Enhancing Counterfactual Explanations

Anushka Kulkarni[1] **Jurjen de Haan**[1]

[1]Student of EEMCS, University of Twente, Enschede, The Netherlands.
{a.h.kulkarni,j.dehaan-3}@student.utwente.nl

Abstract

1 Introduction

As Machine Learning (ML) models have achieved higher predictive performance, their size, complexity and opacity have also increased, preventing the assessment, understanding and potential correction of these models [5]. To address this issue, Explainable AI (XAI) aims to provide transparent models by various techniques while maintaining predictive performance [4]. Next to global model understanding, local explanations aim to offer insight and understanding of individual model predictions [4].

Counterfactual Explanations (CFs) are a local explanation method that, for a given instance, return an alternative instance that indicates a minimal change to one or multiple of the feature values in order to obtain a different output by the model [6]. CF generation generally includes optimizing a loss function with the instance of interest, counterfactual instance and desired model outcome as inputs [6]. The obtained CF instance helps users understand 'what-if' scenario's [6]. As a loan example, if user X would earn an additional 500 euro's, the model decision would change from loan rejected to the desired outcome of loan accepted. Note that this does not imply causality for loan approval data: it is merely the model behavior that is explained rather than real-world causality for loan approvals [6]. CFs are contrastive as they explain an prediction to occur relative to another that did not [2], and are selective as they usually focus on a small number of feature changes [6].

Beyond understanding model decisions for an input, end users should also be able to act to reach the provided counterfactual instance, referred to as the feasibility of CFs [2]. Similarly, one can consider counterfactual explanations as algorithmic recourse [2]. For algorithmic recourse, the focus is on which actions the person is *recommended* to take in order to achieve the desired model outcome [7; 8], whereas counterfactual explanations are limited to only providing a profile (instance) that lead to that outcome [2]. Specifically for algorithmic recourse, actionability of counterfactual instances is of importance, which refers to variables that either the user cannot change, cannot directly control or can change. [2; 9]. For example, variables that the user cannot change are age and gender. Thus, a given CF instance that includes changes in these variables is not actionable. An example of a variable that the user cannot directly control is health status.

Finally, CFs suffer from the 'Rashomon effect', meaning that CF generation produces multiple valid CF instances which potentially contradict eachother, and do not match the popular human preference of simplicity over real-world complexity [6]. On the other hand, diversity can be a desired property of generated CFs [3]. User feedback may play a crucial role in regulating diversity by posing feedback that aims to return feasible CFs only [3].

But since user feedback depends on domain experts and suffers from subjectivity, we propose using ontologies

1.1 Motivation and Research Questions

Despite many works in counterfactual explanations within XAI literature, most do not generate CFs that offer feasible actions that users can perform [2; 11]. Feasibility related problems of CFs, among others, include lack of clear mapping between actions and proposed feature value changes when domain knowledge is absent, and model assumptions such as stable data distribution or lacking monotonicity constraints, which may not reflect real-world effects of feature value changes [11]. Furthermore, as mentioned earlier, CFs suffer from the 'Rashomon effect' which may conflict with the human preference over simple explanations.

We envision that user-feedback constrained CFs may limit the number of generated CFs to those that satisfy user preferences, while still allowing diversity of CFs. Furthermore, user feedback ensures only CFs are returned that are actionable according to the user's domain knowledge, clarifying the mapping between feature value changes and user actions, enhancing feasibility of CFs.

As a limitation, we envision that user feedback constrained CFs requires user input which may not always be available, either due to user availability or lacking resources to collect and incorporate user feedback. Furthermore, as user-feedback is subjective, this may lead to setting unrealistic preferences that are not actionable to anyone. Posing ontological rules automates the CF generation process while also preventing humans from placing impractical constrains when used in combination with user-feedback. The small body of work that combines XAI with ontologies makes it worth investigating how ontologies as an alternative method can help providing feasible counterfactual explanations.

Based on this motivation, this research aims to answer the following research questions:

- RQ: How can the feasibility of counterfactual explanations be enhanced based on ontological context?
- Sub-RQ1: How can integration of ontological contexts help with selecting the most appropriate counterfactual explanations?
- Sub-RQ2: Can we improve the likelihood of the datapoints proposed by the counterfactual explanations compared to the existing method

2 Related Work

There exists little work that combines ontologies with Explainable AI (XAI). In [1], the authors describe the potential of XAI for resolving data quality problems. More specifically, they describe the role of ontologies in describing a context of the data that counterfactual explanations should adhere to. With this, CF generation can be guided by ontologies in the form of semantic constraints to produce counterfactual explanations that are semantically correct. Thus, this would lead to more realistic counterfactual explanations, but not per se feasible ones. Nevertheless, this paper motivated our use of ontologies in this research, as we argue that feasible CFs must in any case be realistic to start with. Moreover, the authors argue that this is an area that deserves much more investigation.

To counteract feasibility of CFs, we identified two broad directions of solutions: solutions based on causality and solutions based on user-feedback. [2] demonstrates an implementation for generating CFs based on a pre-defined causal graph with exogenous (unobserved) and endogeneous (observed) variables. In the causal graph, the value of an endogenous variable can be either informed by only an exogenous variable or a combination of exogeneous and another endogeneous variable. The derived structural equations allow the classifier to be considered in latent space, as the model predicts the target variable based on observed variables that are nested in the structural equations from the unobserved variables. The counterfactual instances are generated based on perturbing the unobserved variables, incorporating causality relationships into the generation of CFs while also providing causal-aware recommendations for algorithmic recourse, and are then transformed to feature space via the structural equations, so they can be delivered to the user. By making use of the latent space, the authors found that their approach could indirectly alter features that were set as non-actionable (non changeable) by taking into account causal influences. Furthermore, the authors found that CFs produced by their approach had lower cost (distance to reach action) compared to a baseline, again due to using causal influences. Most importantly, the approach by the authors provided more realistic and reachable counterfactuals, opposed to the baseline.

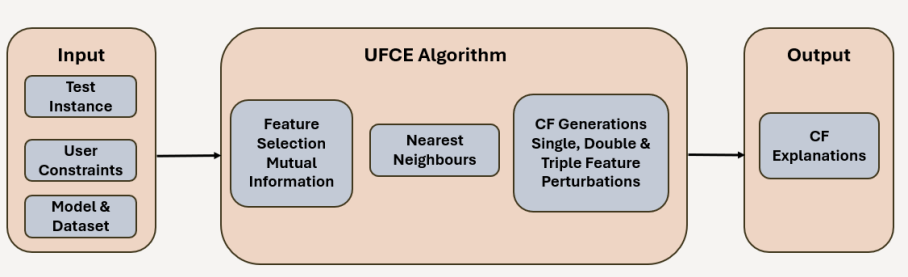

Figure 1: Counterfactual explanation generation pipeline of UFCE - existing algorithm

On the other hand, in [3], the authors' approach to feasible CFs revolves around incorporating actionability constraints based on user-feedback in the generation of CFs. Our work is based on this approach and the original algorithm is further discussed in section 3. The authors found that their approach outperforms two well-known counterfactual explanation methods regarding proximity, sparsity, and feasibility and that user-constraints influence the generation of feasible CFs.

3 Overview of the existing algorithm: UFCE pipeline

As discussed in Section 2, our approach is primarily based on the UFCE algorithm presented in [3]. The main purpose of the UFCE is to facilitate the generation of feasible and actionable counterfactual explanations by using domain expert/user feedback. These user inputs act as constraints that the feature values must adhere to, thereby ensuring a limited acceptable perturbation to the data. An overview of the pipeline is shown in Figure 1.

3.1 Flow of the process

To understand the process, we divide the whole pipeline into 3 parts (with corresponding subsections). Input, main algorithm, and the output.

3.1.1 Input

- Test instance x
- Trained Machine Learning (ML) model f
- **User Feedback** in the form of constraints: These constraints control which features can be changed and how much they are allowed to change. Example 3.1 shows user constraint for one possible use case.

Example 3.1. Assume $A1$ and $A2$ are 2 numerical features and let $A3$ be a categorical feature in a dataset D. User feedback for these features as per the UFCE algorithm would be a map with keys as the feature/attribute names and the corresponding values as the minimum and maximum range values for the numerical features. For categorical features, the algorithm assumes boolean values and hence simply flips the values without any constraint.

$user\_constraint = \{A1 : (10, 20), A2 : (1, 1.5)\}$

3.1.2 Main Algorithm

- Feature selection using **Mutual Information** Calculation (MI): In this, feature dependencies are computed using MI values. Followed by selecting tuples (pairs/triples) of features with high MI for joint perturbation.
- **Nearest Neighbors**: This step is essential to ensure that we stay in close proximity (within a radius) of the test instance. This radius is called the **desired space**. A KD-tree is used to efficiently find instances similar to x in that desired class.
- **Creating a subspace**: Subspace is a bounded feature space which is created using user-specified feature constraints as shown in example 3.1. This ensures that the generated counterfactuals are within actionable and plausible limits.

- **Perturbation Methods**: [3] uses three perturbation techniques:
  - Single Feature Perturbation (UFCE-1): Only one user defined feature is changed by performing a binary search over the specified interval/
  - Double Feature Perturbation (UFCE-2): In this, two dependent features can be changed simultaneously wherein the second feature is predicted using a regressor on the first feature.
  - Triple Feature Perturbation (UFCE-3): In this technique, three features are changed using the MI ranking on the user defined constraints.

3.1.3 Output

The output of the algorithm are possible counterfactual explanations. Out of a given set of counterfactuals, the best ones are selected by using evaluation metrics like **feasibility**, **proximity**, and **sparsity** which are defined in section 7.

3.2 Critical Analysis of the UFCE Algorithm

By imposing user-defined constraints, UFCE limits the number of features that can be modified to generate a favorable outcome. Furthermore, these constraints set a limit on the amount of perturbation permitted. This limitation on the number of counterfactual explanations generated is particularly useful in real-life applications.

Main benefits of the existing algorithm are as follows:

- **User-Centered**: Incorporates user feedback in the algorithm. Apart from the restricted perturbations, another advantage is for business specific applications, users have independence in decided what changes can be allowed.
- **Model-Agnostic**: Works very well with any black box classifier.
- **Data-Aware**: Considers data distribution - nearest neighbors and mutual information.
- **Multiple strategies for perturbations**: Single, double and triple perturbations can be performed.
- **Feasibility Focused**: Ensures that the counterfactuals generated are feasible, plausible, and realistic.

Despite the advantages, the current algorithm is not full proof and sufficient. On critically analyzing the algorithm, follow are areas of concern:

- **User Input not always available**: While in an ideal scenario, domain experts would be available and have user feedback would be an advantage. But this cannot always be guaranteed.
- **Cost overheads**: Having domain experts needs time, money and other resource investment which is not always possible.
- **User feedback is subjective**: Since defined constraints are subjective, they would differ from user to user, making universal generalization impossible. For example, if we consider the scenario of loan application to a bank, we would want this to be as uniform as possible across multiple branches. If user feedback is received from a bank employee, it would then depend on the employee to define the constraints.
- **MI is not used in single perturbations**: In the current UFCE algorithm, MI is used only to make double or triple perturbations. But, we realized that MI could also be useful in making single perturbations. In case of single perturbations, MI can be calculated against the target variable to understand if the feature is important.
- **Binary assumption for categorical features**: Constraints are defined only for numerical features. This is because the algorithm works only with binary categorical feature values. So a perturbation to a categorical feature means a simple flip to the feature value.

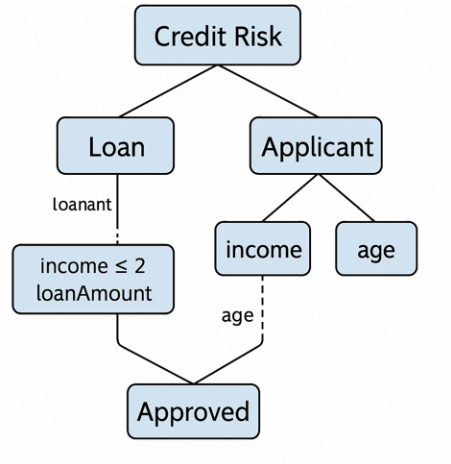

Figure 2: Example of an ontology with entities like credit risk, loan and applicant. Applicant has attributes like income and age. We also define the relationship between attributes income and loan amount.

3.3 Potential area of improvements

Considering the concerns discussed in the current UFCE algorithm in section 3.2, we improve each of the points mentioned in the disadvantages, focusing mainly on the user feedback part. Our main goal is to automate the user feedback such that there would be no dependence on the domain experts. Hence addressing the subjectivity issue. Further, we also address the issue of not using MI for single-feature perturbance by using permutation importance instead. The main idea for replacing user constraints is - Ontology instead [1]. Important concepts and the implementation details are discussed in further sections.

4 Ontology

4.1 What is Ontology

Ontology is a form of knowledge representation in a structured format that formally defines the concepts, entities, and relationships within a specific domain. The main purpose of using ontologies is to provide a shared vocabulary and semantic structure so that domains can be used and applied consistently. It also helps in semantic reasoning and validation. Figure 2 gives and example ontology for the domain of loan approval. This ontology formally defines entities like credit risk, loan and applicant. Each of these entities have various attributes. Constraints can now be imposed using the relationships and properties of these entities and attributes.

4.2 How can it be used in UFCE?

We decided to use ontologies in the UFCE framework to automate the user feedback generation and to remove the domain expert dependency. This would mean using the data distribution to automatically generate semantic rules which can be used as constraints on the features and feature values.

Ontologies can define allowed changes to features based on their conceptual roles and dependencies. For example, an ontology may define that "age" is immutable or that "income" must fall within a certain class hierarchy (e.g., low-income, middle-income, high-income). This can support automated validation and enrichment of user constraints. Further, using data-centric AI, permissible values for each feature modification can also be generated to include in the ontology.

4.3 Potential scalability of Ontology

Using ontology also has a very big potential for scalability. Since ontologies are universal and exist for most of the entities, the algorithm can be extended to universal ontologies like Wikidata, DBPedia,

etc. This would mean a universal and more generic approach to solving the problem of subjectivity in user constraints. Further, ontologies would make sure that the algorithm is scalable and reusable. Extending to global ontologies is out of the scope of our current work. However, it is important to note that these factors influenced our decision to consider using ontologies specifically for explainable AI and counterfactual explanations.

5 Technical Contribution: Our Approach

Our approach is based on the existing UFCE pipeline, addressing the critical concerns raised in section 3.2. To begin with, we implement all the algorithms in [3] for our test dataset. But instead of having manual user feedback, we automate the user constraints by using data-centric XAI techniques. Modified flow has been highlighted in Figure 3. This flow can be compared to the original UFCE pipeline in Figure 1 with the improvements introduced by us in a different color. These enhancements have been explained in detail in the following sub-sections.

5.1 Automating User Constraints

The main purpose of this enhancement is to automate the manual user feedback by generating user constraints using the data. To do so, we use data-centric AI techniques. So, the user constraint as depicted in 3.1 can be generated automatically using only the dataset without requiring any domain expert intervention.

5.2 Data Centric XAI

We make use of following data-centric XAI techniques to generate the user constraints:

- **Feature Summarization (Data Profiling)**: For numerical features, we compute the mean, min, and max For categorical features, the existing UFCE pipeline allows only binary feature values - essentially flipping the values for a perturbation. We instead use frequency distribution for categorical features, thus allowing multiple possible feature values.
- **Statistical Rule Extraction**: This is done to derive interpretable rules from the positive class by comparing distributions like quantiles for numeric features; frequency dominance for categorical features. So, the output of this step is human-readable rule-based explanations, highlighting class-specific patterns.
- **Feature Importance via Permutation**: This is used to assess how feature shuffling impacts model performance. We use it for two main use-cases:
  - Evaluate how good the statistical rules are by combining them with the rules derived using permutation importance. Hence there is a priority of features - giving an order of perturbations.
  - In the existing UFCE pipeline, as highlighted in section 3.2, single feature perturbations do not use mutual information. We instead use permutation feature importance for single feature perturbations as well - enhancing the counterfactuals produced by single feature changes.

6 Design Choices and Implications

To make enhancements in the existing UFCE pipeline, we made a few important design choices that we discuss in this section.

- **Backward compatibility** with the UFCE pipeline can easily replace ontological rules with human feedback. We keep the structure of derived constraints using data-centric AI techniques that are identical. Our goal is to ensure that in cases where users need to provide additional feedback, they can just append their constraints to the existing list and the pipeline works as it is.
- **Automated pipeline**: No human / domain expert intervention needed.

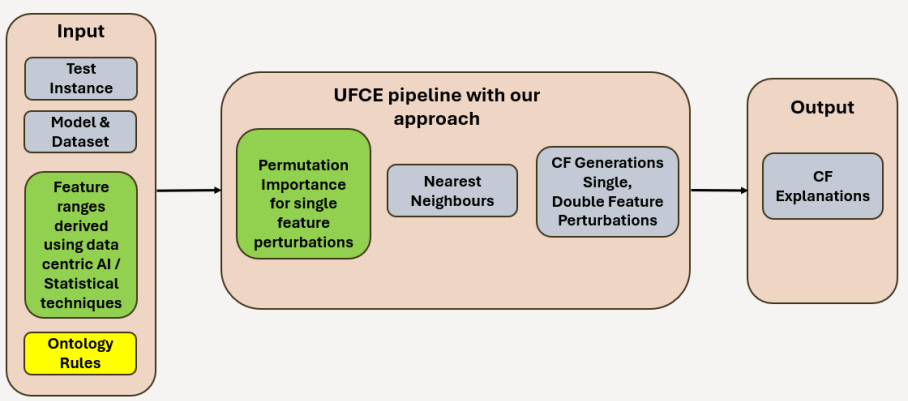

Figure 3: Our Approach: Modified UFCE pipeline with additional / improved steps using Data centric AI methods and Ontology concepts

- **Permutation combination**: As emphasized earlier in section 5.2, current UFCE pipeline does not use MI or any other technique for single feature perturbations and rely only on user feedback. We improve this by using permutation importance to get the most important features.
- **Scalability**: Future potential of being extended to universal ontologies like: Wikidata, DBPedia etc., thus connecting every entity. This step can be used to validate the generated counterfactuals against a set of constraints / rules.
- **Support non-binary categorical features**: Current UFCE pipeline only allows binary categorical features. This implies a change in feature value means flipping the original value. We instead use frequency distribution, hence support multiple possible values for categorical features.

7 Evaluation Metrics

We use similar evaluation metrics as defined in [3]. These metrics are as follows:

Feasibility: [3] defines it as a combined property of any counterfactual explanation which has a boolean value. It is $True$ when a counterfactual is valid, plausible and actionable.

- ***Validity***: A counterfactual is valid if it has a prediction flipped (different) than the original prediction.
- ***Plausibility***: A counterfactual is plausible is it belongs to the plausible space, that is within the dataset and is not an outlier.
- ***Actionability***: This refers to the percentage of changes made to the features that actually belong to the user defined list of features.

Sparsity: The percentage of feature changes between the original test instance and the generated counterfactual explanation.

Proximity: How close is the counterfactual to the original test instance. We use Euclidean distance for numerical feature and Jaccard similarity for categorical features.

8 Results and Inferences

We evaluate our pipeline in two stages. We first use a random dataset to check if counterfactuals are getting generated and other functionalities. We then use the dataset from the original paper [3] to compare the evaluation metrics for our algorithm with the results obtained for the original algorithm of UFCE.

8.1 Average Evaluation Metrics

8.1.1 Our Dataset

For testing purposes, we use the German Credit dataset[1] [10]. This dataset has 1000 instances and 20 features out of which 7 are numerical and the rest are categorical. This dataset is used for a binary classification problem of predicting whether a customer is good(1) or bad(2). Average results for feasibility, sparsity, and proximity for 90 negative (binary classification of 2) test samples is summarized in Table 1.

Table 1: Averaging evaluation metrics over all the negative test samples for counterfactual explanations on the German Credit (Our test dataset) for single and double perturbations.

Perturbation	Avg Feasibility	Avg Sparsity	Avg Proximity Num	Avg Proximity Cat
Single	0.54	0.016	0.0	0.913
Double	1.0	0.041	0.84	1.0

8.1.2 Dataset from the UFCE paper

For comparing the results obtained on an average for feasibility, sparsity, and proximity in our modified algorithm, we compare it against the same dataset as used in original algorithm in the paper [3]. This is a 'Bank Loan' dataset with a binary classification of 0 or 1. In the paper, the authors use 50 test instances to calculate the feasibility, sparsity, and proximity metrics (Table 7 in [**?**]). We replicate a similar experiment for our enhanced pipeline which is shown in Table 2. And for a comparison, we show the results presented in the paper for bank loan dataset in Table 3 (taken as it is from Table 7 in [**?**]).

Table 2: Evaluation metrics for 50 test samples to replicate the experiment shown in Table 7 of [3] for Bank Loan dataset used for single and double perturbations - using our modified algorithm

Perturbation	Feasibility	Sparsity	Proximity Num	Proximity Cat
Single	30.00	2.99	62.63	28.00
Double	0.98	0.17	1.86	0.64

Table 3: Evaluation metrics for 50 test samples - this is taken as it is from the paper for UFCE's original algorithm [3] for Bank Loan dataset used for single and double perturbations

Perturbation	Feasibility	Sparsity	Proximity Num	Proximity Cat
Single	14.00	1.00	0.60	10.00
Double	30.00	2.00	0.00	23.10

8.1.3 Analysis of the results

8.2 Ranking counterfactuals

One important use of the metrics we use is that we can use them to score the multiple counterfactuals obtained, and these scores can then be used to rank the counterfactuals, thus helping us select the best possible explanation for a given test instance. To do this, we maintain a map with counterfactual ID and the corresponding scores for each of the metrics. Then, as per the requirement, best counterfactual is chosen. A good counterfactual should have low proximity and sparsity score and high feasibility score. So when we calculate the average evaluation metrics over all the test instances, for each instance we only choose the best counterfactual explanation with lowest proximity and sparsity score.

9 Limitation and Future work

There are several limitations to our implementation. First, our implementation only perturbs instances based on single or double features, while the original implementation also supports triple feature

[1] `https://archive.ics.uci.edu/dataset/144/statlog+german+credit+data`

perturbations. Second, running the double feature perturbation method is timely, as its algorithm includes binary search and a prediction step by the feature value predictor model at each iteration, following the authors' implementation. However, in order to run the method locally in reasonable time, we set the maximum number of binary search iterations to 20. This limits the maximum amount of counterfactual explanations found for the given instance of interest. Future research is encouraged to address these limitations.

10 Conclusion

References

[1] Bertossi, L., & Geerts, F. (2020). Data quality and explainable AI. Journal of Data and Information Quality, 12(2), 1–9. https://doi.org/10.1145/3386687

[2] Crupi, R., Castelnovo, A., Regoli, D., & Gonzalez, B. S. M. (2021, June 14). Counterfactual explanations as interventions in latent space. arXiv.org. http://arxiv.org/abs/2106.07754

[3] Suffian, M., Alonso-Moral, J. M., & Bogliolo, A. (2024, February 26). Introducing user Feedback-based Counterfactual Explanations (UFCE). arXiv.org. https://arxiv.org/abs/2403.00011

[4] A. Adadi and M. Berrada, "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)," in IEEE Access, vol. 6, pp. 52138-52160, 2018, doi: 10.1109/ACCESS.2018.2870052. keywords: Conferences;Machine learning;Market research;Prediction algorithms;Machine learning algorithms;Biological system modeling;Explainable artificial intelligence;interpretable machine learning;black-box models,

[5] Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. 2023. From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI. ACM Comput. Surv. 55, 13s, Article 295 (December 2023), 42 pages. https://doi.org/10.1145/3583558

[6] Molnar, Christoph. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. 3rd ed., 2025. ISBN: 978-3-911578-03-5. Available at: https://christophm.github.io/interpretable-ml-book.

[7] Joshi, S., Koyejo, K., Vijitbenjaronk, W., Kim B., & Ghosh, J. (2019, July 22). Towards Realistic Individual Recourse and Actionable Explanations in Black-Box Decision Making Systems. arXiv.org. https://arxiv.org/abs/1907.09615

[8] Suresh Venkatasubramanian and Mark Alfano. 2020. The philosophical basis of algorithmic recourse. In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT* '20). Association for Computing Machinery, New York, NY, USA, 284–293. https://doi.org/10.1145/3351095.3372876

[9] Karimi, Amir-Hossein Schölkopf, Bernhard Valera, Isabel. (2021). Algorithmic Recourse: from Counterfactual Explanations to Interventions. 353-362. 10.1145/3442188.3445899.

[10] Hofmann, H. (1994). Statlog (German Credit Data) [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5NC77.

Solon Barocas, Andrew D. Selbst, and Manish Raghavan. 2020. The hidden assumptions behind counterfactual explanations and principal reasons. In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT* '20). Association for Computing Machinery, New York, NY, USA, 80–89. https://doi.org/10.1145/3351095.3372830