Quantifying the Endogenous Domain and Model Shifts Induced by the CLUE Recourse Generator

Karol Dobiczek
Supervisor(s): Cynthia C. S. Liem, Patrick Altmeyer
EEMCS, Delft University of Technology, The Netherlands

June 18, 2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

Employing counterfactual explanations in a recourse process gives a positive outcome to an individual, but it also shifts their corresponding data point. For systems where models are updated frequently, a change might be seen when recourse is applied, and after multiple rounds, severe shifts in both model and domain may occur. Algorithmic recourse frameworks such as CARLA compare the counterfactual generators based on the effectiveness and cost of employing recourse, but little to no previous work has been done on analyzing the shifts in dynamics of the systems. In this paper, we propose a set of metrics aimed at measuring shifts in the domains and models employed in those systems, as well as an experiment framework built on top of CARLA. These metrics allow us to analyze experimentally the characteristics of shifts in dynamics induced by the CLUE and Wachter generators.

1 Introduction

Machine learning models as a decision-making and decision support tool are increasing in popularity in domains such as medical diagnoses [1], credit score evaluation [2] or job candidate screening [3]. The nature of these domains can mean that a prediction made by a model used for this task may influence a person's life directly, for example in a case of an incorrect medical diagnosis. A decision made by a black-box model can have a strong influence on one's life. It is thus crucial, especially in cases where a human is involved, to provide explainable decisions and constructive means of changing the decision outcome to the advantage of the end user[4].

One of the ways to address these problems is through counterfactual explanations (CEs). These can take a form of a sentence similar to *"You have received a negative credit score assessment because your monthly pay is 10,000€. If you had a monthly pay of 15,000€, you would have received a positive credit score."* In the given example the counterfactual shows one of the possible feature changes that lead to a favourable outcome for the end-user [5].

If a human is involved in the process of providing counterfactual explanations, the process is then referred to as *recourse* [4]. There, the counterfactual explanations are provided to the subjects and if successfully employed, they essentially shift the data point corresponding to the subject to the other side of the classifier's decision boundary. When enough data points in the dataset have been moved as an effect of the recourse process, they can cause shifts in both the dataset and the classifier model. We have chosen to refer to those shifts in the dynamics of the system as endogenous shifts, due to them stemming from within the process.

Numerous methods providing counterfactual explanations were developed over the years, 11 of which have been included in *Counterfactual And Recourse Library - CARLA* [4], a benchmarking framework with a specific focus on comparing recourse methods. Apart from recourse methods, this framework contains standardized data sets that can be used to evaluate the analyzed effects. The recourse method selected for this research project, *Counterfactual Latent Uncertainty Explanations - CLUE* [6] is a recourse generator which aims to provide the smallest change to an input, such that it stays on the data manifold and the model's decision on said input is certain.

In this work, we aim to answer the question ***"What are the characteristics of shifts induced by the CLUE recourse generator?"*** by comparing it to a baseline *Wachter* generator. We provide answers for the following sub-questions:

- Does the magnitude of observed shifts differ compared to the baseline generator?
- What might be playing a role in the observed differences?
- What appear to be good ways of mitigating endogenous shifts?

To this end, we explore methods of quantification of the endogenous model and domain shifts induced by using recourse generators. We analyze and discuss results found by quantifying the dynamic shifts induced by two of the generators included in the *CARLA* framework. The shifts induced by recourse generators can stem from various variables of the system. The research entails

experimentally analyzing the distribution shifts of the datasets, shifts in models' performance and decision boundaries, and the quality of recourse.

Using the results gained from the experiments, we define the characteristics of the endogenous model shifts induced by the *CLUE* generator. The results found by performing experiments on the *CLUE* recourse generator have been compared to those found by analyzing the recourse process of the generator by Wachter et. al [5]. The dynamics were compared to test whether the process generates new, distinct clusters in the dataset and to see how it affects the model's performance on the original data. The analysis also helped define factors that play a role in increasing the magnitude of the shifts, such as the models' or generators' hyperparameters.

Previous work on the topic has been done by Upadhyay et al. [7] in their paper introducing the *ROAR* generator and by Pawelczyk et al. [4] in the *CARLA* framework. These papers compare recourse generators based on metrics such as validity, costs, redundancy, or closeness of the counterfactual to the correct cluster. These metrics although useful for evaluating the effectiveness of the generated CEs, do not capture their short- and long-term influence on the system.

The scope of our research is to provide metrics for analyzing the shifts in dynamics and to use them to characterize and compare shifts created by using certain recourse generators. We contribute to the field of algorithmic recourse in two ways. We create a framework capable of executing algorithmic recourse experiments and analyzing the shifts in dynamics of the underlying systems. This allows the future work in the field to easily evaluate recourse generators in terms of the induced shifts. We also perform experiments on two recourse generators and analyze the characteristics and causes of the shifts they induce. By providing information on how different types of recourse generators perform on certain types of domains and with different black-box models, we aim to show and explain the dynamics governing the processes. This can help with better use of the generators for different tasks.

The paper is structured as follows. First, in Section 2 we discuss the previous research in the field and we show the relevancy of the undertaken research given the knowledge gaps in the field. Secondly, in Section 3, we show the steps taken to answer the research question by stating the experimental setup, the resources and techniques used in the research. In Section 4, we show and describe the obtained results. In Section 5 we discuss the obtained results and explaining possible shortcomings of the research. Section 6 reflects on the possible ethical issues of the conducted research and the steps taken to ensure reproducibility. Finally, in Section 7, we conclude by stating the steps taken in the research process, the crucial findings and contributions, and the possible future work in this field.

2 Related work

Algorithmic Recourse

Algorithmic recourse generators aim to provide alternatives to decisions made by black-box machine learning models through *counterfactual explanations*, such that the counterfactual outcome is positive from the point of view of the end-user [4]. As discussed in Section 1, numerous implementations of recourse generators have been produced over the years [5–9]. This paper focuses on comparing the effects of applying recourse generated by two generators: the *CLUE* generator was compared to a baseline generator created by Wachter et al. [5].

The recourse generator formulated by **Wachter** et al. [5] has been chosen as a baseline to compare the results obtained by generating recourse with *CLUE*, since it is one of the first recourse generators created and it has a relatively simple objective function. For factuals x it optimizes generated CEs x' using the following objective function: $\arg\min_{x'} \max_{\lambda} \lambda(f_w(x') - y)^2 + d(x', x)$ with $f_w(x')$ being the CE label, y being the original label and $d(\cdot, \cdot)$ a distance function. Due to the inclusion of d in the formula, it generates CEs to be as close to the original factuals as possible. This means that it is very likely for the CEs to appear close to the decision boundary of the model, as seen on Figure 1.

The generator tested against *Wachter* in this paper, ***CLUE*** [6] is a generative recourse model. It uses a variational autoencoder (VAE) [10] to estimate the generative model of the domain. The CEs are generated by performing a search in the latent space of the encoder for points which generate inputs similar to the original observations in the dataset but are assigned a low uncertainty by the model. The uncertainty is measured as the noise inherent to the generative process of the data and it can stem from class overlap. This approach allows *CLUE* to generate counterfactual explanations that are close to the original distribution, as opposed to generating noisy inputs that get classified as the

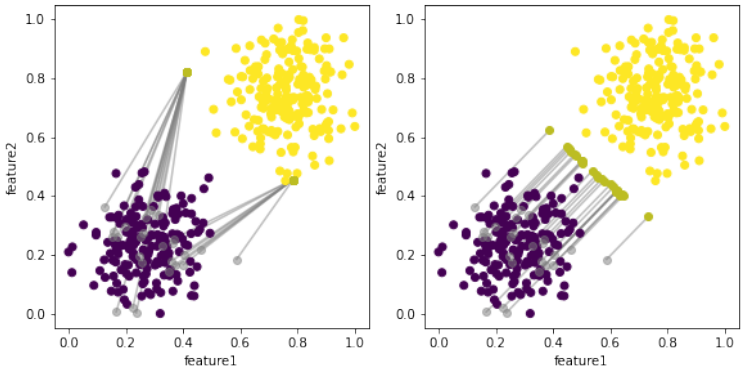

Figure 1: Dataset after a single round of recourse, 25 CEs generated. Left: *CLUE*, right: *Wachter*. The negative class is colored purple, the positive class is colored yellow. Each factual (light grey) and its corresponding counterfactual (olive) are linked with a light grey line. *CLUE* can generate multiple CEs in the same location.

positive class. Figure 1 shows CEs generated by *CLUE*, they fit in to the positive class distribution, while CEs generated by *Wachter* tend to approximate the decision boundary of the model.

Evaluating counterfactual explanations

The *CARLA* framework contains a benchmarking capability for evaluating the CEs found by the generators included in the framework. It employs metrics such as the distance between a factual and its counterfactual explanation, the cost, constraint violations for categorical or immutable features, redundancy - how many proposed feature changes were not necessary, or yNN - a measure of how well a counterfactual falls into the positive class [4].

In a recent work, Upadhyay et al. [7] introduced Robust Algorithmic Recourse (ROAR), a generator whose objective function minimizes the cost of recourse for a maximal possible model shift. The authors evaluate their framework and compare it to other existing generators based on counterfactual cost and validity - the success rate of the recourse.

These measures are useful for evaluating the generators based on the quality of recourse they provide, but they provide little information in terms of how employing the may CEs affect the domain and the model. Given these circumstances, metrics capable of providing information about shifts need to be developed.

Towards quantifying shifts

The notion of shifts in the context of machine learning models has been studied thoroughly in recent decades. Concept drift is one of the earlier ways of describing systems in which changing domains affect the performance or classifications of learning algorithms. Widmer and Kubat (1996) [11] describe concept drifts as the products of radical changes in target concepts caused by changes in the hidden context. A more specific concept of dataset shifts is described as ”a situation where the joint distribution of inputs and outputs differs between training and test stage“ (Quinonero-Candela et al.) [12, p. 29]. This type of shifts can be noticed in the recourse process, where in each round several data points gain a new location in the dataset.

Numerous methods of correcting dataset shifts have been developed, methods including estimation of biased distributions or selection probabilities [13, 14], prior knowledge of class probabilities [15], or nonparametric methods such as Kernel Mean Matching [16]. The latter approach is designed to deal with a particular case of dataset shift called covariate shift. It minimizes a distance function called Maximum Mean Discrepancy [17] which calculates a distance of probability distribution embeddings in an Reproducing Kernel Hilbert Space. This metric can be generalized to be used on any probability distribution by providing samples to the kernel function, and furthermore, a distribution shift p-value can be calculated using the bootstrap method [17].

3 Methods

To characterize the shifts in dynamics of recourse, we propose an experiment which uses numerous metrics to quantify shifts for both the domain and the model. This section describes the experiment, the metrics and datasets used in the experiment as well as the motivation behind their use.

3.1 Experiment

We propose an experiment in which we measure the dynamics of the system undergoing numerous subsequent rounds of algorithmic recourse. The experiment consists of a set number of n iterations of the following sequence. In iteration i, with generators G using dataset D_{i-1} and model M_{i-1} do:

Algorithm 1: Experiment iteration

foreach *generator* $g \in G$ **do**
- Generate factuals F_i using model M_{i-1}
- Generate counterfactuals using the recourse generator for factuals in F_i
- Create D_i by updating D_{i-1} with counterfactuals
- Train a new model M_i on dataset D_i
- Measure shifts in the dataset by comparing D_i to the initial dataset D_0
- Measure shifts in the model by comparing M_i to the initial model M_0
- Generate counterfactual benchmarks using *CARLA*

This setup allows us to gather information about the shifts in varying scenarios such as different datasets, model and generator hyperparameters, or the parameters of the recourse generators. In later iterations the datasets and the models can differ, thus all recourse methods must have a separate copy of the dataset and the model. This copy then gets updated with counterfactuals generated in each round of recourse. The factuals used in the recourse process are the instances in the dataset that have been predicted by a model to be members of the negative class. Since all recourse generators have separate models, an intersection of their factuals has to be taken to ensure all generators work on the same factual instances.

3.2 Measurements

3.2.1 Domain shifts

Maximum Mean Discrepancy (MMD) is a statistic used to calculate a distance between two distributions. MMD is an unbiased empirical estimate of the distance $||\mu_p - \mu_q||$ of two mean embeddings μ_p and μ_q in the Hilbert space [17]. By embedding the probability distributions in characteristic Reproducing Kernel Hilbert Spaces (RKHS), the metric can capture differences in higher-order moments of distribution. Those differences are encoded in the differences of means of nonlinear features of the embeddings [18].

The squared maximum mean discrepancy for two random variables x and y defined on a topological space $\mathcal{X}$ with probability distributions p and q respectively, and $\mathcal{F}$ a class of functions $f : \mathcal{X} \to \mathbb{R}$ is defined as:

$$\begin{aligned} MMD^2[\mathcal{F}, p, q] &:= ||\mu_p - \mu_q||^2 \\ &= \left[\sup_{f \in \mathcal{F}} (\mathbb{E}_x[f(p)] - \mathbb{E}_y[f(q)]) \right]^2 \end{aligned} \tag{1}$$

In an RKHS the squared MMD can be estimated in terms of kernel functions k:

$$MMD^2[\mathcal{F}, X, Y] = \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i}^{m} k(x_i, x_j) + \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} k(y_i, y_j) - \frac{2}{mn} \sum_{i=0}^{m} \sum_{j=0}^{n} k(x_i, y_j) \tag{2}$$

where $X = \{x_1, ..., x_m\}$ and $Y = \{y_1, ..., y_n\}$ are observations randomly sampled from distributions p and q respectively[17] and with a squared exponential kernel $k(x, y) = e^{-\frac{1}{\sigma}||x-y||^2}$ with σ normally being the median distance for all points in X and Y aggregated [19], though for the sake of consistency between experiments and in order to avoid time consuming calculations a set value of $\sigma = 0.15$ was used. The characteristic kernel of the characteristic RKHS guarantees that for $P = Q$ the value of $\text{MMD}^2[P, Q]$ will be equal to zero [18].

MMD is used at the end of each iteration to compare the initial data distribution to the newly updated one and in model shift calculation in form of probability MMD and decision boundary mesh MMD described in the following subsection. An estimated *p-value* for domain shift can be obtained with MMD using bootstrap with null hypothesis $p = q$ for the original data distribution and the distribution after the last round of recourse.

3.2.2 Model shifts

Disagreement is a pseudometric used in our experiments to express a notion of distance between two machine learning models. It is defined as the probability of two models predicting a different label for an arbitrary data point [20] and it is calculated as the fraction of data points for which the two models disagree. For two models a and b the metric is calculated as:

$$\frac{1}{n} \sum_{x \in X}^{n} |f_a(x) - f_b(x)| \tag{3}$$

with $f_a(x)$ and $f_b(x)$ being the predicted labels for data point x in domain X of length n.

MMD for model shifts is calculated using the algorithm for Maximum Mean Discrepancy described above. We propose two metrics that use MMD for measuring model shifts:

- Model MMD
- Probability MMD

Model MMD generates a grid of sample points over the space of the dataset. For each point, we generate its predicted probabilities using the initial model and the model at the current round. The probability distributions of the two meshes are then compared using the MMD to generate a measurement of the model shift. Due to the high runtime of the MMD algorithm for high numbers of samples, we limited model MMD to only generate $10,000$ sample points, meaning it will generate $\lceil \sqrt[n]{10000} \rceil$ points per dimension for a dataset with n dimensions. This metric ensures a global measurement of the shift of the decision boundary for the entire domain.

In the case of the **probability MMD**[1], we propose the calculation of MMD with the predicted probabilities of the initial model and the newly trained models. The motivation behind using this metric is to calculate the magnitude of the shift of the models' decision boundaries. As the decision boundary shifts or changes its gradient, the distribution of predicted probabilities for the data points in the original dataset will also change. This shift is then calculated by MMD.

Prediction mean and variance are the last metrics used for quantifying the model shifts. There are two reasons which justify the use of those. Firstly, the decision boundary shift towards the negative class expected in the course of recourse might mean higher prediction values for instances of the positive class that were once close to the boundary. This can be an unwanted side effect of recourse as it might mean that the prediction certainty for the individuals represented by those data points has been artificially elevated. Secondly, one of the features of the *CLUE* generator is the high certainty of the generated counterfactuals. These metrics can be used to analyze whether the predictions for the data points generated by *CLUE* are higher than for those generated by the baseline generator.

[1] Suggested by a colleague, Aleksander Buszydlik

3.2.3 Benchmarks

The *CARLA* framework provides methods for benchmarking the generated counterfactuals [4]. The metrics used in these benchmarks are **y-Nearest-Neighbours**, a metric signifying whether the counterfactual explanation has been generated near the members of the positive class, the **average time** for generating a counterfactual, the **constraint violations** caused by the counterfactual for categorical and immutable features, **redundancy** of the proposed changes and the average **success rate** of the recourse process.

The **y-Nearest-Neighbours** metric is implemented in the *CARLA* framework using the k-Nearest-Neighbours algorithm. The metric is calculated using the following formula:

$$1 - \frac{1}{ny} \cdot \sum_{i \in C}^{n} \sum_{j \in kNN(x_i)}^{y} (p(x_i) - p(x_j)) \tag{4}$$

with n being the total number of counterfactuals C, y being the number of neighbours used in the kNN algorithm and $p(x_i)$ being the predicted label of the ith data point in the domain.

3.3 Datasets

We select a number of fixed datasets with both synthetic and real-world data to analyze the characteristics of the dynamics of shifts in a wide range of domains.

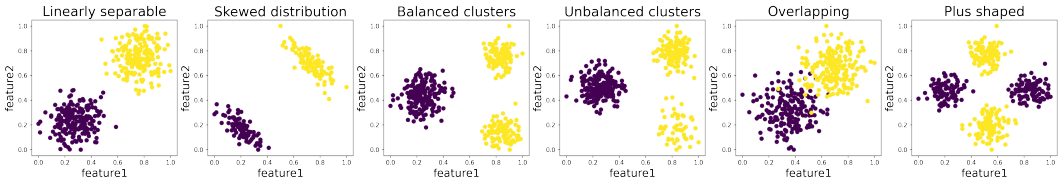

Figure 2: The synthetic datasets used in the experiments. From left: linearly separable, skewed distribution, balanced positive clusters, unbalanced positive clusters, overlapping, plus shaped.

Various synthetic datasets shown in Figure 2 have been generated. The datasets have been generated using settings stated in Appendix A. They range from simple, linearly separable single-cluster distributions, to non-linearly separable overlapping or multi-cluster datasets. With linearly separable datasets, we aim to capture and verify the expected behaviours of the recourse generators for simple shifts in the dynamics of the system. They also serve as means of verifying the correctness and debugging the implementations of the metrics used in the experiment. This type of datasets contains normally distributed data, skewed data, and data with two balanced and unbalanced positive clusters. The non-linearly separable sets contain a dataset with overlapping normally distributed clusters and a dataset with a plus-shaped arrangement of clusters. With these, we aim to test how the generators perform on datasets which require more complex models.

Although analyzing recourse on synthetic datasets can bring useful insights into the process, our main goal is to perform research on real-world data. Thus, real-world datasets have been used in the experiments. The CARLA framework contains 4 datasets loaded from an online repository: Adult Census, COMPAS Recidivism Racial Bias, Home Equity Line of Credit (HELOC) and Give Me Some Credit. As credit score assessment is one of the domains in which decision-making using machine learning models is widespread [2], we decided to use Give Me Some Credit and German Credit as the real-world datasets in our experiments. The information about preprocessing of these datasets is included in Appendix B.

Give Me Some Credit (GMSC) is a dataset published in a 2011 Kaggle prediction competition [21] and contains 250,000 samples of credit borrowers for which the goal is to predict the probability of them being in financial distress within the next two years. The dataset has an input dimension of 11 with features containing the pay, history, balance and loans of a person. Due to the high amount of samples in the original dataset, we generated a downsampled version of the dataset (n = 3K) to be used in the experiments to ensure reasonable execution times. The dataset also contains immutable

and categorical features, which for some recourse generators such as *CLUE* can be problematic due to lack of support for these kinds of features and can lead to constraint violations.

German Credit (GC) dataset has been published by the UCI Machine Learning Repository [22]. It contains 1,000 instances of individuals classified by their good or bad credit risk score. The input dimension of the dataset is 20 with 7 numerical and 13 categorical features. As this dataset contains even more features than GMSC, by using GC we aim to test whether this change in dimensionality will also require appropriate adjustments in *CLUE* hyperparameters. *Wachter* generators and is repeated 5 times.

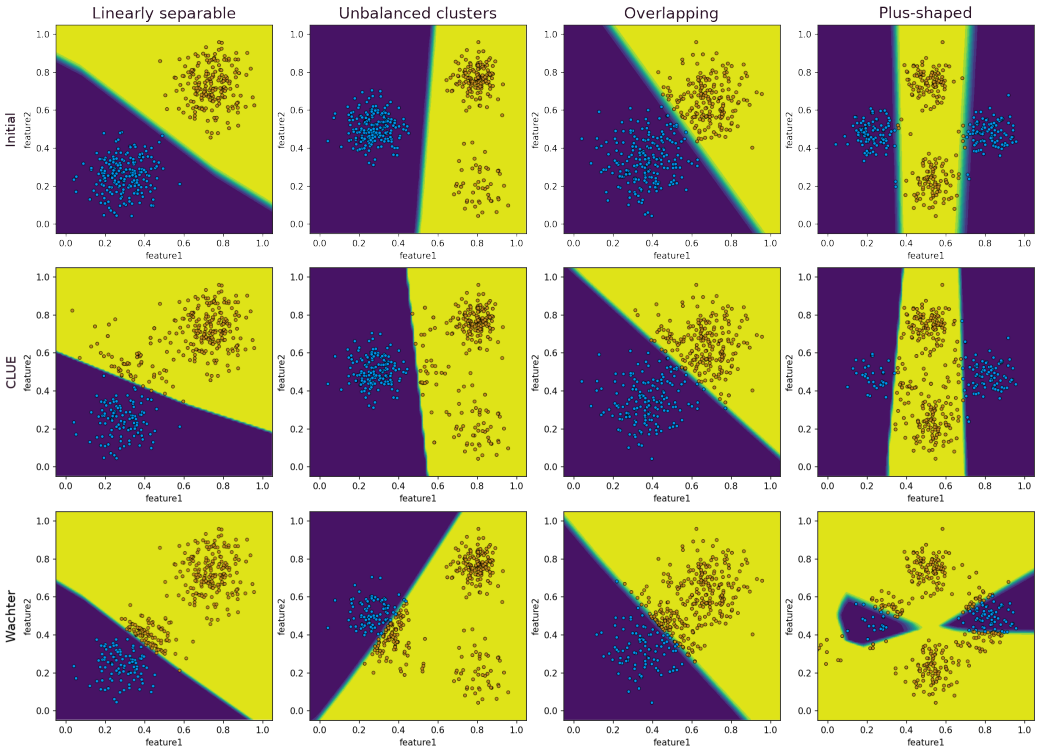

Figure 3: Effects of 20 rounds of recourse on 4 synthetic datasets; from left: linearly separable, unbalanced positive clusters, overlapping, plus shaped. Top row: Initial dataset, model. Recourse generators: middle: *CLUE*, bottom: *Wachter*. The background represents the decision boundary of the model. Blue represents negative class, yellow represents positive class for both model and datapoints.

4 Experiment

We propose a set of experiments testing how various characteristics of the domains, models and CE generators affect the dynamics of recourse. Each experiment (except for the experiment specific to *CLUE*) is ran on both *CLUE* and *Wachter* generators.

The experiments use a variety of classifiers to accommodate various systems that arise in the experiment settings. In total, we use 4 classifiers all implemented in the *CARLA* framework with PyTorch neural networks:

1. C1 - A neural network with one hidden layer - hidden size: $[5]$
2. C2 - A neural network with two hidden layers, for simpler datasets - hidden size: $[10, 10]$
3. C3 - A logistic regression classifier implemented as a neural network with no hidden layers
4. C4 - A neural network with two hidden layers, for complex datasets - hidden size: $[15, 10]$

4.1 Varying datasets

Experiment consisting of 8 runs in 6 synthetic and 2 real-world datasets. The first experiment we established tests how the data distribution affects the recourse generator performance for low-dimensional, synthetic, and complex, real-world datasets. Artificial neural networks with two hidden layers (C2, C4) were used to provide models that can perform well on both linearly separable and non-linearly separable domains.

The results of the first experiment shown in Table 1 show that the *CLUE* generator causes less severe shifts for both domain and the model than the baseline. This is consistent for all of the synthetic datasets. The differences in domain shifts measured in MMD are higher for simpler, linearly separable datasets (linearly separable, skewed, balanced and unbalanced positive clusters) than for the non-linearly separable ones. A possible explanation for this observation is the complexity of the decision boundary for those datasets. As the model gets more complex, higher dimensionality of *CLUE*'s VAE can be necessary. This can also cause the success rate for *CLUE* to drop for the more complicated domains such as unbalanced positive clusters since a single *CLUE* configuration has been used in this experiment.

Dataset	Linearly separable		Skewed distribution		Balanced clusters	
Method	CLUE	Wachter	CLUE	Wachter	CLUE	Wachter
MMD ↓	**0.08(0.01)**	0.12(0.02)	**0.11(0.01)**	0.16(0.00)	**0.05(0.01)**	0.11(0.01)
Disagreement ↓	**0.02(0.01)**	0.05(0.03)	**0.00(0.00)**	0.03(0.01)	**0.02(0.01)**	0.03(0.03)
Model MMD ↓	**0.04(0.03)**	0.05(0.03)	**0.05(0.02)**	0.08(0.00)	**0.02(0.01)**	**0.02(0.01)**
CE Pred. Prob. ↑	**0.90(0.14)**	0.58(0.05)	**0.86(0.16)**	0.57(0.06)	**0.90(0.13)**	0.59(0.08)
yNN ↑	**0.99(0.01)**	0.99(0.02)	**1.00(0.00)**	**1.00(0.00)**	**1.00(0.00)**	0.94(0.17)
Distance ↓	0.50(0.25)	**0.24(0.11)**	0.48(0.22)	**0.20(0.10)**	0.34(0.15)	**0.25(0.10)**
Succ. Rate ↑	1.00	1.00	1.00	1.00	0.78	1.00

Dataset	Unbalanced clusters		Overlapping		Plus shaped	
Method	CLUE	Wachter	CLUE	Wachter	CLUE	Wachter
MMD ↓	**0.04(0.01)**	0.14(0.00)	**0.04(0.01)**	0.07(0.02)	**0.02(0.01)**	0.03(0.01)
Disagreement ↓	**0.01(0.01)**	0.02(0.02)	**0.04(0.03)**	0.12(0.06)	**0.03(0.02)**	0.07(0.06)
Model MMD ↓	**0.01(0.01)**	0.02(0.01)	**0.01(0.01)**	0.02(0.01)	**0.06(0.01)**	0.09(0.03)
CE Pred. Prob. ↑	**0.91(0.14)**	0.62(0.08)	**0.81(0.17)**	0.55(0.03)	**0.88(0.15)**	0.58(0.06)
yNN ↑	**0.96(0.08)**	0.88(0.13)	**0.95(0.06)**	0.88(0.11)	**0.94(0.05)**	0.89(0.13)
Distance ↓	0.41(0.18)	**0.22(0.10)**	0.38(0.13)	**0.19(0.12)**	0.29(0.14)	**0.08(0.05)**
Succ. Rate ↑	0.47	1.00	0.65	1.00	0.86	1.00

Table 1: Varying datasets experiment results for the 6 synthetic datasets. Values show mean and standard deviation (in parentheses). The *CLUE* generator causes less shifts than *Wachter* for all datasets. The differences are more pronounced for simpler datasets.

Table 1 also shows that the differences among the generators for model shifts are lower than those for the domain. The disagreement metric clearly shows that in all cases the baseline generator causes more shifts that result in misclassifications by the model. The Model MMD shows the least difference across datasets except for the plus-shaped dataset, where a large difference is observed, which corresponds with the results in Figure 3, where *Wachter* drastically changes the decision boundary, while for *CLUE* the changes are minimal. These observations can be attributed to the nature of CEs generated by *CLUE* and *Wachter*. By generating CEs along the decision boundary, the latter of the generators accelerates the shift of the decision boundary towards the negative class. This behaviour can later result in new CEs being generated within the negative cluster, thus increasing the disagreement metric.

The *CLUE* generator consistently created CEs with high yNN values. This means that the explanations provided by the generator fall into the desired class more frequently than the ones provided by the baseline generator. This and the fact that the generator aims to produce CEs that have low uncertainty for the classifier results in higher CE predicted probabilities for all experiments. Another effect caused by the inherent characteristics of the two generators is the lower distance for CEs generated by *Wachter*. In its objective function *Wachter* aims to generate CEs with the lowest distance to its factuals, while *CLUE* does not optimize such an objective.

Dataset	Give Me Some Credit		German Credit	
Generator	CLUE	Wachter	CLUE	Wachter
MMD ↓	0.017 (0.001)	**0.006 (0.002)**	0.0007 (0.0002)	**0.0004 (0.0001)**
MMD p-value ↓	—	**0.003 (0.006)**	1.00 (0.00)	1.00 (0.00)
Disagreement ↓	0.22 (0.06)	**0.18 (0.02)**	0.19 (0.02)	**0.15 (0.04)**
Probability MMD ↓	**0.25 (0.07)**	0.27 (0.05)	0.14 (0.07)	**0.08 (0.05)**
CE Predicted Prob. ↑	**0.99 (0.04)**	0.87 (0.15)	**0.95 (0.10)**	0.52 (0.03)
Distance ↓	0.92 (0.27)	**0.03 (0.01)**	2.83 (0.39)	**0.27 (0.21)**
yNN ↑	**1.00 (0.00)**	0.96 (0.02)	**0.84 (0.08)**	0.61 (0.12)

Table 2: 30 rounds of recourse for 25 CEs ran on a reduced version of the Give Me Some Credit (n = 3K) dataset with equally populated classes and 10 rounds for 15 CEs on German Credit. Values show mean and standard deviation (in parentheses). Probability MMD was used due to model MMD limitations discussed in Section 5

The experiment ran on a large, dataset containing real-world data, Give Me Some Credit [21] and German Credit [22], result in *CLUE* causing more domain shifts than *Wachter*. As seen in Table 2, for both datasets *CLUE* causes a higher MMD and disagreement, and for German Credit, it causes a more noticeable shift in the model. While it causes more shifts, the generator is still able to generate CEs that fall better into the target class and have higher predicted probabilities.

4.2 Varying models

Experiment consists of 3 runs on 3 models. In this experiment, we test how the shifts differ when the recourse is being generated for models with varying parameters. Three types of models were tested in the experiments C3, a linear model, in this case, logistic regression implemented as a PyTorch neural network without hidden layers and a sigmoid activation function C1, an artificial neural network with one hidden layer and C2, an artificial neural network with two hidden layers.

Classifier	C1 - 1 layer ANN		C2 - 2 layer ANN		C3 - logistic	
Generator	CLUE	Wachter	CLUE	Wachter	CLUE	Wachter
MMD ↓	**0.04(0.01)**	0.06(0.00)	**0.04(0.01)**	0.05(0.02)	**0.05(0.00)**	0.08(0.00)
Disagreement ↓	**0.30(0.22)**	0.31(0.23)	**0.05(0.03)**	0.08(0.06)	**0.05(0.01)**	0.12(0.02)
Model MMD ↓	**0.04(0.01)**	0.11(0.04)	**0.02(0.02)**	**0.02(0.01)**	**0.12(0.01)**	0.14(0.01)
CE Pred. Prob. ↑	**0.64(0.13)**	0.51(0.04)	**0.89(0.12)**	0.54(0.02)	**0.55(0.05)**	0.50(0.00)
yNN ↑	**0.96(0.06)**	0.91(0.15)	**0.95(0.05)**	**0.95(0.04)**	**0.92(0.07)**	0.68(0.31)

Table 3: Mean and standard deviation (in parentheses) for MMD, disagreement, model MMD, CE predicted probability, yNN for 10 rounds of recourse for 10 CEs ran on the overlapping dataset. Three classifiers used.

When comparing the two generators for systems using different classifiers, the first observation that can be made using the data in Table 3 is that *CLUE* outperforms the baseline generator, similarly to the case in the previous experiment. Here, recourse with *CLUE* generates less domain and model shifts for all cases except for the 2-layer ANN, where its model MMD and yNN metrics are the same as for the baseline classifier.

The largest differences among the generators happen when using the logistic regression classifier - C3. There, *Wachter* causes noticeably more domain shift and a higher disagreement with relatively low yNN. A possible cause of those characteristics is the fact that the model could be highly susceptible to data shifts as well as not being able to fit well to the non-linearly separable data. As seen in Figure 4 recourse on a system using this model has also caused the most severe domain and model shifts.

The one-layer ANN classifier - C1 shows a high initial growth of model MMD which later flattens out. For the baseline classifier, the recourse using C1 leads to a noticeably higher model drift than for *CLUE*, which performs relatively well. Another observation worth noting in this experiment is the overall high disagreement metric results. This metric stays at an average value of around 0.3 throughout the whole duration of the experiment. This is not likely to stem from the recourse process. A possible explanation for the observation is that the classifier does not fit well into the domain and each retraining of the model causes it to change its predictions for a large part of the data points.

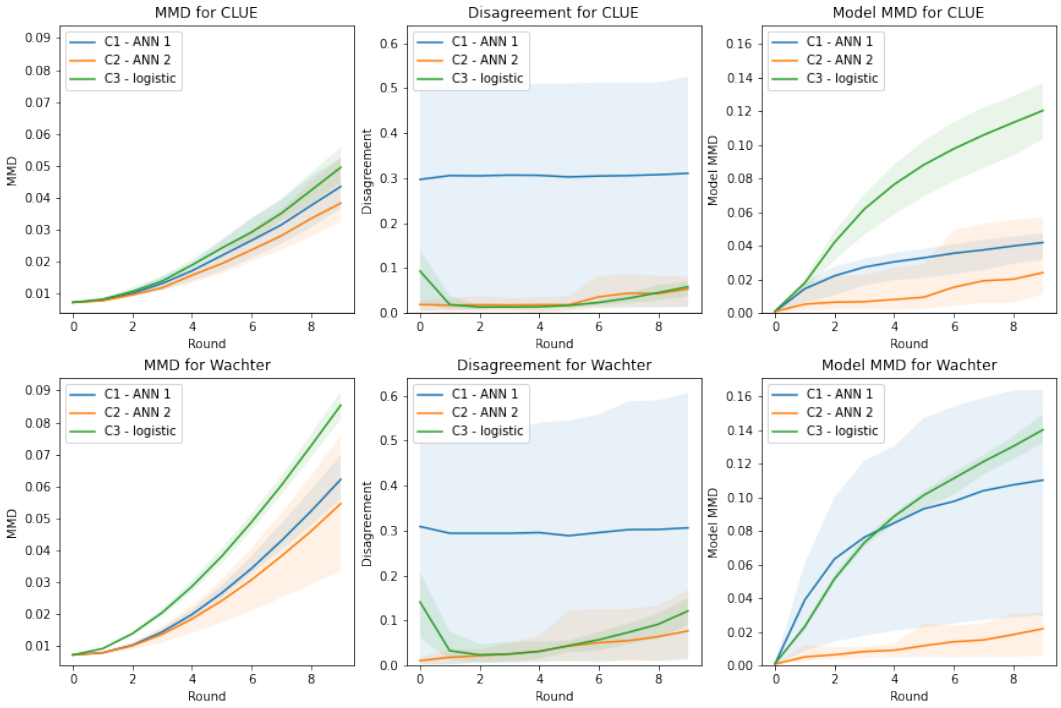

Figure 4: Varying models experiment for 3 models: linear regression, 2 hidden layer ann, 1 hidden layer ann. Top row: CLUE, bottom row: Wachter. Graph contains mean line and the value range (shaded).

One characteristic apparent in the recourse process using C2 and C3 classifiers in Figure 4 is a difference in the disagreement metric for the two generators. While for *Wachter* the disagreement metric starts growing immediately after the start of the process, for *CLUE* the value of the metric on average stays close to zero until the fifth round of recourse. This means that the *CLUE* generator manages to keep the models' original predictions for half of the duration of the process. The recourse using the two-layer ANN classifier - C2 shows the least shifts overall, both generators perform relatively well in this system setting.

4.3 Varying *CLUE* hyperparameters

Experiment consists of 17 runs with 6 VAE latent dimension, depth and width settings on 2 real-world datasets and with 5 VAE latent dimension, depth, width and epoch settings on a synthetic dataset. A separate experiment aimed to compare shifts induced by recourse employed by the *CLUE* generator was executed. The *CLUE* generator uses the VAE decoder as means of generating counterfactual explanations [6]. By increasing the dimensions of the VAE and its training parameters, we aim to test whether it allows *CLUE* to generate CEs that cause less domain shift and stay certain for the model.

CLUE par.	(10, 16, 6)	(16, 12, 4)	(16, 16, 6)	(16, 20, 4)	(16, 20, 8)	(20, 20, 8)
MMD ↓	0.13 (0.02)	0.11 (0.01)	**0.06 (0.02)**	0.03 (0.00)	0.10 (0.03)	0.01 (0.01)
Disagr. ↓	0.37 (0.12)	0.30 (0.14)	**0.21 (0.05)**	0.22 (0.03)	0.21 (0.06)	0.19 (0.04)
Mod. MMD ↓	3.07 (0.01)	3.09 (0.02)	3.08 (0.03)	3.09 (0.02)	3.08 (0.02)	3.08 (0.03)
Prob MMD ↓	0.50 (0.28)	0.44 (0.06)	**0.29 (0.08)**	0.32 (0.05)	0.43 (0.09)	0.06 (0.03)
yNN ↑	0.90 (0.19)	0.98 (0.03)	**0.99 (0.01)**	—	0.88 (0.10)	—
Succ. rate ↑	1.00	1.00	1.00	**0.53**	1.00	**0.03**

Table 4: 100 rounds of recourse generated by 6 configurations of *CLUE* generator on the downsampled Give Me Some Credit dataset. Values show mean and standard deviation (in parentheses). The values for model MMD have been scaled by a factor of 10^4. The *CLUE* parameters are displayed as (latent dimensions, width, depth) of the VAE.

The results of the experiment shown in Table 4 partially confirm the previous assumptions about the parameters of the VAE. As the amount of latent space dimensions grows, the MMD at the last round of recourse for *CLUE* decreases, but after a certain point, the generator is unable to generate enough CEs, possibly due to a mismatch between the VAE width and depth or too high latent space dimensionality. Due to this decrease in success rate, the results for the 4th and 6th configurations are not comparable to the rest of the results, as their domains and therefore models did not undergo the same magnitudes of shifts.

It is worth noting that the model MMD metric fails to capture any differences between the tested parameter configurations, while probability MMD produces results that roughly correspond to the domain shifts. The reason for that is the fact that the sample points of the metric are distributed uniformly throughout the space of the domain. While for synthetic, low-dimensional datasets, the metric gives relatively good results, it is limited by the number of samples it can take. This fact in conjunction with the existence of outliers in real-world data decreases the effectiveness of this metric. The probability MMD, on the other hand, by using existing data points as samples for the MMD, makes it less likely that the changes will not be captured and produces realistic results.

One notable *CLUE* configuration, 16 latent dimension, 20 width, 8 depth, has a success rate of 1, meaning that this set of hyperparameters does not impact the generator's ability to create CEs, although the results suggest a higher shift in dynamics of the system than the 16 latent dimensions, 16 width, 6 depth configuration. This result might stem from the mismatch between the width and the depth of the VAE.

CLUE par.	(10, 16, 6)	(16, 12, 4)	(16, 16, 6)	(16, 20, 4)	(16, 20, 8)	(20, 20, 8)
MMD * 10^2 ↓	0.11 (0.02)	0.13 (0.03)	0.12 (0.02)	0.10 (0.01)	**0.08 (0.01)**	0.11 (0.01)
Disagr. ↓	0.20 (0.01)	0.21 (0.04)	**0.18 (0.02)**	0.21 (0.03)	0.21 (0.03)	0.19 (0.01)
Prob MMD ↓	0.16 (0.05)	0.19 (0.11)	0.29 (0.08)	**0.15 (0.03)**	0.18 (0.06)	0.17 (0.04)
Distance ↓	2.85 (0.31)	2.62 (0.28)	3.26 (0.38)	2.81 (0.33)	2.75 (0.04)	2.74 (0.39)
yNN ↑	0.68 (0.09)	**0.96 (0.04)**	0.85 (0.11)	**0.96 (0.04)**	0.94 (0.04)	0.70 (0.16)
Succ. rate ↑	1.00	1.00	1.00	1.00	1.00	0.99

Table 5: 100 rounds of recourse generated by 6 configurations of *CLUE* generator on the downsampled Give Me Some Credit dataset. Values show mean and standard deviation (in parentheses). The values for MMD have been scaled by a factor of 10^2. The *CLUE* parameters are displayed as (latent dimensions, width, depth) of the VAE.

An experiment has been run using the same settings on the German Credit dataset. The results in Table 5 show that the obtained MMD values are relatively low, similarly to the experiment on the same dataset against *Wachter* in the varying datasets experiment. For this experiment, the lowest shifts in MMD and Probability MMD are observed for hyperparameter configurations with higher VAE width and depth. This corresponds with the prior assumptions about the experiment, that as the dimensionality of the domain rises, the *CLUE* generator will require a more complex VAE to be able to generate better CEs.

The probability MMD for the 3rd configuration, which caused the least shifts for the earlier experiment now obtained the highest value. Although its disagreement value is low, the results for this metric are similar for all configurations. Moreover, this configuration provides CEs that require the highest distance on average to be employed.

CLUE par.	(6, 10, 3, 2)	(12, 10, 3, 2)	(12, 10, 3, 5)	(12, 10, 8, 2)	(12, 10, 8, 5)
MMD ↓	0.01 (0.01)	0.10 (0.01)	**0.06 (0.01)**	0.08 (0.01)	0.07 (0.01)
Disagreement ↓	0.00 (0.00)	0.04 (0.01)	**0.01 (0.01)**	0.03 (0.01)	0.03 (0.01)
Model MMD ↓	0.11 (0.01)	0.12 (0.01)	**0.11 (0.01)**	0.13 (0.01)	0.13 (0.01)
CE Pred. Prob. ↑	0.64 (0.17)	0.64 (0.12)	0.65 (0.13)	0.73 (0.17)	**0.74 (0.17)**
yNN ↑	—	**0.97 (0.06)**	0.93 (0.08)	0.92 (0.18)	0.95 (0.08)
Succ. Rate ↑	0.10	0.98	0.90	1.00	1.00

Table 6: 10 rounds of recourse generated by 5 configurations of *CLUE* generator on the linearly separable dataset. Values show mean and standard deviation (in parentheses). The *CLUE* parameters are displayed as (latent dimensions, width, depth, number of epochs) of the VAE.

The final experiment has been conducted on the linearly separable dataset with the C3 logistic regression classifier. Here, we tested the recourse process for 5 configurations of *CLUE* hyperparameters. The 2nd configuration in Table 6, VAE width 10, depth 3, latent dimension 12 with 2 training epochs has been used in most of the previous experiments with synthetic datasets. In configuration 1, where the latent dimension of search is decreased to 6, the success rate of the generator drops to 0.1, meaning that the generator generates data points that are not counterfactual. By increasing the number of training epochs to 5 in the 3rd configuration, we see lower values for shift metrics, although the success rate of the generator decreases. In configurations 4 and 5, we notice an increase in the success rate to 1, meaning that the VAE is well fitted to the domain and can always generate a counterfactual data point. Those configurations produce CEs that have higher predicted probabilities, although they also generate slightly higher shifts.

5 Discussion

After performing and analyzing the experiments, our findings suggest that we have developed a set of metrics that can measure shifts within a system under recourse. Using our suite of metrics we can test the magnitude and the significance of the shifts induced by the recourse process. When it comes to classifier models the metrics allow us to measure the magnitude of decision boundary shifts and how their predictions change over time. Moreover, using *CARLA* we can also assess the generated CEs. With these metrics and the results of experiments in Section 4, we can answer the research question: ***"What are the characteristics of shifts induced by the CLUE generator?"***. To that end, we first discuss the subquestions introduced in Section 1.

Does the magnitude of observed shifts differ compared to the baseline generator?

Our experiments show that the *CLUE* generator performs better than the baseline *Wachter* generator in terms of both domain and model shifts on smaller, synthetic datasets. There, it generates CEs that better fit the underlying data distribution, and therefore it shifts the system less than the baseline generator. These experiments also showed that the CEs generated with *CLUE* fall better into the target class clusters. This causes *CLUE* to gain better results for the yNN and CE predicted probability metric and can be explained by the difference in the objective functions of the generators described in Section 2.

This situation changes when recourse is applied to a more complicated dataset with **real-world data**. For both the GMSC and GC real-world datasets, we experience higher domain shifts as well as higher results for the disagreement metric with the *CLUE* generator. The probability MMD resulted in a lower value for this generator, although the difference was not high. This somewhat surprising result shows that a simpler objective function of the *Wachter* generator may cause less severe shifts than *CLUE*. This could be caused by the characteristics of the dataset or by the VAE used by *CLUE* not being trained sufficiently.

The real-world data experiment has also revealed a limitation regarding one of the metrics used for quantifying model shifts. The model MMD metric has been omitted from the results in Table 2 due to it giving small (in the order of 10^{-4}) and very similar results for both generators. This was also noticed for the results of experiment 4.3, where the results had to be scaled up and since the experiment was run on the same dataset, it confirmed that the metric is not effective for high-dimensional, real-world datasets. This limitation is described further in the following subsection.

When the generators were tested on different classifiers, *CLUE* again outperformed *Wachter* in most of the metrics related to shifts. It has also become apparent that for recourse generated with *CLUE* the shifts might appear later than for the baseline generator.

What might be playing a role in the observed differences?

The main cause of the differences observed between the two generators is their objective functions. The objective function of the *Wachter* generator described in Section 2 includes a distance function which causes the generated CEs to be as close to the factuals as possible. Meanwhile, *CLUE* uses a VAE to generate CEs that are probable to exist in the original dataset by estimating the generative model of the domain. Moreover, it optimizes the CEs to have a low uncertainty for the model [2]. With these characteristics in mind, we can define how they affect the results of the experiment.

[2] *CLUE* normally expects the models to be Bayesian classifiers [6]

One of the simplest metrics in which the difference in generator design is apparent is the distance between a factual and a counterfactual. There, *Wachter* always gets the lowest mean value due to its prioritization of the low distance. On the other hand, values for metrics such as yNN and the CE predicted probabilities are always high for *CLUE*, indicating that it can generate counterfactuals that fall into the target class and can be more certain to the generator.

The differences between the magnitudes of induced shifts on synthetic and real-world datasets for both generators can be explained by the performance of the VAE. For small, low-dimensional, synthetic datasets we used, the VAE can be fitted with a low amount of epochs and relatively low dimension settings. On these types of datasets, the *CLUE* generator with a well-fitted VAE outperformed *Wachter* by inducing both domain and model shifts of lower magnitude. For real-world datasets with high amounts of features, the situation reverses and *CLUE* induces more severe shifts.

What appear to be good ways of mitigating endogenous shifts?

The last set of experiments we performed focuses on the way VAE's hyperparameter configurations affect the recourse provided by *CLUE*. Although not all optimal configurations were tested in the limited time of the project, our results show that they greatly affect the performance of the generator. If the configuration is mismatched with regards to the domain, e.g. the domain requires a larger VAE, the success rate of the generator might decrease greatly. On the other hand, when the configuration is matched well with the characteristics of the domain, we notice that the CEs are generated close to or within the target class and therefore cause less endogenous shifts. This can be seen in the results of the first experiment, where a well-fitted VAE enables *CLUE* to generate recourse which causes less severe shifts than the baseline generator.

Another factor that plays role in the magnitudes of induced shifts is the choice of classifier in the system. As seen in Subsection 4.2 the domain, as well as the model shifts induced in the process, vary depending on the type of classifiers used. When acting on a non-linearly separable domain, simpler models such as C1 - single hidden layer ANN or C3 - logistic regression classifiers cause more shifts than a more complex C2 - a two-layer ANN. One has to therefore take into account that a model well-fitting the domain is also needed to minimize the shifts.

5.1 Limitations

The limitations we encountered along the process of the experiment creation and execution regarding the execution times of the recourse and the measurements as well as issues specific for certain metrics. The counterfactual generation times play a rather small part in the overall runtime of the recourse process, taking from fractions of seconds up to a couple of seconds.

Execution times The processes that cause the most overhead on the execution time of an experiment iteration are training the machine learning models and training CLUE's VAE. Both have to be accommodated to the domain under recourse. The training times for models with lower complexities averaged around a second while the more complex ones took up to 30 seconds to complete training. On top of this, the models have to be trained multiple times - once per generator, and there is a risk of a model being mistrained and not being able to generate any factuals - this case requires a retraining of the model. The training of CLUE's VAE is by far the most time-consuming in the whole process, taking tens of seconds to train for simple models and up to a minute for complex ones.

With all these factors taken into account, the runtimes for full experiment suites can make them. For simple domains, 100 experiment iterations took 14 minutes. The experiments mostly require multiple combinations of parameters to be analyzed and they need to be repeated numerous times to obtain believable results. This causes the runtimes for full experiments to grow into an order of multiple hours, while for complex datasets even reaching tens of hours in runtime.

Lack of parallelization in *CARLA* Another limitation is caused by the implementation of the recourse benchmarking framework CARLA. The main limitation caused by the implementation of the framework and one which also affects the previous limitation is the lack of parallelization possibilities. With the current implementation, the multithreading capabilities of a CPU are not accessible and tests using the PyTorch with Cuda GPU mode enabled showed a major decrease in performance of the experiments. The latter could also be affected by the type of operations performed during the process of recourse, which could be incapable to be optimised by a GPU.

Metrics Numerous limitations regarding metrics have been found over the course of developing the framework and running the experiments. The metrics based on the **MMD** metric have also been sources of limitations. Firstly the calculation of the σ value for the MMD requires a lot of resources. The process of calculation of the median distances for all samples would in many cases exceed the memory capacity of the devices on which the experiments were run. For that reason, we decided to use a set value of σ for the MMD. Although this can affect the sensitivity of the metric, it still provided comparability for the results.

Secondly, the process of generating the p-value for MMD using bootstrapping requires numerous iterations to calculate the value. This causes the process to be very time-consuming and thus we limited the number of iterations to 1000. Ideally, more iterations could be used but that would further slow the experiment process down.

Finally, a limitation in the **model MMD** metric appeared during tests on real-world datasets. As mentioned in experiments in Sections 4.1 and 4.3, the metric performs well on synthetic datasets, but the high-dimensionality and complexity of a real-world dataset cause the metric to not pick up any dynamics. This is because the metric uses a fixed number of sample points. The number of sample points per dimension decreases rapidly as the number of dimensions grows, e.g. 3 samples for GMSC with 10 dimensions. To increase the sensitivity of the metric, an enormous amount of sample points would be required, making the execution times unfeasible.

6 Responsible research

6.1 Reproducible research

A crucial part of research aimed at extending the domain under analysis is the reproducibility of results of the described experiments. In order to fulfil the reproducibility of the results of our research, we share the source code of the framework that was developed and used in the course of the research. A GitHub repository [3] containing the code and the experiment data is made public. Apart from the framework, full documentation with instructions on setting up the experiments and running them in a correct environment, as well as several ready examples are provided. To ensure reproducibility of the obtained results, full metadata of the experiments described in our paper is described. This includes the number of recourse rounds, the number of counterfactuals per round, the classes of recourse generators, machine learning models and their parameters, and the datasets.

Moreover, to increase reproducibility, we include the randomness seeds in the experiment metadata. Although, at the time of writing this paper our framework supports both seeding the NumPy and PyTorch libraries, the experiments presented in this paper only contain the seed for the NumPy library. Seeding NumPy ensures that operations like sampling the data are deterministic and it increases reproducibility for metrics using MMD. In the current version, with processes like training models or VAEs seeded, the MMD results are fully reproducible. The metric results generated by *CARLA* cannot be reproduced as the framework uses different random seeds.

6.2 Ethical research

Although the problem of algorithmic recourse can influence an individual in either a positive or a negative way, as mentioned in Section 1, our research focuses on measuring and quantifying the factors that influence these effects. The counterfactual explanations found through algorithmic recourse generators aim to provide individuals with ways to improve their situation in a system with a machine learning model as means of decision making. By applying these explanations an individual can affect the whole domain as well as the underlying model and by doing so, influence other individuals involved in the system.

Our research aims to provide means of measuring the dynamics of the aforementioned shifts and experimentally test how different generators influence the different dynamics of the systems. We hope to provide useful information to researchers analyzing and developing recourse generators. Our work can also enable entities such as banks, who need to employ algorithmic recourse as one of their services, to choose a recourse method that would perform best given their system. In that case, the entities become stakeholders of the research, but they can merely gain knowledge about the

[3] https://github.com/drobiu/recourse_analysis

characteristics of a generator; they stay responsible for choosing those that provide characteristics valuable for them or their clients.

7 Conclusions and Future work

The solutions for evaluating CEs in the process of algorithmic recourse lack the ability to evaluate the dynamics of the underlying system. In this paper, we address this issue and contribute to the field by proposing a framework with a set of metrics aimed at capturing the dynamics of the domain and the model under recourse. We built our metrics upon *CARLA*, a framework for executing and evaluating recourse. We used MMD, a non-parametric measure of distance between two probability distributions as our main domain shift metric. We also proposed model MMD, a metric that estimates the decision boundary of the classifier using a meshgrid and computes the shift with the MMD.

Apart from methods for measuring the domain and model shifts, our framework contains metrics that measure the quality of generated recourse. Moreover, the framework can execute experiments, generate results and even visualizations of the process. The process of recourse generated by *CLUE* and *Wachter* generators was tested on a multitude of synthetic datasets modelling different possible domain scenarios, and two real-world datasets from the credit score assessment domain.

Using this framework we were able to run numerous experiments testing different aspects that might play a part in the characteristics and magnitudes of the shifts. The results of these experiments enabled us to capture the differences between the two generators, which we related to their designs. The differences seem to stem from the objective functions of the generators, which affect the way the CEs are generated. We also analyzed different aspects of the generators, their hyperparameters, or aspects of the system such as the type of classifiers that affect the shifts and the performance of recourse. With that, we were also able to formulate possible ways of mitigating the shifts regarding the previous findings.

Future work on the subject can take into account the limitations stated in Subsection 5.1. The main hurdle in properly analyzing the configurations of the VAE used by *CLUE* are the long runtimes and high amount of resources needed to train the encoder. Due to these limitations, we were not able to find the optimal hyperparameter configurations for *CLUE* for the real-world data in the time span of this project. The question of whether with optimised parameters *CLUE* can perform better than *Wachter* on these domains remains open.

The two metrics proposed by us, used for capturing model decision boundary shifts, model MMD and probability MMD both turned out to perform well, but they both have issues. As mentioned in Subsection 5.1, model MMD uses a meshgrid to estimate the decision boundary over the whole domain. This makes the metric extremely computationally expensive for high-dimensional domains. The probability MMD, on the other hand, uses just the predicted probabilities of a sample of data points in the dataset. This means that even if the model does not change, but points in the dataset get updated, its value will change. Future work can take into account these limitations and develop a metric that combines these two to create a robust but computationally viable model shift metric.

References

[1] S. Safdar, S. Zafar, N. Zafar, and N. F. Khan, "Machine learning based decision support systems (dss) for heart disease diagnosis: A review," *Artificial Intelligence Review*, vol. 50, no. 4, pp. 597–623, 2017. DOI: 10.1007/s10462-017-9552-8.

[2] J. Galindo and P. Tamayo, "Credit risk assessment using statistical and machine learning: Basic methodology and risk modeling applications," *Computational Economics*, vol. 15, no. 1/2, pp. 107–143, 2000. DOI: 10.1023/a:1008699112516.

[3] C. C. S. Liem *et al.*, "Psychology meets machine learning: Interdisciplinary perspectives on algorithmic job candidate screening," *The Springer Series on Challenges in Machine Learning*, pp. 197–253, 2018. DOI: 10.1007/978-3-319-98131-4_9.

[4] M. Pawelczyk, S. Bielawski, J. van den Heuvel, T. Richter, and G. Kasneci, "CARLA: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms," *CoRR*, vol. abs/2108.00783, 2021. arXiv: 2108.00783. [Online]. Available: https://arxiv.org/abs/2108.00783.

[5] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harvard journal of law & technology*, vol. 31, pp. 841–887, Apr. 2018. DOI: 10.2139/ssrn.3063289.

[6] J. Antoran, U. Bhatt, T. Adel, A. Weller, and J. M. Hernández-Lobato, "Getting a CLUE: A method for explaining uncertainty estimates," in *International Conference on Learning Representations*, 2021. DOI: 10.48550/ARXIV.2006.06848. [Online]. Available: https://arxiv.org/abs/2006.06848.

[7] S. Upadhyay, S. Joshi, and H. Lakkaraju, "Towards robust and reliable algorithmic recourse," *CoRR*, vol. abs/2102.13620, 2021. arXiv: 2102.13620. [Online]. Available: https://arxiv.org/abs/2102.13620.

[8] R. K. Mothilal, A. Sharma, and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, ACM, Jan. 2020. DOI: 10.1145/3351095.3372850. [Online]. Available: https://doi.org/10.1145%5C%2F3351095.3372850.

[9] S. Joshi, O. Koyejo, W. Vijitbenjaronk, B. Kim, and J. Ghosh, *Towards realistic individual recourse and actionable explanations in black-box decision making systems*, 2019. DOI: 10.48550/ARXIV.1907.09615. [Online]. Available: https://arxiv.org/abs/1907.09615.

[10] D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, 2013. DOI: 10.48550/ARXIV.1312.6114. [Online]. Available: https://arxiv.org/abs/1312.6114.

[11] G. Widmer and M. Kubat, "Learning in the Presence of Concept Drift and Hidden Contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996. DOI: 10.1023/a:1018046501280.

[12] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, *Dataset Shift in Machine Learning*, ser. Neural Information Processing series. MIT Press, 2008, ISBN: 9780262170055. [Online]. Available: https://books.google.nl/books?id=qJ0jEAAAQBAJ.

[13] M. Dudík, S. Phillips, and R. E. Schapire, "Correcting sample selection bias in maximum entropy density estimation," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds., vol. 18, MIT Press, 2005. [Online]. Available: https://proceedings.neurips.cc/paper/2005/file/a36b0dcd1e6384abc0e1867860ad3ee3-Paper.pdf.

[14] B. Zadrozny, "Learning and evaluating classifiers under sample selection bias," *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, vol. 2004, Sep. 2004. DOI: 10.1145/1015330.1015425.

[15] Y. Lin, Y. Lee, and G. Wahba, "Support Vector Machines for Classification in Nonstandard Situations," *Machine Learning*, vol. 46, no. 1/3, pp. 191–202, 2002. DOI: 10.1023/a:1012406528296.

[16] A. Gretton *et al.*, "Covariate shift by kernel mean matching," *Dataset Shift in Machine Learning, 131-160 (2009)*, Jan. 2009.

[17] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *J. Mach. Learn. Res.*, vol. 13, pp. 723–773, Mar. 2012, ISSN: 1532-4435.

[18] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schlkopf, and G. R. Lanckriet, "Hilbert space embeddings and metrics on probability measures," *Journal of Machine Learning Research*, vol. 11, Apr. 2010. [Online]. Available: http://www.jmlr.org/papers/volume11/sriperumbudur10a/sriperumbudur10a.pdf.

[19] S. Rabanser, S. Günnemann, and Z. C. Lipton, "Failing loudly: An empirical study of methods for detecting dataset shift," *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. DOI: 10.48550/ARXIV.1810.11953.

[20] M.-F. Baclan, "Lecture 20 & 21: Active learning," *10-806 Foundations of Machine Learning and Data Science*, Nov. 2015. [Online]. Available: http://www.cs.cmu.edu/~avrim/ML07/lect1116-18.pdf.

[21] Kaggle Competition, *Give me some credit. improve on the state of the art in credit scoring by predicting the probability that somebody will experience financial distress in the next two years*.

[22] D. Dua and C. Graff, *UCI machine learning repository*, 2017. [Online]. Available: http://archive.ics.uci.edu/ml.

A Synthetic datasets

We generated the synthetic datasets using the following settings:

Dataset	`a` means	`b` means	`a` num.	`b` num.	covariance	NumPy seed
Linearly separable	$\begin{pmatrix}-5\\-5\end{pmatrix}$	$\begin{pmatrix}10\\10\end{pmatrix}$	200	200	$\begin{pmatrix}9&0\\0&9\end{pmatrix}$	9
Skewed distribution	$\begin{pmatrix}-5\\-5\end{pmatrix}$	$\begin{pmatrix}8\\8\end{pmatrix}$	100	100	$\begin{pmatrix}5&-4\\-4&5\end{pmatrix}$	1
Balanced positive clusters	$\begin{pmatrix}-7.5\\0\end{pmatrix}$	$\begin{pmatrix}5\\-6\end{pmatrix}$ $\begin{pmatrix}5\\6\end{pmatrix}$	200	100 100	$\begin{pmatrix}3&0\\0&3\end{pmatrix}$ $\begin{pmatrix}4&0\\0&4\end{pmatrix}$	5
Unbalanced positive clusters	$\begin{pmatrix}-7.5\\0\end{pmatrix}$	$\begin{pmatrix}5\\-6\end{pmatrix}$ $\begin{pmatrix}5\\6\end{pmatrix}$	200	50 150	$\begin{pmatrix}3&0\\0&3\end{pmatrix}$ $\begin{pmatrix}4&0\\0&4\end{pmatrix}$	5
Overlapping	$\begin{pmatrix}-5\\-5\end{pmatrix}$	$\begin{pmatrix}5\\5\end{pmatrix}$	200	200	$\begin{pmatrix}12&0\\0&12\end{pmatrix}$	2
Unbalanced positive clusters	$\begin{pmatrix}-6\\0\end{pmatrix}$ $\begin{pmatrix}6\\0\end{pmatrix}$	$\begin{pmatrix}0\\-6\end{pmatrix}$ $\begin{pmatrix}0\\6\end{pmatrix}$	100 100	100 100	$\begin{pmatrix}2&0\\0&2\end{pmatrix}$ $\begin{pmatrix}2&0\\0&2\end{pmatrix}$	3

Source code used for generating the datasets using these settings: https://github.com/abuszydlik/model-shifts-with-dice/blob/b1a52eec1097c24fe74e7f8f9fc7a67cb7044e27/notebooks/synthetic_datasets.ipynb

B Real-world dataset preprocessing

The preprocessing of the Give Me Some Credit dataset consisted of downsampling the dataset to a size of $n = 3K$ and providing equal target class population ($n = 1.5K$ each).

Source code: https://github.com/drobiu/recourse_analysis/blob/f3233c57f1c374f04cfaf7f43d8d7ec85a3c93b4/notebooks/Dataset_subsampling.ipynb

The preprocessing steps for the German Credit dataset consisted of encoding the categorical features as binary features for frequently appearing values and aggregating some of the features.

Source code: https://github.com/abuszydlik/model-shifts-with-dice/blob/4b651d971700664a558746f9bf14c001fe81140b/notebooks/GC_processing.ipynb