The endogenous dynamics induced by Algorithmic Recourse

Giovan Angela
Supervisor(s): Cynthia Liem, Patrick Altmeyer
EEMCS, Delft University of Technology, The Netherlands

June 19, 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

Machine learning classifiers have become a household tool for banks, companies, and government institutes for automated decision-making. In order to help explain why a person was classified a certain way, a solution was proposed that could generate these counterfactual explanations. Several generators have been introduced and tested but include several side effects. One of these side effects is making it easier to be classified incorrectly after sufficient recourse has been applied. Dynamics, a.k.a. shifts in both the domain and model, cause these side effects. We aimed to quantify these dynamics induced by two generators, Wachter *et al.* and REVISE, and compare them against each other. We performed three experiments with both generators and looked at the effect a different dataset, model, or hyper-parameter may have had on the dynamics. We found that REVISE induces a slight model shift while the domain shifts increase with each round of recourse.

1 Introduction

In recent years we have become more dependent on Machine Learning algorithms to decide if a person can get a loan, has a high risk of performing a crime, or to determine their risk in health by healthcare systems [1]. The benefits that Machine Learning brings to various fields may also come with some risks [2]. The black-box nature of traditional ML models for human classification caused a push toward the explainability of a model. The 'why' in the decision-making of a model is of great importance, specifically when people's lives are directly affected by those decisions.

A counterfactual explanation (CFE) was introduced to give insight into '*how the world would have to be different for a desirable outcome to occur*' [3, p. 6]. However, what CFEs provide in understanding what set of features results in the desired outcome, they lack in recommending the steps to take to realize this outcome. Algorithmic recourse (AR) builds on CFEs by adding a cost function '*to shift the focus from explaining a decision to providing recommendable actions to achieve recourse*' [4, p. 2]. Venkatasubramanian and Alfano define AR as '*the systematic process of reversing unfavorable decisions by algorithms and bureaucracies across a range of counterfactual scenarios*' [5, p. 284]. Steps have been made with algorithmic recourse, but this was not without introducing a new problem: model and domain shifts caused by the algorithmic recourse [6].

These dynamics, shifts occurring in the model and domain, may influence how the classification model works. It may become more accessible for people to be classified positively, or worse, people who initially got rejected may eventually get accepted without having to perform any of the actions suggested by the recourse method. These possible effects are why it is essential to quantify the dynamics a recourse method may induce and figure out what controls these dynamics and if we can mitigate them.

As of writing, no published research about the effects of the dynamics introduced by repeated algorithmic recourse and how to measure it seems to exist. With this paper, we aim to introduce metrics that can be used to compare the Wachter *et al.* [3] and REVISE [1] recourse methods and answer the following question: *What are the characteristics of the dynamics induced by REVISE?* To help answer this question, we have strived to answer if (1) the magnitude of induced dynamics differ compared to the baseline generator? (2) If so, what factors might be playing a role here? (3) What appear to be good ways to mitigate endogenous shifts? (4) How can we quantify the dynamics?

2 Related work

For this research, we looked at two AR generators, Wachter *et al.* and REVISE. Both share the same goal of finding counterfactuals for the given factuals, but their reasoning for *how* to find them differs. Wachter *et al.* use an optimization approach that, for a factual x, tries to find '*counterfactuals $\hat{x}$ which are as close as possible to x*' [7, p. 6]. REVISE, on the other hand, uses a variational autoencoder (VAE) to estimate the generative model. With this model, REVISE suggests counterfactuals '*that are likely to occur under the data distribution*' [7, p. 6].

Both Wachter *et al.* and REVISE use a gradient based-model and algorithm and are able to handle binary categorical features. Wachter *et al.* assumes inputs are pairwise statistically independent, while

REVISE assumes that the input is generated by a generative model [7]. Another difference between the two generators is the ability to handle immutables. Wachter *et al.* cannot handle immutable features, but REVISE can only handle immutable features if the features are binary.

Little to no work has been done on the dynamics induced by these recourse generators. A first mention of the dynamics in counterfactual explanation (CE), a.k.a. recourse, is found in a paper by S. Verma *et al.* [8], which lists it as one of CE's challenges that should be taken into account. Dynamics are a challenge because the assumption that an underlying model does not change over time may not be valid in all applications where recourse is applied. However, they did not provide a solution to prevent these dynamics from occurring.

3 Methodology

3.1 Characteristics

To help find a way to quantify the dynamics induced by applying recourse, we performed a couple of experiments using various synthetic datasets in which we have measured multiple characteristics outlined in this subsection.

3.1.1 Accuracy / F1-score

In order to provide meaningful data, the models trained should be accurate enough, or the experiments will fail. The F1-score is used to see how an induced shift may influence the accuracy of a model. The F1-score is calculated using the test set.

3.1.2 Mean

For each feature and class, the mean is calculated. The mean may indicate that a shift in the domain has occurred. However, the amount of plots required to visualize this for a dataset with n features is not feasible ($n \cdot 2$ plots).

3.1.3 Number of clusters

Ideally, there should be two clusters at the start of the experiment, and at the end of each experiment, multiple clusters may have developed due to the dynamics induced by the recourse method. K-means clustering is used to find the centers of each cluster for a range of values for k. The number of clusters may indicate a movement in the dataset. A Python library called kneed [9] was used to find an optimal amount of clusters within the range of k.

3.1.4 Maximum Mean Discrepancy (MMD)

The MMD is a multivariate two-sample test proposed by Gretton *et al.* [10] to determine if two samples originate from different distributions. In our case, we use it to measure the distance between our original model and dataset and the modified model and dataset.

An unbiased estimator for the squared MMD is given by:

$$MMD^2(X,Y) = E[\kappa(X,X)] + E[\kappa(Y,Y)] - 2E[\kappa(X,Y)]$$

Because we use a linear kernel, for its simplicity in implementation and usage, the calculation we do in our measurement is done by:

$$MMD^2(X,Y) = E[XX^T] + E[YY^T] - 2E[XY^T]$$

The MMD is used for three calculations: The MMD Domain, MMD Model, and MMD Probabilities.

MMD Domain - Calculates the MMD between the new and original dataset, where A is the new dataset and B the original:

$$MMD^2(A,B) = E[AA^T] + E[BB^T] - 2E[AB^T]$$

MMD Model - Calculates the MMD between the probability values of a mesh grid on the new and original model. First, we generate a mesh grid of 100^2 for our two feature datasets. Then we convert this grid to a list of points and input this list into the new and original model. The results of both models, a list of probabilities for each point on the mesh grid, are then compared using the MMD. In this calculation, U contains the results of the new model and V that of the original model:

$$MMD^2(U,V) = E[UU^T] + E[VV^T] - 2E[UV^T]$$

MMD Probabilities - Calculates the MMD between the probability values of the new dataset on the new and original model. Each point of the new dataset is used as input in the new and original model. The results of both models, a list of probabilities for each point in the new dataset, are then compared using the MMD. Here P is a set that contains the results of the new model, and set Q contains the results of the original model:

$$MMD^2(P,Q) = E[PP^T] + E[QQ^T] - 2E[PQ^T]$$

3.1.5 Disagreement coefficient

The disagreement coefficient measures how far off the updated model has drifted from the original regarding different classifications. It is calculated as follows:

First, we input the current dataset in the original and new models. We then compare the given classifications for each point. When both models disagree, a counter is increased. Afterward, this value is normalized by dividing it by the size of the dataset.

3.1.6 Boundary width

Boundary width is a measure introduced by our colleague, A. Buszydlik, to quantify the width of the decision boundary by summing the squared centered probabilities for each class and dividing it by the size of the set. This width is calculated by: $(\sum_{x \in X} (p(x) - 0.5)^2)/n$, where X is the set of points containing all points classified as either negative or positive, and n is the number of points in this set.

3.1.7 Mean probability of counterfactuals

To help indicate the quality of a recourse generator and its effect on itself when reapplying, we calculate the mean probability of the counterfactuals found per round. The closer this value is to 1, the higher the 'quality' of the generated counterfactuals.

3.2 Datasets

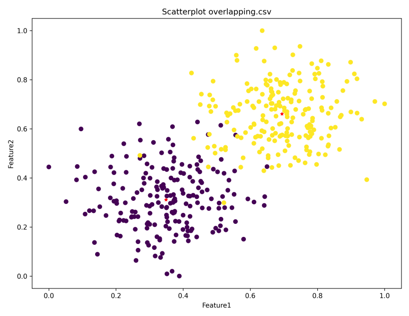
Four static synthetic datasets were selected for our experiments, as shown in Figure 1. They were chosen due to variance, mean, overlap, and overall shape differences. Each dataset has two feature columns, with a 50/50 distribution of points in the positive and negative classes. We use various datasets to see their influence on a recourse generator by structure, spread, and density. Using these datasets will give the results more credibility and allow us to get an idea of how a generator would perform on any arbitrary two-feature dataset and maybe an n-feature dataset. The parameters used to generate the datasets can be found in table 1, the code, and the sets can be found in our GitHub repository[1].

One dataset that is not listed in the table is the moons dataset. This set was created using the `make_moons` method from the SciKit Learn Python library [11]. The following two parameters were used for creating the moons dataset: `n_samples` = 200 and `noise` = 0.15.

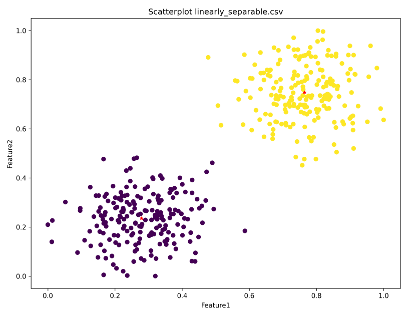
[1]URL to the repository: https://github.com/deoxys/CSE3000-REVISE-dynamics-quantifier/datasets
URL to the Jupyter Notebooks used to generate the datasets: https://github.com/abuszydlik/model-shifts-with-dice

Dataset name	Clusters	Negative class			Positive class		
		n	Mean(s)	Variance(s)	n	Mean(s)	Variance(s)
overlapping	2	200	(-5, -5)	12	200	(5, 5)	12
linearly separable	2	200	(-5, -5)	9	200	(10, 10)	9
balanced positive clusters	3	200	(7.5, 0)	3	200	(5, -6), (5, 6)	4, 2

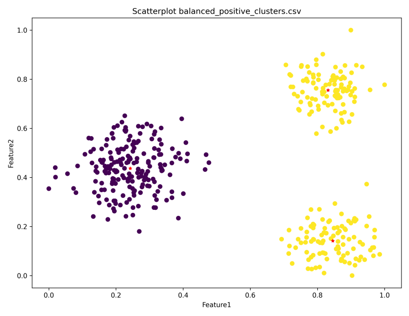
Table 1: Parameters used to generate the datasets, which can be found in our GitHub repository



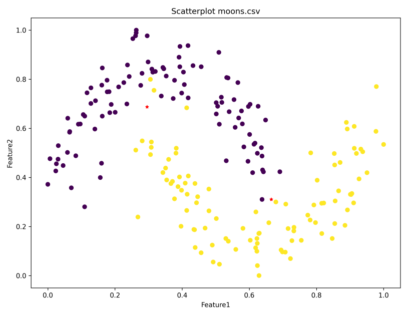
(a) overlapping, $n = 400$



(b) linearly separable, $n = 400$



(c) balanced positive clusters, $n = 400$



(d) moons, $n = 200$

Figure 1: Scatterplots of the used datasets where yellow is the positive class, purple the negative class, and the red star indicates the center of each cluster

The last dataset originates from a Kaggle competition [12] named Give Me Some Credit (GMC) that contains real-life historical data on $150\,000+$ borrowers and ten feature columns. We sub-sampled this set into an $n = 3\,000$ dataset with a 50/50 positive-negative ratio.

3.3 Experiments

3.3.1 General setup

Each experiment type will be repeated five times to increase the results' reliability. Due to the limited time for the project and the extended run-time of each experiment, only five repetitions were done per experiment.

Each of these experiments follows roughly the same steps. We first start with a grid search algorithm to find the optimal parameters for the model on the given dataset. Then we train the classification model on the selected dataset with the optimal parameters. After training, the state of the model and dataset is measured.

When the initial first steps, grid search, training, and measuring, are complete, we repeat the following steps until 50% of the samples in the negative class have been changed. The negatively predicted factuals are retrieved and used as input for the counterfactual generator. For each factual, a counterfactual is produced by the generator, and a list of counterfactuals is returned. From this list, five counterfactuals are randomly selected to be used to update their respective factual. After updating the dataset with the counterfactuals and the model is retrained. Finally, the state of this updated dataset and model is measured.

In the end, for each experiment, we end up with a measurement of the start and after each round.

3.3.2 Difference between the models

Description	Label	Hidden layers	Neurons per hidden layer
Logistic Regression	LR	None	None
NN with one hidden layer	NN1	1	Layer 1: 4 neurons
NN with two hidden layers	NN2	2	Layer 1: 8 neurons Layer 2: 4 neurons

Table 2: Three configurations used for the classification models

Three different models are used in this experiment: one Logistic Regression (LR) model and two different Neural Network (NN) models. The configuration for the three models is shown in table 2. All models were used with the best parameters found using a grid-search operation before training and running the experiment. Each generator used the same model at the start. For this experiment, we have used the overlapping dataset 1a.

3.3.3 Different datasets with the same model

This experiment aimed to find how well each generator would perform with differently shaped datasets.

We have used the following three datasets: linearly separable 1b, balanced positive clusters 1c and moons 1d. Each dataset was processed with the same type of model: NN1. Before each model was trained, a grid-search operation was performed to find the optimal parameters for the model on the given dataset. Each generator starts with the same model variant.

The original plan was to use another dataset instead of moons, which had the same size (n=400) as the other datasets. The problem, however, was that we could not complete at least one iteration of this experiment with this dataset. The cause may lie within the shape of the set and the hidden layer setup of the NN model we used.

3.3.4 Changing the hyperparameters

To see if it can influence the dynamics induced by the generators, we test the effect of some of the hyperparameters on their respective generator.

Wachter *et al.* has several hyperparameters which can be used to change the behavior of the recourse. Three of those parameters are `lr`, `lambda_param`, and `n_iter`. The `lr` controls the learning rate of the gradient descent that Wachter *et al.* uses. The `lambda_param` is used as a weight factor for the `feature_cost`. Lastly, we have the `n_iter` parameter, which indicates the maximum of iterations for optimizing the counterfactuals.

For REVISE, the usage of the hyperparameters may differ. The VAE is controlled by several of these hyperparameters, but we focused on the `lambda`, `lr`, and `max_iter` parameters. The `lambda` is used to decide how similar the counterfactual has to be to the factual. The `lr` controls the learning rate of REVISE. The value for `max_iter`, similar to the `n_iter` of Wachter *et al.*, handles the maximum number of iterations used to optimize the counterfactuals.

As shown in table 3, the values for the learning rate and maximum number of iterations may not influence the result of this experiment. However, the lambda parameter for Wachter *et al.* increases the feature cost, which may provide counterfactuals requiring fewer actions. For REVISE, increasing this value may produce counterfactuals with a smaller distance to their facts.

	Learning rate		Lambda parameter		Maximum number of iterations	
Wachter *et al.*	0.01	0.05	0.01	0.05	1000	500
REVISE	0.1	0.05	0.5	0.6	1000	500

Table 3: The default and new values for each of the modified hyperparameters

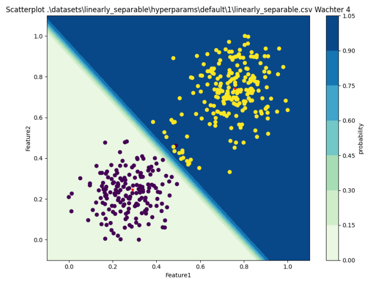
3.3.5 Performance of generators on real-life data

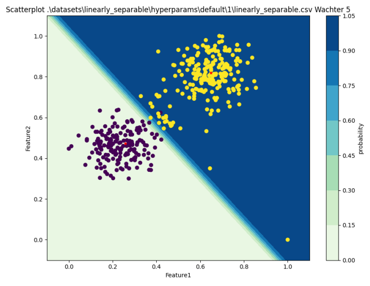
The experiment aimed to see if the generators perform equivalently in real-life as on synthetic datasets. We used a sub-sampled set of the GMC dataset for this experiment. For the classification, we trained a NN with two hidden layers. The first layer had ten neurons and the second five neurons. We found that this configuration for the model performed the best in terms of accuracy by at least 75%.

Due to the enormous number of calculations required for this metric, we were unable to calculate the MMD of the model. We had, therefore, disabled the MMD model metric for this dataset.

4 Results

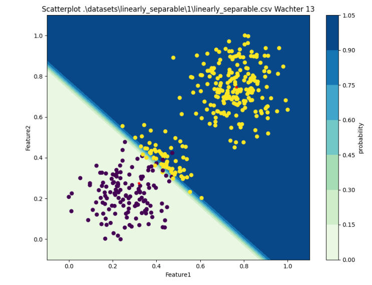
Each experiment in this section was performed five times to get the average scores visualized in the figures. There have been two instances, shown in Figure 2, during the experiments where the baseline generator, Wachter *et al.*, did not perform as expected due to something we can only describe as an error. This error may be due to how the Wachter *et al.* generator is implemented in the CARLA library, but further research must be done to ensure this is the case.

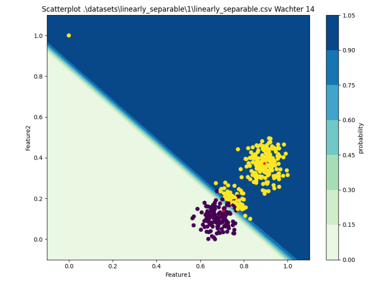




(a) Different datasets experiment with Wachter *et al.* at round 4

(b) Different datasets experiment with Wachter *et al.* error at round 5





(c) Default hyperparameters experiment with Wachter *et al.* at round 13

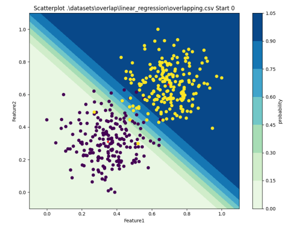
(d) Default hyperparameters experiment with Wachter *et al.* error at round 14

Figure 2: Scatterplots of the errors that occurred during an iteration of two experiments

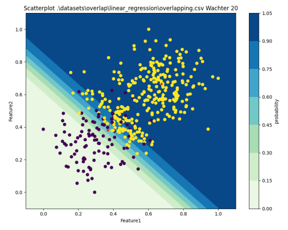
Figures 2a and 2b show that Wachter *et al.* moved one or two points from the negative class to the lower right corner on the positive side. This error can be seen more clearly in 2c and 2d, where Wachter *et al.* is expected to move the datapoint over the decision boundary between the negative and positive clusters.

The two iterations containing this error were discarded and run again to give a valid representation of the performance for each generator.

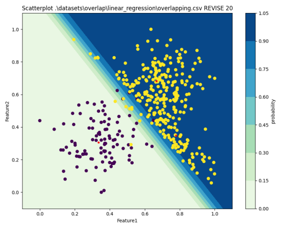
4.1 Difference between the models



(a) Start



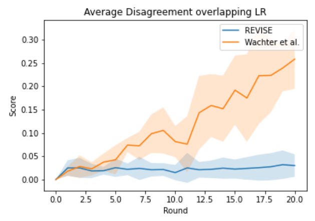
(b) Wachter *et al.* after 20 rounds



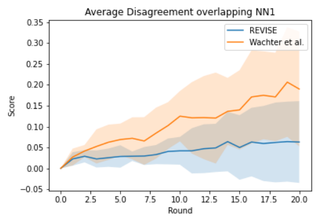
(c) REVISE after 20 rounds

Figure 3: A scatterplot of one of the iterations of this experiment for the LR model visualizing the dataset, model, and their shifts. Purple indicates the negative class and yellow the positive class. Similar behavior is seen in the scatterplots for both NN's

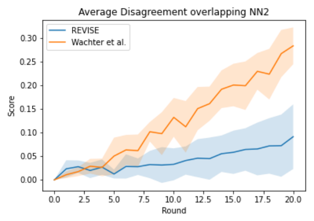
Figures 4, 7, and 8 show some of the experiment results for the three types of models, LR, NN1, and NN2, that we used. Each started with the same dataset, `overlapping.csv` 1a, after which the recourse generator started its work. Figure 3 indicates how Wachter *et al.* and REVISE may influence the given datasets and models.



(a) Disagreement LR



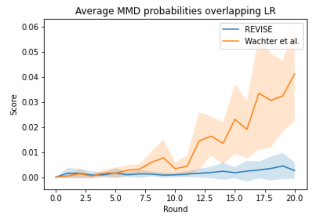
(b) Disagreement NN1



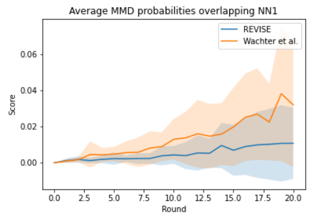
(c) Disagreement NN2

Figure 4: The plots for the average disagreement score of LR, NN1, and NN2

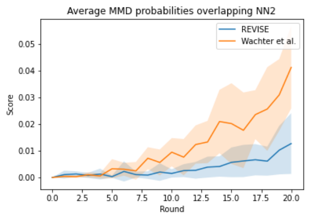
In Figure 4, we can see that retrained model of Wachter *et al.* seems to disagree more often with the original model than REVISE. This disagreement can be seen in Figure 3, where Wachter *et al.* (3b) has shifted the decision boundary of the model closer to the negative class while positioning the generated counterfactuals close to the negative class. This set of points, mainly classified as positive by the current model, is more likely to be classified as negative by the original model. This classification, in turn, increases the disagreement between the two models over time. REVISE, however, tightened the decision boundary and rotated it slightly. The tightening of the decision boundary does not affect the classification of the counterfactuals with the original model that much, compared to Wachter *et al.*, and will therefore result in a lower disagreement score.



(a) MMD probabilities LR

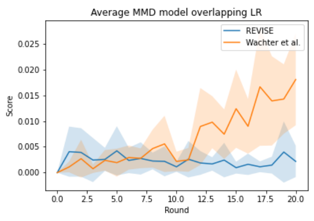


(b) MMD probabilities NN1

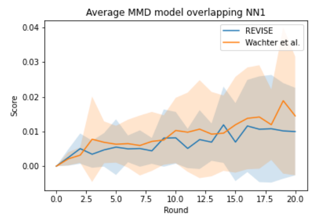


(c) MMD probabilities NN2

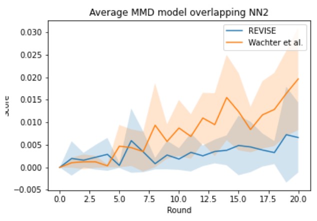
Figure 5: The plots for the MMD of the average probabilities mesh grid of LR, NN1, and NN2



(a) MMD model LR



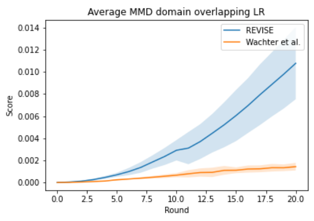
(b) MMD model NN1



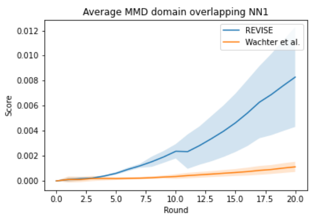
(c) MMD model NN2s

Figure 6: The plots for the average MMD of the model probabilities of LR, NN1, and NN2

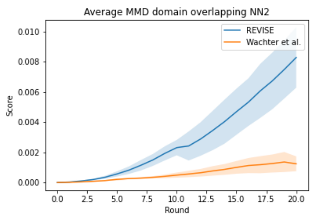
Now let us look at the MMDs of the model (3.1.4) and the probabilities (3.1.4), shown in 6 and 5. When we compare the MMD probability plots with the MMD model plots, we can see that they look very similar and calculate the same score. The MMD probability uses the probabilities of a mesh grid to calculate the MMD, whereas the MMD model uses the probabilities of the dataset. We can assume that the MMD of the probabilities better represents the shifts in a model.



(a) MMD domain LR



(b) MMD domain NN1



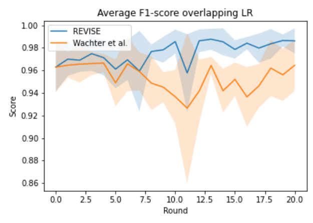
(c) MMD domain NN2

Figure 7: The plots for the MMD of the domain of LR, NN1, and NN2

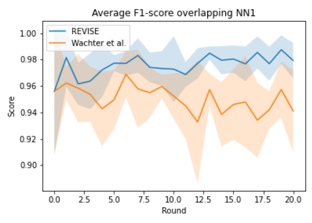
When we look at the MMD of the domain 7 for each model, we see that Wachter *et al.* causes a minuscule shift in the domain compared to REVISE. Despite the model used, REVISE seems to grow further and further away from the original domain after every round. This shift can be explained by looking at the difference between what REVISE and Wachter *et al.* end up in Figure 3.

REVISE (3c) positions the counterfactuals close to the positive cluster and spreads them near the decision boundary.

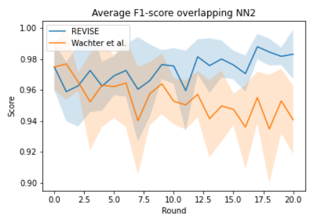
Wachter *et al.* (3b) places the counterfactuals on the decision boundary between the positive and negative clusters due to how Wachter *et al.* operates. The boundary moves closer and closer to the negative class due to this placement until the last round. This new positive cluster "merges" with the negative cluster and causes the domain shift, as calculated by the MMD domain, to be minimal compared to REVISE.



(a) F1-Score LR



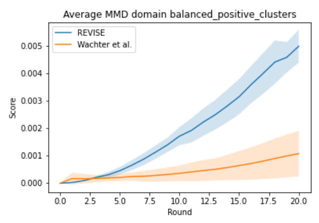
(b) F1-Score NN1



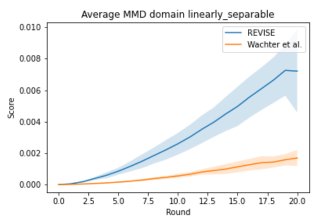
(c) F1-Score NN2

Figure 8: The plots for the average F1-score of LR, NN1, and NN2

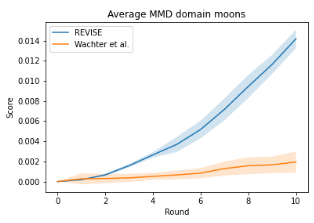
The average F1-score 8 for this experiment is in favor of REVISE, which seems to indicate that REVISE makes it 'easier' for the ML model to train itself on the updated dataset than Wachter *et al.*



(a) balanced positive clusters



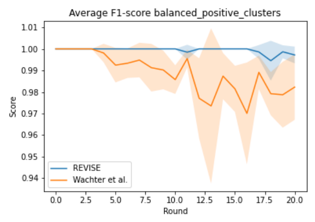
(b) linearly separable



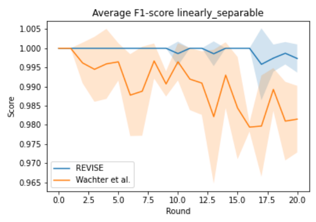
(c) moons

Figure 11: The average MMD domain score after five iterations of this experiment on each dataset

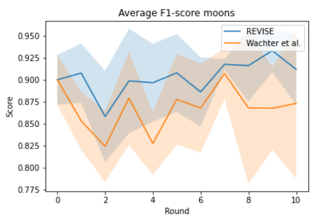
4.2 Different datasets with the same model



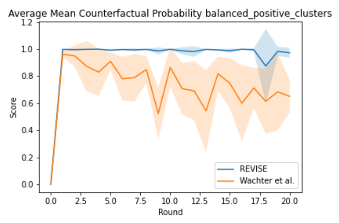
(a) balanced positive clusters



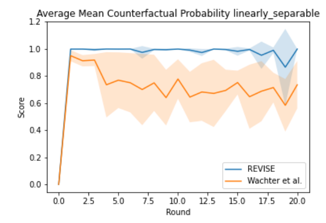
(b) linearly separable



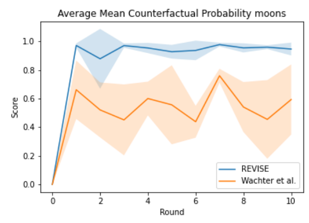
(c) moons



(d) balanced positive clusters



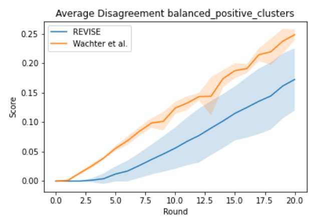
(e) linearly separable



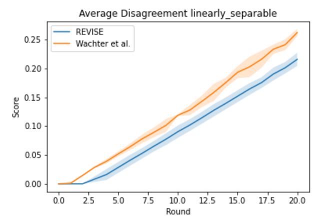
(f) moons

Figure 9: The average F1-score and average of the mean counterfactual probability after five iterations of this experiment on each dataset

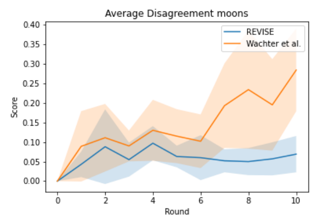
We can see, in Figure 9 that REVISE remains accurate and generates counterfactuals that are more likely to be classified positively compared to Wachter *et al.* We expected to see this because of how Wachter *et al.* finds its counterfactuals in contrast to REVISE's.



(a) balanced positive clusters



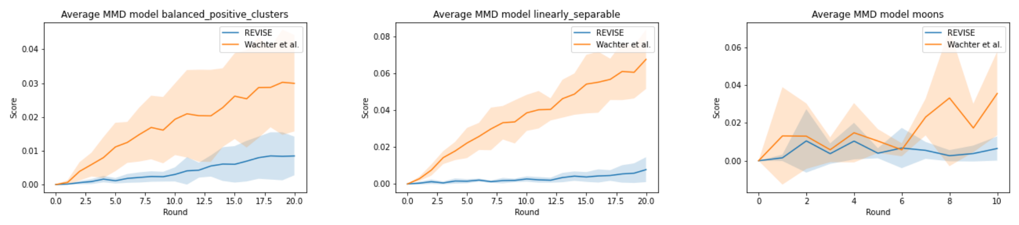
(b) linearly separable



(c) moons

Figure 10: The average disagreement score after five iterations of this experiment on each dataset

The disagreement scores shown in 10 are not entirely what we expected when we compare this Figure with the results of the previous experiment 4 and the accuracy scores 9. This discrepancy may be due to different datasets in both experiments. We expected REVISE to perform better than Wachter *et al.* by scoring lower on the disagreement metric.

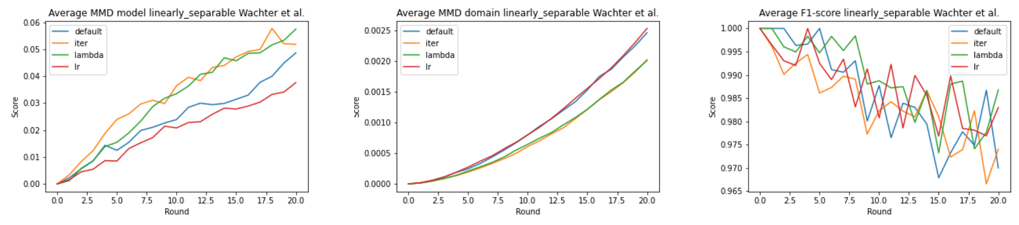


(a) balanced positive clusters (b) linearly separable (c) moons

Figure 12: The average MMD model score after five iterations of this experiment on each dataset

The same shifts we saw in the domain and model, of our previous experiment, in figures 7 and 6 can be seen in figures 11 and 12. This similarity is in line with our expectations since a different dataset should not affect the influence of a generator on the dataset and model. Both generators still suggest the same type of counterfactuals we expect.

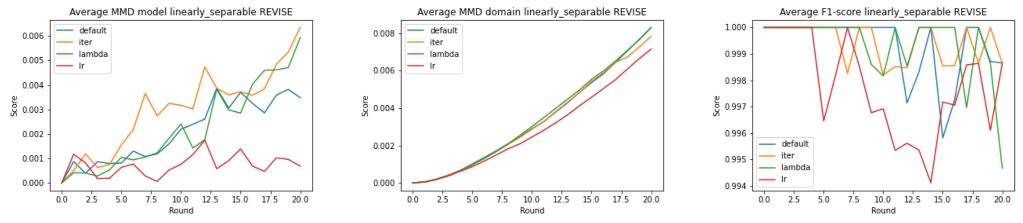
4.3 Changing the hyperparameters



(a) Average MMD model scores (b) Average MMD domain scores (c) Average F1-scores

Figure 13: The effect of changing hyperparameter values compared to the default value on the MMD model, domain, and F1-score for Wachter *et al.*

For Wachter *et al.*, increasing the `lr` value from its default (0.01) to 0.05 seems to decrease the shift in the model compared to the default values in Figure 13a while staying close to the default shift in the domain as seen in Figure 13b. However, changing the `iter` and `lambda` value for *Wachter et al.* seems to increase the shift in the model and decrease the shift in the domain. This is may be caused by the higher feature cost that the `lambda` parameter controls, forcing the counterfactuals to recommend less changes. Changing the values does not seem to affect the F1-score much compared to the default values, but more iterations are required to see if this is true.

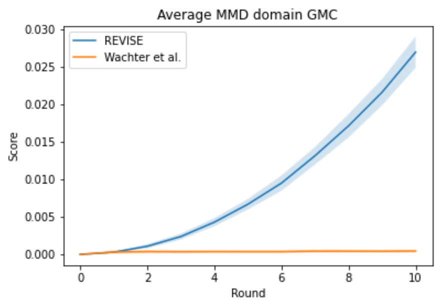


(a) Average MMD model scores (b) Average MMD domain scores (c) Average F1-scores

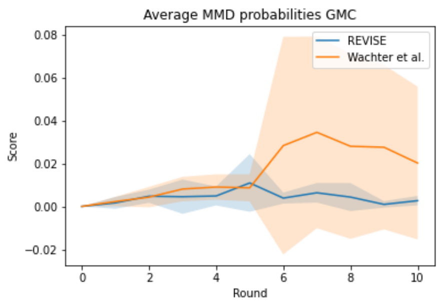
Figure 14: The effect of changing hyperparameter values compared to the default value on the MMD model, domain, and F1-score for REVISE

Changing the `lr` hyperparameter seems to have the same effect on decreasing the shift in the model 14a for REVISE as it does for Wachter *et al.*, but it also seems to decrease the shift in the domain 14b slightly. Decreasing the learning rate for REVISE seemed to positively mitigate the shifts while having minimal influence on the F1-score 14c.

4.4 Performance of generators on real-life data



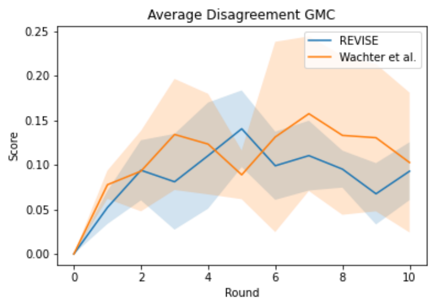
(a) Average MMD model scores



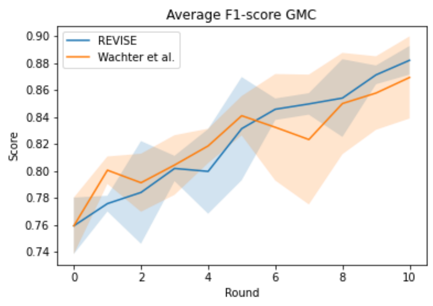
(b) Average MMD domain scores

Figure 15: The average MMD of the domain and probabilities of Wachter *et al.* and REVISE on the sub-sampled GMC dataset

In Figure 15a we see the same domain shifts as in 7 and 11 which is as expected. The models' shifts, shown in Figure 15b, look almost the same for the first five rounds of REVISE and Wachter *et al.* However, after the first five rounds, the deviation of the average result for Wachter *et al.* seems to increase, while the deviation for REVISE seems the same.



(a) Average disagreement scores



(b) Average F1 scores

Figure 16: The average disagreement and F1-score of Wachter *et al.* and REVISE on the sub-sampled GMC dataset

Both REVISE and Wachter *et al.* seem to follow the same average disagreement and F1 scores in Figure 16.After the first round, Wachter *et al.* shows a higher deviation in Figure 16a and round six in Figure 16b compared to REVISE.

Overall the performance of REVISE and Wachter *et al.* on a real-life dataset looks similar. However, REVISE performs better when we consider the deviation and model shift.

5 Responsible Research

In this research, we have looked at the dynamics induced by the REVISE recourse generator and how well it performs against Wachter *et al.*, the baseline generator. For this purpose, we have used synthetic datasets generated randomly using the parameters provided in 1.

The real-life dataset used in our last experiment is a publicly available set containing historical data from thousands of borrowers. No personal data, e.g., address, nationality, gender, or phone number, is saved in this set; therefore, it cannot be used to harm.

This research aims to provide the institutes that use recourse some information about how far their classification model has shifted from its original state. Secondly, it shows the dangers of using recourse without considering the dynamics beforehand.

During the experiments, an iteration returned an erroneous result in two instances. We have mentioned this in section 4 and explained why we think this was incorrect and how we proceeded to correct the result.

To ensure the reproducibility of this research, we have used synthetic datasets that, in combination with the code, can be accessed freely through our GitHub repository[2]. The code can be used on any machine running Docker instances, except for machines running on ARM chipsets. This limitation is due to an old library used by the CARLA framework that this work builds on. Furthermore, the experimental setup has been provided with the information required to reproduce the results.

6 Discussion

We found from the experiments that the model's MMDs and probabilities are similar, as seen in 6 and 5. The MMD of the model seems an accurate way to quantify the dynamics in a model. It seems accurate due to its usage of a mesh grid. One enormous drawback of this measure is that, for an n feature dataset, the calculations required increase by 100^n. Decreasing the 'resolution' of this mesh grid would result in an inaccurate portrayal of the shifts in the model. For this reason, we think that the MMD of the probabilities may be a better characteristic in real-life datasets than the MMD of the model. However, using the MMD probabilities will result in a less accurate representation of the shift in the model. The difference between the MMD of the model and the probabilities seems to be in the accuracy in the portrayal of the model shift and the run-time complexity.

Another exciting measure we found to be of use is the MMD of the domain to represent the shift in the domain. This metric is where Wachter *et al.* performs best in minimizing this shift. Sadly, the low score in domain shift does not result in a lower model shift. A reason for this can be seen in Figure 3, where the positive cluster slowly merges with the negative cluster. This 'pushes' the decision boundary closer and closer to the negative class compared to what REVISE produces.

During the third experiment, where we changed some hyper-parameters, we found that tweaking the learning rate for both generators could decrease the dynamics both produce. While we did not test various values for the learning rate, we do think that choosing the correct value for each dataset could help mitigate some of the dynamics.

7 Conclusions and Future Work

From the results, we can conclude that the characteristics of the dynamics induced by REVISE seem to be an increasingly higher domain shift while maintaining a low model shift. Compared to Wachter *et al.*, REVISE performs better with model shifts but worse with domain shifts. This performance could be explained by the difference in implementation of both generators, but REVISE may still outperform Wachter *et al.* if the optimal hyperparameters are used. This assumption does require more research into the various parameters and VAE of REVISE.

The dynamics were measured by calculating the MMD on the domain, model, and probabilities. It allowed us to quantify the dynamics induced by both generators and compare how well they performed in several experiments. While the MMD of the model provides an accurate display of the shifts in the model, the drawback is too enormous for it to be of use in real-life datasets. Since the MMD for the model and probabilities seem almost equivalent, the MMD of the probabilities could be an excellent substitute for use with n-feature datasets.

One question we are still unable to answer is if there exists a decent way to mitigate the endogenous shifts induced by both generators. A quick experiment with changing the hyperparameters indicates that there may be some improvement when increasing the learning rate of Wachter *et al.* compared to its default values. More testing with multiple values and datasets is required to see if this indication holds any merit.

[2]https://github.com/deoxys/CSE3000-REVISE-dynamics-quantifier

7.1 Shortcomings & suggestions

Due to time constraints, not every question has an entirely satisfactory answer. For that purpose, we give the following suggestion in case any further research in this field is done.

One improvement in this research would be to use more extensive or real-life datasets with more feature columns and see if the same conclusions can be made. This suggestion is because we used datasets of size $n = 400$ with only two feature columns. These datasets are trivial to plot and visualize, but when more dimensions are added, the visualization becomes unfeasible, and the required recourse becomes complex. Even though these 'simple' datasets can indicate what may occur when using larger datasets, one can only be sure when it is tested extensively with big or real-life data.

Another suggestion is to see if generating a different amount of counterfactuals has any effect on the outcome of each generator. During the experiments, we generated counterfactuals for each factual in the dataset. From this generated set of counterfactuals, five are randomly sampled. What if we generated counterfactuals for a subset of factuals. Does this influence the dynamics that the recourse generators may induce?

Through all the experiments we conducted, the counterfactuals generated have always been selected based upon a fixed probability value inside the CARLA library. If a found counterfactual exceeds the 0.5 probability threshold, it is selected and used. What could be interesting is to see how this threshold influences the dynamics induced by the recourse generators.

While we changed the hyperparameters from their default values, the VAE used in REVISE remained unchanged. We think optimizing the VAE hyperparameters may decrease the domain shift of REVISE seen in each experiment. We also recommend using several configurations for the generative model to see which performs the best in various datasets.

Our last suggestion would be to look at the predicted negative factuals and see if there is a quantitative difference in using the predicted negative factuals over the actual factuals. Something that may have occurred during this research is that one factual with a positive class in the dataset but is predicted to be in the negative class may go through for recourse multiple times. While this should be unnecessary in a real-life situation, it should be considered when looking at the results of this research.

References

[1] 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. [Online]. Available: https://arxiv.org/abs/1907.09615

[2] S. M. Julia Angwin, Jeff Larson and L. Kirchner, "Machine bias," May 2016. [Online]. Available: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

[3] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harvard Journal of Law & Technology, 2018*, 11 2017. [Online]. Available: https://arxiv.org/pdf/1711.00399.pdf

[4] A. Karimi, B. Schölkopf, and I. Valera, "Algorithmic recourse: from counterfactual explanations to interventions," *CoRR*, vol. abs/2002.06278, 2020. [Online]. Available: https://arxiv.org/abs/2002.06278

[5] S. Venkatasubramanian and M. Alfano, "The philosophical basis of algorithmic recourse," *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020.

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

[7] M. Pawelczyk, S. Bielawski, J. v. d. Heuvel, T. Richter, and G. Kasneci, "Carla: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms," 2021. [Online]. Available: https://arxiv.org/abs/2108.00783

[8] S. Verma, J. Dickerson, and K. Hines, "Counterfactual explanations for machine learning: A review," 2020. [Online]. Available: https://arxiv.org/abs/2010.10596

[9] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a "kneedle" in a haystack: Detecting knee points in system behavior," in *2011 31st International Conference on Distributed Computing Systems Workshops*, 2011, pp. 166–171.

[10] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, “A kernel two-sample test,” *Journal of Machine Learning Research*, vol. 13, no. 25, pp. 723–773, 2012. [Online]. Available: http://jmlr.org/papers/v13/gretton12a.html

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] K. 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,” https://www.kaggle.com/competitions/GiveMeSomeCredit/overview, accessed: 2022-05-29.