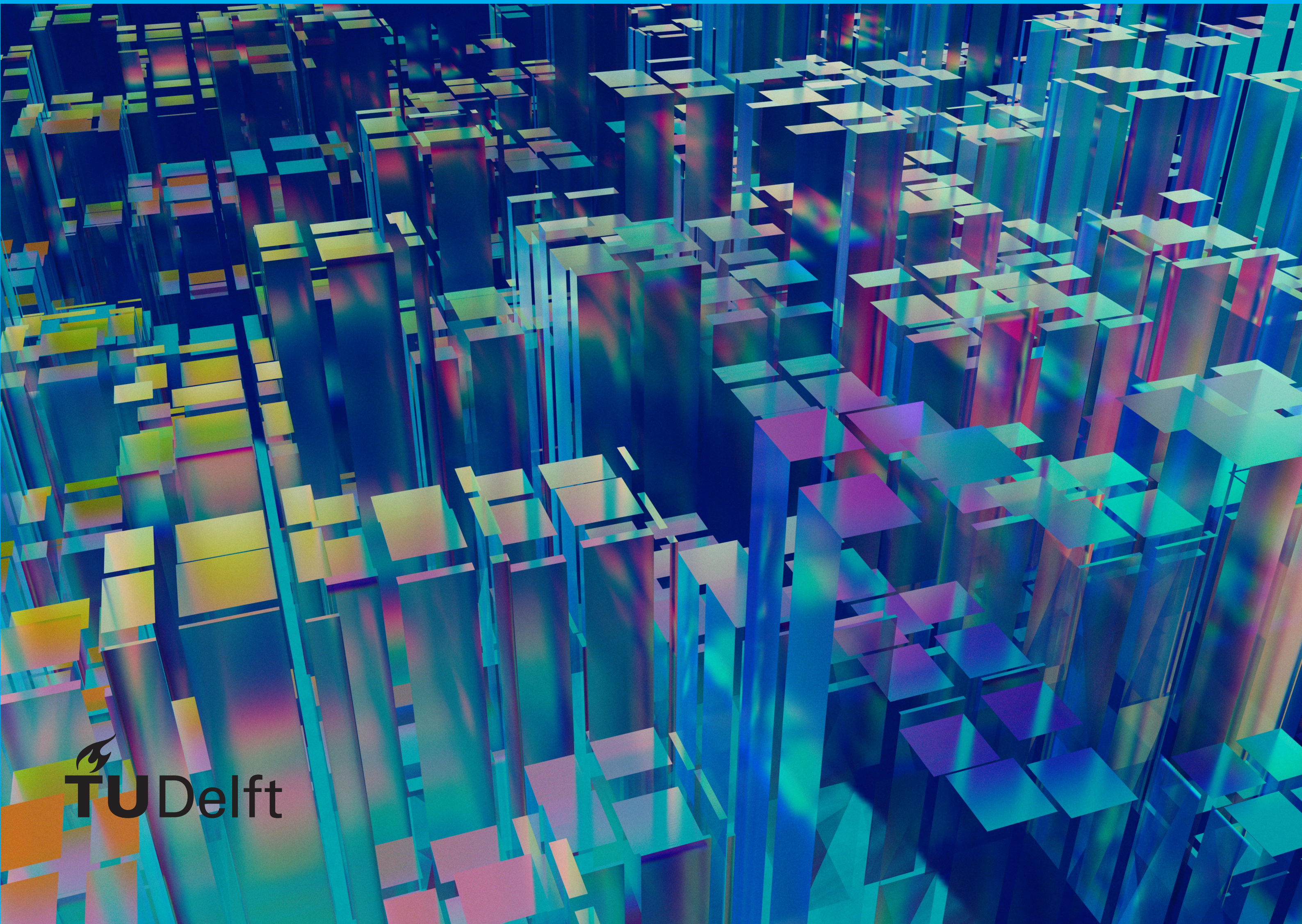
Generating Diverse Counterfactuals through Evolutionary Multi-Objective Optimization

M.R. Tromp



Generating Diverse Counterfactuals through Evolutionary Multi-Objective Optimization

by

M.R. Tromp

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 29, 2024 at 14:00.

Student number:	4814495	
Project duration:	November 13, 2023 – August 29, 2024	
Thesis committee:	Prof. dr. P.A.N. Bosman	TU Delft, Algorithmics, supervisor
	Dr. T. Alderliesten	LUMC, Radiation Oncology, supervisor
	Dr. C.C.S. Liem	TU Delft, Multimedia Computing
	E.M.C. Sijben, MSc.	CWI, Daily supervisor

TUDelft

Preface

With this thesis, my time at Delft University of Technology comes to an end. Ever since my bachelor's thesis in genetic programming I knew that I wanted my master's thesis to be in evolutionary algorithms. This was confirmed when I followed the evolutionary algorithms course, and I am grateful for the opportunity to be able to follow up on this during my thesis project. I would like to thank some people for their guidance and support during my thesis project.

First I want to thank Peter Bosman and Tanja Alderliesten for their expertise and flexibility. I also would like to thank Evi Sijben for her feedback, insights, and the many meetings. Additionally, I want to thank Marharyta Domnich for her guidance and feedback. Furthermore, I would like to thank Cynthia Liem for being on my thesis committee.

Finally, I want to thank my family and friends for their support throughout my time in university.

M.R. Tromp
Delft, August 2024

Abstract

Counterfactual explanations are a useful tool to explain trained models. They are based on counterfactual thoughts, which are a natural human thought process that helps us reason about the past. When applied to trained models they show how to make minimal changes to a data point in order to obtain a desired output.

Most methods find these counterfactuals by optimizing a set of objectives. Previously these objectives were often combined into a loss function using an aggregation operator. This operator implicitly decides the priority between the objectives, but this ordering is not always in line with the user's preferences.

To mitigate this the Multi-Objective Counterfactuals (MOC) method was introduced. MOC turns counterfactual generation into a multi-objective optimization problem and presents the user with a diverse set of counterfactuals that have different trade-offs for the objectives. It optimizes the set of objectives with an evolutionary algorithm called Nondominated Sorting Genetic Algorithm II.

In this thesis we optimize this problem using Multi-Objective Real-Valued Gene-Pool Optimal Mixing Evolutionary Algorithm, which is a different evolutionary algorithm. We present a single-modal method and two multi-modal methods. We compare the performance of our methods to a counterfactual generation method named Diverse Counterfactual Explanations (DiCE), which focusses on feasibility and diversity within a set of generated counterfactuals. Additionally, we also present a visualization tool for sets of counterfactuals.

The single-modal method generates counterfactuals that are realistic, but do not consistently perform well in other areas. The first multi-modal method generates diverse sets of counterfactuals, but overall performs worse. The second multi-modal method generates counterfactuals that perform similarly to the single-modal method, but are more diverse.

Contents

1 Introduction

With the ever-increasing prevalence of Artificial Intelligence (AI) in our daily lives, for example in healthcare [1], the workplace [17], or the military [33], the global population has grown more nervous of products and services using AI [25]. The European Union has introduced the AI Act to try to increase the trust in and transparency of decisions and predictions made by AI models [9]. Similarly, this need for trustworthy AI is also visible in other governments, like in an Executive Order in the United States of America [24]. The field that aims to make AI models more explainable is called eXplainable AI (XAI). One such XAI technique consists of counterfactual explanations.

Counterfactual thoughts help us explain our own past actions and sequences of past events [7]. Imagine you made the new year's resolution that you will cycle to work instead of driving. The first and second day the weather is nice, and you commit to the resolution. However, on the third day it so happens to be raining so you decide to drive instead. Later that day, when you ask yourself why you already broke your new year's resolution on day three, you tell yourself: "If it had been sunny, I would have cycled to work". This is an example of a counterfactual thought.

When applied to XAI, counterfactuals take the form of changes to an original data point such that the prediction of the model changes. To generate useful counterfactuals, we need to take many different objectives into account. Each of these objectives tries to ensure a different element we would like to see in a counterfactual. These objectives can include how similar the counterfactual is to the original data point, or the number of feature values that are changed in the counterfactual with regard to the original data point [21, 27, 28]. These objectives then have to be combined to generate counterfactuals. Many methods do this by combining the chosen objectives into an aggregated cost function, but because the aggregation operator implicitly chooses how the objectives are prioritized this can have a significant impact on the returned counterfactuals [28]. It is often the case that methods provide the user with only one counterfactual, like the methods introduced by Wachter et al. [43], FACE [35], and DACE [26]. This can be a problem, because this implicit priority can be different from the user's preferences. Say we have two different but equally plausible counterfactuals, one which changes all feature values a small amount and one that changes only two feature values a large amount. It is impossible to decide which counterfactual the user would prefer without knowledge about the user.

To mitigate both of these problems, multi-objective counterfactual generation was introduced in the form of Multi-Objective Counterfactuals (MOC) [10]. Instead of aggregating the objectives, the objectives are optimized with a multi-objective evolutionary algorithm. The algorithm returns a set of solutions that each have different trade-offs for the optimized objectives. This allows the user to choose which counterfactual is best for their use case.

We use a different multi-objective evolutionary algorithm called MO-RV-GOMEA [5] to generate counterfactuals in a multi-objective way. Each generation it performs clustering within the population. Within these clusters it exploits linkage information between variables to perform better variation and keeps track of a set of individuals that is diverse in objective values within elitist archives. This allows it to find a diverse set of solutions that is close to being optimal.

We define the following research question: *Can we use MO-RV-GOMEA to present the user with a set of diverse and relevant counterfactuals that are generated in a multi-objective way?*

This research question can be divided into the following sub-questions:

- Q1 - Is the standard MO-RV-GOMEA implementation combined with the objectives from MOC enough to guarantee diversity within the generated counterfactuals while keeping them relevant for the user?
- Q2 - Can we create a diversity objective for MO-RV-GOMEA to optimize in combination with the other objectives?
- Q3 - How can we visualize the set of returned counterfactuals for the user?

In this thesis we present three different methods that generate counterfactuals using MO-RV-GOMEA. One of these methods is a single-modal method, the other two are multi-modal methods. The single-modal method optimizes objectives based on MOC. Both multi-modal methods optimize (a subset of) these same objectives, while additionally optimizing a new diversity objective. We also create a visualization tool to visualize the generated counterfactuals.

The single-modal method is able to generate realistic counterfactuals but can lack diversity within the generated set of counterfactuals. The first multi-modal method generates sets of counterfactuals that are relatively diverse but under-performs on all other metrics. The second multi-modal method is able to generate sets of counterfactuals that perform similar to the single-modal method on all metrics, while also being more diverse.

This thesis is structured as follows. In Chapter 2 we discuss background and literature relevant to this research. Then, Chapter 3 describes our methodology. This is followed by Chapter 4, which explains our experimental setup and design. The results of these experiments are shown and discussed in Chapter 5. We then present limits to this research in Chapter 6 and draw conclusions in Chapter 7. Lastly, we present possible future work in Chapter 8.

2 Background and Literature Overview

In this chapter we provide the necessary background for this thesis and discuss some relevant literature. Section 2.1 discusses the concept of counterfactuals, why they are necessary, and how they can be generated. In Section 2.2 we describe what Evolutionary Algorithms are, and in Section 2.3 we describe the family of Evolutionary Algorithms that we use in this thesis. The last Section, Section 2.4, touches on symbolic regression.

2.1. Counterfactuals

In this Section we give the background and literature overview related to counterfactuals. First, we give some background on what Explainable Artificial Intelligence (XAI) is and why it is necessary. Then we explain what counterfactual thoughts are and how they can be applied to XAI. After this we talk about how we can generate these counterfactuals and the common objectives that are optimized for. Then we touch on some methods that generate counterfactuals in a multi-objective way, and last we give a short overview of some methods that are used to visualize these counterfactuals.

2.1.1. Explainable Artificial Intelligence

Artificial Intelligence (AI) has made its way into large parts of our daily lives, often in the form of complex models that are not understandable to humans. Some of these applications include healthcare [1], the workplace [17], and the military [33]. While the reported understanding of AI amongst the global population has increased over the past years, this increased understanding also comes with an increased nervousness of products and services using AI [25].

To try to increase the trust in AI, the European Union introduced the AI Act, which will become fully applicable in the coming years [9]. They state that a reason that the legislation in the AI Act is necessary is that ”it is often not possible to find out why an AI model has made a decision or prediction and taken a particular action” [9]. This need for trustworthy AI is also reflected by the governments of other countries, like in an Executive Order in the United States of America [24].

One of the key dimensions of responsibly developing and deploying AI is the explainability of said models [32]. The field that is concerned with this explainability is called XAI. XAI aims to make AI models more understandable using explanations for decisions made [22]. Thus, the aim of XAI techniques is to explain how an AI model has made a certain decision and therefore hopefully increase the trustworthiness of these models.

2.1.2. Counterfactual thoughts for XAI

Have you ever wondered what would have happened if you had taken a different action in a certain situation? These thoughts, like ”If I had worked harder, I would have received a higher bonus” or ”If it had been sunny, I would have cycled to work” are examples of counterfactual thoughts.

Counterfactual thoughts are thoughts about the past that provide alternatives to reality, where these alternatives often evaluate outcomes that are either better or worse than what actually happened [13]. When we reason about alternatives to the past, these counterfactuals serve many purposes, including explaining why events happened or why we took certain actions [7]. Our previous examples explain why the person did not get a raise; and why the person decided not to cycle to work.

This type of thinking can also be applied to the future, resulting in thoughts like ”If I work hard this month, I will receive a higher bonus” or ”If I plant an apple tree now, it will grow fruit a year earlier”. These types of thoughts are called prefactual thoughts, which are used by people to plan for the future [7]. Prefactuals were originally defined to be a variant of counterfactuals [14].

Applying the concept of counterfactual thoughts to explain AI models is part of XAI. These counterfactuals capture the change in model output when an original data point changes. When the model output sufficiently changes, e.g. the predicted class changes for classification tasks or the predicted value is within a certain range for a regression task, that (changed) data point is a counterfactual for the original data point. Because counterfactuals reflect a human thought process, they can be very intuitive when used to explain AI models.

Counterfactuals are a form of post-hoc local explanations, meaning that they are used on trained models and are meant to explain a single individual person or data point [21]. Therefore, counterfactual explanations are able to explain any type of model and thus are a model-agnostic method. Note that some people use the term contrastive explanation interchangeably with counterfactuals [39]. Others define contrastive explanations to focus on the difference between possible outcomes, while they define counterfactuals to focus on changes to past situations that then lead to a different outcome [39].

2.1.3. Counterfactual generation

There are many surveys that give an overview of recent developments in counterfactual generation. Using some of these surveys we will first explain several visions on what the most important parts of counterfactuals are, then we will discuss different objectives that are often mentioned in literature, and last we will go through how these objectives are often optimized to generate counterfactuals.

In [21] Guidotti argues that a counterfactual should make the minimal changes possible to the original data point in order to become a counterfactual, but they do not define what minimal means. Karimi et al. [27] agree that a counterfactual needs to make the minimal changes possible to the original data point. While counterfactual thoughts are generally applied to both positive and negative situations [13], these authors, Epstude and Roese, focus on using counterfactuals to help users change their negative situation for the better. They call this algorithmic recourse. They argue that counterfactuals consist of two parts: a counterfactual explanation and a counterfactual recommendation. The counterfactual explanations are the changes made to the original input point and the counterfactual recommendation are the actions that the user has to take to get there. While Laugel et al. [28] state that minimal changes are most commonly optimized when generating counterfactuals, they focus on diversity in counterfactuals and argue that it is beneficial to present the user with multiple diverse counterfactuals because needs are specific to that user.

Other than making as few changes as possible to the original data point, there are other objectives that can be optimized when generating counterfactuals. In [21] Guidotti describes the following common types of objectives:

- Validity - Whether the counterfactual results in a prediction within the desired range or class;
- Sparsity - The number of features with different values when comparing the counterfactual and the original data point;
- Proximity - The distance between the counterfactual and the original input point;
- Plausibility - Whether the feature values for the counterfactual are coherent with the data set;
- Discriminative power - When comparing the counterfactual and the original data point humans should be able to classify them differently;
- Actionability - The number of changed features of the counterfactual that are actionable. E.g. if the counterfactual tells the user to change reduce their age, this is not actionable;
- Causality - Whether the counterfactual captures causal relationships between features;
- Diversity - How different the generated counterfactuals are.

In this thesis we will use these definitions to refer to these types of objectives.

Karimi et al. [27] describe objectives and constraints that are similar to some of the ones in this list. They state that the contrastive explanations are found using a proximity objective, while the minimal consequential recommendations are found by minimizing a cost function. This cost function tries to capture the effort it would take for the user to follow the recommendation. The other objectives that they describe are plausibility, actionability, diversity, and sparsity.

In [28] Laugel et al. create different groups of types of objectives. In their general type they include the proximity and sparsity objectives. Their data-contextualisation criteria group consists of objectives that try to make sure counterfactuals are realistic and understandable to humans and include objectives similar to plausibility which are aimed to keep the counterfactuals close to the original data set. The last group of objectives they define is called user-centered contextualisation criteria, which are objectives that capture how well suited a counterfactual is for the user it was generated for. This group includes the actionability and causality objectives.

Laugel et al. also describe three different types of diversity. The first type is diversity in criteria, which is diversity that is based on the different objectives that are optimized for. The second type is diversity in the feature space, which is based on diversity of where the counterfactuals are located in the feature space. The third type is diversity in actions, which is based on diversity in the actions it takes for the user to get to the situation described by the counterfactual.

To generate counterfactuals, models have to somehow combine the objectives. Laugel et al. [28] describe that it is often the case that (a subset of) these objectives are combined into an aggregated cost function. Both Laugel et al. [28] and Guidotti [21] state that it is not possible to guarantee all objectives at once. Therefore Laugel et al. [28] argue that the aggregation operator has a large impact on the returned counterfactual(s), since that operator decides how the objectives are prioritised. They also state that while many counterfactual generation methods return only one counterfactual, it is beneficial to provide multiple counterfactuals to the user to give the user a choice in which counterfactual they prefer based on what they find important in a counterfactual.

DiCE is a counterfactual generation method focused on the feasibility and diversity of a set of generated counterfactuals [34]. They focus on the trade-off between diversity and proximity and allow the user to give domain knowledge, like feature weights and constraints, as inputs. In their paper they provide metrics to evaluate their method. These are the metrics that we use to evaluate our method as well.

The authors of DiCE filter the generated counterfactuals by creating causality constraints based on user input to ensure that the generated counterfactuals are feasible. They optimize for diversity by building on determinantal point processes and for proximity using the negative average distance to the original data point for a set of counterfactuals. These are combined into one loss function, with a separate multiplier for each. The values of these multipliers can be set by the user. While they do take sparsity into account, they do not optimize for it, instead opting to modify generated counterfactuals.

We use DiCE as a baseline to compare our method against. However, unlike our method, DiCE works for both classification and regression models. We also use the metrics used to evaluate DiCE to evaluate our method.

2.1.4. Multi-objective counterfactual generation

Instead of optimizing for an aggregated objective, like a weighted sum of objectives, the authors of [10] changed counterfactual generation into a multi-objective optimization problem and propose the Multi-Objective Counterfactuals (MOC) methods. They optimize a multi-objective minimization task with the following objectives:

1. Distance to becoming a counterfactual (similar to Validity);
2. Proximity;
3. Sparsity;
4. Distance to k nearest observed data points.

Whenever they are calculating the distance between two data points, they use the Gower distance [20]. The algorithm they use to find counterfactuals is called Nondominated Sorting Genetic Algorithm II (NSGA-II) [12]. NSGA-II is a multi-objective Evolutionary Algorithm (EA) that uses fitness based on non-dominated sorting to ensure elitism and then uses crowding-distance sorting to preserve diversity. The authors of MOC modified NSGA-II to work for both discrete and continuous values using mixed integer evolutionary strategies (MIES) [29]. Some of the modifications the authors made are used to achieve more diversity in feature space and actionability. They use a different crowding-distance sorting algorithm for the crowd-comparison. To improve diversity, they compute the crowding distance in both objective space and feature space. They show that MOC outperforms other state-of-the-art methods on all objectives and the number of non-dominated solutions returned. However, we do have to note that these results could be skewed in favour of MOC, since MOC optimizes for these objectives and the other methods might not.

The base set of objectives that we optimize are the same as the objectives presented in the MOC method, including the way they are calculated, except for the first objective. We describe this difference further in the Methodology chapter. Another difference is that instead of optimizing using NSGA-II, we use MO-RV-GOMEA [5]. Where the creators of MOC have made quite a few changes specifically for counterfactual search, we use the standard unchanged version of MO-RV-GOMEA. In our method users are not able to set non-actionable features and we do not penalize counterfactuals that are above a certain distance from the original data point. Another difference between our method and MOC is that MOC works for classification and regression tasks, while our method is only designed for regression tasks. However, unlike MOC, we do optimize for diversity in two of the versions of our method. More on this in Chapter 3.

In [36] the authors present their state-of-the-art multi-objective counterfactual generation method, named Coherent Actionable Recourse based on sound counterfactual Explanations (CARE). They focus on providing the user with a diverse list of counterfactuals that the user can actually act on, which is called actionable recourse. The method also provides the user with a temporal action sequence for a generated counterfactual, which is a list of steps the user has to take to achieve the result of the counterfactual.

CARE optimizes for seven objectives, which are divided into four modules, using NSGA-III [11]. NSGA-III is a multi-objective EA which is similar to NSGA-II, but it keeps resulting solutions diverse using predefined reference points. There is a hierarchy between the modules, which is VALIDITY, SOUNDNESS, COHERENCY, and ACTIONABILITY. The generated counterfactuals are ranked based on this hierarchy, making sure that they first adhere to the VALIDITY objectives before adhering to the SOUNDNESS objectives, etc. This hierarchy also makes it possible to add or remove modules.

The objectives within the VALIDITY module are validity, proximity to the original data point, and sparsity. Within the SOUNDNESS module the objectives are proximity to ground truth data points, and connectedness to the data set using a continuous path. The COHERENCY and ACTIONABILITY modules only have one objective each, which are coherency between correlated features within the counterfactual and adherence to preferences set by the user respectively.

A point of note is that, like DiCE and MOC, CARE also works for both classification and regression models.

2.1.5. Visualizing counterfactuals

To convey the meaning of generated counterfactuals to the user, we can use a visualization tool. These tools can help the user understand the difference between the counterfactuals, and the difference between the counterfactuals and the original data point. When we look at these visualization tools, we can see that they are quite complex. This complexity only increases when we increase the number of generated counterfactuals. While this does not have to be a problem, especially when these tools are used by experts, it can be difficult for end-users to understand how these tools work and what it is that they display.

Some examples of these complex visualization tools are the interactive visual analytics tool aimed at end-users called ViCE [19] in Figure 2.1; the visualization tool aimed at developers of models called AdViCE [18] shown in Figure 2.2; DECE [8], the interactive visualization tool aimed at both end-users and developers in Figure 2.3; the tool aimed at exploring decision boundaries shown in Figure 2.4 called CoFFi [38]; and COFVEE [23] in Figure 2.5, which is a visualization tool based on multiple others.

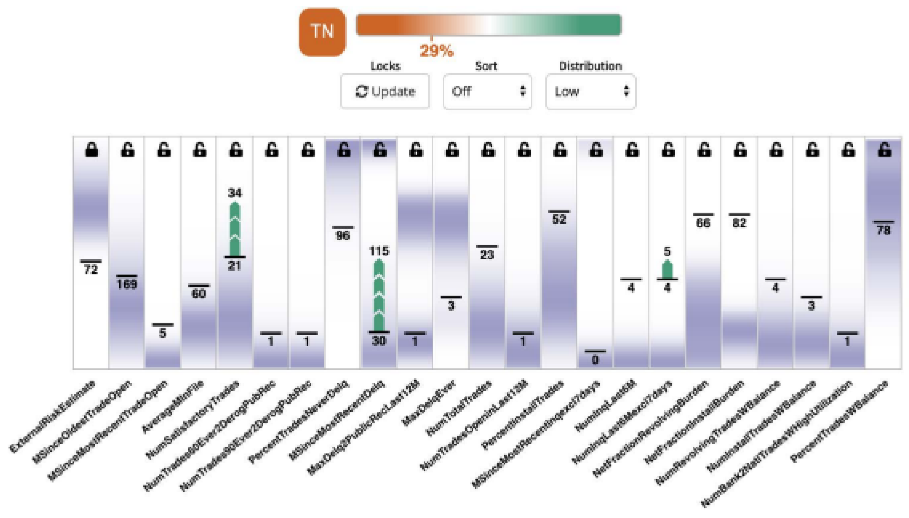


Figure 2.1: ViCE [19].

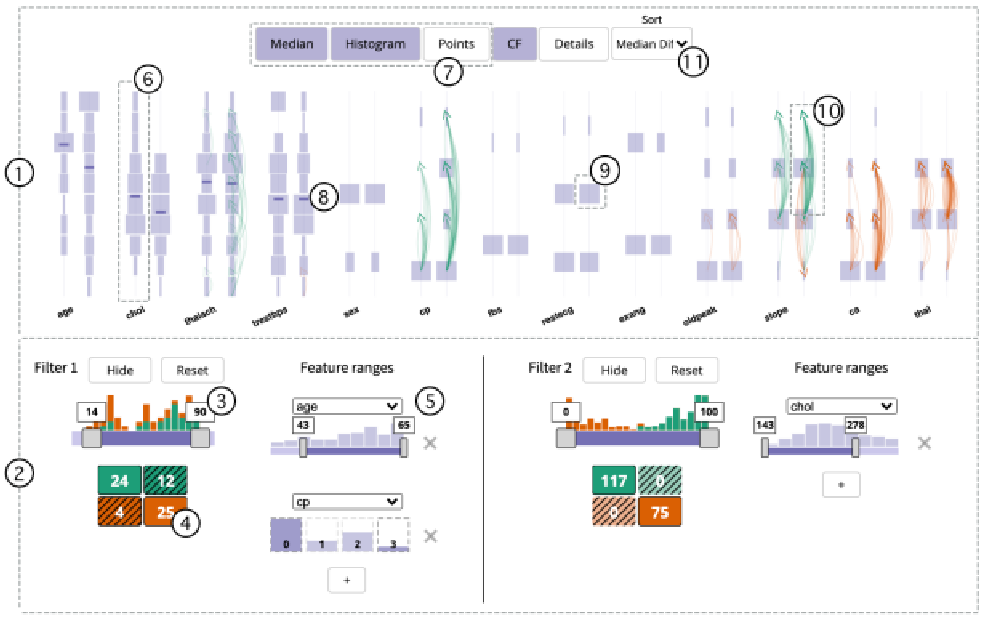


Figure 2.2: AdViCE [18].

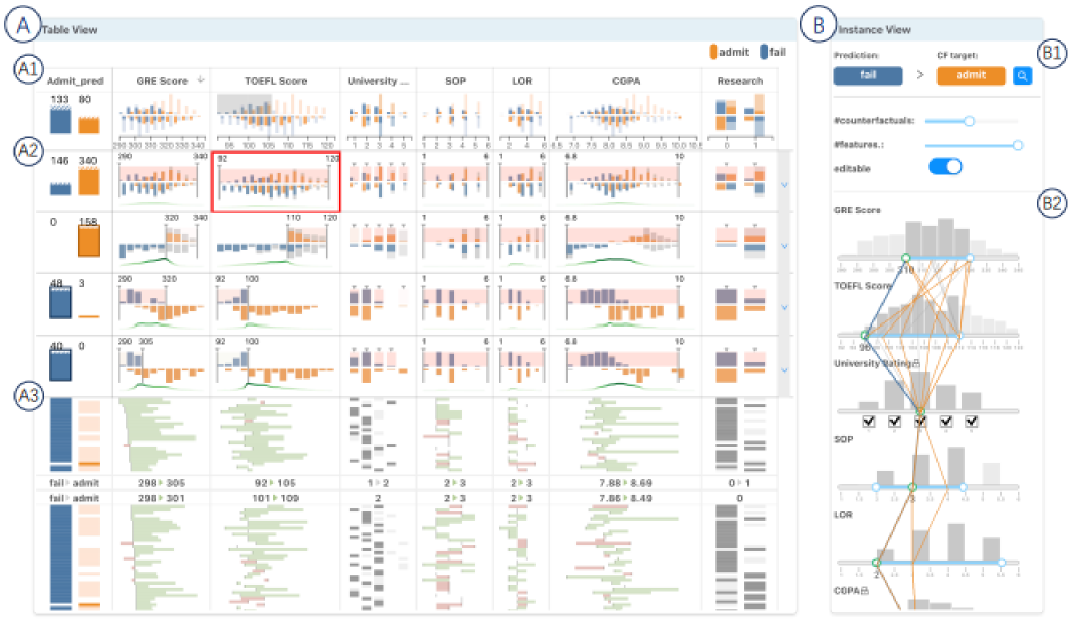
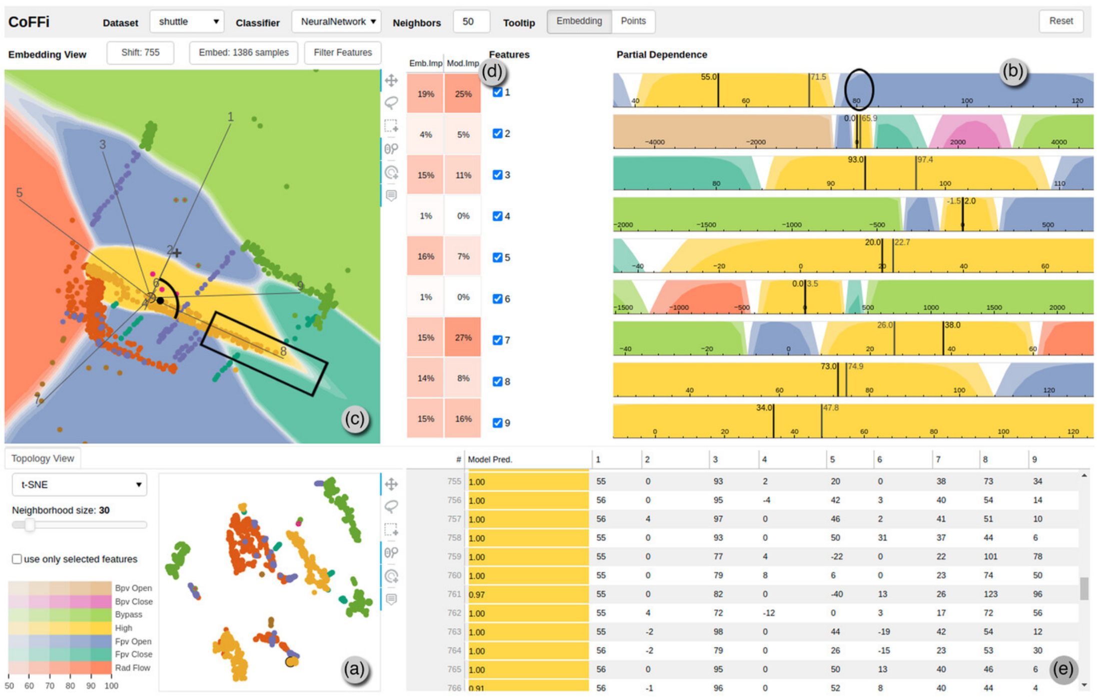


Figure 2.3: DECE [8].



Dataset: shuttle; Classifier: NeuralNetwork; Neighbors: 50; Tooltip: Embedding | Points

Shift: 755; Embed: 1386 samples; Filter Features

t-SNE; Neighborhood size: 30; use only selected features

Emb.Imp	Mod.Imp	Features
19%	25%	☑ 1
4%	5%	☑ 2
15%	11%	☑ 3
1%	0%	☑ 4
16%	7%	☑ 5
1%	0%	☑ 6
15%	27%	☑ 7
14%	8%	☑ 8
15%	16%	☑ 9

#	Model Pred.	1	2	3	4	5	6	7	8	9
755	1.00	55	0	93	2	20	0	38	73	34
756	1.00	56	0	95	-4	42	3	40	54	14
757	1.00	56	4	97	0	46	2	41	51	10
758	1.00	55	0	93	0	50	31	37	44	6
759	1.00	55	0	77	4	-22	0	22	101	78
760	1.00	55	0	79	8	6	0	23	74	50
761	0.97	55	0	82	0	-40	13	26	123	96
762	1.00	55	4	72	-12	0	3	17	72	56
763	1.00	55	-2	98	0	44	-19	42	54	12
764	1.00	56	-2	79	0	26	-15	23	53	30
765	1.00	56	0	95	0	50	13	40	46	6
766	0.91	56	-1	96	0	52	8	40	44	4

Figure 2.4: CoFFi [38].

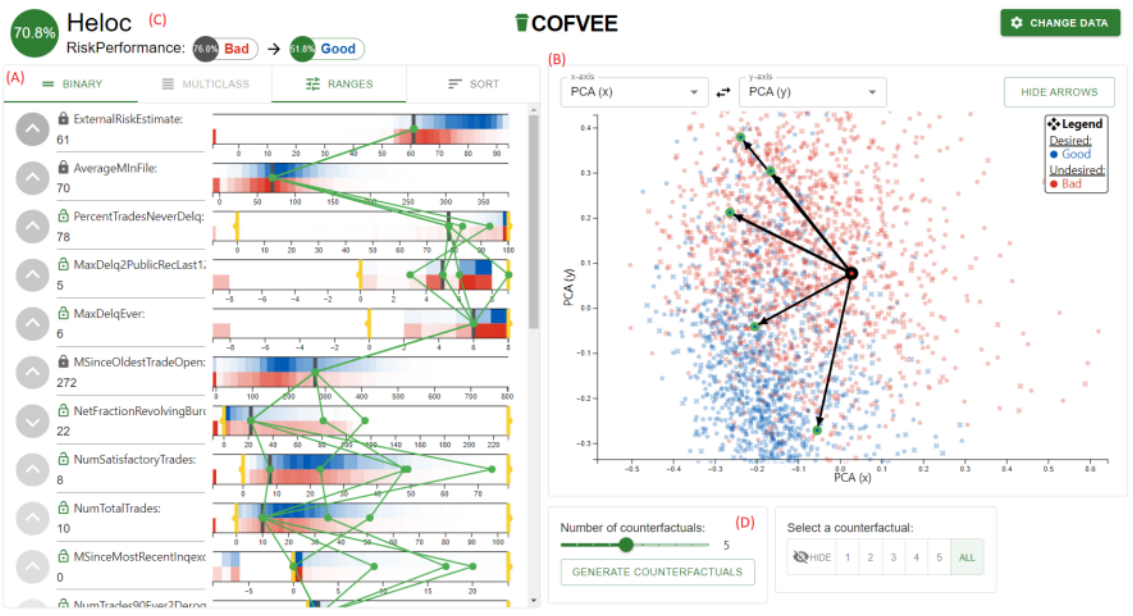


Figure 2.5: COFVEE [23].

2.2. Evolutionary Algorithms

Evolutionary algorithms (EAs) is a family of optimization algorithms that are based on Darwinian evolution. These algorithms work in iterations, where each iteration is a generation. In turn these generations then consist of populations, which then consist of individuals. These individuals are what we are ultimately looking for: (encodings of) solutions to the problem.

The population evolves over the generations. This is done by selecting individuals from the population and combining them to create offspring, which are the individuals that make up the next generation. By making sure that the best individuals have a higher chance of generating this offspring we ensure that the population evolves to create better solutions over the generations. To compare how good each individual is, we use the fitness of each individual. This fitness consists of one or more objectives.

2.2.1. Multi-Objective Evolutionary Algorithms

It is often the case that we need to optimize for more than one objective, which are likely conflicting. An example of this is when we want to cycle somewhere. Two possible objectives for this would be the time it takes to get to the location we want to go to, and the pleasantness of the route.

Imagine we have four possible routes. The first route only takes 20 minutes, but of those 20 minutes we have to cycle 15 of them next to a highway. The second route takes 40 minutes, but it takes us through meadows and along a nice river. The third route takes 30 minutes and while we also get to cycle along the river, we do have to cycle next to the highway for 5 minutes. The fourth route takes 40 minutes, of which we have to cycle 10 of them next to a highway.

Each of these routes have their advantages and disadvantages. The first route is fastest, but cycling next to a highway is unpleasant and causes us to take in a lot of air pollution. While route 2 is a very pleasant cycling route and limits our air pollution intake, it takes us twice as long to get to our destination. Route 3 sits perfectly between the other two routes with the time that it takes, and it also lets us cycle a part of the more pleasant route. However, we still have to cycle next to the highway. The only route that is worse than the others is route 4. This route takes longer than the others, and we have to cycle next to the highway for longer as well. If we were choosing a route for ourselves, we can pick which we find more important: speed or pleasantness. However, we cannot make this choice for every other person. Except for route 4, none of these routes are objectively better than the other, and we are not able to assign weights to both objectives to make an objective choice.

Therefore, we want to present a set of different routes, or more generally, a set of different solutions. We do this in the form of an approximation front. The aim of an approximation front is to approximate the optimal Pareto front. This approximation front consists of all non-dominated solutions that were found by the algorithm. A solution is non-dominated if there are no other solutions that have better objective values for all objectives. So, in our earlier example, route 4 is dominated by the other routes. Therefore, it would not be part of the approximation front.

2.3. The GOMEA family

Gene-Pool Optimal Mixing Evolutionary Algorithm (GOMEA) [40] is an EA that models how groups of variables could be linked, which can be used during variation. This linkage information is captured in a family of subsets (FOS). Because this linkage information is captured based on variable indices, GOMEA only works with fixed length solutions. A FOS consists of a subset of the power set of the set of all variables, where each variable is contained in at least one FOS element. This linkage information can be provided a priori. If it is not provided it is learned during evolution. GOMEA evolves one generation into the next using this FOS. Every individual in the population is evolved using Gene-pool Optimal Mixing (GOM). GOM works as follows. For every set in the FOS, we pick a random donor and copy the values of the variables in that set from the donor to our individual. If this increases the fitness, we keep the changes. Otherwise, we revert the changes and move on to the next set in the FOS. This is repeated for every individual in the population.

The FOS structure that we use in this thesis is the full linkage tree (LT). As described in [40], a LT is constructed as follows. First, we add all sets of a single variable to the FOS. Then we start clustering the sets. Each iteration we combine two sets and add the newly combined set to the FOS. Note that the FOS will now contain both the combined sets and the two sets individually. Which sets are combined is decided by maximizing the mutual information. However, the mutual information is costly to compute. Therefore, it is approximated using the unweighted pair group method with arithmetic mean instead for large sets of variables. This is repeated until there are only two sets left which together contain all variables. The resulting FOS is then treated as a stack.

2.3.1. GP-GOMEA

The Genetic Programming variant of GOMEA is called GP-GOMEA [42]. It maps the nodes of the trees to a fixed-length string, therefore allowing for the learning and usage of a FOS to take place. GP-GOMEA was created for symbolic regression and generates perfect r-ary trees with a maximum height h, where both r and h are set by the user. GP-GOMEA focuses on generating small solutions to increase interpretability.

GP-GOMEA is the algorithm that we use to generate the models that we create counterfactuals for.

2.3.2. MM-GP-GOMEA

The Multi-Modal variant of GP-GOMEA is called MM-GP-GOMEA [37]. In MM-GP-GOMEA each individual represents multiple trees, instead of only one tree like in GP-GOMEA. It uses multi-objective optimization with two objectives to provide the user with multiple diverse solutions to the problem. The first objective is the sum of the trees Mean Squared Errors, and the second objective is the mean of the minimum squared errors of the trees.

2.3.3. RV-GOMEA

The Real-Valued variant of GOMEA is called RV-GOMEA [4]. It builds on top of GOMEA by using mechanisms from AMaLGaM [2].

Instead of choosing a random donor every time a set in the FOS is used, RV-GOMEA learns k-variate normal distribution for every FOS element from the population, where k is the size of that set. This normal distribution is then used to sample from evolution to replace the values of the variables in that element of the FOS. RV-GOMEA also uses Adaptive Variance Scaling (AVS) according to the Standard Deviation Ratio to increase variance that is decreased because of selection; Anticipated Mean Shift (AMS) to speed up search; and Forced Improvements (FI) to escape local minima.

RV-GOMEA also uses an Interleaved Multistart Scheme (IMS) to automatically find a good population size for the problem. IMS interleaves multiple instances of an EA, each with a different population size, usually starting with a small base population size and working its way up.

2.3.4. MO-GOMEA

The Multi-Objective variant of GOMEA is called MO-GOMEA [30]. It differs from GOMEA in that it uses an elitist archive. Another difference is that it uses *k*-leader-means clustering, based on MAMaLGaM [3]. The elitist archive stores the non-dominated solutions found up until that point, using a diversity metric to make sure the elitist archive stays diverse. The clustering picks *k* well spread leaders based on the objective values. These leaders are used to cluster the population into equal sized clusters. Note that because the clusters are of equal size, it is possible that they overlap. This ensures that the approximation front does not become disconnected.

After clustering, MO-GOMEA learns a FOS for and performs evolution on each of these clusters. For each objective the cluster with the best mean value for that objective is chosen to optimize for only that objective. The other clusters optimize for the combination of all objectives. This process of clustering and then evolving repeats every generation. Because the evolution only takes place within one cluster, different parts of the front can be handled differently, and it allows MO-GOMEA to find and return a diverse approximation front that is in high proximity to the optimal front.

2.3.5. MO-RV-GOMEA

The Real-Valued variant of MO-GOMEA is called Multi-Objective Real-Valued Gene-Pool Optimal Mixing Evolutionary Algorithm (MO-RV-GOMEA) [5]. It extends MO-GOMEA to the real-valued domain using mechanisms from MAMaLGaM [3]. Like MO-GOMEA, MO-RV-GOMEA uses an elitist archive and clustering, where each cluster has its own FOS and goes through evolution independently. Each generation goes through the clustering process. It also uses AVS, AMS, and FI.

As described in Subsection 2.3.4, the multi-objectiveness is handled as follows. For each objective the objective with the best mean objective value is optimized for that objective only. The other clusters are optimized for all objectives. For each of the objectives the objective value is computed and kept track of separately.

It is also possible to provide constraints within the computation of the objectives. These constraints tell the algorithm that a solution is infeasible. This means that if x has a better total objective value than solution y, but x has a constraint value above 0, y still dominates x.

MO-RV-GOMEA is the algorithm that we use to generate the counterfactuals.

2.4. Symbolic regression

We test our method on Symbolic Regression (SR) models generated using GP-GOMEA. The goal of SR is to fit a model to data using a set of symbols.

Generally speaking, symbolic models are easier to interpret than sub-symbolic models like neural networks. SR models are often represented as mathematical equations. Models are inherently more interpretable if they are represented by mathematical equations [31], but this interpretability reduces as the equation grows. Small symbolic models are indeed relatively interpretable, and when the models become larger it is no longer possible to interpret them [42].

3 Methodology

In this chapter we describe our method. In Section 3.1 we describe what objectives we optimize for, and how we use MO-RV-GOMEA to optimize for these. After that we describe our visualization tool in Section 3.2.

3.1. Counterfactual Generation

In this Section we describe our counterfactual generation method. First, we talk about a single-modal variant in Section 3.1.1, followed by a multi-modal variant described in Section 3.1.2. In the last section, Section 3.1.3, we explain how we select a final set of *k* counterfactuals from the counterfactuals that were generated by our method.

3.1.1. Single-Modal MO-RV-GOMEA

We implement our method by implementing objectives and optimizing for them using MO-RV-GOMEA. In the definitions of our objectives we use x to denote the original data point, m to denote a model, and $c = \{c^1, c^2, ..., c^d\}$ to denote a (potential) counterfactual, with *d* being the number of features in the dataset. So, c^1 then is the value of the first feature in the counterfactual. Following the same notation we can then define x as $x = \{x^1, x^2, ..., x^d\}$. The desired range of output to be considered a counterfactual is *S*, where *s* is the lowest threshold of this range. We also define *T* to be the train dataset.

The objectives that we implemented are based on MOC [10]. Similar to MOC the following is our multi-objective minimization task:

$$\min_c o(c, m, s, x, T) = \min_c (o_1(m(c), s), o_2(x, c), o_3(x, c), o_4(c, T)),$$

where o is four dimensional and real-valued. So, o contains the objectives we optimize for.

Our first objective measures the distance to being a counterfactual. This is the objective that ensures that our method actually generates counterfactuals that lead to predictions within our desired range instead of returning random data points. The objective measures the difference between the predicted value resulting from that data point and the threshold we need to reach to cross into the desired range. When the predicted value is within the desired range, we set the objective value to 0. We can formalize this as follows:

$$o_1(m(c), s) = \begin{cases} |s - m(c)|, & \text{if } m(c) < s \\ 0, & \text{otherwise} \end{cases}$$

If the threshold has not been reached, we add a constraint to ensure that the method prioritizes finding valid counterfactuals over the other objectives.

Our second objective measures the distance between the original data point *x* and our (potential) counterfactual *c*. This objective tries to ensure that the counterfactual is as relevant to the original data point as possible. Therefore, it is defined as:

$$o_2(x,c) = dist(x,c)$$

As our distance metric *dist* we use the Gower distance for mixed features [20], which is defined as follows:

$$dist(p_1,p_2) = \frac{1}{d}\sum_{j=1}^{d}\begin{cases}\frac{1}{R^j}|p_1^j - p_2^j| & \text{if feature j is numerical}\\ \mathbb{I}_{p_1^j \neq p_2^j} & \text{if feature j is categorical}\end{cases}$$

where R^j is the value range of feature j based on the dataset and p_1 and p_2 are two data points.

Our third objective measures the similarity between the original data point *x* and the (potential) counterfactual *c*. This means that we measure the number of features that have different values and normalize this count, which results in the following definition:

$$o_3(x,c) = \frac{1}{d}\sum_{j=0}^{d} c^j \neq x^j,$$

where *d* is the number of features in that dataset.

Our fourth and last objective measures the distance between a (potential) counterfactual *c* and the closest data point in the dataset. We use this objective to make sure that the counterfactual is similar to the existing data points. When a counterfactual is close to the dataset it is more likely that it is realistic. We define this as follows:

$$o_4(c,T) = \min\{\mathrm{dist}(c,T)\}$$

Since MO-RV-GOMEA is designed for real-valued problems, it does not inherently work for integers and categorical features. To ensure that it does work for these values, we modified the algorithm.

The modification rounds numbers to the nearest integer for non-real-valued feature values. This rounding happens any time a feature value is initialized or changed. While this allows the method to run for non-real-valued features, this does mean that it is likely we lose a lot of nuance during our evolution process.

Imagine we have an integer-valued feature for which the optimal value is 3 and we have an individual that has value 2 for that feature. If MO-RV-GOMEA then changes the value of that feature to 2.4 this is a step in the right direction, but because we round it to the nearest integer this is then rounded back down to 2. This means we immediately lose this step in the right direction. It is possible that this has a significant impact on the performance of the method, especially when we have many non-real-valued features.

3.1.2. Multi-Modal MO-RV-GOMEA

Our single-modal implementation of the method does not optimize for diversity in the feature space. While MOC [10] does not optimize for diversity directly either, the authors have made a modification to NSGA-II to improve feature diversity within the returned counterfactuals by computing the crowding distance in both objective space and feature space instead of just the objective space. Instead of making such a fundamental change to MO-RV-GOMEA we want to be able to optimize for diversity. To do this, we create a multi-modal version of MO-RV-GOMEA, inspired by MM-GP-GOMEA [37].

In the Multi-Modal implementation of our method each individual in the population represents multiple counterfactuals. The number of counterfactuals within one individual depends on the number of counterfactuals we ask the method to generate. So, if we ask the method to generate *k* counterfactuals, each individual will consist of *k* counterfactuals. Because of this we can guarantee that as long as the method is able to find an approximation front of at least one individual, we can return *k* diverse counterfactuals to the user.

When the method finishes evolution we take all individuals in the approximation front, remove the ones that have non-counterfactuals in them, and split all other individuals into separate counterfactuals to create a final list of generated counterfactuals. This list can then be used the same way as the approximation front generated by the single-modal method.

The major difference compared with the single-modal version is in the objectives. We define an individual as $i = \{i_1, i_2, ..., i_k\}$, which means that i_1 denotes the first counterfactual within the individual.

Similar to the single-modal method we can then define these counterfactuals as $i_j = \{i_j^1, i_j^2, ..., i_j^d\}$, where *d* is the number of features in the dataset. So, i_j^1 is the value of the first feature for the j^{th} counterfactual within the individual.

We added a new objective specifically to optimize for diversity within the multi-modal individuals. This objective measures the sum of the distances between each pair of counterfactuals within the individual and is defined as follows:

$$o_5(i) = \sum_{a=1}^{k-1} \sum_{b=a+1}^{k} dist(i_a, i_b),$$

with *i* being the individual we are evaluating, which then consists of *k* counterfactuals.

So, when using this objective our multi-objective optimization task now changes to:

$$\min_i o(i) = \min_i \left(\sum_{j=1}^{k} o_1(m(i_j), s), \sum_{j=1}^{k} o_2(x, i_j), \sum_{j=1}^{k} o_3(x, i_j), \sum_{j=1}^{k} o_4(i_j, T), o_5(i) \right),$$

where o is five dimensional and real-valued.

However, this way of optimizing for diversity could lead to a potential problem. By adding an objective targeting diversity, it could be the case that the diversity will come at the cost of the other objectives. It could also be the case that the counterfactuals will just become diverse for diversity's sake, without actually adding value for the end user. This could happen if the method starts changing features that do not impact the final predicted value. If this feature changes between counterfactuals it does add diversity, but it is not meaningful or of importance for explaining decisions made by the model. It could even negatively impact the final result, since it then no longer is clear that this feature does not impact the model's prediction.

To prevent this, we introduce another objective and multi-objective optimization task for the multi-modal method, which looks like this:

$$\min_i o(i) = \min_i \left(\sum_{j=1}^{k} o_1(m(i_j), s), \sum_{j=1}^{k} o_2(x, i_j), \sum_{j=1}^{k} o_3(x, i_j), o_6(i_j, T) \right),$$

where o is four dimensional and real-valued.

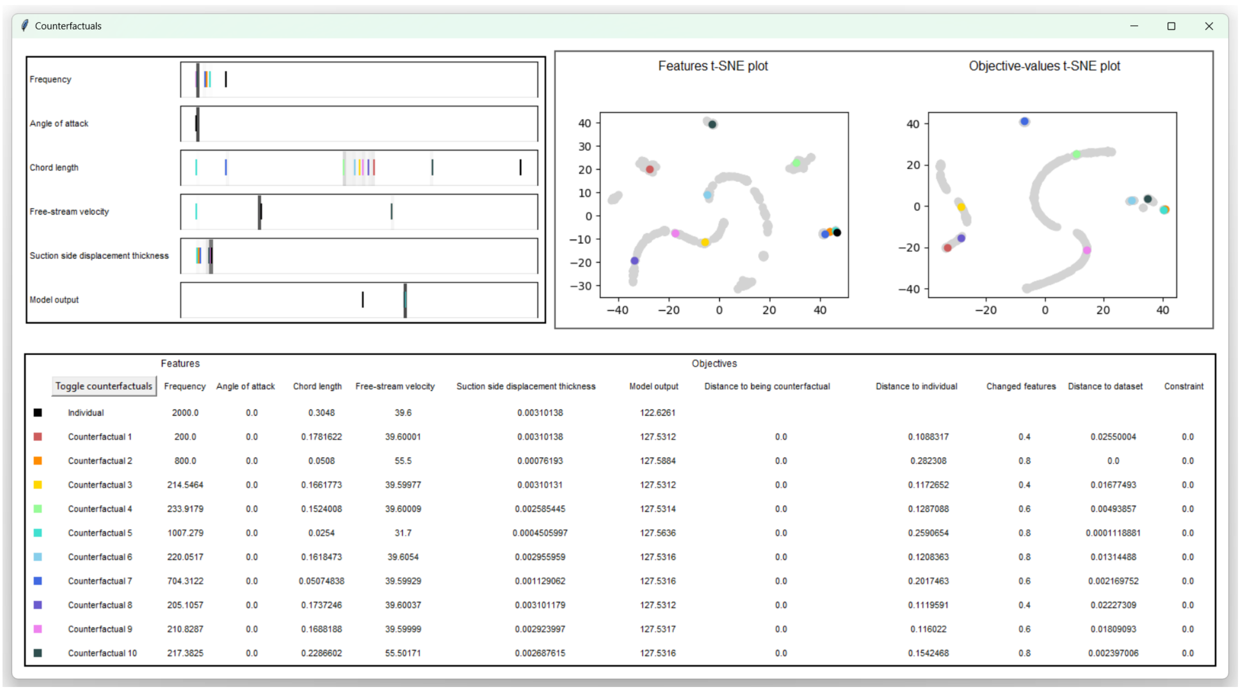
Our new objective combines objective 4 and objective 5, which are the distance to the dataset and diversity objectives respectively. The new objective captures the distance from the counterfactuals within the individual to distinct data points within the dataset. We calculate this by considering the counterfactuals from front to back. For the first counterfactual we calculate the distance to the nearest data point within the dataset and keep track of which data point this is. Then, for the next counterfactual we find the closest data point in the dataset. However, if at any point during this search the closest point would be the same as the one for the first counterfactual, we ignore that data point. We repeat this for every counterfactual within the individual and sum these values. This results in the following definition:

$$o_6(i, T) = \sum_{\substack{j=1 \\ t_a \neq t_b \text{ if } a \neq b}}^{k} \min\{\text{dist}(i_j, T)\},$$

with t_a and t_b being the data point closest to counterfactual *a* and counterfactual *b* within the individual.

By only measuring the distance to distinct data points within the data set, we aim to create meaningful diversity. Instead of counterfactuals being diverse for diversity's sake, they need to stay close to the data points within the dataset while also moving towards distinct data points to introduce diversity.

As shown in the multi-objective optimization tasks for the multi-modal methods, we change the existing objectives for both multi-modal methods compared to the single-modal method. Instead of calculating the objectives for the individual, we sum the objective values calculated for each counterfactual separately. So, if we have an individual that consists of four counterfactuals, and their separate objective values are 0.2, 0.1, 0.5, and 0.7, the objective value for that individual would be 1.5. We



		Features						Objectives				
	Toggle counterfactuals	Frequency	Angle of attack	Chord length	Free-stream velocity	Suction side displacement thickness	Model output	Distance to being counterfactual	Distance to individual	Changed features	Distance to dataset	Constraint
■	Individual	2000.0	0.0	0.3048	39.6	0.00310138	122.6261					
🟥	Counterfactual 1	200.0	0.0	0.1781622	39.60001	0.00310138	127.5312	0.0	0.1088317	0.4	0.02550004	0.0
🟧	Counterfactual 2	800.0	0.0	0.0508	55.5	0.00076193	127.5884	0.0	0.282308	0.8	0.0	0.0
🟨	Counterfactual 3	214.5464	0.0	0.1661773	39.59977	0.00310131	127.5312	0.0	0.1172652	0.4	0.01677493	0.0
🟩	Counterfactual 4	233.9179	0.0	0.1524008	39.60009	0.002585445	127.5314	0.0	0.1287088	0.6	0.00493857	0.0
🟦	Counterfactual 5	1007.279	0.0	0.0254	31.7	0.0004505997	127.5636	0.0	0.2590654	0.8	0.0001118881	0.0
🟦	Counterfactual 6	220.0517	0.0	0.1618473	39.6054	0.002955959	127.5316	0.0	0.1208363	0.8	0.01314488	0.0
🟦	Counterfactual 7	704.3122	0.0	0.05074838	39.59929	0.001129062	127.5316	0.0	0.2017463	0.6	0.002169752	0.0
🟪	Counterfactual 8	205.1057	0.0	0.1737246	39.60037	0.003101179	127.5312	0.0	0.1119591	0.4	0.02227309	0.0
🟪	Counterfactual 9	210.8287	0.0	0.1688188	39.59999	0.002923997	127.5317	0.0	0.116022	0.6	0.01809093	0.0
■	Counterfactual 10	217.3825	0.0	0.2286602	55.50171	0.002687615	127.5316	0.0	0.1542468	0.8	0.002397006	0.0

Figure 3.1: Our visualization tool for sets of counterfactuals. Top left: Feature Densities view. Top right: t-SNE Plot view. Bottom: Counterfactual view.

do this for every objective. Note that because of this the maximum possible objective value for each objective now is k times larger than the maximum possible objective value for each objective in the single-modal version of the method, since we sum the objective values for k individuals.

3.1.3. Selecting Diverse Counterfactuals

The number of counterfactuals that both the single-modal and multi-modal methods return can be anywhere between zero to over a thousand. This means we have so many counterfactuals that it is no longer reasonable to look at them all, especially for an end user that wants an explanation for their own situation. Therefore, we want to select a few counterfactuals from this list of counterfactuals that are diverse enough to give different options. To return the best possible results we want to select a set of counterfactuals that best describes the spread of the approximation front.

To select counterfactuals from the approximation front we use a method based on greedy scattered subset selection. It works as follows. First, remove all points that are not counterfactuals. Then, if the number of counterfactuals that we are selecting, defined as k, is larger than or equal to the number of features, pick the counterfactual where the difference in feature value is the largest between that counterfactual and the original data point. If k is smaller than the number of features, pick the point that has the largest feature value for a random feature. After this iteratively select counterfactuals, based on largest Gower distance in feature values with the already selected set, until we have selected k counterfactuals.

By using this method to pick counterfactuals we try to select counterfactuals that describe the approximation front well and give the user a diverse list of counterfactuals to choose from.

3.2. Visualization

Because of the complexity of the several existing visualization methods, we implemented our own simple visualization tool based on the aforementioned more complex tools. By reducing the complexity, we give a clearer and easier to understand overview of generated counterfactuals.

However, this means that our visualization has two drawbacks compared to other existing tools. The first drawback is that we provide less information. The second drawback is that we do not provide customization settings that can be used by the user to tailor the counterfactuals to their preferences.

Figure 3.1 shows this new tool. It consists of three different views. At the bottom we have the

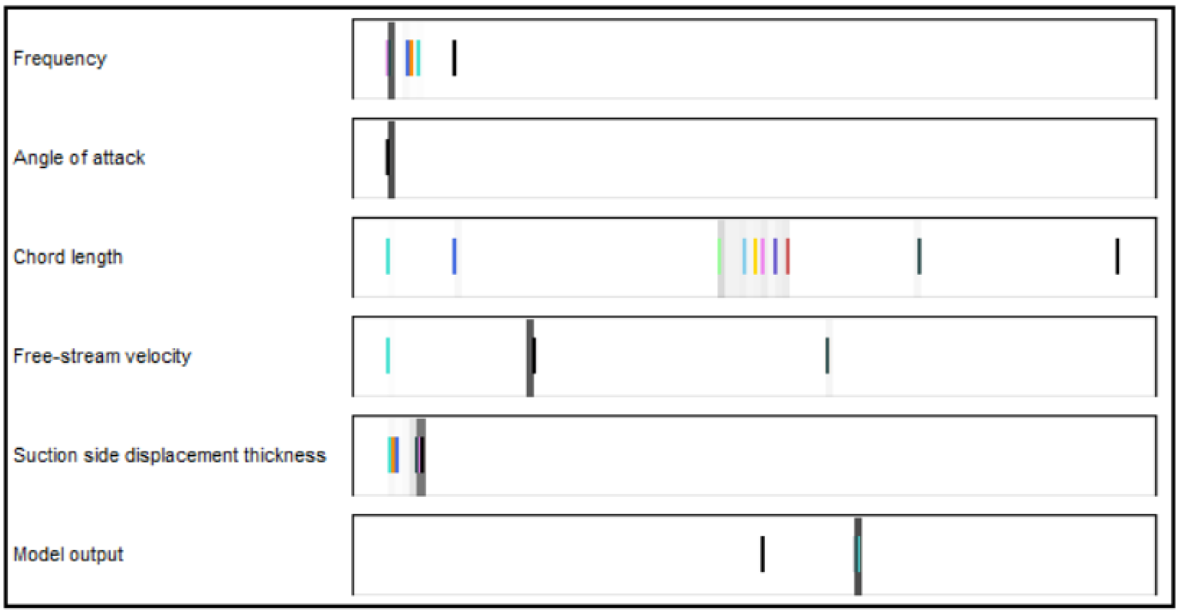


Figure 3.2: Feature Density view in our visualization tool.

Counterfactual view, at the top left we have the *Feature Densities view*, and at the top right we have the *t-SNE Plot view*.

Counterfactual view. The *Counterfactual view* consists of the original data point for which we generated counterfactuals and a list of selected counterfactuals. Each counterfactual is assigned a distinct colour. For the original data point we list all feature values, and for the counterfactuals we list all feature and objective values. The colours shown in this view correspond to the colours shown in the other views. So, the red counterfactual in this view is the same counterfactual as the red counterfactual in the *Feature Densities view* and the *t-SNE Plot view*.

This view contains a ”Toggle counterfactuals” button. This button allows us to toggle which counterfactuals we want to see in the other views. This allows a user to focus on the counterfactuals they are interested in and provides clarity in case counterfactuals overlap.

We include this overview to provide the exact values of both the features and the objectives for each counterfactual.

Feature Densities view. The *Feature Densities view*, shown in Figure 3.2, shows the densities of the generated counterfactuals per feature. For each feature we take the lowest and highest values in the training set and divide the space between these into 30 bins. Then we count how many generated counterfactuals fall in each bin. Each bin that does not contain any counterfactuals is white, each bin that does contain counterfactuals is shaded grey. The darker the shade of grey, the more counterfactuals are in that bin.

These bins are overlaid with the selected counterfactuals shown in the *Counterfactual view*. This means that if a coloured counterfactual is shown on a shaded bin, this counterfactual is in said bin.

We include this view to provide a clear overview of where the counterfactuals are in the feature space. The shaded bins provide insight into what is important for most counterfactuals and allow the user to see whether there is a certain value, which might or might not differ from the original data point, that all counterfactuals have for a certain feature; or whether there is a broad spectrum of values that can still provide counterfactuals. The coloured overlays allow for insight into how different features might depend on each other. For example, it shows that if the value of one feature is low (Chord length) the value for another feature will likely also be low (Free-stream velocity).

t-SNE Plot view. The *t-SNE Plot view*, shown in Figure 3.3, shows t-SNE plots for both the feature and objective values. All generated counterfactuals are shown in light grey, and again are overlaid with the selected counterfactual in their corresponding colours. It uses the t-SNE implementation from scikit-learn.

We include this view to provide an overview of how diverse the selected counterfactuals are.

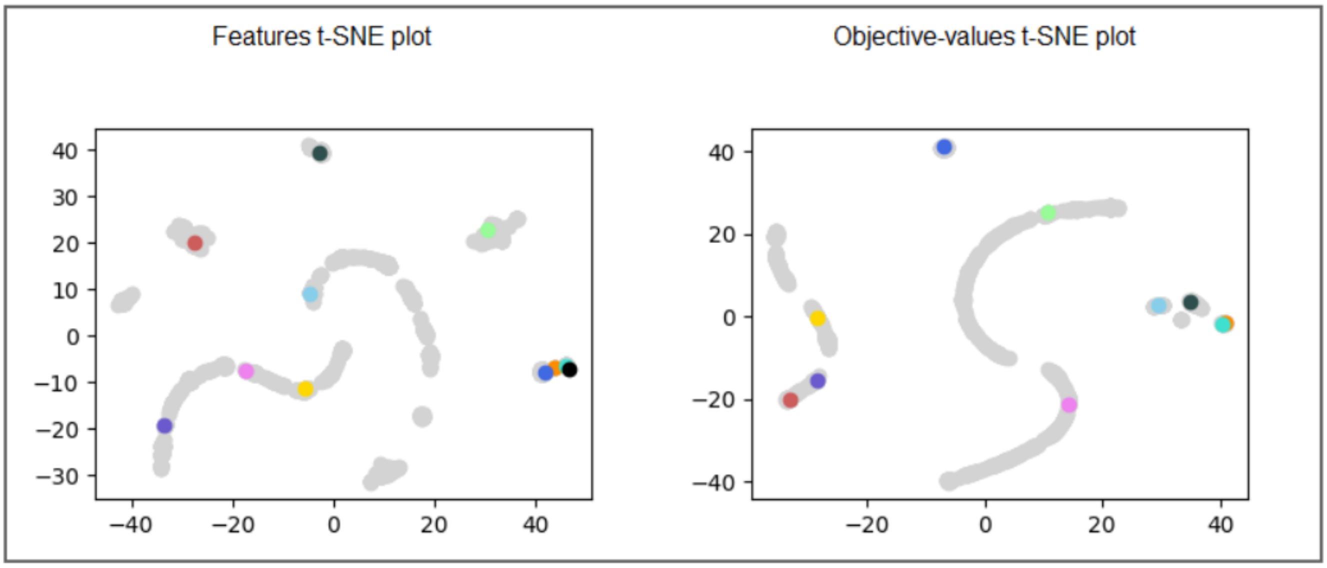


Figure 3.3: t-SNE Plot view in our visualization tool.

4 Experimental Setup and Design

In this chapter we describe the experimental setup and the experiment design. First, we describe the experimental setup, including the metrics, datasets, models, and baselines we use, as well as the actual parameters used in these experiments. Second, we describe the different experiments we do.

4.1. Experimental setup

4.1.1. Baselines

We use DiCE [34] as a baseline to compare our method to. DiCE is implemented in Python and has support for Sklearn models, which makes it simple to run it for the same instances we use to evaluate our method. For this comparison we use the same models and random seeds as for our own method. To set the random seed for DiCE we use the random method. We set the proximity weight and diversity weight to 0.5 and 1.0 respectively, as the authors describe in the paper.

To gain a better understanding of the performance of this method we also tried to use other baselines. However, we were not able to get either of these to work with our chosen models.

The first method we tried to use as a baseline is Multi-Objective Counterfactual Explanations (MOC) [10], because this is the original multi-objective counterfactual generation method. We were unable to use this method, because their method is implemented in R. The models that we use are only available in Python and C++. The authors of MOC do compare their method to DiCE. This is possible because they train their models in R, and then use the models in Python.

The second method we tried to use as a baseline is CARE [36], because it is a state-of-the-art counterfactual generation method. However, we were not able to do this, because the CARE implementation is incompatible with the models we use. The predictions made by the models we use differ too much from the target values, which then in turn causes CARE to not be able to generate counterfactuals.

4.1.2. Metrics

The quantitative metrics are based on the ones used to evaluate DiCE [34]. We define *k* as the number of counterfactuals the method was asked to generate. To generate *k* counterfactuals from DiCE we set it as a hyperparameter. For MO-RV-GOMEA we select *k* counterfactuals from the approximation front.

Then we define $C = \{c_1, c_2, ..., c_n\}$ to be the set of generated counterfactuals, where *n* is the number of generated counterfactuals. Depending on the model and difficulty of finding counterfactuals k and n can be equal, and we can define their relationship as $n <= k$. If we ask a model to generate 10 counterfactuals it could be the case that it only returns 6 counterfactuals. In that case $k = 10$ and $n = 6$.

A single counterfactual consists of a set of feature values. We define this as $c = \{c^1, c^2, ..., c^d\}$, where *d* is defined as the number of dataset features. So, the first feature value for the first generated counterfactual is then denoted as c_1^1. Following this same notation, we can define the original data point *x* as $x = \{x^1, x^2, ..., x^d\}$ and the first feature value of that data point as x^1.

As the distance measure we once more use the Gower distance [20], which is defined as follows:

$$dist(p_1, p_2) = \frac{1}{d}\sum_{j=1}^{d}\begin{cases}\frac{1}{R^j}|p_1^j - p_2^j| & \text{if feature j is numerical}\\ \mathbb{I}_{p_1^j \neq p_2^j} & \text{if feature j is categorical}\end{cases} \tag{4.1}$$

for data points p_1 and p_2.

Validity Validity measures how many actual counterfactuals the method returned versus how many we ask it to generate. If we define *v* as the number of unique counterfactuals in C that lead to a predicted value within the desired range, then we define the validity as follows:

$$\%\text{Valid-CFs} = \frac{\text{v}}{k}$$

When *n* == *v* all counterfactuals in *C* are unique and lead to predictions within the desired range. In this case it holds that:

$$\%\text{Valid-CFs} = \frac{\text{n}}{k}$$

If not all counterfactuals in C are unique or lead to predictions within the desired range this does not hold.

Proximity Proximity measures the distance between the original data point *x* and each counterfactual within *C*. We therefore aim to measure how similar the counterfactuals are to the original data point. This is calculated as the average distance between the counterfactuals and the original data point. Therefore, proximity is defined as:

$$\text{Proximity} = -\frac{1}{n}\sum_{j=1}^{n}\text{dist}(c_j, x)$$

We multiply the actual proximity by -1 to create more intuitive graphs. By doing this it holds for every metric that a higher value is better.

Sparsity Sparsity measures the number of features that differ between a counterfactual and the original data point.

$$\text{Sparsity} = 1 - \frac{1}{nd}\sum_{a=1}^{n}\sum_{b=1}^{d}1_{[c_a^b \neq x^b]}$$

By subtracting the actual value from 1 we once again make sure that a higher value means better sparsity.

Diversity Diversity is measured in the average pair-wise distance between all counterfactuals in *C*. It is similar to proximity, but instead of using the original data point we use another counterfactual from *C*.

$$\text{Diversity} = \frac{1}{\binom{n}{2}}\sum_{a=1}^{n-1}\sum_{b=a+1}^{n}\text{dist}(c_a, c_b)$$

We also measure count-diversity, which is similar to the sparsity. It measures the number of feature values that are different between each pair of counterfactuals in *C*.

$$\text{Count-Diversity} = \frac{1}{\binom{n}{2} * d}\sum_{a=1}^{n-1}\sum_{b=a+1}^{n}\sum_{l=1}^{d}1_{[c_a^l \neq c_b^l]}$$

Distance to dataset This metric was not used in the evaluation of DiCE [34]. It measures the distance between a counterfactual and the closest data point in the train dataset. We include it, because we want a way to measure how realistic a counterfactual is. Since the data points in the dataset are real measurements, these data points are realistic. If we then measure the distance to these realistic data points, we can use that as a proxy for this realism.

We can define this distance as:

Functions	+, -, *, and aq
Generations limit	75
Evaluations limit	-
Time limit	-
Initial tree height	4
Maximum of tries for solutions in the initial population to be syntactically unique	1000
FOS	Linkage tree
Ephemeral random constants	true
Interleaved multi-start scheme	5 sub-generations and early stopping criterion 1
Parallel	1

Table 4.1: Parameter values for GP-GOMEA.

$$\text{Distance to dataset} = -\frac{1}{n}\sum_{j=1}^{n} min\{\text{dist}(c_j, T)\}$$

where T is the train set. Because all data points in the train dataset are examples of real data points, measuring the distance to the dataset can be seen as a proxy for how realistic the counterfactuals are.

Of these metrics, we optimize for all of them except Diversity and Count-Diversity using the single-modal method, and for all of them using the multi-modal methods. When calculating these metrics for multiple runs we can simply average the measurements over the total number of runs.

4.1.3. Datasets

We evaluate our method using the following four datasets. Because MO-RV-GOMEA is a real-valued method, we only use regression datasets.

Airfoil Self-Noise This dataset contains measurements of many different NACA 0012 airfoils in a wind tunnel [6]. It consists of 8 different features, and the model's task is to predict the scaled sound pressure.

Yacht Hydrodynamics This dataset contains measurements of 22 different types of yacht hulls [16]. It consists of 6 features, and the model's task is to predict the residuary resistance per unit weight of displacement.

Concrete Compressive Strength This dataset contains 8 features consisting of different ingredients used to mix concrete and the age in days [44]. The model's task is to predict the compressive strength of the concrete.

Bike Sharing This dataset contains information about bike rentals for the years 2011 and 2012 [15]. There are two versions of this dataset, one that contains hourly measurements and one that contains daily measurements. We only use the day part of the dataset and remove 2 variables, namely the count of casual users and the count of registered users. This leaves us with 12 features describing measurements like weather, day of year, and whether it is a working day. The model's task is to predict the total count of bike rentals.

Each of these datasets contain a number of continuous features. Both the Airfoil Self-noise dataset and the Yacht Hydrodynamics dataset consist entirely of continuous features. The Concrete Compressive Strength dataset contains one integer feature, namely age. The only dataset that contains majority non-continuous features is the Bike Sharing dataset, in which only 4 features are continuous.

To run the experiments we use two data points per dataset. These data points are the first line of the train set and the first line of the test set.

4.1.4. Models

We generate the models to explain using Genetic Programming Gene-pool Optimal Mixing Evolutionary Algorithm (GP-GOMEA) [42]. The models we use are Symbolic Regression models. To create these models we use 80% of each dataset as a train set and 20% as test set.

Forced improvements	True
Boundary repair	True
FOS	Full linkage tree
Value-to-reach	True
User range	[-1000000,1000000]
Maximum number of generations	3,000,000
Time limit	120 seconds

Table 4.2: MO-RV-GOMEA parameters we set to run our experiments.

Rotation angle	0
Tau	0.35
Base population size	30
Maximum number of populations	25
Base number of mixing components	5
Distribution multiplier decrease	0.9
Standard deviation ratio threshold	1.0
Elitist archive size target	100
Value to reach	0.001
maximum no improvement stretch	100

Table 4.3: MO-RV-GOMEA parameters we did not change.

GP-GOMEA has quite a few parameters to set. The values for these are shown in Table 4.1, where aq is a protected form of division. All other parameters are set to their default values.

Using these parameters we create 5 models with seeds 1-5. For each experiment we use these same 5 models and average over them.

Besides the implementation in C++, GP-GOMEA is also available as a python package. The package allows it to be used as a Sklearn model. This makes it easier to use for multiple different methods implemented in different programming languages.

4.1.5. Parameters

Like GP-GOMEA, MO-RV-GOMEA also has many different parameters. To run our experiments we use the parameters shown in Table 4.2.

While it might seem that setting a time limit would hamper experiment reproducibility, practice shows that if a run exceeded this time it, would not find any counterfactuals within the maximum number of evaluations. When this was the case the runtime was too long to be able to run all experiments. However, it is still possible that this influences the results.

Other parameters that we did not experiment with are set as shown in Table 4.3.

4.1.6. Definition of a counterfactual

Because we deal only with regression tasks, it is difficult to define when a data point is considered to be a counterfactual. Each of these datasets are significantly different and we have no domain knowledge in any of them. For the purposes of this thesis, we therefore define a counterfactual as follows.

A data point *c* is a counterfactual if $m(x) \leq t * m(c)$, where *m* is the model, *x* is the original data point we want to generate counterfactuals for, and *t* is a constant. In this thesis *t* is set to 1.025, unless stated otherwise. This means that for the purposes of this thesis an data point is a counterfactual if the its predicted value by the model is 2.5% larger than the predicted value for the original data point.

4.2. Experimental Design

We have done four experiments. In this section we describe these experiments and why we have done them. To run these experiments we use the same five models. These models are created with GP-GOMEA, using the parameters described above and the seeds set to 1-5. For each point in experiment one and two we also run five runs. So, when combining the number of models and extra runs, each point in the graph is averaged over twenty-five runs.

4.2.1. Single Instance

The first type uses only the Bike Sharing dataset. We compare the results from a single run from both our single-modal method and DiCE, using our new visualization without any of the quantitative metrics described above. By walking through a single example for one dataset it allows us to get a more intuitive feel for the method's performance before we look at metrics averaged over multiple runs. To achieve this, we evaluate the performance of the methods on the day in the dataset that contains data for all hours in the day and has the lowest target value. We use the model with seed 1 for this experiment and set the threshold for being a counterfactual to 4.0 to allow for more drastic changes.

4.2.2. Single-Modal Performance

After this the second type of experiment uses all datasets. We compare how the methods perform when asking for an increasing number of counterfactuals using the quantitative metrics. This experiment allows us to gain an understanding of how our method performs in a typical setting where people use it to explain their personal circumstances. For this experiment we ask the methods to generate 1, 2, 4, 6, 8, and 10 counterfactuals.

We perform this experiment to see how our single-modal method performs compared to the baseline DiCE.

4.2.3. Objectives Effects

Our third type of experiment evaluates how each objective influences the performance of our method. We use all datasets and quantitative metrics. Once again we ask our method to generate 1, 2, 4, 6, 8, and 10 counterfactuals. For this experiment we only evaluate the performance of MO-RV-GOMEA using the data point from the test set to prevent using a data point that is the model has been trained on and that is contained in the dataset we use for the Distance to the dataset objective. To evaluate the effect of each objective we compare a version of MO-RV-GOMEA using all objectives to versions that are missing one objective. The only objective that is not evaluated this way is objective 0, because that objective is necessary to guarantee that the method will return any counterfactuals if possible.

We perform this experiment to see how the different objectives interact with each other, and to see how they influence diversity within the generated counterfactuals.

4.2.4. Multi-Modal Performance

Our fourth and last experiment measures the performance of our multi-modal methods compared to our single-modal method and DiCE. We use all quantitative metrics and all datasets except for the bike sharing dataset, and also ask the methods to generate 1, 2, 4, 6, 8, and 10 counterfactuals. The bike sharing dataset is excluded because of its number of features. Since it has 12 features this causes the multi-modal individuals to become very large very quickly. By excluding this dataset we limit the runtime. To decrease runtime more we make a few changes to our parameter settings. We set the maximum number of generations to 30,000 and the time limit to 240 seconds. The experiment only uses the individual from the test set.

The aim of this experiment is to compare the performance of our multi-modal methods to both the basline DiCE and our single-modal method.

5 Results and Discussion

In this Chapter we present the results of our experiments described in 4.2 and discuss them.

5.1. Single Instance

In this experiment we generate counterfactuals for a single day of bike rentals in Washington, D.C., USA. The day we evaluate is December 26th, 2012. It was a Wednesday and since it was not a holiday it was a working day. The weather was partly cloudy, quite cold at a temperature of 3.44°C with a feeling temperature of 2.36 °C, and humid with 82% humidity. With a wind speed of 21.2 mph, it was windy enough for small trees to begin to sway [41].

Figure 5.1 shows the counterfactuals that were generated by the methods for this specific day. Instead of showing the feature values we have translated the values into their real world meaning. This allows for a more intuitive way to interpret the results.

When we look at the counterfactuals there are a few things that stand out. The first is that MO-RV-GOMEA makes more changes than DiCE. Where DiCE only makes changes to three features, MO-RV-GOMEA makes changes to all of them. The changes made by MO-RV-GOMEA are more realistic. Whenever it changes the date, it also changes the season, year, month, weekday, etc. accordingly. While this is interesting to see, we do have to note that our counterfactual selection method could play a large role in this. It is likely that there are counterfactuals within the approximation front that are closer to the ones produced by DiCE. However, since this counterfactual selection method is part of our counterfactual generation method as a whole, it is important to compare these produced sets.

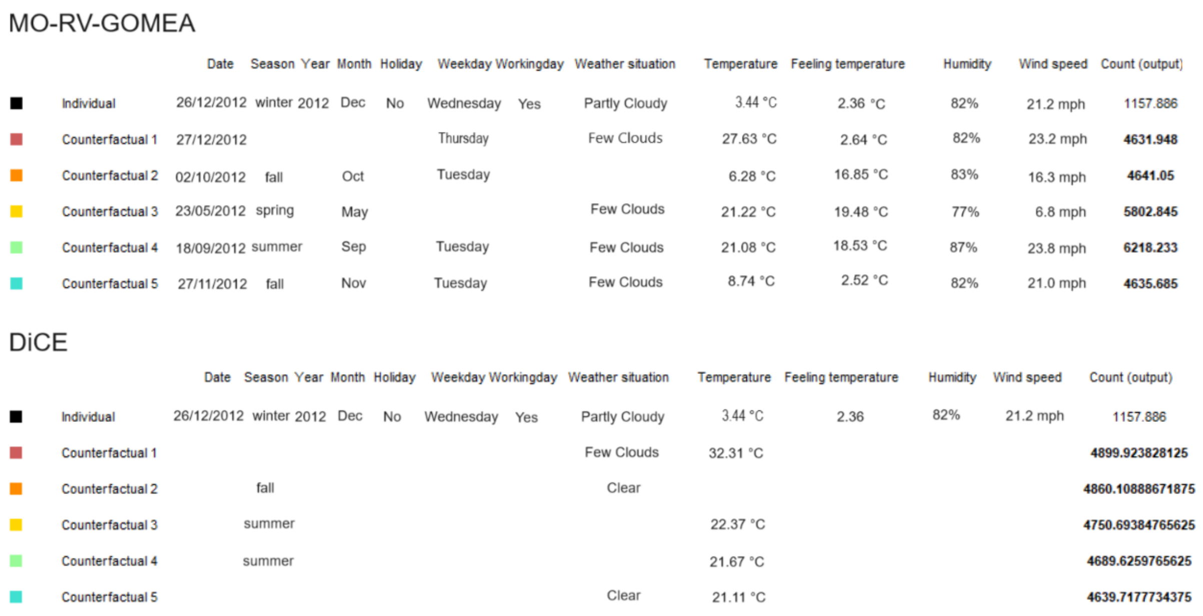
If we compare the counterfactuals that were generated by MO-RV-GOMEA to the instances in the dataset with the same date, we can see why these examples look more realistic. For all date changes the non-continuous features change to the exact values of those days as well. The difference in the continuous features varies per counterfactual. For counterfactual 5 all continuous features are significantly different, but counterfactual 3 is the exact same as the instance in the dataset. The other counterfactuals are anywhere in between.

Figure 5.2 shows the densities per feature for the generated counterfactuals from these methods. We can see that the counterfactuals generated by MO-RV-GOMEA indeed seem to be more diverse than the ones generated by DiCE, which in turn seem to be closer to the original data point than the ones generated by MO-RV-GOMEA. However, when we look at the grey shaded densities for MO-RV-GOMEA, we can see that there are counterfactuals within the approximation front that are closer to the original data point.

This is confirmed by Table 5.1, which shows the values for each of the metrics calculated over the returned sets of counterfactuals. MO-RV-GOMEA performs equal to or better than DiCE on all metrics

	%-Valid counterfactuals	Proximity	Sparsity	Diversity	Count-Diversity	Distance to dataset
MO-RV-GOMEA	100%	-0.40	0.32	0.39	0.67	-0.02
DiCE	100%	-0.13	0.83	0.16	0.22	-0.12

Table 5.1: Metrics for a single run of the methods on the Bike Sharing dataset. All values are rounded to two decimals.



MO-RV-GOMEA

		Date	Season	Year	Month	Holiday	Weekday	Workingday	Weather situation	Temperature	Feeling temperature	Humidity	Wind speed	Count (output)
⬛	Individual	26/12/2012	winter	2012	Dec	No	Wednesday	Yes	Partly Cloudy	3.44 °C	2.36 °C	82%	21.2 mph	1157.886
🟥	Counterfactual 1	27/12/2012					Thursday		Few Clouds	27.63 °C	2.64 °C	82%	23.2 mph	**4631.948**
🟧	Counterfactual 2	02/10/2012	fall		Oct		Tuesday			6.28 °C	16.85 °C	83%	16.3 mph	**4641.05**
🟨	Counterfactual 3	23/05/2012	spring		May				Few Clouds	21.22 °C	19.48 °C	77%	6.8 mph	**5802.845**
🟩	Counterfactual 4	18/09/2012	summer		Sep		Tuesday		Few Clouds	21.08 °C	18.53 °C	87%	23.8 mph	**6218.233**
🟦	Counterfactual 5	27/11/2012	fall		Nov		Tuesday		Few Clouds	8.74 °C	2.52 °C	82%	21.0 mph	**4635.685**

DiCE

		Date	Season	Year	Month	Holiday	Weekday	Workingday	Weather situation	Temperature	Feeling temperature	Humidity	Wind speed	Count (output)
⬛	Individual	26/12/2012	winter	2012	Dec	No	Wednesday	Yes	Partly Cloudy	3.44 °C	2.36	82%	21.2 mph	1157.886
🟥	Counterfactual 1								Few Clouds	32.31 °C				**4899.923828125**
🟧	Counterfactual 2		fall						Clear					**4860.10888671875**
🟨	Counterfactual 3		summer							22.37 °C				**4750.69384765625**
🟩	Counterfactual 4		summer							21.67 °C				**4689.6259765625**
🟦	Counterfactual 5								Clear	21.11 °C				**4639.7177734375**

Figure 5.1: The counterfactuals from a single run on the Bike Sharing dataset, with the values translated to readable values. For the counterfactuals only values that differ from the original data point are shown. All values are rounded. Some values are the same as the value in the original data point after rounding.

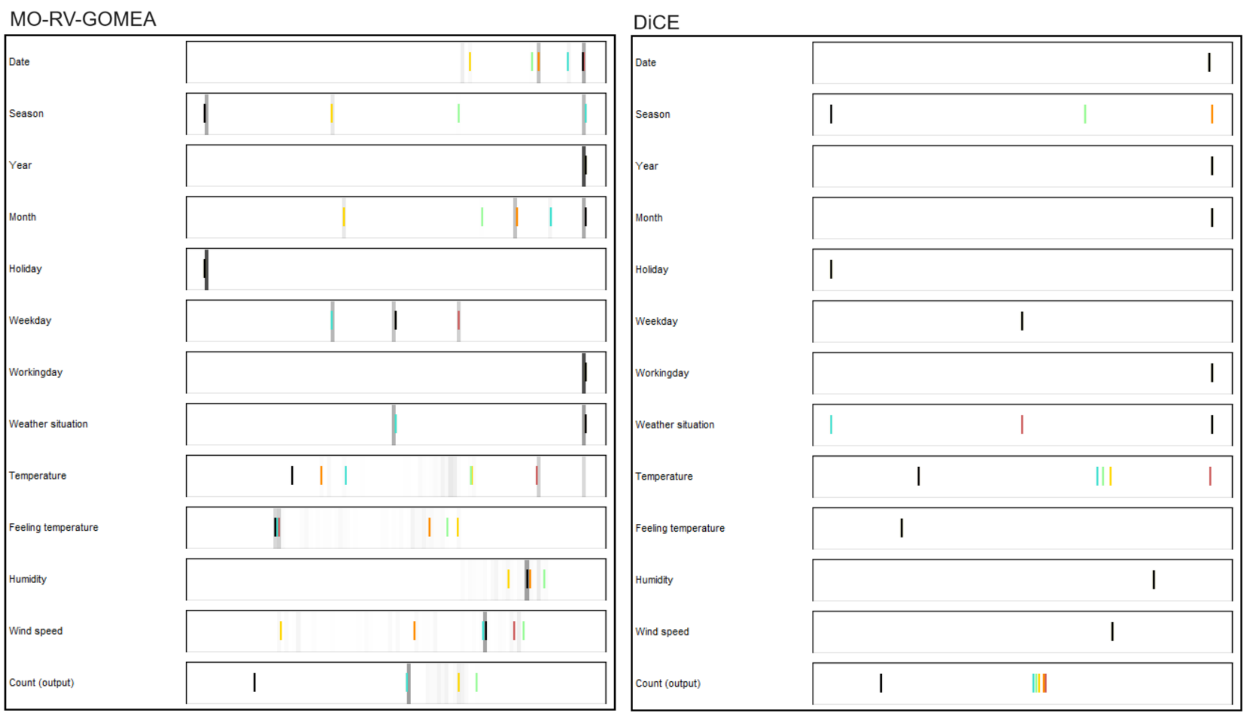


Figure 5.2: Densities per feature for a single run on the Bike Sharing dataset shown for both methods.

except for Proximity and Sparsity. The similarity between the counterfactuals and actual instances in the dataset is also visible in Table 5.1. With an average distance of 0.02 to the closest point in the dataset for MO-RV-GOMEA it is clear that they are incredibly similar.

The changes made by DiCE make some sense, when the weather is clear or the temperature is higher it is likely that more people will rent a bike. However, the temperature and feeling temperature do not correspond. To a lesser extent this is also the case for MO-RV-GOMEA.

The second point of note has to do with the output of the different methods. For the counterfactuals generated by DiCE, the model output is much closer to the counterfactual threshold than for the ones generated by MO-RV-GOMEA. Where the output values for DiCE counterfactuals varies between 4639 and 4899, the output values for MO-RV-GOMEA vary between 4631 and 6218. Again, this is partially the result from our counterfactual selection method, because there are some grey shaded feature density areas closer to the predicted value of the original data point.

The third point of note is that the changes made by MO-RV-GOMEA are much smaller than the ones made by DiCE. These changes can even be so small that they no longer show up after rounding for our translated values. An example of this is the humidity for counterfactual 1, which is the same after rounding as the humidity of the original data point. This could be caused by the way in which both methods search for counterfactuals. Both methods start with randomly initialized data points. MO-RV-GOMEA is an evolutionary algorithm and performs evolution on a relatively large population of these data points by making (small) changes to the feature values. DiCE only looks for as many counterfactuals as it needs by optimizing both diversity and proximity using gradient descent [34]. This could lead to MO-RV-GOMEA not finding the exact feature values of the original data point and other data points in the population, which then in turn shows up in the returned set of counterfactuals.

Because this experiment is based on only one run for one data point and one model, we cannot draw any conclusions yet. However, this experiment does show that when compared to DiCE, MO-RV-GOMEA seems to generate counterfactuals that make smaller changes to more features, the changes made are more realistic, and the generated counterfactuals are further away from the original data point.

5.2. Single-Modal Performance

Figure 5.3 shows the results for the increasing number of counterfactuals experiments for all datasets. For each method we show the results for both the train data point and the test data point. In this section we will discuss the results in order of the metrics.

5.2.1. %-Valid counterfactuals

The percentage of valid counterfactuals that are returned varies per dataset but stays the same even if the methods have to return an increasing amount of counterfactuals. For the yacht and concrete dataset both methods are able to return all requested counterfactuals. However, this is not the case for the bike dataset or the airfoil dataset. DiCE did not return any counterfactuals for the test data point from the bike dataset. For the airfoil dataset the methods were able to find at least some counterfactuals for each data point. These percentages are higher for the train data point than for the test data point. For the test data point we can see that DiCE only returns 40% of the counterfactuals compared to the 60% returned by MO-RV-GOMEA. When looking at the runs themselves, we see that this is caused by the methods only being able to find counterfactuals for that percentage of the models. For the models that the methods are able to generate counterfactuals for, they generate the number of counterfactuals that we ask to generate.

The methods struggle to find counterfactuals because of the data points that we are evaluating. When we look at Table 5.2 we can see why this is. For both the train instance and the test instance, the value to reach to become a counterfactual is very close to the maximum prediction on the entire train set. This is even more so the case for the test instance than for the train instance. For model 2 and 4 the value to reach is higher than the maximum value predicted on the train set. This means that it is possible that the value we need to reach to become a counterfactual is unreachable for that model.

So, MO-RV-GOMEA is able to find an equal number of or more counterfactuals than DiCE.

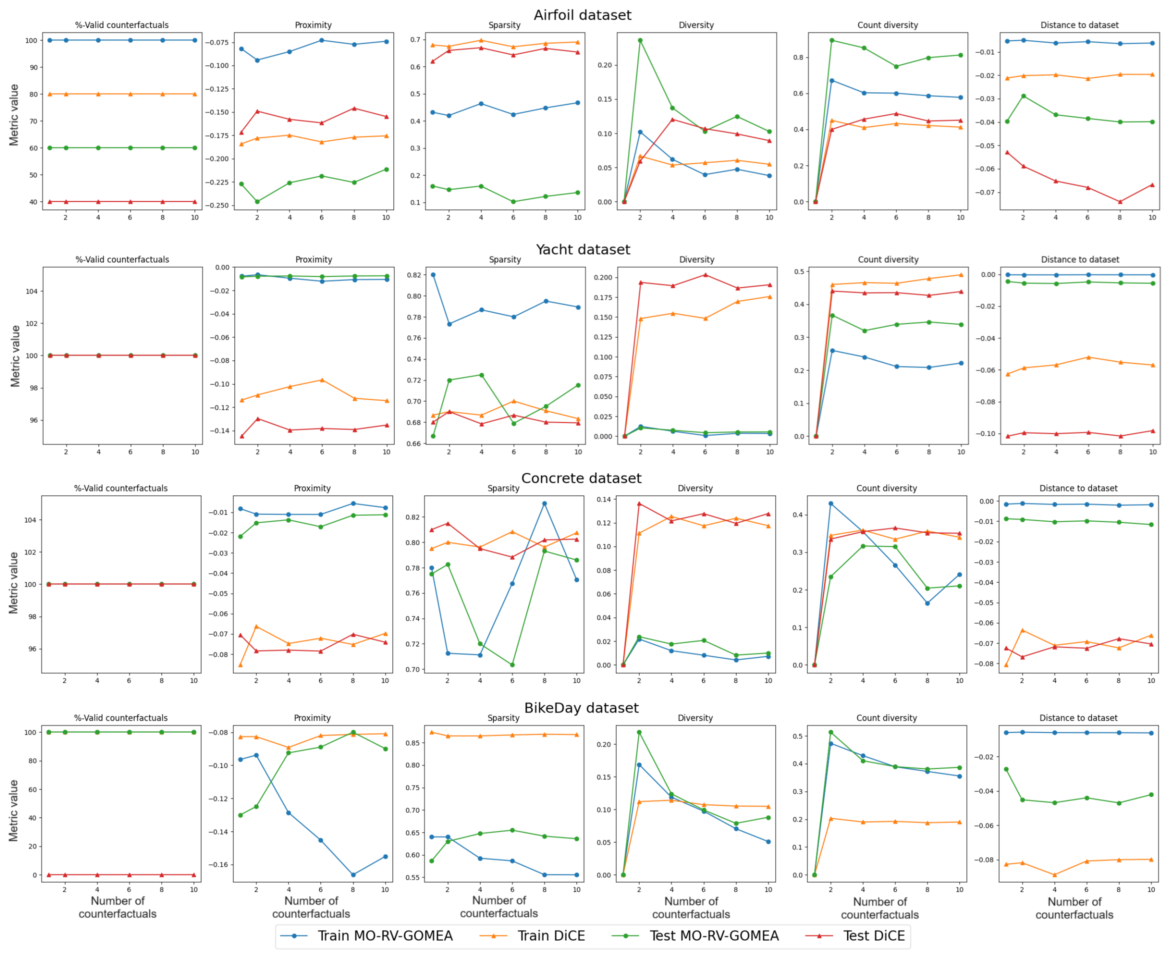


Figure 5.3: The results for the increasing number of counterfactuals experiments for all datasets using our single-modal method. Note that for the bike sharing dataset only the %-Valid counterfactuals graph shows the red line, because DiCE was not able to generate any counterfactuals for that data point. The other lines are all at the 100% value.

	Predicted value train instance	Value to reach train instance	Predicted value test instance	Value to reach test instance	Maximum on train set
Model 1	123.52	126.61	127.92	131.12	128.57
Model 2	123.62	126.71	126.76	129.93	127.32
Model 3	123.75	126.85	125.74	128.88	131.88
Model 4	125.83	128.98	127.30	130.478	130.43
Model 5	124.53	127.64	123.50	126.59	128.65

Table 5.2: Predicted value, value to reach to become a counterfactual, and maximum possible value on the train set for all 5 models used on the Airfoil dataset.

5.2.2. Proximity

The same divide that we saw for the %-valid counterfactuals, we also see for the proximity. On the yacht and concrete dataset the counterfactuals generated by MO-RV-GOMEA have a higher proximity than the ones generated by DiCE. This divide is also the case for the train data point from the airfoil dataset, but not for the test data point from that same data set. MO-RV-GOMEA has a lower proximity than DiCE on the bike dataset. However, when we look at the metric values this divide is not as large as it looks in the graph.

Because DiCE returns fewer counterfactuals for the airfoil dataset we cannot say with certainty that it is actually higher in proximity than MO-RV-GOMEA. It is possible that MO-RV-GOMEA contains a subset of counterfactuals that are similar to DiCE. If the other counterfactuals that were not found by DiCE are much farther away from the input point, this could mean that the results are skewed.

When we look at the proximity scores for the increasing number of counterfactuals we cannot see an overall trend in the performance of our single-modal method. On the airfoil and concrete dataset the proximity seems to increase slightly for higher numbers of counterfactuals, while it decreases slightly for the yacht dataset. On the bike dataset we can see that it increases for the test data point while it decreases for the train data point. When looking at the actual values this increase and decrease is not as large as it looks, with the differences being around 0.05 and -0.07 respectively. The increase is similar to the increase shown in the airfoil dataset.

5.2.3. Sparsity

The results in terms of sparsity are not as clear as the other metrics. While DiCE performs similarly on both the train and test data points from all datasets, this is not the case for MO-RV-GOMEA. Dice scores significantly better for the airfoil and bike datasets, hovering around 0.65 and 0.85 respectively. MO-RV-GOMEA has much lower metric values, with around 0.45 and 0.15 for the train and test instance on the airfoil dataset and between 0.55 and 0.65 for both instances on the bike sharing dataset. The performance of MO-RV-GOMEA on the airfoil dataset is much lower than on the yacht and concrete datasets, where the proximity does not drop under 0.66. DiCE generally outperforms MO-RV-GOMEA on the concrete dataset, while the reverse is generally true for the yacht dataset. On the yacht dataset and the concrete dataset MO-RV-GOMEA looks like it behaves much erratically, with high peaks and low valleys in the lines. However, this behavior is similar to the performance on the airfoil dataset. It is exaggerated because the scale of the graphs is different between the different datasets. As stated, the metric values for MO-RV-GOMEA on the airfoil dataset are very low. However, like for the proximity metric, we cannot conclude that DiCE actually outperforms MO-RV-GOMEA for the airfoil dataset, because we are evaluating a smaller set of counterfactuals.

Generally, there seems to be a slight upward trend when increasing the number of counterfactuals that the single-modal method has to generate. The performance of MO-RV-GOMEA is similar to or worse than DiCE for the sparsity metric.

5.2.4. Diversity and Count diversity

For the diversity metric the diversity is 0 when we ask to generate 1 counterfactual, because there are no pairs of counterfactuals to calculate the diversity between. For each dataset we do see a slight decline in diversity as the number of counterfactuals increases. This makes sense, because if we have an approximation front that spans a certain feature space, and then we select two counterfactuals from this approximation front, the diversity is high. However, depending on how the first two counterfactuals were picked, this pair-wise distance will now decrease for each counterfactual that we select. This is because there will be more counterfactuals located in the same feature space, thus making the pair-wise distances smaller.

The performance on the yacht and concrete datasets are similar. The diversity within the sets of counterfactuals generated by DiCE is higher than the diversity for the ones generated by MO-RV-GOMEA, with the diversity of DiCE being between 0.150 and 0.200 and the diversity of MO-RV-GOMEA being between 0 and 0.025. This is not the case for the airfoil and bike datasets, where the diversity is much higher within sets of size two generated by MO-RV-GOMEA. After this the diversity becomes similar to the ones generated by DiCE. For these datasets MO-RV-GOMEA's diversity is much higher than for the other datasets, namely between around 0.05 and 0.25.

There also is a difference in count diversity of MO-RV-GOMEA between the datasets. On the airfoil and bike dataset the count diversity is higher than DiCE, while the opposite is the case for the yacht and concrete dataset. This high count diversity, and the sparsity performance, is because of how MO-RV-GOMEA generates the counterfactuals. As we can see in Figure 5.1 it can make both big and small changes. While small changes do not increase the distance between the counterfactual and the input individual or the dataset significantly, they do count for both the Sparsity and Count diversity.

5.2.5. Distance to dataset

MO-RV-GOMEA outperforms DiCE on the distance to the dataset for every dataset. For MO-RV-GOMEA the distance to the dataset is smaller than -0.01 for the train instance and and smaller than -0.05 for the test instance on all datasets. Because these distances are so small, it is likely that, as for the single instance experiment, the generated counterfactuals are probably more realistic. Given that the distance is just below zero for all datasets except the airfoil dataset, it is also likely that many of these are exact, or near exact, replicas of existing data points in the data sets.

However, we do have to note that DiCE does not optimize for the distance to the dataset and our method does. While we use the distance to the dataset as a metric to capture how realistic the counterfactuals are, it makes sense that MO-RV-GOMEA outperforms DiCE on this metric.

5.2.6. General remarks

We do want to note that the performance of MO-RV-GOMEA on the bike dataset is similar to the performance on the airfoil dataset. As described earlier, the performance of MO-RV-GOMEA on the airfoil dataset is influenced by the difficulty of generating counterfactuals for the data points from that data set. This is not the case for the data points from the bike dataset. It could be that this difference in performance compared to the yacht and concrete datasets comes from a different difficulty in generating counterfactuals. The bike dataset is the dataset that has the most non-real-valued features. While we cannot conclude that this is the cause of the lowered performance, it is possible.

Overall, we see that our single-modal method outperforms DiCE on proximity for most datasets and on the distance to the dataset for all datasets. On sparsity and (count) diversity the single-modal method performs similar or worse to DiCE.

5.3. Counterfactual Selection Method Effects

While this is not part of our experiments in and of itself, we do want to touch on the effect of the counterfactuals selection method.

Figure 5.4 shows what the metrics look like if we take the runs from our previous experiment, but treat the set of all generated counterfactuals as if that is the returned set of counterfactuals. Comparing the performance of this set to the selected set of counterfactuals gives us an indication of general performance of the selection method. Note that the %-valid counterfactuals metric increases drastically because we are now dividing the size of the set of all counterfactuals by the number of counterfactuals we wanted to generate. As described in 5.2.4 the diversity also becomes smaller, because we are evaluating larger sets of counterfactuals.

When we compare the results of the entire list of counterfactuals to the results of the single-modal method, we can see that for the performance of the set of selected counterfactuals and the average performance of all generated counterfactuals are relatively similar. There are some minor differences, but these are so minor that they are not worth pointing out specifically. This means that our selection method seems to capture the average of all of the counterfactuals.

Although the impact of the selection method looks to be negligible on these instances, more evaluation is needed to be able to conclude this.

5.4. Objectives Effects

We evaluate the influence of the optimized objectives by removing one and comparing the performance to a version of MO-RV-GOMEA that optimizes for all objectives. The only objective we do not evaluate is validity, because this is the objective that makes sure that the returned counterfactuals are actually counterfactuals.

Figure 5.5 shows the results of running single-modal MO-RV-GOMEA optimizing the different objectives for only the test data point. Comparing the performance of the method optimizing all objectives to the ones that optimize three out of the four objectives allows us to evaluate how the objectives influ-

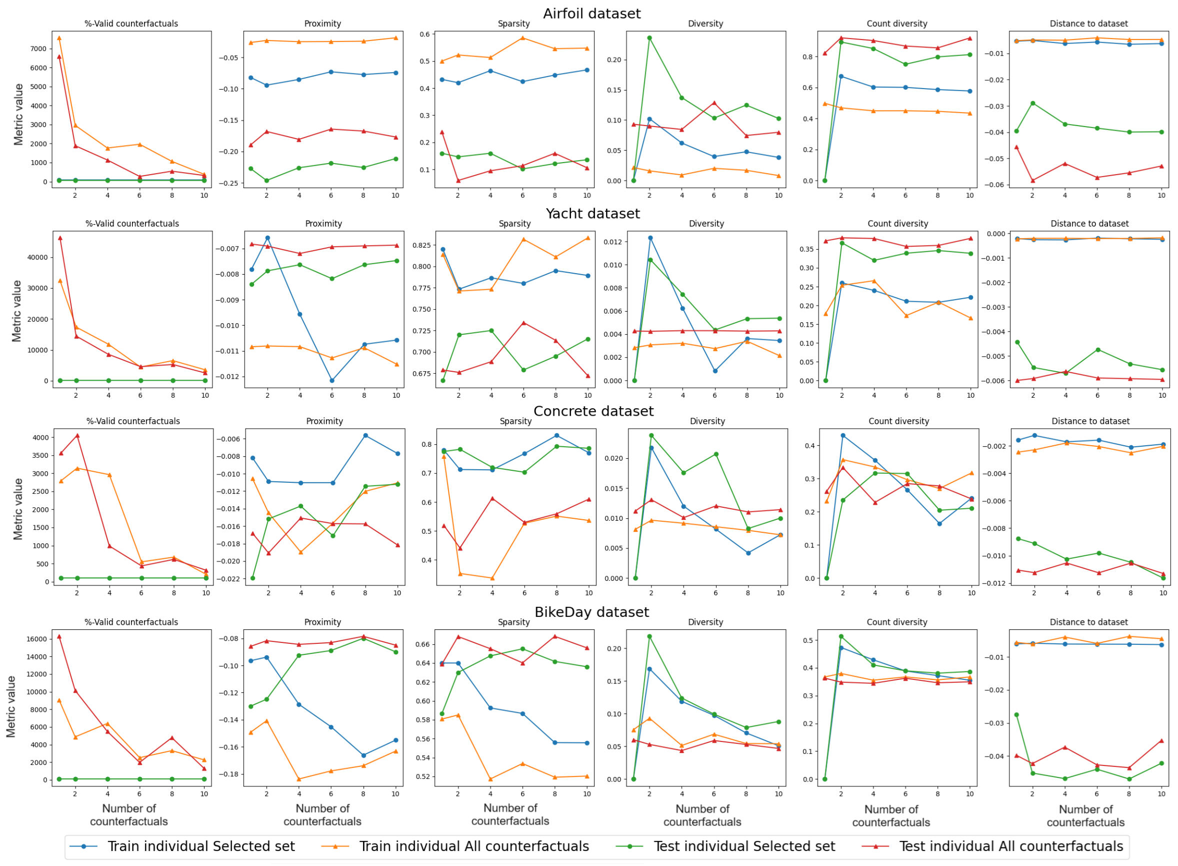


Figure 5.4: The results for the increasing number of counterfactuals experiments for all datasets using our single-modal method when calculating over all generated counterfactuals instead of the returned set.

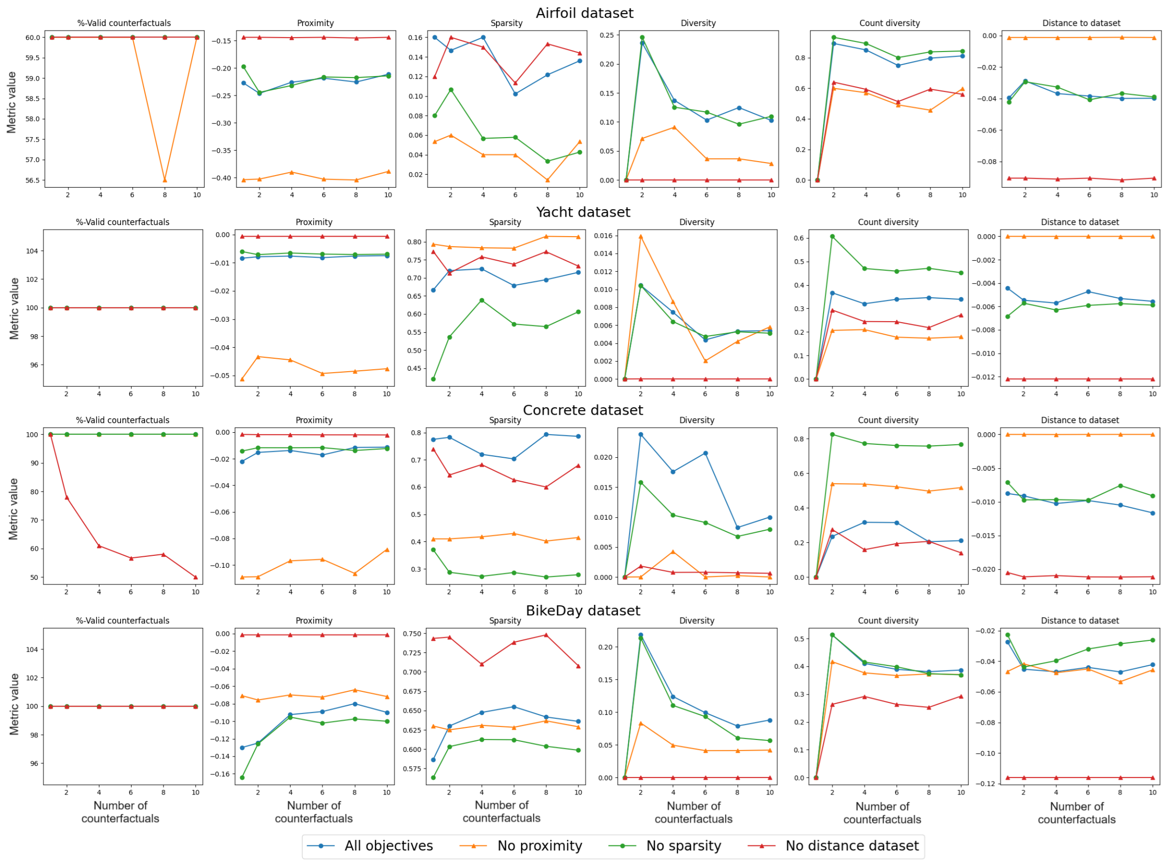


Figure 5.5: The performance of single-modal MO-RV-GOMEA on the datasets for the first instance of the test dataset. We evaluate the influence of the different objectives we optimize for.

ence each other. It also allows us to evaluate how they influence the diversity within the set of returned counterfactuals.

First we look at the influence of the proximity objective. In Figure 5.5 we can see that it influences the proximity metric. For the airfoil, yacht, and concrete dataset the proximity plummets without this objective. The largest difference is on the airfoil dataset, where it drops from between -0.20 and -0.25 to around -0.40. It also has some influence on the sparsity metric. Without optimizing for proximity, the sparsity is similar to or worse than the sparsity of optimizing for all objectives. This same change is visible in the diversity metric. What is interesting is the fact that for these datasets the distance to the dataset becomes incredibly small, showing as just below 0 in the graphs. This means that there is a trade-off between these two objectives. Another point of note is that when we do not optimize for proximity, the diversity within the generated counterfactuals reduces for all datasets except for the yacht dataset.

The trade-off makes sense given that this evaluation is on an instance of the test set. When the method has to optimize for both proximity and distance to the dataset for an individual that is not in the train set, it is likely that there is at least some-what of a distance between the individual and the train set. If we optimize to be close to both instances, there is an inherent trade-off. When one of these objectives is taken away, it becomes much easier to optimize for the other.

Second we look at the influence of the sparsity objective. As expected, the performance on the sparsity metric becomes worse when we do not optimize for sparsity. However, the performance on the other metrics stays very similar to when we optimize for all objectives. There is a slight difference on the count diversity for the yacht and concrete datasets. When the counterfactuals are less sparse, they have fewer feature values in common with the original data point, and therefore also fewer feature values in common with each other; and thus have a higher count-diversity.

Third we look at the influence of the distance to the dataset objective. When we do not optimize for distance to dataset the proximity metric is much better than for any of the other combinations of objectives. On the airfoil dataset it is just above -0.15, and on all other datasets it is just below 0. Sparsity is also higher than most others, since it performs similar to or better than when we optimize for all objectives. We also see the opposite effect of not optimizing for proximity. Now the proximity is better while the distance to the dataset is consistently much worse. With one of the objectives gone, the method is free to optimize for the other objective.

The trade-off between proximity and distance to dataset is also expressed in the diversity metric. When optimizing for both proximity and distance to dataset the diversity is generally higher than when optimizing for only one of these. Again this makes sense, because when optimizing for both, the method is not able to hone in and get as close as possible to only one of these. This effect is not as large when we do not optimize for proximity, and thus still optimize for the distance to the dataset. It is likely that this is because the train set consists of many different individuals that the counterfactual can get close to instead of just one input individual. This allows for more diversity in the set of counterfactuals while still being very close to a single instance in the train dataset.

One point of note is visible in the %-valid counterfactuals metric. Leaving out objectives can have an influence on the number of counterfactuals that the method was able to generate. For the airfoil dataset the method is only able to generate 56.5% of the asked counterfactuals when not optimizing for proximity and asking for eight counterfactuals. The percentage of valid returned counterfactuals drops from 100% down to 80% and finally around 50% when not optimizing for the distance to the dataset on the concrete dataset. With fewer objectives it is easier for one solution to be dominated by another, because we remove an entire dimension which could have a different value. This means that we can lose solutions that would otherwise be kept.

From this experiment we can conclude that the proximity, sparsity, and distance to dataset objectives each greatly influence the performance of the method on their respective metrics. We can also see that there is a trade-off between proximity and distance to dataset. It shows that the sparsity objective only has a big impact on the sparsity metric and a minor impact on the count diversity. The proximity and especially the distance to the dataset objectives greatly influence the diversity within the returned set of counterfactuals.

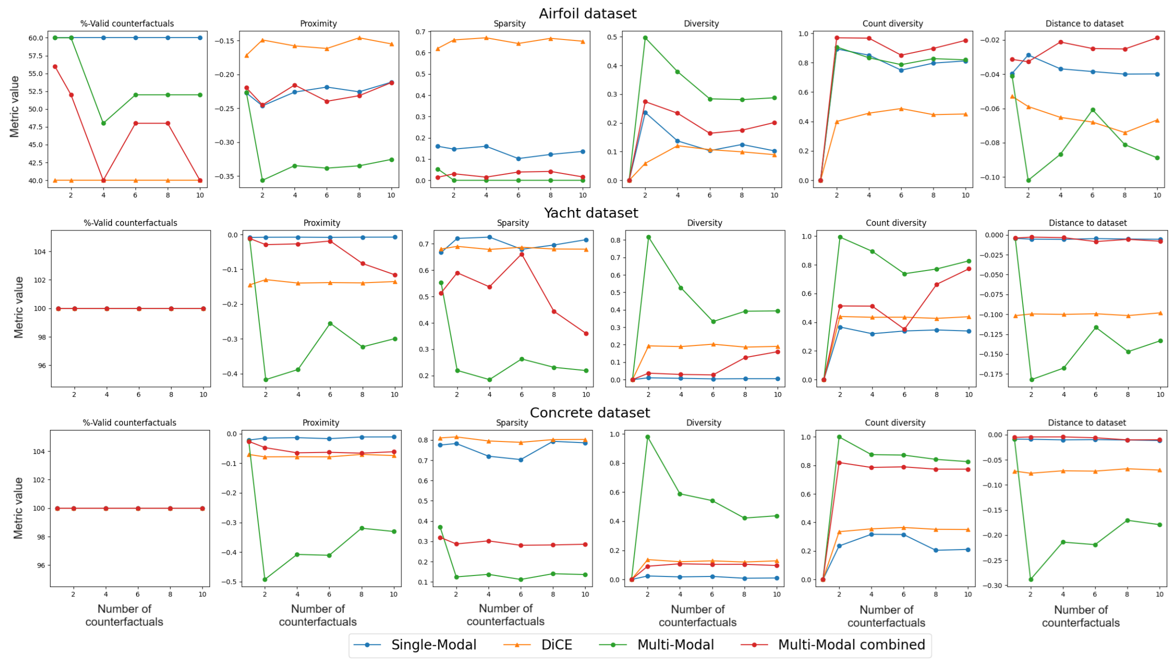


Figure 5.6: The performance of single-modal and both versions of multi-modal MO-RV-GOMEA on the datasets for the first instance of the test dataset.

5.5. Multi-Modal Performance

Figure 5.6 shows the results that compare our single-modal method, our multi-modal methods, and DiCE. In this experiment we only use the test data point.

First, we consider our base multi-modal method that uses the same objectives as our single-modal method but adds the diversity objective. In Figure 5.6 we can see that with values of 0.3 and up this method has much higher diversity, but performs far worse than the other methods in all other metrics. Therefore, it is likely that this method creates diverse counterfactuals for diversity's sake, instead of creating meaningfully diverse counterfactuals.

This is not the case for our second multi-modal method, which is the method that combines the distance to the dataset objective with the diversity objective. Figure 5.6 shows that this combined multi-modal method creates counterfactuals that are more diverse than our single-modal method, while performing much better on the other metrics than our other multi-modal method. The one objective that it performs significantly worse in is the sparsity. Overall, it seems to be a happy medium between our single-modal and other multi-modal method, and it outperforms DiCE on the proximity and distance to dataset metrics while also coming close on the diversity metric.

However, there is a discrepancy in the performance of this method, and the others, on the Airfoil dataset. This is because the combination of the data point and the tested models makes the airfoil dataset difficult to generate counterfactuals for, as described in Section 5.2. Given that this is the only dataset on which this method performs this way, it is likely that the difference in performance is caused by this.

So, this experiment shows that our first multi-modal method produces very diverse sets of counterfactuals, but at the cost of all other metrics. Our second multi-modal method generates counterfactuals that perform better in diversity than our single-modal method, while also performing similarly or slightly worse in the other metrics. When we compare both multi-modal methods the first method performs best in the diversity metric and the second method performs best in the other metrics.

6 Limitations

In this Chapter we discuss the limitations to our research. We can divide these limitations into three different groups. In Section 6.1 we describe general limitations to our method. Following that we describe a limitation that is specific to the single-modal method specifically in Section 6.2. The last Section, Section 6.3, talks about limitations specific to the multi-modal method.

6.1. General Limitations

6.1.1. Counterfactual selection procedure

When generating counterfactuals, MO-RV-GOMEA creates an entire approximation front of counterfactuals. However, the size of these approximation fronts is usually larger than the number of counterfactuals we are looking for. Therefore, we have to choose which of the counterfactuals within the approximation front we return as a final set of counterfactuals.

The way we choose the final set of counterfactuals can greatly influence the values for our metrics. Section 5.3 shows that our method seems to capture the average of the set of all generated counterfactuals within the returned set. However, it is possible that somewhere within that set there exists a set that has both a higher proximity and a higher diversity than the one we returned. This would mean that that set of counterfactuals is closer to the original data point while also offering the user with more variety to choose from.

Finding this hypothetical better set is a combinatorial optimization problem. For a small set of generated counterfactuals or a small number of counterfactuals that need to be selected this is doable. However, the number of generated counterfactuals is usually within the hundreds and sometimes even within the thousands. When we then ask for more than a few counterfactuals, the number of possible subsets becomes too large to evaluate within a reasonable time. Therefore, we have to use a different method to choose a set, like the greedy method we use now.

Instead of choosing a set using our current algorithm, we could let the user choose the type of counterfactuals we select from the approximation set. This can be used as a constraint on our search space. The types of counterfactuals could be in terms of objectives or feature values.

6.1.2. Time limit on experiments

To limit the runtime we use a time limit on our experiments. While practice showed that runs that exceeded this runtime only find data points that were not counterfactuals, it is possible that the timelimit influenced the results. This would be the case if runs that did generate counterfactuals were actually cut short by this timelimit. Depending on the machine this could mean that some runs would be able to run for less or more evaluations, thus producing slightly different results. Therefore it can have an impact on the reproducability of the experiments. While this impact would not be big, we do find it important to note.

6.1.3. Distance to dataset

Another issue that arises when looking at the single instance experiment is the fact that the method was able to reproduce an instance from the dataset. While this is not necessarily a problem within the scope of this research, it does become a problem when working with more sensitive or confidential data. For example, if we were to use a method like this to explain model predictions on patient data, we would not want the method to return other patient's data as a counterfactual, even if the records are anonymized.

6.1.4. Non-Real-Valued Features

When we look at the performance of our method on the different datasets, we can see that it performs differently on the bike sharing dataset. It is possible that this is because of the fact that that dataset consists of mostly non-real-valued features. MO-RV-GOMEA is not designed to be used on non-real-valued values. While our changes allow it to work with them, it is likely that performance on these features is nowhere near optimal. As described in 3.1.1, the reason for this is that we lose a lot of nuance due to the rounding of feature values. This could have influenced the generated counterfactuals.

6.1.5. Model quality

Model quality can have a big influence on the set of generated counterfactuals. The fact that we were unable to generate counterfactuals using CARE because the model output differs too much from the data points' labels. This points to the fact that our model quality could be too low. It is possible that our method performs worse on other models. This is not a problem for this particular research. We do however still want to note this, because this method theoretically should be able to generalize to other models.

6.1.6. Only Regression Tasks

The last general limitation we want to note is the fact that our method has only been implemented and tested for regression tasks. Like for the previous limitation this is not a problem for research itself since we only test on symbolic regression models, but we do want to note this as a potential problem. Many other methods, like DiCE, MOC, and CARE, are capable of handling both regression and classification tasks. This makes them more general and thus applicable in more situations. With some changes to the implementation, it should be possible to apply this method to classification tasks as well. However, because we have not tested this we do not know how it would perform compared to other methods like DiCE.

6.2. Single-Modal Method

6.2.1. No optimization for diversity

The limitation for the single-modal method is the fact that we do not optimize for diversity. While some results show that this there sometimes is diversity within the set of returned counterfactuals, there are also many cases where this is not the case. The method could therefore return a set of counterfactuals that are all so similar that they are of no use to the user.

6.3. Multi-Modal Method

6.3.1. Memory

When the number of counterfactuals that we want to generate increases, i.e. when k increases, the number of counterfactuals within the individuals increases with it. These individuals can become very large very quickly. This is especially the case for datasets that have a large number of features. Take the bike sharing dataset for example. This dataset has 12 features, so if we want to generate 10 counterfactuals this results in individuals consisting of 120 real-valued numbers. Depending on hardware and population size this can cause memory issues. These problems arise sooner than for the single-modal method.

6.3.2. Runtime

The number of counterfactuals we want to generate can also greatly influence the runtime. This is because each objective is calculated for each counterfactual separately. So, if our individual consists of 10 counterfactuals, we calculate the objective value for 10 times the population.

It can also potentially cause problems after evolution, when we are trying to choose k counterfactuals from the final approximation set. Since each individual consists of k counterfactuals we have to multiply the final approximation set by *k* to get the total number of generated counterfactuals. This number can also become very large very quickly, thus increasing computation time. While this process does not take as long as the generating of the counterfactuals, it does impact in the total computation time. This is especially the case if we were to use a different selection algorithm.

6.3.3. Points in Dataset

In one of the datasets we used, the Concrete dataset, there are multiple instances that have the same feature values but different target values. This is a problem for our objective that optimizes for the distance to distinct points in the dataset. The purpose of this objective is to introduce diversity by pushing the counterfactuals to distinct data points, but when these data points can still be equal the diversity within the individual can decrease. While this is an easy problem to remedy with pre-processing, this is another step that has to be taken.

7 Conclusion

To conclude, we presented one single-modal and two multi-modal methods to generate counterfactuals using MO-RV-GOMEA. We also present a visualization tool to visualize our generated counterfactuals. This tool is less complex than existing tools. For both types of counterfactual generation methods we have modified MO-RV-GOMEA to work for non-real-valued features by rounding feature values to the nearest integer whenever it can be changed to a non-integer. We evaluate these methods using the following metrics: %-valid counterfactuals, proximity, sparsity, diversity, count diversity, and distance to dataset.

The single-modal method optimizes the objectives from MOC using the standard version of MO-RV-GOMEA. Our objectives therefore are: validity, proximity, sparsity, and distance to the dataset. While examining a single instance shows that the single-modal method is able to generate counterfactuals that are realistic, further experimentation shows that we cannot guarantee that there is much diversity within the generated counterfactuals. The counterfactuals are generally higher in proximity to the original data point, but lower in sparsity when compared to the original data point.

We also present two multi-modal methods. These methods optimize (a subset of) the objectives from MOC, and one new diversity objective each. In both multi-modal methods the individuals represent a set of counterfactuals. The existing objectives are then calculated for this entire set of counterfactuals and summed to find the total objective values for that individual.

The first multi-modal method optimizes all objectives that are optimized by the single-modal method. In addition to this it also optimizes the diversity in the form of the total distance between all counterfactuals within the individual. While this multi-modal method generates sets of counterfactuals that are high in diversity, it performs much worse on the other metrics.

The second multi-modal method optimizes all objectives that are optimized by the single-modal method, except for the distance to the dataset objective. To introduce more relevant diversity instead of diversity for diversity's sake, this method optimizes a different diversity objective. This diversity objective is similar to the distance to the dataset objective. Instead of all counterfactuals within an objective being allowed to move towards the same data point, this objective pushes the counterfactuals to different data points. It does this by summing the distance to distinct data points within the dataset. This method is able to generate sets of counterfactuals that are more diverse than our single-modal method, while also keeping similar performance on other metrics.

So, we have created a method that is able to present the user with a set of diverse and relevant counterfactuals that are generated in a multi-objective way for regression datasets.

8 Future Work

In this Chapter we discuss possible future work for this research.

8.1. Other Objectives

As described in Section 2.1.3 there are many different types of objectives that are used for counterfactual generation. In this research we only used a few different options and do not cover all types of objectives. Some types of objectives that we do not optimize for include discriminative power, actionability, and causality. These different types of objectives could greatly influence the performance of the method. Evaluating more and different types of objectives could improve the methods' performance on the metrics and the relevance and usability to the user.

8.2. Counterfactual Selection from Approximation Front

The way we choose our counterfactuals from the generated approximation front can influence the performance of our method. Our current method works and seems to capture the average of the entire approximation front. However, Figure 5.2 shows that it does not compare the spread of the feature values as well as it could. While for each feature the outer grey bins have selected counterfactuals in them, we can see that between these grey bins the selected counterfactuals are not spread evenly. If we can improve this, it is possible that the method will have a high diversity while also performing better on the other metrics.

This is especially an open question with respect to the multi-modal methods. Currently we split all individuals into a list of counterfactuals, after which we use the same method to select the *k* counterfactuals for the single-modal method. However, this means that we are not taking advantage of the fact that each individual consists of *k* counterfactuals.

Instead of using the current method we could select one individual to split up and return as output, divide all generated counterfactuals into groups based on the closest data point in the dataset, or any other methods.

To compare our current method and other potential methods it would be good to evaluate all possible sets of *k* counterfactuals within a returned approximation front. This would allow us to see the metrics for the best possible set, the average set, and the worst possible set. These can then be compared to the results of the used methods. However, since the approximation fronts can become very large this is already incredibly computationally intense for even small numbers of *k* like $k = 4$.

8.3. Interactivity Within Visualization Tool

Our current visualization tool is only interactive in the sense that the user can choose which of the *k* counterfactuals to show within the tool. However, this interactivity can be extended to make more use of the sheer size of the returned approximation fronts. Examples of this are that we could allow the user to pick a region within the feature space that they want to explore by selecting regions for each feature, set values for certain features, or let them explore counterfactuals that are very similar or dissimilar to a certain counterfactual.

8.4. Other Models

While we know how our methods perform on symbolic regression models generated by GP-GOMEA, we do not know how they would perform on other types of models. However, our method is not specific to these models. With some conversion of the code the method should be able to generalize to other models as well, and therefore it would be interesting to see how the methods perform on other models.

8.5. Feature and Task Type

With the current implementation the methods are able to run with non-real-valued features, but they are not designed for these. This leads to worse performance for datasets with many non-real-valued features. Since most datasets and real-world applications will likely have a mixture of real-valued and non-real valued features, it is interesting to see if there are modifications that can be made to the method to better support these types of features.

As mentioned before, it is also the case that many methods work for both classification and regression models, while our methods are only implemented and tested for regression models. While this is not nearly as much of a wide-spread problem for our methods, there are many more regression tasks than there are real-valued feature only datasets, it could still be interesting to see how our methods perform on classification tasks as well.

8.6. Multi-Modal Individual Size

In our current multi-modal implementation, the size of the individual is dependent on k. This number of counterfactuals within the individual could have a large effect on the diversity of the final list of generated counterfactuals, because the diversity objectives are optimized within these individuals. However, when k becomes larger, this increases the size of the individual. This can become a problem for both the space and time taken to run the method. So, it might be beneficial to experiment with the number of counterfactuals within an individual, to find out what the optimal size of the individuals is. This could include using a minimum and/or a maximum on the number of counterfactuals within an individual, multiplying k with a constant to get the number of counterfactuals within an individual, or always setting it to a certain number independent of k.

8.7. Elitist Archive

MOC changes the crowding distance computation to, in addition to taking the objective space into account, also take the feature space into account [10]. They say that this allows the method to keep individuals even if they do not vary greatly in objective values but do vary in feature values.

It could be beneficial to make a similar change for MO-RV-GOMEA. This could be done in the elitist archive. Currently the elitist archive is created by checking domination between individuals on objective and constraint values, but this could potentially be changed to also take the feature space into account. Like with MOC, counterfactuals could be kept in the elitist archive if they differ enough from the other counterfactuals while also being dominated. By doing this it could lead to better diversity for the single-modal implementation of our method.

8.8. Expert Evaluation

The metrics that we use to evaluate the methods capture objective data, but they cannot necessarily tell us how realistic and achievable the generated counterfactuals are. While we can personally look at an instance for the bike sharing dataset and judge these factors ourselves, we are not able to do that for the other datasets. It could be very informative to do a user study with experts within these fields, or other fields for other datasets, that can better judge the quality of the generated counterfactuals.

8.9. Real-World Data

As described in the previous section, our evaluation metrics do not capture user preferences. We also only evaluate our method using existing datasets. While the bike share dataset consists of real-world data, the others were all captured in experimental settings. This means that we have mainly evaluated our methods using non-real-world data.

It would be beneficial to evaluate the methods on real-world applications. Not only would this allow us to further investigate how the methods perform on real-world data, but it also allows us to take a human-in-the-loop approach to evaluation. This also allows us to take expert evaluation into account and non-expert user evaluation.

Bibliography

[1] Ahmed Al Kuwaiti et al. “A Review of the Role of Artificial Intelligence in Healthcare”. In: *Journal of Personalized Medicine* 13.6 (2023). ISSN: 2075-4426. DOI: `10.3390/jpm13060951`. URL: `https://www.mdpi.com/2075-4426/13/6/951`.

[2] Peter A. N. Bosman, Jörn Grahl, and Dirk Thierens. “Benchmarking Parameter-Free AMaLGaM on Functions With and Without Noise”. In: *Evolutionary Computation* 21.3 (Sept. 2013), pp. 445–469. ISSN: 1063-6560. DOI: `10.1162/EVCO_a_00094`. eprint: `https://direct.mit.edu/evco/article-pdf/21/3/445/1501050/evco\_a\_00094.pdf`. URL: `https://doi.org/10.1162/EVCO%5C_a%5C_00094`.

[3] Peter A.N. Bosman. “The anticipated mean shift and cluster registration in mixture-based EDAs for multi-objective optimization”. In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. GECCO ’10. Portland, Oregon, USA: Association for Computing Machinery, 2010, pp. 351–358. ISBN: 9781450300728. DOI: `10.1145/1830483.1830549`.

[4] Anton Bouter et al. “Exploiting linkage information in real-valued optimization with the real-valued gene-pool optimal mixing evolutionary algorithm”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO ’17. Berlin, Germany: Association for Computing Machinery, 2017, pp. 705–712. ISBN: 9781450349208. DOI: `10.1145/3071178.3071272`.

[5] Anton Bouter et al. “The multi-objective real-valued gene-pool optimal mixing evolutionary algorithm”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO ’17. Berlin, Germany: Association for Computing Machinery, 2017, pp. 537–544. ISBN: 9781450349208. DOI: `10.1145/3071178.3071274`. URL: `https://doi.org/10.1145/3071178.3071274`.

[6] Thomas Brooks, D. Pope, and Michael Marcolini. *Airfoil Self-Noise*. UCI Machine Learning Repository. 2014. DOI: `https://doi.org/10.24432/C5VW2C`.

[7] R. M. J. Byrne. “Counterfactual thought”. In: *Annual Review of Psychology* 67 (1 2016), pp. 135–157. DOI: `10.1146/annurev-psych-122414-033249`.

[8] Furui Cheng, Yao Ming, and Huamin Qu. *DECE: Decision Explorer with Counterfactual Explanations for Machine Learning Models*. 2020. arXiv: `2008.08353 [cs.LG]`.

[9] European Commission. *AI Act*. URL: `https://digital-strategy.ec.europa.eu/en/policies/regulatory-framework-ai`.

[10] Susanne Dandl et al. “Multi-Objective Counterfactual Explanations”. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2020, pp. 448–469. ISBN: 9783030581121. DOI: `10.1007/978-3-030-58112-1_31`. URL: `http://dx.doi.org/10.1007/978-3-030-58112-1_31`.

[11] Kalyanmoy Deb and Himanshu Jain. “An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints”. In: *IEEE Transactions on Evolutionary Computation* 18.4 (2014), pp. 577–601. DOI: `10.1109/TEVC.2013.2281535`.

[12] Kalyanmoy Deb et al. “A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II”. In: *Parallel Problem Solving from Nature PPSN VI*. Ed. by Marc Schoenauer et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 849–858. ISBN: 978-3-540-45356-7.

[13] Kai Epstude and Neal J. Roese. “The Functional Theory of Counterfactual Thinking”. In: *Personality and Social Psychology Review* 12.2 (2008). PMID: 18453477, pp. 168–192. DOI: `10.1177/1088868308316091`. eprint: `https://doi.org/10.1177/1088868308316091`. URL: `https://doi.org/10.1177/1088868308316091`.

[14] Kai Epstude, Annika Scholl, and Neal J. Roese. "Prefactual Thoughts: Mental Simulations about What Might Happen". In: *Review of General Psychology* 20.1 (2016), pp. 48–56. DOI: `10.1037/gpr0000064`. eprint: `https://doi.org/10.1037/gpr0000064`. URL: `https://doi.org/10.1037/gpr0000064`.

[15] Hadi Fanaee-T. *Bike Sharing*. UCI Machine Learning Repository. 2013. DOI: `https://doi.org/10.24432/C5W894`.

[16] J. Gerritsma, R. Onnink, and A. Versluis. *Yacht Hydrodynamics*. UCI Machine Learning Repository. 2013. DOI: `https://doi.org/10.24432/C5XG7R`.

[17] Lorentsa Gkinko and Amany Elbanna. "The appropriation of conversational AI in the workplace: A taxonomy of AI chatbot users". In: *International Journal of Information Management* 69 (2023), p. 102568. ISSN: 0268-4012. DOI: `https://doi.org/10.1016/j.ijinfomgt.2022.102568`. URL: `https://www.sciencedirect.com/science/article/pii/S0268401222001025`.

[18] Oscar Gomez et al. *AdViCE: Aggregated Visual Counterfactual Explanations for Machine Learning Model Validation*. 2021. arXiv: `2109.05629 [cs.HC]`.

[19] Oscar Gomez et al. "ViCE: Visual Counterfactual Explanations for Machine Learning Models". In: *Proceedings of the 25th International Conference on Intelligent User Interfaces*. IUI '20. Cagliari, Italy: Association for Computing Machinery, 2020, pp. 531–535. ISBN: 9781450371186. DOI: `10.1145/3377325.3377536`.

[20] J. C. Gower. "A General Coefficient of Similarity and Some of Its Properties". In: *Biometrics* 27.4 (1971), pp. 857–871. ISSN: 0006341X, 15410420. URL: `http://www.jstor.org/stable/2528823` (visited on 08/06/2024).

[21] Riccardo Guidotti. "Counterfactual explanations and how to find them: literature review and benchmarking". en. In: *Data Mining and Knowledge Discovery* (Apr. 2022). ISSN: 1573-756X. DOI: `10.1007/s10618-022-00831-6`. URL: `https://doi.org/10.1007/s10618-022-00831-6` (visited on 11/20/2023).

[22] David Gunning et al. "XAI—Explainable artificial intelligence". In: *Science Robotics* 4.37 (2019), eaay7120. DOI: `10.1126/scirobotics.aay7120`. eprint: `https://www.science.org/doi/pdf/10.1126/scirobotics.aay7120`. URL: `https://www.science.org/doi/abs/10.1126/scirobotics.aay7120`.

[23] Bart Hofmans. *Interactive visualization for the exploration and comparison of counterfactual explanations*. Apr. 2023. URL: `https://research.tue.nl/en/studentTheses/interactive-visualization-for-the-exploration-and-comparison-of-c`.

[24] The White House. *FACT SHEET: President Biden Issues Executive Order on Safe, Secure, and Trustworthy Artificial Intelligence*. URL: `https://www.whitehouse.gov/briefing-room/statements-releases/2023/10/30/fact-sheet-president-biden-issues-executive-order-on-safe-secure-and-trustworthy-artificial-intelligence/`.

[25] Ipsos. *GLOBAL VIEWS ON A.I. 2023*. 2023. URL: `https://www.ipsos.com/`.

[26] Kentaro Kanamori et al. "DACE: Distribution-Aware Counterfactual Explanation by Mixed-Integer Linear Optimization". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. Ed. by Christian Bessiere. Main track. International Joint Conferences on Artificial Intelligence Organization, July 2020, pp. 2855–2862. DOI: `10.24963/ijcai.2020/395`. URL: `https://doi.org/10.24963/ijcai.2020/395`.

[27] Amir-Hossein Karimi et al. "A Survey of Algorithmic Recourse: Contrastive Explanations and Consequential Recommendations". In: *ACM Comput. Surv.* 55.5 (Dec. 2022). ISSN: 0360-0300. DOI: `10.1145/3527848`. URL: `https://doi.org/10.1145/3527848`.

[28] Thibault Laugel et al. "Achieving Diversity in Counterfactual Explanations: a Review and Discussion". In: *2023 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '23. ACM, June 2023. DOI: `10.1145/3593013.3594122`. URL: `http://dx.doi.org/10.1145/3593013.3594122`.

[29] Rui Li et al. "Mixed integer evolution strategies for parameter optimization". In: *Evolutionary computation* 21.1 (2013), pp. 29–64.

[30] Ngoc Hoang Luong, Han La Poutré, and Peter A.N. Bosman. “Multi-objective gene-pool optimal mixing evolutionary algorithms”. In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. GECCO ’14. Vancouver, BC, Canada: Association for Computing Machinery, 2014, pp. 357–364. ISBN: 9781450326629. DOI: 10.1145/2576768.2598261.

[31] Nour Makke and Sanjay Chawla. “Interpretable scientific discovery with symbolic regression: a review”. In: *Artificial Intelligence Review* 57.2 (Jan. 2024). DOI: 10.1007/s10462-023-10622-0.

[32] Nestor Maslej et al. *The AI Index 2024 Annual Report*. AI Index Steering Committee, Institute for Human-Centered AI, Stanford University, Apr. 2024.

[33] Forrest E Morgan et al. “Military applications of artificial intelligence”. In: *Santa Monica: RAND Corporation* (2020).

[34] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. “Explaining machine learning classifiers through diverse counterfactual explanations”. In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. FAT* ’20. ACM, Jan. 2020. DOI: 10.1145/3351095.3372850. URL: http://dx.doi.org/10.1145/3351095.3372850.

[35] Rafael Poyiadzi et al. “FACE: Feasible and Actionable Counterfactual Explanations”. In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. AIES ’20. ACM, Feb. 2020. DOI: 10.1145/3375627.3375850. URL: http://dx.doi.org/10.1145/3375627.3375850.

[36] Peyman Rasouli and Ingrid Chieh Yu. “CARE: Coherent Actionable Recourse based on Sound Counterfactual Explanations”. In: *International Journal of Data Science and Analytics* (2022), pp. 1–26.

[37] E. M. C. Sijben, T. Alderliesten, and P. A. N. Bosman. *Multi-modal multi-objective model-based genetic programming to find multiple diverse high-quality models*. 2022. arXiv: 2203.13347 [cs.NE]. URL: https://arxiv.org/abs/2203.13347.

[38] Jan-Tobias Sohns, Christoph Garth, and Heike Leitte. “Decision Boundary Visualization for Counterfactual Reasoning”. In: *Computer Graphics Forum* 42.1 (2023), pp. 7–20. DOI: https://doi.org/10.1111/cgf.14650.

[39] Ilia Stepin et al. “A Survey of Contrastive and Counterfactual Explanation Generation Methods for Explainable Artificial Intelligence”. In: *IEEE Access* 9 (2021), pp. 11974–12001. DOI: 10.1109/ACCESS.2021.3051315.

[40] Dirk Thierens and Peter A.N. Bosman. “Optimal mixing evolutionary algorithms”. In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. GECCO ’11. Dublin, Ireland: Association for Computing Machinery, 2011, pp. 617–624. ISBN: 9781450305570. DOI: 10.1145/2001576.2001661.

[41] NOAA US Department of Commerce. *Beaufort wind scale*. Sept. 2016. URL: https://www.weather.gov/mfl/beaufort.

[42] Marco Virgolin et al. “A Model-based Genetic Programming Approach for Symbolic Regression of Small Expressions”. In: *CoRR* abs/1904.02050 (2019). arXiv: 1904.02050. URL: http://arxiv.org/abs/1904.02050.

[43] Sandra Wachter, Brent Mittelstadt, and Chris Russell. *Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR*. 2018. arXiv: 1711.00399 [cs.AI]. URL: https://arxiv.org/abs/1711.00399.

[44] I-Cheng Yeh. *Concrete Compressive Strength*. UCI Machine Learning Repository. 2007. DOI: https://doi.org/10.24432/C5PK67.