Multi-Objective Counterfactual Explanations*

Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl

Ludwig-Maximilians-Universität München, 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'. Current approaches can compute counterfactuals only for certain model classes or feature types, or they generate counterfactuals that are not consistent with the observed data distribution. To overcome these limitations, we propose the Multi-Objective Counterfactuals (MOC) method, which translates the counterfactual search into a multi-objective optimization problem and solves it with a genetic algorithm based on NSGA-II. It returns a diverse set of counterfactuals with different trade-offs between the proposed objectives, enabling either a more detailed post-hoc analysis to facilitate better understanding or more options for actionable user responses to change the predicted outcome. 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 and 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 few features. Therefore,

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

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 [32]. 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 [28]. Attribution methods such as Local Interpretable Model-agnostic Explanations (LIME) [25] and Shapley Values [20] 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 [16,9,6,14,23,27,33]. Apart from Grath et al. [9], these approaches are limited to classification. Unlike the other methods, the method of Poyiadzi et al. [23] can obtain plausible counterfactuals by constructing feasible paths between data points with opposite predictions. Their approach only works for numerical features.

A model-specific approach was proposed by Wachter et al. [32], who also introduced and formalized the concept of counterfactuals in predictive modeling. Like many model-specific methods [13,18,21,26,30] their approach is limited to differentiable models. The approach of Tolomei et al. [29] generates explanations for tree-based ensemble binary classifiers. As with [32] and [18], 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., 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 represent 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 age') or counterproductive in more general contexts (e.g., 'raise your blood pressure 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. Due to space constraints here, we refer the reader to Appendix A for two concrete examples illustrating the above.

Compared to other counterfactual methods, MOC is model-agnostic, handles classification, regression and mixed feature spaces, which (in our opinion) furthermore increases its practical usefulness in general applications.

4 Methodology

Before we explain the counterfactual search in more detail, we define a counterfactual explanation and propose four objectives associated with this definition.

Definition 1 (Counterfactual Explanation). *Let $\hat{f} : \mathcal{X} \to \mathbb{R}$ be a prediction function, $\mathcal{X}$ the feature space and $Y' \subset \mathbb{R}$ a set of desired outcomes. The latter can either be a single value or an interval of values. Then, a counterfactual explanation $\mathbf{x}' \in \mathcal{X}$ for the data point $\mathbf{x}^{orig}$ is a close neighbor of $\mathbf{x}^{orig}$ whose prediction $\hat{f}(\mathbf{x}')$ is in, or sufficiently close to, the desired outcome set Y'. To improve interpretability, $\mathbf{x}'$ should differ from $\mathbf{x}^{orig}$ only in a few features and should be 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).

4.1 Counterfactual Objectives

Definition 1 of a counterfactual explanation can be translated into a multi-objective optimization task with four objectives to be minimized:

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

[1] Rashomon effect [4]

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

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

The second component o_2 quantifies the distance between $\mathbf{x}^{orig}$ and $\mathbf{x}$. using the Gower distance to account for mixed features [8]:

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

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

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

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:

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

To obtain counterfactuals close to the observed data $\mathbf{X}^{obs}$, the fourth objective o_4 measures the weighted average Gower distance between $\mathbf{x}$ and the k nearest observed data points $\mathbf{x}^{[1]}, ..., \mathbf{x}^{[k]} \in \mathbf{X}^{obs}$:

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

Further procedures to increase the plausibility of the counterfactuals are integrated into the optimization algorithm and are described in Section 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 $\mathbf{x}^{orig}$ (o_2 and o_3) and to the observed data (o_4).

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

4.2 Counterfactual Search

The *Nondominated Sorting Genetic Algorithm II* (NSGA-II) [5] is an evolutionary multi-objective genetic algorithm consisting of two steps that are repeated in each generation: generating new candidates by selection, recombination and mutation of candidates of the current population, and selecting the best candidates by nondominated sorting and crowding distance sorting. Nondominated sorting favors solutions near the Pareto front and crowding distance sorting preserves the diversity of the candidates.

MOC directly uses the NSGA-II with modifications to optimize its four objectives. First, unlike the original NSGA-II, it uses *mixed integer evolutionary strategies* (MIES) [17] 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). Crowding distance sorting reflects the diversity of each candidate as the sum of distances to its neighbors across the four objectives. In contrast 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 features 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 By default, MOC randomly initializes the features values of a candidate in the first population and randomly selects if they differ from the values of $\mathbf{x}^{orig}$. However, if a feature has a large influence on the prediction, it should be more likely that a counterfactual will propose changes to this feature. The importance of a feature for an entire dataset can be measured as the standard deviation of the partial dependence plot [10]. Analogously, we propose to measure the feature importance for a single prediction with the standard deviation of the Individual Conditional Expectation (ICE) curve of $\mathbf{x}^{orig}$. 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 [7]. 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 $\mathbf{x}^{orig}$. Therefore, the standard deviation σ_j^{ICE} of each feature x_j is transformed into probabilities within $[p_{min}, p_{max}] \cdot 100\%$:

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

with $\boldsymbol{\sigma}^{ICE} := (\sigma_1^{ICE}, ..., \sigma_p^{ICE})$. p_{min} and p_{max} are hyperparameters 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 age. The value of these features are permanently set to the values of $\mathbf{x}^{orig}$ 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 solution that violates the constraint least will be reassigned to the 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. [5]. Since the constraint violators are in the last fronts, they are less likely to be selected for the next generation.

Mutation Features of the candidates are mutated with a pre-defined (low) probability – one of the hyperparameters of MOC. By default, numerical features are altered by the scaled Gaussian mutator, which adds a normally distributed random value to the current feature value [17]. Discrete features are mutated by uniformly sampling their possible values. Binary and logical features are flipped. Instead, we propose to sample a new feature from its conditional distribution given all other features, to obtain counterfactuals that are consistent with the observed data distribution. The order in which the features are changed is randomized since a mutation depends on the previous mutations. The conditional distribution is estimated by conditional transformation trees derived from the observed dataset [12].

4.4 Evaluation Metric

A common measure to evaluate the set of nondominated solutions (Pareto set) returned by genetic algorithms is the dominated hypervolume indicator [34]. It is defined as the volume of the region of the objective space – bounded by a reference point – which contains candidates that are weakly dominated by at least one of the members of the Pareto set. Higher values of the hypervolume are preferred. For MOC, the values of the reference point are

$$\mathbf{s} = (\inf_{y' \in Y'} |\hat{f}(\mathbf{x}^{orig}) - y'|, 1, p, 1)$$

representing the maximal values of the objectives. The dominated hypervolume should monotonically increase over the generations since MOC improves the adaptive fit of the population of candidates to the Pareto front.

4.5 Tuning of Parameters

The dominated hypervolume measures also the performance for tuning the hyperparameters of MOC using iterated F-racing [19]. The parameters are the population size and the probabilities of recombining and mutating the features of a candidate. The parameters were tuned on six binary classification datasets from OpenML [31] – which were not used in the benchmark – and on single desired targets. A summary of the tuning setup and results can be found in Table 5 in Appendix B. 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 [11]. The dataset has 522 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 tuned a support vector machine (with radial-basis (RBF) kernel) on 80 % of the 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. On the 20 % hold-out set, the model had an accuracy of 0.6, which is enough for illustration purposes. We chose the first observation of the dataset as $\mathbf{x}^{orig}$, with a predicted probability for 'good' credit risk of 0.41 and the following feature values:

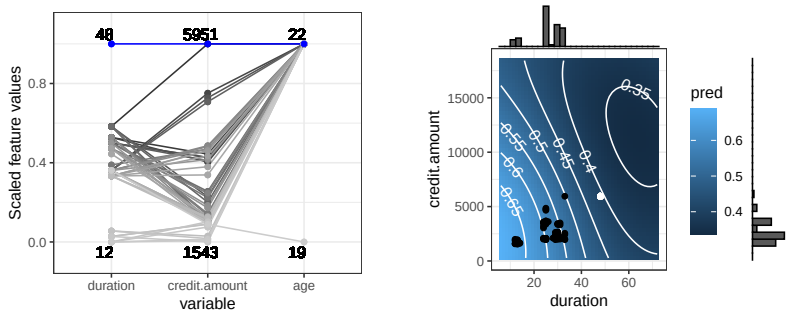
Feature	age	sex	job	housing	saving	checking	credit.amount	duration	purpose
Value	22	female	2	own	little	moderate	5951	48	radio/TV

We set the desired outcome interval to $[0.5, 1]$ which indicates a change to a 'good' credit risk. We generated counterfactuals using MOC with the modified initialization and mutation strategies and the parameters tuned by iterated F-racing. Candidates with a prediction below 0.5 were penalized.

A total of 59 counterfactuals were found by MOC. All had a prediction greater than 0.5 for class 'good'. Credit duration was changed for all counterfactuals, followed by credit amount (98 %). The person's age was only changed for one counterfactual. 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 Figure 1, either as a parallel coordinate plot or a response surface plot along two numerical features.[3] All counterfactuals had values equal to or smaller than the values of $\mathbf{x}^{orig}$ for duration, credit amount and age.

The response surface plot illustrates why the counterfactuals with changes in features duration, amount or both are consistent with our multi-objective task. The colour gradient and contour lines indicate that either duration or both

[3] This is equivalent to a 2-D ICE-curve through $\mathbf{x}^{orig}$ [7]. We refer to Section 4.3 for a general definition of ICE curves.



(a) Feature values (b) Response surface plot

Fig. 1. Visualization of counterfactuals for the prediction of the first data point $\mathbf{x}^{orig}$ of the credit dataset. **Left:** Feature values of counterfactuals. Only changed features are shown. The blue line corresponds to the values of $\mathbf{x}^{orig}$. The given numbers are the minimum and maximum values of the features over $\mathbf{x}^{orig}$ and the counterfactuals. **Right:** Response surface plot for the model prediction along features duration and credit amount, holding other feature values constant at the value of $\mathbf{x}^{orig}$. Colors and contour lines indicate the predicted value. The white point is $\mathbf{x}^{orig}$ 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.

amount and duration must be decreased to reach the desired outcome. Due to the fourth objective and the modified mutator, we obtained counterfactuals in high density areas (indicated by the histograms), even though $\mathbf{x}^{orig}$ is in a low density area. Counterfactuals in the lower left corner seem to be in a less favorable region far from $\mathbf{x}^{orig}$ but they are close to the observed data points.

6 Experimental Setup

In this section, the performance of MOC is evaluated in a benchmark study for binary classification. The datasets for this benchmark are from the `OpenML` platform [31] and are briefly described in Table 1. We selected datasets with no missing values, with up to 1500 observations and a maximum of 40 features. For each dataset, we tuned and trained the following models on 80 % of the data: logistic regression, random forest, xgboost, RBF support vector machine and a one-hidden-layer neural network. The tuning parameter set and the performance on the 20 % hold-out are in Table 8 in Appendix C. Each model returned only the probability for one class. Ten observed data points per dataset were randomly selected as $\mathbf{x}^{orig}$. The desired target for each $\mathbf{x}^{orig}$ was set to the opposite of the predicted class:

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

The benchmark study aimed to answer two research questions:

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* (16)	0.7 (10)	1* (8)
cmc	0.95* (20)		1* (9)
diabetes	1* (65)	0.8* (20)	1* (7)
ilpd	1* (18)	0.95* (22)	1* (6)
kc2	1* (47)	0.79* (24)	1 (1)
kr-vs-kp	1* (4)		0.14 (7)
no2	1* (45)	1* (25)	1* (10)
pc1	1* (42)	0.5 (14)	
plasma_retinol	1* (6)		0.88* (8)
tic-tac-toe	1* (2)		0 (7)

Q1) How does MOC perform compared to other state-of-the-art methods for counterfactuals?

Q2) How do the proposed initialization and mutation strategies of Section 4.3 influence the performance of MOC?

For the first part, we compared MOC – once with and once without the proposed initialization and mutation strategies – with 'DiCE' by Mothilal et al. [21], 'Recourse' by Ustun et al. [30] and 'Tweaking' by Tolomei et al. [29]. We chose DiCE, Recourse and Tweaking because they are implemented in general open source code libraries.[4] Some 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 closest observed data point to $\mathbf{x}^{orig}$ (according to the Gower distance) that has a prediction equal to our desired outcome. Since this approach is part of the *What-If Tool* of Google's PAIR team [22], we call this approach 'Pair'.

The parameters of DiCE, Recourse and Tweaking were set to the default values recommended by the authors (Appendix C.2). To allow for a fair comparison, we initialized MOC with the parameters of iterated F-racing which were tuned on other binary classification datasets and on single desired targets (instead of ranges) (Appendix B). We set the number of considered neighbors k for the fourth objective (o_4) to 1. While MOC returned up to 59 counterfactuals, the other methods are designed to either return one or few. Therefore we limited

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

the maximum number of counterfactuals to 10 for all approaches.[5] Tweaking and Pair generated only one counterfactual by design. For MOC we reduced the number of counterfactuals by crowding distance sorting (Section 4.2).

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 [34]. The coverage rate is the relative frequency with which counterfactuals of a method are dominated by MOC's counterfactuals for a certain model and $\mathbf{x}^{orig}$. 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 hypervolume over the generations of MOC with and without the proposed initialization and mutation strategies. As a baseline, we used a random search approach that has the same population size (59) and number of generations (60) as MOC. In each generation, some feature values were uniformly sampled from their set of possible values derived from the observed data, while other features were set to the values of $\mathbf{x}^{orig}$. The hypervolume for one generation was computed over the newly generated candidates combined with the candidates of the previous generations.

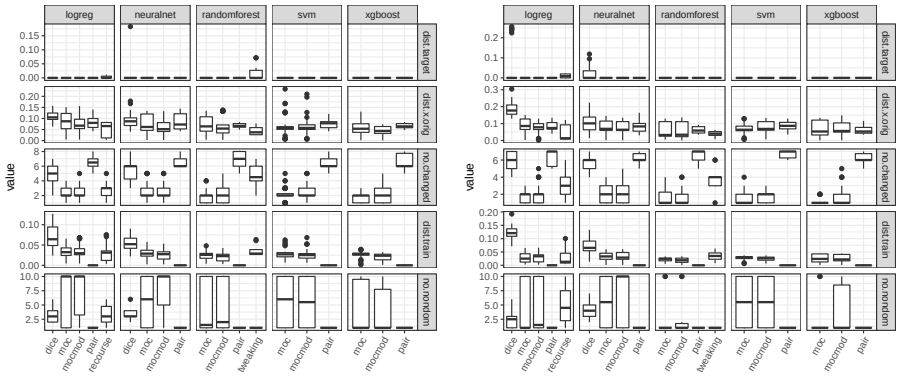
7 Results

Q1) MOC vs. state-of-the-art counterfactual methods

Table 2 shows the coverage rate of the methods to be compared by the modified 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 nearly all counterfactuals of DiCE for all datasets. The same holds for Tweaking except for the datasets kr-vs-kp and tic-tac-toe. The coverage rate for these datasets is (close to) 0 because the counterfactuals of Tweaking had the same objective values as the ones of MOC. MOC's coverage rate of Recourse was greater than 70 % except for pc1 where only 50 % of the counterfactuals were dominated.

Figure 2 compares MOC (with (*mocmod*) and without (*moc*) the modifications) with the other methods for the diabetes and no2 datasets and for each model separately. The resulting boxplots for all other datasets are shown in Figures 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, always met the target (except for two counterfactuals of kr-vs-kp) and

[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 (*no.nondom*) per model for MOC, MOC with modified initialization and mutation strategies (*mocmod*), Pair, DiCE, Recourse and Tweaking for the diabetes and no2 datasets. Lower values are better except for *no.nondom*.

changed the least features. DiCE performed worse than MOC in all objectives, especially in the number of changed features and the distance to the nearest observed data point. Pair, Tweaking and Recourse performed better than MOC for the distance to $\mathbf{x}^{orig}$, but more features were changed and fewer nondominated solutions were found. Recourse's counterfactuals were on average closer to the observed data, but for kc2, boston and no2 some counterfactuals did not meet the target. Tweaking's counterfactuals sometimes did not reach the desired outcome because they stayed close to $\mathbf{x}^{orig}$. The modified MOC (*mocmod*) often found more nondominated counterfactuals and its counterfactuals were often closer to $\mathbf{x}^{orig}$ and the observed data compared to MOC without the modifications.

Q2) MOC initialization and mutation strategies

Figure 3 shows the ranks of the dominated hypervolumes for MOC without modifications, for each modification of MOC and random search. Ranks were calculated per dataset, model, $\mathbf{x}^{orig}$ and generation, and were averaged over all datasets, models and $\mathbf{x}^{orig}$. We transformed hypervolumes to ranks because the achieved hypervolumes are not comparable across datasets. It can be seen that the proposed modifications clearly outperforms the standard MOC. The ranks of the initial population were higher when the ICE curve variance was used to initialize the candidates. The use of the conditional distributions for the mutation of features led to higher dominated hypervolumes over the generations. We received best performance over the generations when both modifications were used. At each generation, all versions of MOC clearly outperformed random search. In Figure 6 in the Appendix, the dominated hypervolumes over the generations are separately shown for each dataset. They largely agree with the results shown here. The performance gains of MOC compared to random search were particularly evident for higher-dimensional datasets.

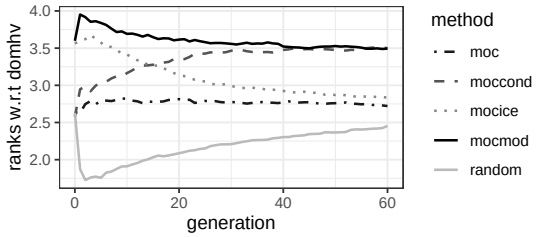


Fig. 3. Comparison of the ranks w.r.t. the dominated hypervolume (*domhv*) per generation averaged over all models and datasets. For each approach, the population size of each generation was 59. A higher hypervolume and therefore a higher rank is better. Legend: *moc*: MOC without modifications; *moccond*: MOC with the mutator based on conditional distributions; *mocice*: MOC with the ICE curve variance initialization; *mocmod*: MOC with both modifications; *random*: random search.

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 is model-agnostic, suited for classification and regression, handles mixed feature spaces and returns a diverse set of counterfactuals with different trade-offs between our four proposed objectives. We modified the algorithm to accelerate the optimization and to obtain plausible counterfactuals according to user preferences and to the distribution of the observed data. We demonstrated the usefulness of MOC to explain a prediction on the German credit dataset and showed in a benchmark study that MOC is able to find more counterfactuals than other counterfactual methods. In addition, the counterfactuals produced by MOC had fewer features changes and their predictions almost always met the desired outcome. Our suggested modifications of MOC resulted in higher performance in less evaluations and in counterfactuals that were closer to the original data point and to the observed data.

MOC has only been evaluated on binary classification, and only with respect to the dominated hypervolume 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 is available at `https://github.com/susanne-207/moc`. The implementation of MOC is based on our implementation of [17], which we also used for [**?**]. MOC will be integrated into the `iml` R package [**?**].

References

1. Allaire, J., Chollet, F.: keras: R Interface to 'Keras' (2019), https://keras.rstudio.com, r package version 2.2.5.0
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. Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G., Jones, Z.M.: mlr: Machine Learning in R. Journal of Machine Learning Research **17**(170), 1–5 (2016), http://jmlr.org/papers/v17/15-066.html
4. Breiman, L.: Statistical Modeling: The Two Cultures. Statistical Science **16**(3), 199–231 (08 2001). https://doi.org/10.1214/ss/1009213726, https://doi.org/10.1214/ss/1009213726
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (April 2002). https://doi.org/10.1109/4235.996017
6. 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
7. Goldstein, A., Kapelner, A., Bleich, J., Pitkin, E.: Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation. Journal of Computational and Graphical Statistics **24**(1), 44–65 (2015). https://doi.org/10.1080/10618600.2014.907095, https://doi.org/10.1080/10618600.2014.907095
8. Gower, J.C.: A General Coefficient of Similarity and Some of its Properties. Biometrics **27**(4), 857–871 (1971)
9. Grath, R.M., Costabello, L., Van, C.L., Sweeney, P., Kamiab, F., Shen, Z., Lécué, F.: Interpretable Credit Application Predictions With Counterfactual Explanations. CoRR (abs/1811.05245) (2018), http://arxiv.org/abs/1811.05245
10. Greenwell, B.M., Boehmke, B.C., McCarthy, A.J.: A simple and effective model-based variable importance measure. arXiv preprint arXiv:1805.04755 (2018)
11. Hofmann, H.: German Credit Risk (2016), https://www.kaggle.com/uciml/german-credit, last access 25.01.2020
12. Hothorn, T., Zeileis, A.: Transformation Forests (2017)
13. 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
14. 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
15. Kingma, D., Ba, J.: Adam: A Method for Stochastic Optimization. International Conference on Learning Representations (12 2014)
16. 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
17. Li, R., Emmerich, M.T., Eggermont, J., Bäck, T., Schütz, M., Dijkstra, J., Reiber, J.H.: Mixed Integer Evolution Strategies for Parameter Optimization. Evolutionary Computation **21**(1), 29–64 (2013)
18. Looveren, A.V., Klaise, J.: Interpretable Counterfactual Explanations Guided by Prototypes. CoRR **abs/1907.02584** (2019), http://arxiv.org/abs/1907.02584

19. López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace Package: Iterated Racing for Automatic Algorithm Configuration. Operations Research Perspectives **3**, 43 – 58 (2016). https://doi.org/https://doi.org/10.1016/j.orp.2016.09.002, http://www.sciencedirect.com/science/article/pii/S2214716015300270
20. Lundberg, S.M., Lee, S.I.: A Unified Approach to Interpreting Model Predictions. In: Advances in Neural Information Processing Systems. pp. 4765–4774 (2017)
21. 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
22. PAIR: What-If Tool (2018), https://pair-code.github.io/what-if-tool/, last access 31.01.2020
23. Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., Bie, T.D., Flach, P.: FACE: Feasible and Actionable Counterfactual Explanations (2019)
24. 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.) Evolutionary Multi-Criterion Optimization. pp. 825–840. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
25. 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)
26. Russell, C.: Efficient Search for Diverse Coherent Explanations. CoRR (abs/1901.04909) (2019), http://arxiv.org/abs/1901.04909
27. 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
28. Su, J., Vargas, D.V., Sakurai, K.: One Pixel Attack for Fooling Deep Neural Networks. IEEE Transactions on Evolutionary Computation **23**, 828–841 (2017)
29. 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. pp. 465–474. KDD '17, ACM, New York, NY, USA (2017). https://doi.org/10.1145/3097983.3098039, http://doi.acm.org/10.1145/3097983.3098039
30. Ustun, B., Spangher, A., Liu, Y.: Actionable Recourse in Linear Classification. In: Proceedings of the Conference on Fairness, Accountability, and Transparency. pp. 10–19. FAT* '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3287560.3287566, http://doi.acm.org/10.1145/3287560.3287566
31. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: Networked Science in Machine Learning. SIGKDD Explorations **15**(2), 49–60 (2013). https://doi.org/10.1145/2641190.2641198, http://doi.acm.org/10.1145/2641190.2641198
32. 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
33. White, A., d'Avila Garcez, A.: Measurable Counterfactual Local Explanations for Any Classifier (2019)

34. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms — a Comparative Case Ctudy. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.P. (eds.) Parallel Problem Solving from Nature — PPSN V. pp. 292–301. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)

A Illustration of MOC's Benefits

This section illustrates the benefits of having a *set* of counterfactuals, as opposed to a single counterfactual. We use the diabetes dataset as a first example, with logistic regression as a classifier as in the benchmark study (Section 6). Table 3 compares the set of counterfactuals returned by MOC to the two returned counterfactuals by Recourse [30] for the prediction of observation 59. The logistic regression predicted a probability of having diabetes of 0.86 for this observation. The desired target is a prediction of less than 0.5. Recourse suggested that age

Table 3. Counterfactuals and corresponding objective values of MOC and Recourse for the prediction of a logistic regression for observation 59 of the diabetes dataset. Orange indicates an increase in the suggested value, blue a decrease compared to the original value of observation 59 (shown in parentheses).

Feature ($\mathbf{x}^{orig}$)	MOC_1	MOC_2	MOC_3	MOC_4	MOC_5	MOC_6	MOC_7	$Recourse_1$	$Recourse_2$
preg (0)	0	0	0	0	0	0	0	0	0
plas (146)	71.96	37.98	74.05	72.30	73.32	139.47	74.21	57	57
pres (82)	82	82	82	82	82	82	82	82	82.680
skin (0)	0	0	0	0	0	0	0	0	0
insu (0)	0	0	0	0	0	0.009	0	0	0
mass (40.5)	40.5	40.5	40.5	40.5	40.5	40.470	40.5	40.5	40.5
pedi (1.781)	1.040	1.781	0.985	1.042	0.988	0.289	0.791	1.781	1.781
age (44)	44	44	44	44	44	30.489	44	34.8	35.260
dist.target	0	0	0	0	0	0	0	0	0
dist.x.orig	0.086	0.068	0.088	0.086	0.088	0.112	0.098	0.075	0.075
nr.changed	2	1	2	2	3	5	2	2	3
dist.train	0.068	0.068	0.066	0.068	0.066	0.017	0.056	0.075	0.075

and plasma glucose concentration (*plas*), should be lowered, and blood pressure (*pres*) should be minimally increased (which is counterproductive) to reduce the probability of diabetes. In contrast, MOC returned a set of seven counterfactuals that provide more options for actionable user responses and are closer to the observed data than Recourse's counterfactuals (*dist.train*). Counterfactual MOC_2, which only suggested a reduction in plasma glucose concentration, has overall lower objective values than both counterfactuals of Recourse. Counterfactual MOC_6 suggested changes to five features so that it is especially close to the nearest training data point (*dist.train*).

A second example is the cmc dataset, where the current "contraceptive method choice" of a woman is predicted with a random forest, based on her and her families characteristics. This task and model were also used in the benchmark study (Section 6). Table 4 compares the set of counterfactuals found by MOC with the single counterfactual found by Tweaking [29] for the prediction of observation 379. The random forest predicted a probability of taking no contraceptive of 0.75

Table 4. Counterfactuals and corresponding objective values given by MOC and Tweaking for the prediction of a random forest for observation 379 of the cmc dataset. Orange indicates an increase in the feature value, blue a decrease compared to the original value of observation 379 (given in parentheses).

Feature ($\mathbf{x}^{orig}$)	MOC_1	MOC_2	MOC_3	MOC_4	$Tweaking_1$
Wifes_age (18)	18	34.17	19.06	18	17.45
Wifes_education (2)	2	2	2	3	2
Husbands_education (3)	3	3	4	3	3
Number_of_children_ever_born (1)	1.5	1.66	1	1	1.71
Wifes_religion (1)	1	1	1	1	1
Wifes_now_working. (1)	1	1	1	1	1
Husbands_occupation (2)	2	2	2	2	2
Standard.of.living_index (3)	3	3	3	2	3
Media_exposure (0)	0	0	0	0	0
dist.target	0	0	0	0	0
dist.x.orig	0.003	0.059	0.115	0.222	0.007
nr.changed	1	2	2	2	2
dist.train	0.003	0.003	0	0	0.007

for this observation. The desired target is a prediction of less than 0.5. Tweaking suggested reducing the woman's age but at the same time having another child to increase the probability of using a contraceptive.[6] This is contradictory and not plausible. In contrast, MOC returns a set of counterfactuals that are more plausible (e.g., increase in births accompanied by an age increase) and indicate various strategies (e.g., increase the husband's or wife's education). In addition, MOC_1 and MOC_2 dominate the counterfactual of Tweaking.

B Iterated F-racing

We used iterated F-racing (irace) [19] to tune the parameters of MOC for binary classification (`irace` R package version 3.3). 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 $\mathbf{x}^{orig}$ and a desired outcome. To obtain 300 instances, we randomly sampled 10 data points as $\mathbf{x}^{orig}$ per trained model. The desired target for each $\mathbf{x}^{orig}$ was the opposite of the predicted class: $Y' = \mathbb{I}_{\hat{f}(\mathbf{x}^{orig})<0.5}$. The trained

[6] By reclassifying age and number of births as integers (instead of decimals) in the cmc dataset, integer changes would be recommended.

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]	59
p.rec	Probability a pair of parents is chosen to recombine	[0.3, 1]	0.61
p.rec.gen	Probability a feature is recombined	[0.3, 1]	0.55
p.rec.use.orig	Probability indicator for feature changes is recombined	[0.3, 1]	0.72
p.mut	Probability a child is chosen to be mutated	[0.05, 0.8]	0.69
p.mut.gen	Probability one feature is mutated	[0.05, 0.8]	0.25
p.mut.use.orig	Probability indicator for feature change is flipped	[0.05, 0.5]	0.19

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 [31] and are briefly described in Table 7. The datasets were not used in the benchmark study (Section 6). Numerical features were standardized (Z-score) and categorical features were one-hot encoded. We used random search (with 200 iterations for neural networks and 100 iterations for all other models) and 5-fold cross-validation with the misclassification error as the performance measure to tune the hyperparameters of the models. The search space is given in Table 6. For neural network and logistic regression, ADAM [15] was the optimizer, the batch size was 128 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, the default values of the `mlr` (version 2.16.0) and `keras` (version 2.2.5.0) R packages were chosen [3,1].

In each step of irace, parameter configurations were evaluated by running MOC on the same selected instance. MOC was configured without the modifications of Section 4.3 and stopped after evaluating 15000 candidates with Eq. (1). We investigated on 150 newly sampled instances (five newly sampled $\mathbf{x}^{orig}$ per trained model and dataset) that 15000 evaluations should be enough to ensure convergence of the hypervolume in most cases. The integral of the first order spline approximation of the dominated hypervolume over 15000 evaluations was

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

the performance criterion [24]. The integral takes into account not only the extent but also the rate of convergence of the dominated hypervolume.

The resulting performance values of each configuration were used for a Friedman-test to discard less promising configurations. The first Friedman test was conducted after initial configurations were evaluated on 15 instances. This means that each configuration was evaluated using 15 different models and $\mathbf{x}^{orig}$ before some configurations were excluded. After the first Friedman-test, tests were conducted after evaluating the remaining configurations on a single instance to accelerate the exclusion process. The best configuration returned by irace is given in Table 5.

To obtain a default parameter for the number of generations, we determined for 150 newly sampled $\mathbf{x}^{orig}$ (five newly sampled $\mathbf{x}^{orig}$ per trained model and dataset) after how many generations of the tuned MOC the dominated hypervolume does not increase for 10 generations. 60 generations were chosen as default which is the 80 % quantile of all 150 returned values.

C Parameters for Benchmark Study

C.1 Hyperparameters of Models

We used random search (with 200 iterations for neural networks and 100 iterations for all other models) and 5-fold cross-validation (with misclassification error as performance measure) to tune the hyperparameters of the models on the training data. We chose the same tuning search space as for iterated F-racing (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 [15] 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` and

`keras` R packages [3,1]. Table 8 shows the accuracies of the trained models on the test data (20 % hold-out).

Table 8. Accuracy on test data per benchmark data set and model. Legend: *Name:* OpenML task name; *rf:* random forest; *logreg:* logistic regression. Logistic regression was only trained on datasets with numerical or binary features.

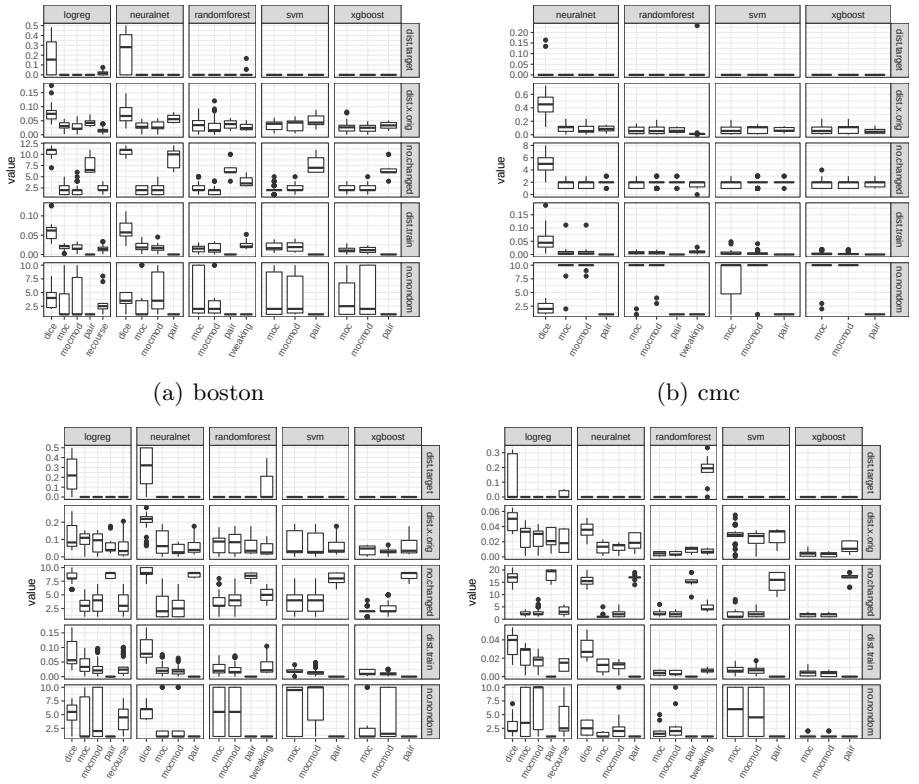
Name	rf	xgboost	svm	logreg	neuralnet
boston	0.87	0.89	0.86	0.91	0.85
cmc	0.69	0.74	0.67		0.61
diabetes	0.75	0.69	0.74	0.75	0.77
ilpd	0.67	0.70	0.70	0.56	0.58
kc2	0.80	0.80	0.81	0.80	0.69
kr-vs-kp	0.99	0.99	0.99		0.99
no2	0.56	0.61	0.48	0.59	0.54
pc1	0.94	0.94	0.88	0.89	0.80
plasma_retinol	0.57	0.59	0.70		0.62
tic-tac-toe	0.98	0.98	0.98		0.96

C.2 Parameters of Counterfactual Methods

For Tweaking [29], we only changed ϵ, a positive threshold that bounds 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 [30], we left all parameters at their default settings. We used the Python implementation of Recourse on Github: `https://github.com/ustunb/actionable-recourse` (commit `aaae8fa`).

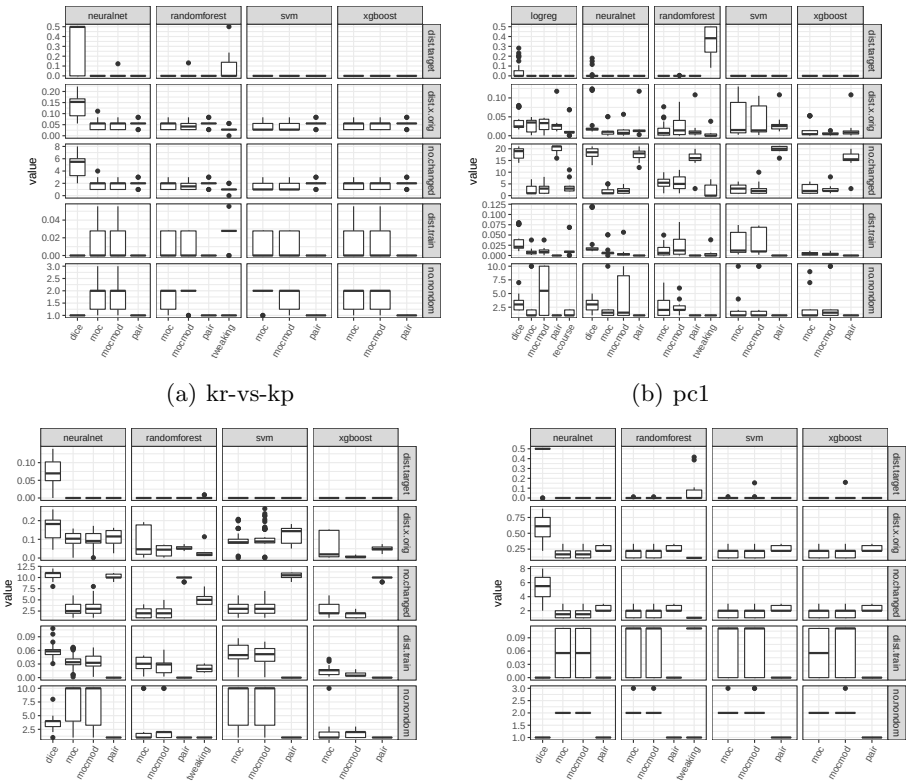
For DiCE [21], we used the 'DiverseCF' version proposed by the authors [21] and left the hyperparameters at their defaults. We used the inverse mean absolute deviation for the feature weights. For datasets where the mean absolute deviation of a feature was zero we set the feature weight to 10. We used the Python implementation of DiCE available on Github: `https://github.com/microsoft/DiCE` (commit `7bcb098`).



(a) boston (b) cmc

(c) ilpd (d) kc2

Fig. 4. Boxplots of the objective values and number of nondominated counterfactuals (*no.nondom*) per model for MOC, MOC with modified initialization and mutation strategies (*mocmod*), Pair, DiCE, Recourse and Tweaking for the boston, cmc, ilpd and kc2 datasets. Lower values are better except for *no.nondom*.



(a) kr-vs-kp

(b) pc1

(c) plasma_retinol

(d) tic-tac-toe

Fig. 5. Boxplots of the objective values and number of nondominated counterfactuals (*no.nondom*) per model for MOC, MOC with modified initialization and mutation strategies (*mocmod*), Pair, DiCE, Recourse and Tweaking for the kr-vs-kp, pc1, plasma_retinol and tic-tac-toe datasets. Lower values are better except for *no.nondom*.

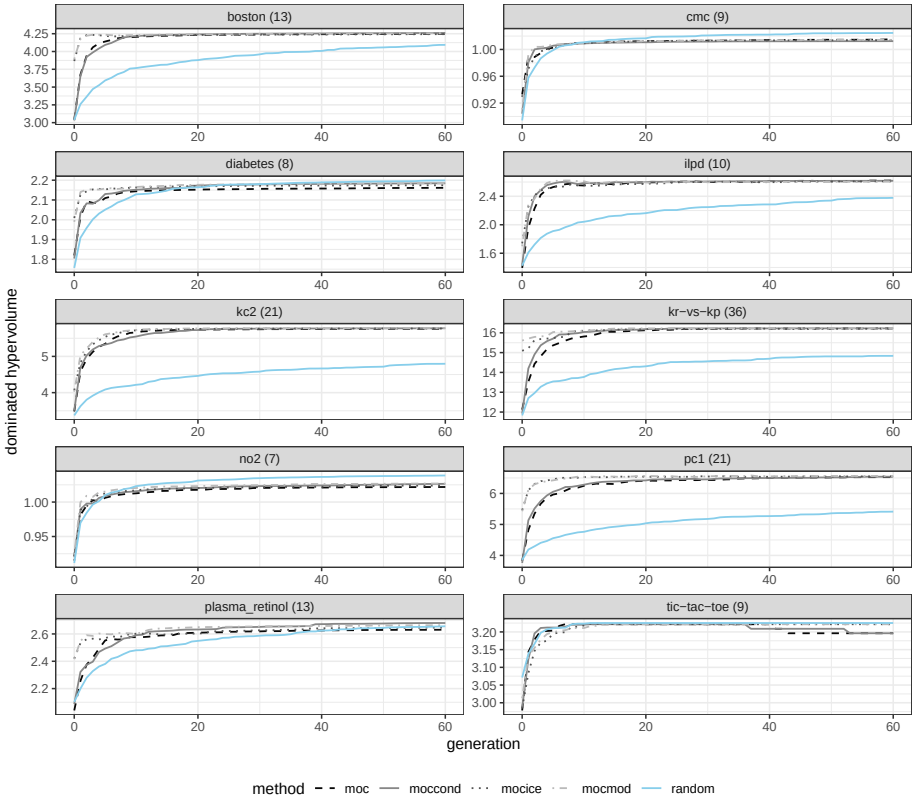


Fig. 6. Comparison of the mean dominated hypervolume per generation and per benchmark dataset averaged over all models. The numbers in parentheses indicate the number of features. For each approach, the population size of each generation was 59. Higher values of the hypervolume are better. Legend: *moc*: MOC without modifications; *moccond*: MOC with the mutator based on conditional distributions; *mocice*: MOC with the ICE curve variance initialization; *mocmod*: MOC with both modifications; *random*: random search.