of the counterfactuals are integrated into the optimization algorithm and are described in Sect. 4.3.

Balancing the four objectives is difficult since the objectives contradict each other. For example, minimizing the distance between counterfactual outcome and desired outcome Y' (o_1) becomes more difficult when we require counterfactual feature values close to $\mathbf{x}^*$ (o_2 and o_3) and to the observed data (o_4).

4.2 Counterfactual Search

Our proposed method MOC uses the *Nondominated Sorting Genetic Algorithm II* (NSGA-II) [7] with modifications specific to the problem considered. First, unlike the original NSGA-II, it uses *mixed integer evolutionary strategies* (MIES) [19] to work with the mixed discrete and continuous search space. Furthermore, a different crowding distance sorting algorithm is used, and we propose some optional adjustments tailored to the counterfactual search in the upcoming section.

For MOC, each candidate is described by its feature vector (the 'genes') and the objective values of the candidates are evaluated by Eq. (1). Features of candidates are recombined and mutated with predefined probabilities – some of the control parameters of MOC. Numerical features are recombined by the simulated binary crossover recombinator [6], all other feature types by the uniform crossover recombinator [31]. Based on [19], numerical features are mutated by the scaled Gaussian mutator. Categorical features are altered by uniformly sampling from their admissible levels, while binary and logical features are simply flipped. After recombination and mutation, some feature values are randomly set to the values of $\mathbf{x}^*$ with a given (low) probability – another control parameter – to prevent all features from deviating from $\mathbf{x}^*$.

Contrary to NSGA-II, the crowding distance is computed not only in the objective space $\mathbb{R}^4$ (L_1 norm) but also in the feature space $\mathcal{X}$ (Gower distance), and the distances are summed up with equal weighting. As a result, candidates are more likely kept if they differ greatly from another candidate in their feature values although they are similar in the objective values. Diversity in $\mathcal{X}$ is desired because the chances of obtaining counterfactuals that meet the (hidden) preferences of users are higher. This approach is based on Avila et al. [2].

MOC stops if either a predefined number of generations is reached (default) or the performance no longer improves for a given number of successive generations.

4.3 Further Modifications

Initialization. Naively, we could initialize a population by uniformly sampling some feature values from their full range of possible values, while randomly setting other features to the values of $\mathbf{x}^*$ to induce sparsity. However, if a feature has a large influence on the prediction, it should be more likely that the counterfactual values differ from $\mathbf{x}^*$. The importance of a feature for an entire dataset can be measured as the standard deviation of the partial dependence plot [12]. Analogously, we propose to measure the feature importance for a single prediction with the standard deviation of the Individual Conditional Expectation (ICE) curve of $\mathbf{x}^*$. ICE curves show for one observation and for one feature how the prediction changes when the feature is changed, while other features are fixed to the values of the considered observation [9]. The greater the standard deviation of the ICE curve, the higher we set the probability that the feature value is initialized with a different value than the one of $\mathbf{x}^*$. Therefore, the standard deviation σ_j^{ICE} of each feature x_j is transformed into probabilities within $[p_{min}, p_{max}] \cdot 100\%$:

$$P(value\ differs) = \frac{(\sigma_j^{ICE} - min(\sigma^{ICE})) \cdot (p_{max} - p_{min})}{max(\sigma^{ICE}) - min(\sigma^{ICE})} + p_{min}$$

with $\boldsymbol{\sigma}^{ICE} := (\sigma_1^{ICE}, ..., \sigma_p^{ICE})$. p_{min} and p_{max} are control parameters with default values 0.01 and 0.99.

Actionability. To get more actionable counterfactuals, extreme values of numerical features outside a predefined range are capped to the upper or lower bound after recombination and mutation. The ranges can either be derived from the minimum and maximum values of the features in the observed dataset or users can define these ranges. In addition, users can identify non-actionable features such as the country of birth or gender. The values of these features are permanently set to the values of $\mathbf{x}^*$ for all candidates within MOC.

Penalization. Furthermore, candidates whose predictions are further away from the target than a predefined distance $\epsilon \in \mathbb{R}$ can be penalized. After the candidates have been sorted into fronts F_1 to F_K using nondominated sorting, the candidate that violates the constraint least will be reassigned to front F_{K+1}, the candidate with the second smallest violation to F_{K+2}, and so on. The concept is based on Deb et al. [7]. Since the constraint violators are in the last fronts, they are less likely to be selected for the next generation.

Mutation. Since the aforementioned mutators do not take the data distribution into account and can potentially generate unlikely new candidates, we suggest a conditional mutator. It generates plausible feature values conditional on the values of the other features. For each input feature, we trained a transformation tree [14] on X^{obs}, which is then used to sample values from the conditional distribution. We mutate the feature in randomized order since a feature mutation now depends on the previous changes.

How our proposed strategies for initialization and mutation affect MOC is later examined in a benchmark study (Sects. 6 and 7).

4.4 Evaluation Metric

We use the popular hypervolume indicator (HV) [38] to evaluate the quality of our estimated Pareto front, with reference point $\mathbf{s} = (\inf_{y' \in Y'} |\hat{f}(\mathbf{x}^*) - y'|, 1, p, 1)$, representing the maximal values of the objectives. We compute the HV always over the complete archive of evaluated solutions.

4.5 Tuning of Parameters

We also use HV, when we tune MOC's control parameters – population size, the probabilities for recombining and mutating a feature of a candidate – with iterated F-racing [21]. Furthermore, we let iterated F-racing decide whether our proposed strategies for initialization and mutation of Sect. 4.3 are preferable. Tuning is performed on six binary classification datasets from OpenML [34] – which were not used in the benchmark. A summary of the tuning setup and results can be found in Table 5 in Appendix B. Iterated F-racing found both our initialization and mutation strategy to be advantageous. The tuned parameters were used for the credit data application and the benchmark study.

5 Credit Data Application

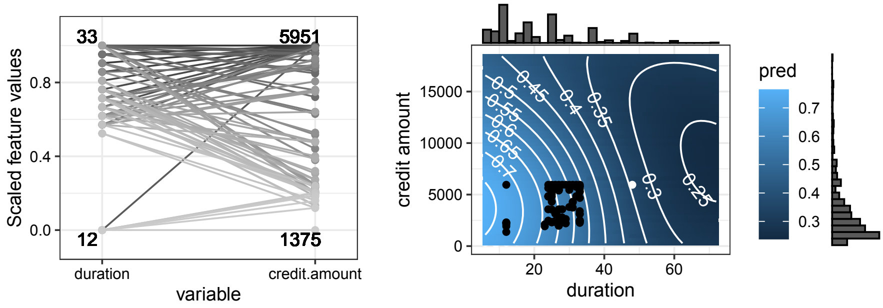
This section demonstrates the usefulness of MOC to explain the prediction of credit risk using the German credit dataset [13]. The dataset has 522 complete observations and nine features containing credit and customer information. Categories with few case numbers were combined. The binary target indicates whether a customer has a 'good' or 'bad' credit risk. We chose the first observation of the dataset as $\mathbf{x}^*$ with the following feature values:

Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose
22	Female	2	Own	Little	Moderate	5951	48	Radio/TV

We tuned a support vector machine (with radial-basis (RBF) kernel) on the remaining data with the same tuning setup as for the benchmark (Appendix C). To obtain a single numerical outcome, only the predicted probability for the class 'good' credit risk was returned. We obtained an accuracy of 0.64 for the model using two nested cross-validations (CV) (5-fold CV in outer and inner loop) and a predicted probability for 'good' credit risk of 0.41 for $\mathbf{x}^*$.

We set the desired outcome interval to $Y' = [0.5, 1]$, which indicates a change to a 'good' credit risk. We generated counterfactuals using MOC with the parameter setting selected by iterated F-racing. Candidates with a prediction below 0.5 were penalized.

A total of 136 counterfactuals were found by MOC. In the following, we focus upon the 82 of them with predictions within $[0.5, 1]$. Credit *duration* was changed



(a) Parallel plot

(b) Response surface plot

Fig. 1. Visualization of counterfactuals for the first data point $\mathbf{x}^*$ of the credit dataset. **(a)** Feature values of the counterfactuals. Only changed features are shown. The given numbers indicate the minimum and maximum feature values of the counterfactuals. **(b)** Response surface plot for the model prediction along features duration and credit amount, holding other feature values constant at the value of $\mathbf{x}^*$. Colors and contour lines indicate the predicted value. The white point is $\mathbf{x}^*$ and the black points are the counterfactuals that only proposed changes in duration and/or credit amount. The histograms show the marginal distributions of the features in the observed dataset.

for all counterfactuals, followed by *credit amount* (86%). Since a user might not want to investigate all returned counterfactuals individually (in feature space), we provide a visual summary of the Pareto set in Fig. 1, either as a parallel coordinate plot or a response surface plot[3] along two features. All counterfactuals had values equal to or smaller than the values of $\mathbf{x}^*$ for *duration* and *credit amount*. The response surface plot illustrates why these feature changes were recommended. The color gradient and contour lines indicate that either *duration* or both *credit amount* and *duration* must be decreased to reach the desired outcome. Due to the fourth objective and the conditional mutator, we obtained counterfactuals in high density areas (indicated by histograms). Counterfactuals in the lower left corner seem to be in a less favorable region far from $\mathbf{x}^*$, but they are close to the training data.

6 Experimental Setup

In this section, the performance of MOC is evaluated in a benchmark study for binary classification. The datasets are from the OpenML platform [34] and are briefly described in Table 1. We selected datasets with no missing values, with up to 3500 observations and a maximum of 40 features. We randomly selected ten observed data points per dataset as $\mathbf{x}^*$ and excluded them from the training data. For each dataset, we tuned and trained the following models: logistic regression, random forest, xgboost, RBF support vector machine and a

[3] This is equivalent to a 2-D ICE-curve through $\mathbf{x}^*$ [9]. We refer to Sect. 4.3 for a general definition of ICE curves.

Table 1. Description of benchmark datasets. Legend: *task:* OpenML task id; *Obs:* Number of rows; *Cont/Cat:* Number of continuous/categorical features.

Task	Name	Obs	Cont	Cat
3718	boston	506	12	1
3846	cmc	1473	2	7
145976	diabetes	768	8	0
9971	ilpd	583	9	1
3913	kc2	522	21	0
3	kr-vs-kp	3196	0	36
3749	no2	500	7	0
3918	pc1	1109	21	0
3778	plasma_retinol	315	10	3
145804	tic-tac-toe	958	0	9

Table 2. MOC's coverage rate of methods to be compared per dataset averaged over all models. The number of nondominated counterfactuals for each method are given in parentheses. Higher values of coverage indicate that MOC dominates the other method. The * indicates that the binomial test with $H_0 : p < 0.5$ that a counterfactual is covered by MOC is significant at the 0.05 level.

	DiCE	Recourse	Tweaking
boston	1* (36)	0.92* (24)	0.9* (10)
cmc	1* (17)		0.75 (8)
diabetes	1* (64)	0.45 (40)	1 (3)
ilpd	1* (26)	1* (37)	0.83 (6)
kc2	1* (53)	0.31 (55)	1 (2)
kr-vs-kp	1* (8)		0.2 (10)
no2	1* (58)	0.5 (12)	0.9* (10)
pc1	1* (60)	0.66* (38)	
plasma_retinol	1* (7)		0.89* (9)
tic-tac-toe	1* (20)		0.75 (8)

one-hidden-layer neural network. The tuning parameter set and the performance using nested resampling are in Table 8 in Appendix C. Each model returned only the probability for one class. The desired target for each $\mathbf{x}^*$ was set to the opposite of the predicted class:

$$Y' = \begin{cases}]0.5, 1] & \text{if } \hat{f}(\mathbf{x}^*) \leq 0.5 \\ [0, 0.5] & \text{else} \end{cases}.$$

The benchmark study aimed to answer two research questions:

Q1) How does MOC perform compared to other state-of-the-art methods for counterfactuals?
Q2) How do our proposed strategies for initialization and mutation of Sect. 4.3 influence the performance of MOC?

For the first one, we compared MOC – once with and once without our proposed strategies for initialization and mutation – with 'DiCE' by Mothilal et al. [24], 'Recourse' by Ustun et al. [33] and 'Tweaking' by Tolomei et al. [32]. We chose DiCE, Recourse and Tweaking because they are implemented in general open source code libraries.[4] The methods are only applicable to certain models: DiCE can handle neural networks and logistic regressions, Recourse can handle logistic regressions and Tweaking can handle random forests. Since Recourse can only process binary and numerical features, we did not train logistic regression on cmc, tic-tac-toe, kr-vs-kp and plasma_retinol. As a baseline, we selected the

[4] Most other counterfactual methods are implemented for specific examples, but cannot be easily used for other datasets.

closest observed data point to $\mathbf{x}^*$ (according to the Gower distance) that has a prediction equal to our desired outcome. Since this approach is part of the *What-If Tool* [36], we call this approach 'Whatif'.

The parameters of DiCE, Recourse and Tweaking were set to the default values recommended by the authors (Appendix D). To allow for a fair comparison, we initialized MOC with the parameters of iterated F-racing which were tuned on other binary classification datasets (Appendix B). While MOC can potentially return several hundreds of counterfactuals, the other methods are designed to either return one or a few. We have therefore limited the maximum number of counterfactuals to ten for all approaches.[5] Tweaking and Whatif generated only one counterfactual by design. For MOC we reduced the number of counterfactuals by preferring the ones that achieved the target prediction Y' and/or the highest HV contribution.

For all methods, only nondominated counterfactuals were considered for the evaluation. Since we are interested in a diverse set of counterfactuals, we evaluate the methods based on the size of their counterfactual set, its objective values, and the coverage rate derived from the coverage indicator by Zitzler and Thiele [38]. The coverage rate is the relative frequency with which counterfactuals of a method are dominated by MOC's counterfactuals for a certain model and $\mathbf{x}^*$. A counterfactual covers another counterfactual if it dominates it, and it does not cover the other if both have the same objective values or the other has lower values in at least one objective. A coverage rate of 1 implies that for each generated counterfactual of a method MOC generated at least one dominating counterfactual. We only computed the coverage rate over counterfactuals that met the desired target Y'.

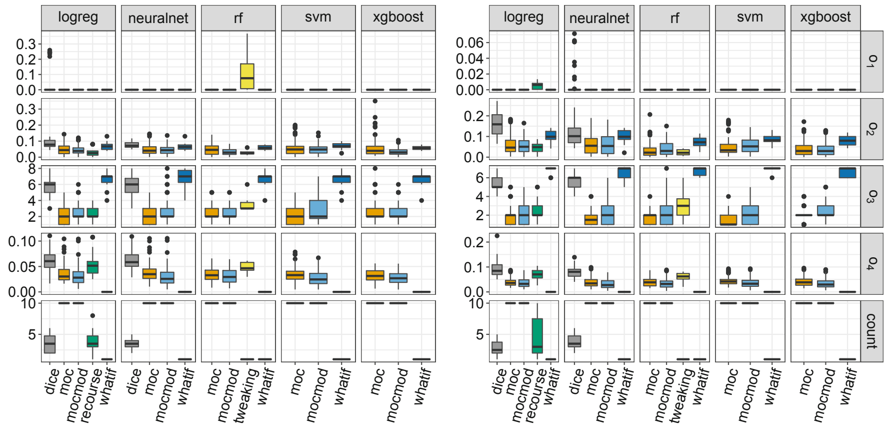
To answer the second research question, we compared the dominated HV over the generations of MOC with and without our proposed strategies for initialization and mutation. As a baseline, we used a random search approach that has the same population size (20) and number of generations (175) as MOC. In each generation, some feature values were uniformly sampled from their set of possible values derived from the observed data and $\mathbf{x}^*$, while other features were set to the values of $\mathbf{x}^*$. The HV for one generation was computed over the newly generated candidates combined with the candidates of the previous generations.

7 Results

Q1) MOC vs. State-of-the-Art Counterfactual Methods

Table 2 shows the coverage rate of each method (to be compared) by the tuned MOC per dataset. Some fields are empty because Recourse could not process features with more than two classes and Tweaking never achieved the desired outcome for pc1. MOC's counterfactuals dominated all counterfactuals of DiCE for all datasets. The same holds for Tweaking except for kr-vs-kp and tic-tac-toe because the counterfactuals of Tweaking had the same objective values as

[5] Note that this artificially penalizes our approach in the benchmark comparison.



(a) diabetes

(b) no2

Fig. 2. Boxplots of the objective values and number of nondominated counterfactuals (*count*) per model for MOC with our proposed strategies for initialization and mutation (*mocmod*), MOC without these modifications, Whatif, DiCE, Recourse and Tweaking for the datasets diabetes and no2. Lower values are better except for *count*.

the ones of MOC. MOC's coverage rate of Recourse only exceeded 90% for boston and ilpd since Recourse's counterfactuals often deviated less from $\mathbf{x}^*$ (but performed worse in other objectives).

Figure 2 compares MOC (with (*mocmod*) and without (*moc*) our proposed strategies for initialization and mutation) with the other methods for the datasets diabetes and no2 and for each model separately. The resulting boxplots for all other datasets are shown in Figs. 4 and 5 in the Appendix. They agree with the results shown here. Compared to the other methods, both versions of MOC found the most nondominated solutions, which met the target and changed the least features. DiCE performed worse than MOC in all objectives. Tweaking's counterfactuals were often closer to $\mathbf{x}^*$, but they were further away from the nearest training data point and more features were changed. Tweaking's counterfactuals often did not reach the desired outcome because they stayed too close to $\mathbf{x}^*$. The MOC with our proposed modifications found counterfactuals closer to $\mathbf{x}^*$ and the observed data, but required more feature changes compared to MOC without the modifications.

Q2) MOC Strategies for Initialization and Mutation

Figure 3 shows the ranks of the dominated HVs for MOC without modifications, for each modification of MOC and random search. Ranks were calculated per dataset, model, $\mathbf{x}^*$ and generation, and were averaged over all datasets, models and $\mathbf{x}^*$. We transformed HVs to ranks because the HVs are not comparable across $\mathbf{x}^*$. It can be seen that the MOC with our proposed modifications clearly

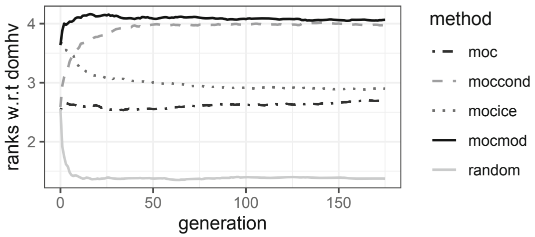


Fig. 3. Comparison of the ranks w.r.t. the dominated HV (*domhv*) per generation averaged over all models and datasets. For each approach, the population size of each generation was 20. A higher HV and therefore a higher rank is better. Legend: *moc*: MOC without our proposed modifications; *moccond*: MOC with the conditional mutator; *mocice*: MOC with the ICE curve variance initialization; *mocmod*: MOC with both modifications; *random*: random search.

outperforms the MOC without these modifications. The ranks of the initial population were higher when the ICE curve variance was used to initialize the candidates. The use of the conditional mutator led to higher dominated HVs over the generations. We received the best performance over the generations when both modifications were used. At each generation, all versions of MOC outperformed random search. Figure 6 in the Appendix shows the ranks over the generations for each dataset separately. They largely agree with the results shown here. The performance gains of MOC compared to random search were particularly evident for higher-dimensional datasets.

8 Conclusion and Outlook

In this paper, we introduced Multi-Objective Counterfactuals (MOC), which to the best of our knowledge is the first method to formalize the counterfactual search as a multi-objective optimization problem. Compared to state-of-the-art approaches, MOC returns a diverse set of counterfactuals with different trade-offs between our proposed objectives. Furthermore, MOC is model-agnostic and suited for classification, regression and mixed feature spaces. We demonstrated the usefulness of MOC to explain a prediction on the German credit dataset and showed in a benchmark study that MOC finds more counterfactuals than other counterfactual methods that are closer to the training data and required fewer feature changes. Our proposed initialization strategy (based on ICE curve variances) and our conditional mutator resulted in higher performance in fewer evaluations and in counterfactuals that were closer to the data point we were interested in and to the observed data.

MOC has only been evaluated on binary classification, and only with respect to the dominated HV and the individual objectives. It is an open question how to let users select the counterfactuals that meet their – a-priori unknown – trade-off between the objectives. We leave these investigations to future research.

9 Electronic Submission

The complete code of the algorithm and the code to reproduce the experiments and results of this paper are available at https://github.com/susanne-207/moc. The implementation of MOC is based on our implementation of [19], which we also used for [3]. We will provide an open source R library with our implementation of the method based on the `iml` package [23].

A Illustration of MOC's Benefits

This section illustrates the benefits of having a *diverse set* of counterfactuals using the diabetes dataset of the benchmark study (Sect. 6). We will compare the counterfactuals returned by MOC with the ones of Recourse [33] and Tweaking [32]. Due to space constraints, we only show the six counterfactuals of MOC with the highest HV contribution for both examples.

Table 3. Counterfactuals and corresponding objective values of MOC and Recourse for the prediction of a logistic regression for observation 741 of the diabetes dataset. Shaded fields indicate values that differ from the value of observation 741 in brackets.

Feature ($\mathbf{x}^*$)	MOC$_1$	MOC$_2$	MOC$_3$	MOC$_4$	MOC$_5$	MOC$_6$	Recourse$_1$	Recourse$_2$	Recourse$_3$
preg (11)	11.00	6.35	11.00	11.00	11.00	6.35	11.00	11.00	10.92
plas (120)	27.78	3.29	79.75	94.85	79.75	3.18	57.00	57.00	57.00
pres (80)	80.00	80.00	80.00	80.00	80.00	80.00	80.00	80.00	80.00
skin (37)	37.00	37.00	37.00	37.00	37.00	37.00	37.00	36.81	37.00
insu (150)	150.00	150.00	17.13	150.00	40.61	150.00	150.00	150.00	150.00
mass (42.3)	42.30	42.30	29.17	15.36	29.17	42.30	42.30	42.30	42.30
pedi (0.78)	0.78	0.78	0.31	0.78	0.17	0.78	0.78	0.78	0.78
age (48)	48.00	41.61	44.42	48.00	48.00	48.00	28.36	28.36	28.36
o_1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
o_2	0.06	0.12	0.10	0.07	0.10	0.11	0.08	0.08	0.08
o_3	1.00	3.00	5.00	2.00	4.00	2.00	2.00	3.00	3.00
o_4	0.10	0.05	0.03	0.07	0.04	0.07	0.09	0.09	0.09

Table 3 contrasts MOC's counterfactuals with the three counterfactuals of Recourse for the prediction of observation 741. A logistic regression predicted a probability of having diabetes of 0.89 for this observation. The desired target is a prediction of less than 0.5, which indicates having no diabetes. All counterfactuals of Recourse suggest the same reduction in *age* and plasma concentration (*plas*), with two counterfactuals additionally suggesting a minimal reduction in the number of pregnancies (*preg*) or the skin fold thickness (*skin*).[6] Apart from that a reduction in *age* or *preg* is impossible, they do not offer many options

[6] By reclassifying *age* and *preg* as integers (instead of decimals), integer changes would be recommended by MOC, Recourse and Tweaking.

Table 4. Counterfactuals and corresponding objective values given by MOC and Tweaking for the prediction of a random forest for observation 268 of the cmc dataset. Shaded fields indicate values that differ from the value of observation 268 in brackets.

Feature ($\mathbf{x}^*$)	MOC$_1$	MOC$_2$	MOC$_3$	MOC$_4$	MOC$_5$	MOC$_6$	Tweaking$_1$
preg (2)	2.00	2.00	2.00	2.00	2.00	2.00	1.53
plas (128)	121.50	90.21	126.83	128.00	88.44	120.64	119.71
pres (64)	64.00	64.00	64.00	64.00	64.00	64.00	64.00
skin (42)	42.00	42.00	42.00	42.00	42.00	42.00	42.00
insu (0)	0.00	0.00	0.00	0.00	0.00	90.93	0.00
mass (40)	40.00	40.00	40.00	40.00	40.00	40.00	40.00
pedi (1.1)	1.10	0.48	1.10	0.17	0.46	1.10	1.10
age (24)	24.00	24.00	24.00	24.00	25.85	24.00	28.29
o_1	0.00	0.00	0.00	0.00	0.00	0.00	0.00
o_2	0.00	0.06	0.00	0.05	0.06	0.02	0.02
o_3	1.00	2.00	1.00	1.00	3.00	2.00	3.00
o_4	0.05	0.02	0.05	0.04	0.01	0.03	0.06

for users. Instead, MOC returned a larger set of counterfactuals that provide more options for actionable user responses and are closer to the observed data than Recourse's counterfactuals (o_4). Counterfactual MOC$_1$ has overall lower objective values than all counterfactuals of Recourse. MOC$_3$ suggested changes to five features so that it is especially close to the nearest training data point (o_4).

Table 4 compares the set of counterfactuals found by MOC with the single counterfactual found by Tweaking for the prediction of observation 268. A random forest classifier predicted a probability of having diabetes of 0.62 for this observation. Again, the desired target is a prediction of less than 0.5. Tweaking suggested reducing the number of children and plasma glucose concentration (*plas*) while increasing the *age* so that the probability of diabetes decreases. This is contradictory and not plausible. In contrast, MOC's counterfactuals suggest various strategies, e.g., only a decrease of *plas*, which is easier to realize. In addition, MOC$_1$, MOC$_3$ and MOC$_6$ dominate the counterfactual of Tweaking. Since five of six counterfactuals suggest changes to *plas*, the user may have more confidence that *plas* is an important lever to achieve the desired outcome.

B Iterated F-racing

We used iterated F-racing (irace) [21] to tune the parameters of MOC for binary classification. The parameters and considered ranges are given in Table 5. The number of generations was not part of the parameter set because it would be always tuned to the upper bound. Instead, the number of generations was determined after the other parameters were tuned with irace. Irace was initialized with a maximum budget of 3000 evaluations equal to 3000 runs of MOC. In every step, irace randomly selected one of 300 instances. Each instance consisted of a trained model, a randomly selected data point from the observed data as $\mathbf{x}^*$

Table 5. Parameter space investigated with iterated F-racing, as well as the resulting optimized configuration (*Result*).

Name	Description	Range	Result
M	Population size	[20, 100]	20
initialization	Initialization strategy	[Random, ICE curve]	ICE curve
conditional	Whether to use the conditional mutator	[TRUE, FALSE]	TRUE
p.rec	Probability a pair of parents is chosen to recombine	[0.3, 1]	0.57
p.rec.gen	Probability a feature is recombined	[0.3, 1]	0.85
p.rec.use.orig	Probability the indicator for feature changes is recombined	[0.3, 1]	0.88
p.mut	Probability a child is chosen to be mutated	[0.05, 0.8]	0.79
p.mut.gen	Probability one feature is mutated	[0.05, 0.8]	0.56
p.mut.use.orig	Probability indicator for a feature change is flipped	[0.05, 0.5]	0.32

and a desired outcome. The desired target for each $\mathbf{x}^*$ was the opposite of the predicted class:

$$Y' = \begin{cases}]0.5, 1] & \text{if } \hat{f}(\mathbf{x}^*) \leq 0.5 \\ [0, 0.5] & \text{else} \end{cases}.$$

The trained model was either logistic regression, random forest, xgboost, RBF support vector machine or a two-hidden-layer neural network. Each model estimated only the probability for one class. The models were trained on datasets obtained from the OpenML platform [34] (without the sampled $\mathbf{x}^*$) and are briefly described in Table 7. While these datasets were not used in the benchmark study (Sect. 6), the same preprocessing steps were conducted and the models were tuned with the same setup (see Sect. C for details).

In each step of irace, parameter configurations were evaluated by running MOC on the same selected instance. MOC stopped after evaluating 8000 candidates with Eq. (1), which should be enough to ensure convergence of the HV in most cases. The integral of the first order spline approximation of the dominated HV over the evaluations was the performance criterion as recommended by [26]. The integral takes into account not only the extent but also the rate of convergence of the dominated HV. A Friedman test was used to discard less promising configurations. The first Friedman test was conducted after initial configurations were evaluated on 15 instances; afterward, the test was conducted after evaluating the remaining configurations on a single instance to accelerate the exclusion process. The best configuration returned is given in Table 5.

To obtain a default parameter for the number of generations for the benchmark study, we determined for the 300 instances after how many generations of the tuned MOC the dominated HV has not increased for 10 generations. We chose the maximum of 175 generations as a default for the study.

Table 6. Tuning search space per model. The hyperparameters *ntrees* and *nrounds* were log-transformed.

Model	Hyperparameter	Range
randomforest	ntrees	[0, 1000]
xgboost	nrounds	[0, 1000]
svm	cost	[0.01, 1]
logreg	lr	[0.0005, 0.1]
neuralnet	lr	[0.0005, 0.1]
	layer_size	[1, 6]

Table 7. Description of datasets for tuning with iterated F-racing. Legend: *Task:* OpenML task id; *Obs:* Number of rows; *Cont/Cat:* Number of continuous/categorical features.

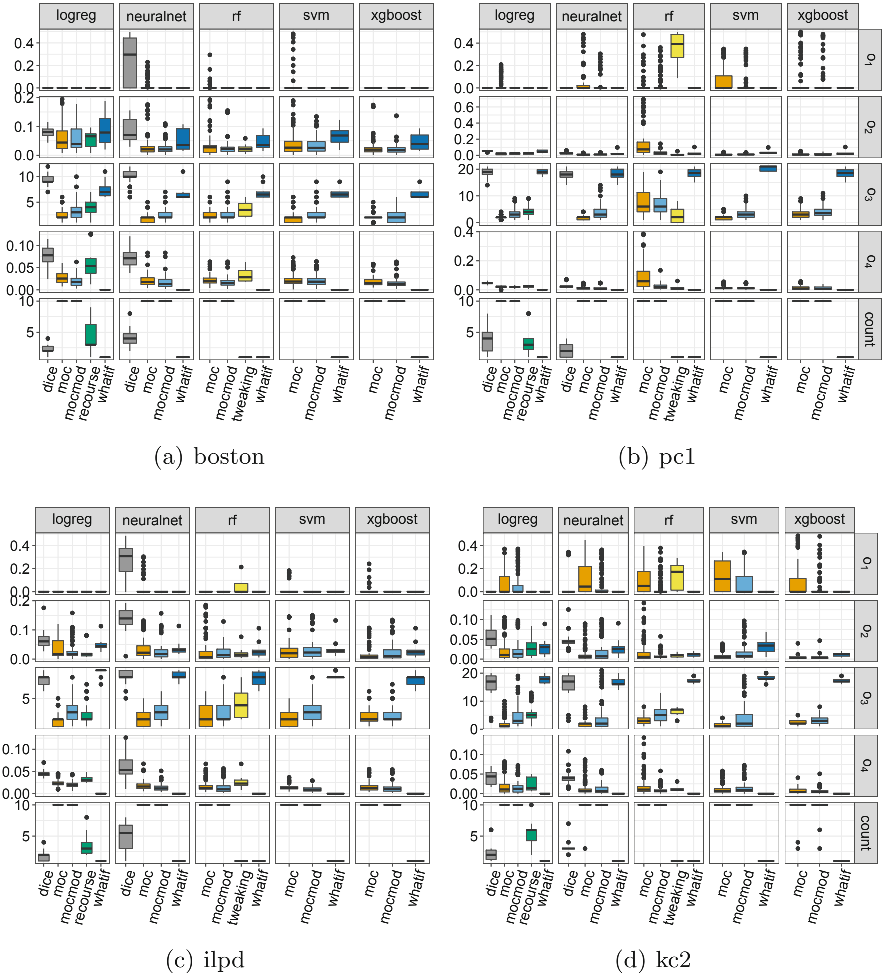
Task	Name	Obs	Cont	Cat
3818	tae	151	3	2
3917	kc1	2109	21	0
52945	breastTumor	277	0	6
3483	mammography	11183	6	0
3822	nursery	12960	0	8
3586	abalone	4177	7	1

C Model Hyperparameters for the Benchmark Study

We used random search (with 200 iterations for neural networks and 100 iterations for all other models) and 5-fold CV (with misclassification error as performance measure) to tune the hyperparameters of the models on the training data. The tuning search space was the same as for iterated F-racing and is shown in Table 6. Numerical features were scaled (standardization (Z-score) for random forest, min-max-scaling (0–1-range) for all other models) and categorical features were one-hot encoded. For neural network and logistic regression, ADAM [17] was the optimizer, the batch size was 32 with a 1/3 validation split and early stopping was conducted after 5 patience steps. Logistic regression needed these configurations because we constructed the model as a zero-hidden-layer neural network. For all other hyperparameters of the models, we chose the default values of the `mlr` [4] and `keras` [1] R packages. Table 8 shows the accuracies of the trained models using nested resampling (5-fold CV in outer and inner loop).

Table 8. Accuracy using nested resampling per benchmark dataset and model. Legend: *Name:* OpenML task name; *rf:* random forest. Logistic regression (*logreg*) was only trained on datasets with numerical or binary features.

Name	rf	xgboost	svm	logreg	neuralnet
boston	0.90	0.89	0.87	0.86	0.87
cmc	0.70	0.72	0.67		0.68
diabetes	0.76	0.74	0.75	0.63	0.68
ilpd	0.69	0.67	0.65	0.53	0.58
kc2	0.81	0.80	0.79	0.75	0.72
kr-vs-kp	0.99	0.99	0.97		0.99
no2	0.63	0.59	0.58	0.55	0.54
pc1	0.93	0.93	0.91	0.91	0.88
plasma_retinol	0.53	0.52	0.58		0.55
tic-tac-toe	0.99	0.99	0.98		0.97



(a) boston

(b) pc1

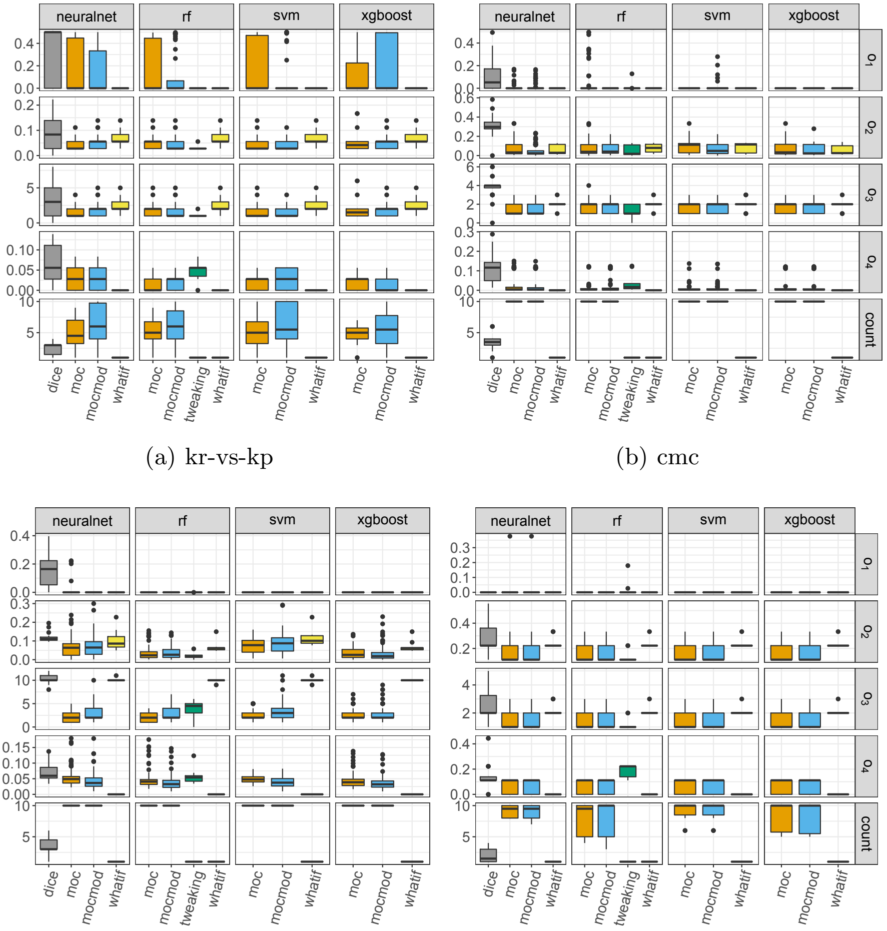
(c) ilpd

(d) kc2

Fig. 4. Boxplots of the objective values and number of nondominated counterfactuals (*count*) per dataset and model for MOC with our proposed strategies for initialization and mutation (*mocmod*), MOC without these modifications, Whatif, DiCE, Recourse and Tweaking. Lower values are better except for *count*.

D Control Parameters of Counterfactual Methods

For Tweaking [32], we only changed ϵ, a positive threshold that limits the tweaking of each feature. It was set to 0.5 because it obtained better results for the authors on their data example on Ad Quality in comparison to the default value 0.1. We used the R implementation of Tweaking on Github: https://github.com/katokohaku/featureTweakR (commit `6f3e614`). For Recourse [33], we left all parameters at their default settings. We used the Python implementation of Recourse on Github: https://github.com/ustunb/actionable-recourse (com-



(a) kr-vs-kp

(b) cmc

(c) plasma_retinol

(d) tic-tac-toe

Fig. 5. Boxplots of the objective values and number of nondominated counterfactuals (*count*) per dataset and model for MOC with our proposed strategies for initialization and mutation (*mocmod*), MOC without these modifications, Whatif, DiCE, Recourse and Tweaking. Lower values are better except for *count*.

mit aaae8fa). For DiCE [24], we used the 'DiverseCF' version proposed by the authors [24] and left the control parameters at their defaults. We used the inverse mean absolute deviation for the feature weights. For datasets where the mean absolute deviation of a feature was zero, we set the feature weight to 10. We used the Python implementation of DiCE available on Github: https://github.com/microsoft/DiCE (commit fed9d27).

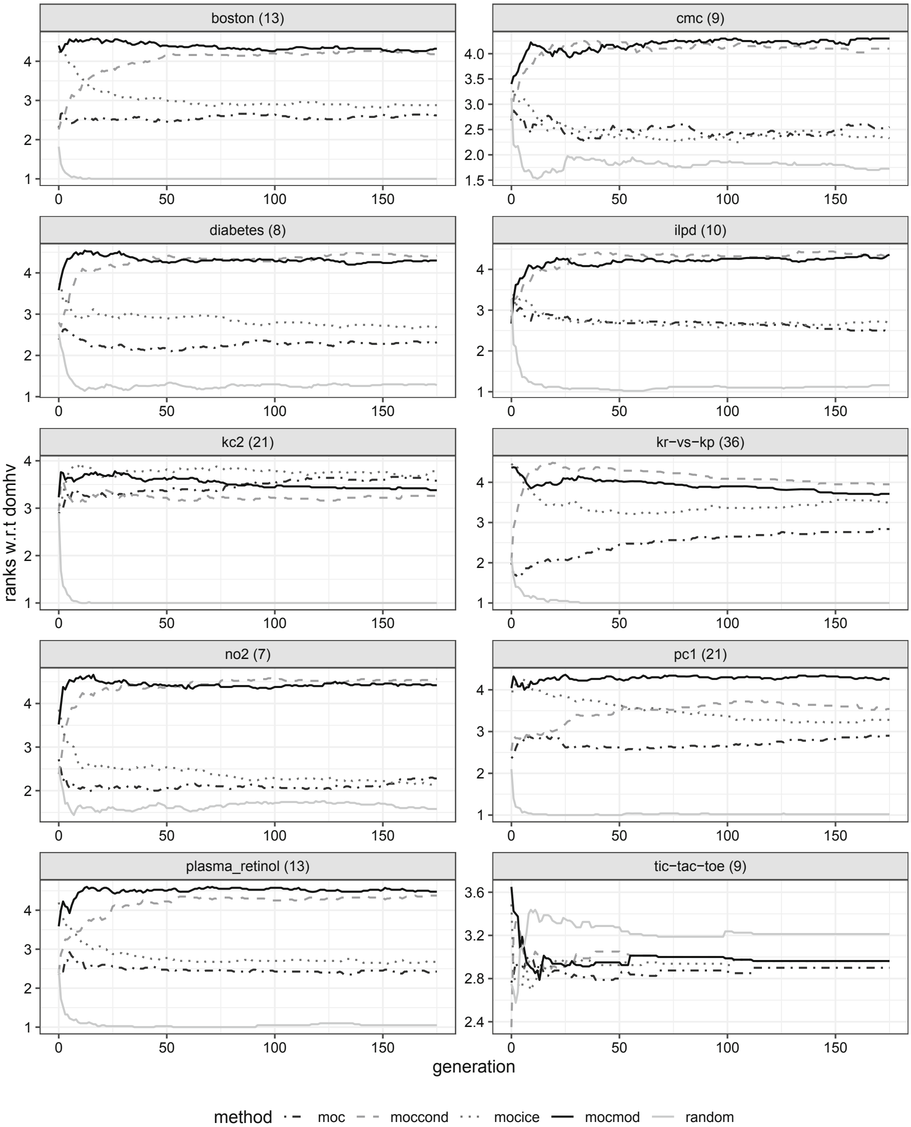


Fig. 6. Comparison of the ranks w.r.t. the dominated HV (*domhv*) per generation and per benchmark dataset averaged over all models. The numbers in parentheses indicate the number of features. For each approach, the population size of each generation was 20. Higher ranks are better. Legend: *moc*: MOC without modifications; *moccond*: MOC with the conditional mutator; *mocice*: MOC with the ICE curve variance initialization; *mocmod*: MOC with both modifications; *random*: random search.

References

1. Allaire, J., Chollet, F.: keras: R Interface to 'Keras' (2019). https://keras.rstudio.com, R package version 2.3.0
2. Avila, S.L., Krähenbühl, L., Sareni, B.: A multi-niching multi-objective genetic algorithm for solving complex multimodal problems. In: OIPE. Sorrento, Italy (2006). https://hal.archives-ouvertes.fr/hal-00398660
3. Binder, M., Moosbauer, J., Thomas, J., Bischl, B.: Multi-Objective Hyperparameter Tuning and Feature Selection using Filter Ensembles (2019). Accepted at GECCO 2020
4. Bischl, B., et al.: mlr: Machine Learning in R. J. Mach. Learn. Res. **17**(170), 1–5 (2016). http://jmlr.org/papers/v17/15-066.html, R package version 2.17
5. Breiman, L.: Statistical modeling: the two cultures. Stat. Sci. **16**(3), 199–231 (2001). https://doi.org/10.1214/ss/1009213726
6. Deb, K., Agarwal, R.B.: Simulated binary crossover for continuous search space. Complex Syst. **9**, 115–148 (1995)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
8. Dhurandhar, A., Pedapati, T., Balakrishnan, A., Chen, P., Shanmugam, K., Puri, R.: Model Agnostic Contrastive Explanations for Structured Data. CoRR abs/1906.00117 (2019). http://arxiv.org/abs/1906.00117
9. Goldstein, A., Kapelner, A., Bleich, J., Pitkin, E.: Peeking inside the black box: visualizing statistical learning with plots of individual conditional expectation. J. Comput. Graph. Stat. **24**(1), 44–65 (2015). https://doi.org/10.1080/10618600.2014.907095
10. Gower, J.C.: A general coefficient of similarity and some of its properties. Biometrics **27**(4), 857–871 (1971)
11. Grath, R.M., et al.: Interpretable Credit Application Predictions With Counterfactual Explanations. CoRR (abs/1811.05245) (2018). http://arxiv.org/abs/1811.05245
12. Greenwell, B.M., Boehmke, B.C., McCarthy, A.J.: A simple and effective model-based variable importance measure. arXiv preprint arXiv:1805.04755 (2018)
13. Hofmann, H.: German Credit Risk (2016). https://www.kaggle.com/uciml/german-credit. Accessed 25 Jan 2020
14. Hothorn, T., Zeileis, A.: Transformation Forests (2017)
15. Joshi, S., Koyejo, O., Vijitbenjaronk, W., Kim, B., Ghosh, J.: Towards Realistic Individual Recourse and Actionable Explanations in black-box decision making systems. CoRR abs/1907.09615 (2019). http://arxiv.org/abs/1907.09615
16. Karimi, A., Barthe, G., Balle, B., Valera, I.: Model-Agnostic Counterfactual Explanations for Consequential Decisions. CoRR (abs/1905.11190) (2019). http://arxiv.org/abs/1905.11190
17. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations, December 2014
18. Laugel, T., Lesot, M.J., Marsala, C., Renard, X., Detyniecki, M.: Comparison-Based Inverse Classification for Interpretability in Machine Learning. CoRR (abs/1712.08443) (2017). http://arxiv.org/abs/1712.08443
19. Li, R., et al.: Mixed integer evolution strategies for parameter optimization. Evol. Comput. **21**(1), 29–64 (2013)

20. Looveren, A.V., Klaise, J.: Interpretable Counterfactual Explanations Guided by Prototypes. CoRR abs/1907.02584 (2019). http://arxiv.org/abs/1907.02584
21. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Perspect. **3**, 43–58 (2016). https://doi.org/10.1016/j.orp.2016.09.002, http://www.sciencedirect.com/science/article/pii/S2214716015300270, R package version 3.4.1
22. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, pp. 4765–4774 (2017)
23. Molnar, C., Bischl, B., Casalicchio, G.: iml: an R package for interpretable machine learning. JOSS **3**(26), 786 (2018). https://doi.org/10.21105/joss.00786
24. Mothilal, R.K., Sharma, A., Tan, C.: Explaining Machine Learning Classifiers through Diverse Counterfactual explanations. CoRR (abs/1905.07697) (2019). http://arxiv.org/abs/1905.07697
25. Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., Bie, T.D., Flach, P.: FACE: Feasible and Actionable Counterfactual Explanations (2019)
26. Radulescu, A., López-Ibáñez, M., Stützle, T.: Automatically improving the anytime behaviour of multiobjective evolutionary algorithms. In: Purshouse, R.C., Fleming, P.J., Fonseca, C.M., Greco, S., Shaw, J. (eds.) EMO 2013. LNCS, vol. 7811, pp. 825–840. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37140-0_61
27. 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, pp. 1135–1144 (2016)
28. Russell, C.: Efficient Search for Diverse Coherent Explanations. CoRR (abs/1901.04909) (2019). http://arxiv.org/abs/1901.04909
29. Sharma, S., Henderson, J., Ghosh, J.: CERTIFAI: Counterfactual Explanations for Robustness, Transparency, Interpretability, and Fairness of Artificial Intelligence models. CoRR abs/1905.07857 (2019). http://arxiv.org/abs/1905.07857
30. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Trans. Evol. Comput. **23**, 828–841 (2017)
31. Syswerda, G.: Uniform crossover in genetic algorithms. In: Proceedings of the 3rd International Conference on Genetic Algorithms, pp. 2–9. Morgan Kaufmann Publishers Inc., San Francisco (1989)
32. Tolomei, G., Silvestri, F., Haines, A., Lalmas, M.: Interpretable predictions of tree-based ensembles via actionable feature tweaking. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2017, pp. 465–474. ACM, New York (2017). https://doi.org/10.1145/3097983.3098039
33. Ustun, B., Spangher, A., Liu, Y.: Actionable recourse in linear classification. In: Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, pp. 10–19. ACM, New York (2019). https://doi.org/10.1145/3287560.3287566
34. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. SIGKDD Explor. **15**(2), 49–60 (2013). https://doi.org/10.1145/2641190.2641198
35. Wachter, S., Mittelstadt, B.D., Russell, C.: Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. CoRR (abs/1711.00399) (2017). http://arxiv.org/abs/1711.00399
36. Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viégas, F.B., Wilson, J.: The What- If Tool: Interactive Probing of Machine Learning Models. CoRR abs/1907.04135 (2019). http://arxiv.org/abs/1907.04135

37. White, A., d'Avila Garcez, A.: Measurable Counterfactual Local Explanations for Any Classifier (2019)
38. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms—a comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0056872