On the Generation of Realistic and Robust Counterfactual Explanations for Algorithmic Recourse

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Martin Pawelczyk

aus Dortmund

Tübingen

2023

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	19.06.2023
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Gjergji Kasneci
2. Berichterstatterin:	Prof. Dr. Ulrike von Luxburg

Abstract

This recent widespread deployment of machine learning algorithms presents many new challenges. Machine learning algorithms are usually opaque and can be particularly difficult to interpret. When humans are involved, algorithmic and automated decisions can negatively impact people's lives. Therefore, end users would like to be insured against potential harm. One popular way to achieve this is to provide end users access to algorithmic recourse, which gives end users negatively affected by algorithmic decisions the opportunity to reverse unfavorable decisions, e.g., from a loan denial to a loan acceptance. In this thesis, we design recourse algorithms to meet various end user needs. First, we propose methods for the generation of realistic recourses. We use generative models to suggest recourses likely to occur under the data distribution. To this end, we shift the recourse action from the input space to the generative model's latent space, allowing to generate counterfactuals that lie in regions with data support. Second, we observe that small changes applied to the recourses prescribed to end users likely invalidate the suggested recourse after being nosily implemented in practice. Motivated by this observation, we design methods for the generation of robust recourses and for assessing the robustness of recourse algorithms to data deletion requests. Third, the lack of a commonly used code-base for counterfactual explanation and algorithmic recourse algorithms and the vast array of evaluation measures in literature make it difficult to compare the performance of different algorithms. To solve this problem, we provide an open-source benchmarking library that streamlines the evaluation process and can be used for benchmarking, rapidly developing new methods, and setting up new experiments. In summary, our work contributes to a more reliable interaction of end users and machine learned models by covering fundamental aspects of the recourse process and suggests new solutions towards generating realistic and robust counterfactual explanations for algorithmic recourse.

Kurzfassung

Der in letzter Zeit weit verbreitete Einsatz von Algorithmen des maschinellen Lernens bringt viele neue Herausforderungen mit sich. Eine dieser Herausforderungen ist, dass Algorithmen des maschinellen Lernens in der Regel undurchsichtig und besonders schwer zu interpretieren sind. Wenn Menschen involviert sind, können sich algorithmische Entscheidungen negativ auf das Leben dieser Menschen auswirken. Daher möchten die Endnutzer gegen mögliche Schäden abgesichert sein und in der Lage sein "Revision" gegen die algorithmische Entscheidung einzulegen. Ein beliebter Weg, dies zu erreichen, besteht darin, Endnutzern Zugang zu "algorithmischer Revision" zu verschaffen, die Endnutzern, die von algorithmischen Entscheidungen negativ betroffen sind, die Möglichkeit geben, ungünstige Entscheidungen rückgängig zu machen; z. B. von einer Kreditverweigerung zu einer Kreditannahme. In dieser Arbeit entwerfen wir Revisionsalgorithmen, die verschiedenen Bedürfnissen der Endnutzer gerecht werden. Zunächst schlagen wir Methoden für die Generierung realistischer, algorithmischer Revision vor. Wir verwenden generative Modelle, um algorithmische Revisionen vorzuschlagen, die auf der Grundlage der Datenverteilung wahrscheinlich auftreten. Zu diesem Zweck verlagern wir die Revisionsaktionen aus dem Eingaberaum in den latenten Raum des generativen Modells und können so kontrafaktische Daten erzeugen, die in Regionen mit Datenunterstützung liegen. Zweitens stellen wir fest, dass kleine Änderungen an den algorithmischen Revisionen nachdem sie in der Praxis unpräzise umgesetzt wurden, die algorithmische Revision wahrscheinlich ungültig machen. Motiviert durch diese Beobachtung entwickeln wir Methoden zur Generierung und Bewertung von robusten algorithmischen Revisionen. Drittens erschweren das Fehlen einer allgemein verwendeten Codebasis für Revisionsalgorithmen sowie die Vielzahl von Bewertungsmaßstäben in der Literatur einen Vergleich der verschiedener Algorithmen. Um dieses Problem zu lösen, stellen wir eine Opensource Benchmarking-

Bibliothek vor, die den Evaluierungsprozess vereinfacht und für das Benchmarking, die schnelle Entwicklung neuer Methoden und den Aufbau neuer Experimente verwendet werden kann. Zusammenfassend lässt sich sagen, dass unsere Arbeit zu einer zuverlässigeren Interaktion zwischen Endnutzern und maschinengelernten Modellen beiträgt, indem sie grundlegende Aspekte des algorithmischen Revisionsprozesses abdeckt und neue Lösungen zur Erzeugung realistischer und robuster kontrafaktischer Erklärungen für algorithmischen Revisionen vorschlägt.

1 Introduction and Overview

1.1 Motivation

The internet and improvements in computing capabilities have made it easy to collect and process vast amounts of data. As a consequence, the use of machine learning algorithms has become widespread and covers many aspects of everyday routines where humans are involved. These routines cover all areas of daily life, such as hiring decisions [28, 129, 151], loan assignments [10, 42, 168], judicial parole decisions [13, 47], and disease risk prediction or diagnosis [95, 113].

This recent widespread development of the use of machine learning algorithms also presents many new challenges. When humans are involved, algorithmic and automated decisions can negatively impact people's lives. For this reason, we want to ensure that algorithmic decision support systems make decisions for the right reasons. As system designers, our hope is that we can ensure that the algorithmic decision support systems are transparent, accountable and can be trusted. However, algorithmic decision support systems are usually opaque and can be particularly difficult to interpret [33, 68, 110]. In addition to that, relative to a single decision maker, algorithmic decision support systems can make decisions at scale impacting thousands or millions of individuals simultaneously [112].

Unsurprisingly, undesirable effects of algorithmic decision support systems have been observed in practice: for example, to assist the human resource allocation team in hiring appropriate computer programmers Amazon rolled out an AI tool to screen promising candidates [21] which quickly turned out to put female applicants at a disadvantage. Similar biases have presumably been found in recidivism risk prediction settings [13, 37]. Simultaneously, policy makers have been working on creating regulations to control these systems to empower end users to gain insights regarding the specific decisions that were made about them.

To safeguard individuals' rights, policymakers have implemented regulations such as the General Data Protection Regulation (GDPR) [60] and the California Consumer Privacy Act (CCPA) [111] to regulate the collection and use of personal data and machine learning models. These regulations arguably provide individuals with the right to recourse, which refers to the ability to take steps to change an unfavorable outcome (such as a denied credit application) caused by a model's prediction.[1] Additionally, these laws empower individuals by giving them more control over their personal data, such as the right to withdraw consent for its use at any time [54].

To safeguard individuals from misconduct, researchers have developed the notion of algorithmic recourse which refers to the ability of individuals to challenge or contest decisions supported by algorithms, and to have those decisions reviewed or revised if necessary. It is a way of ensuring that algorithms are accountable and transparent in their decision support processes, and that individuals have the opportunity to correct any errors or biases that may have influenced the algorithms' decisions [159, 166]. Algorithmic recourse can take many forms, depending on the specific context in which the algorithm is being used. For example, in the context of automated decision-making systems used in hiring or lending, algorithmic recourse might involve the ability to request an explanation for why a particular decision was made, and to provide additional information or context that could influence the decision. In other cases, algorithmic recourse might involve the ability to appeal a decision made by an algorithm, or to have the decision reviewed by a human. Summarizing, the goal of algorithmic recourse is to ensure that algorithms are used in a fair and transparent manner, and to give individuals the opportunity to correct any errors or biases that may have influenced the algorithms' decisions. This can help to build trust in algorithmic systems and to ensure that they are used ethically and responsibly.

In this thesis, we develop algorithmic tools that provide algorithmic recourses to humans and evaluate the quality of these recourses. We approach this problem through the lens of counterfactual explanations, which is a type of explanation that describes what would have happened if some aspect (i.e., some feature values) of the situation had been different. Following Wachter et al. [166] we consider explanations of the following form:

> Score f was assigned since the features $\mathbf{v}$ had values $v_1, v_2, \cdots$. If $\mathbf{v}$ had values $v'_1, v'_2, \cdots$ instead, *and all other variables were constant*, score

[1]We will discuss the legal motivation for the deployment of algorithms to enable recourse in Chapter 2.

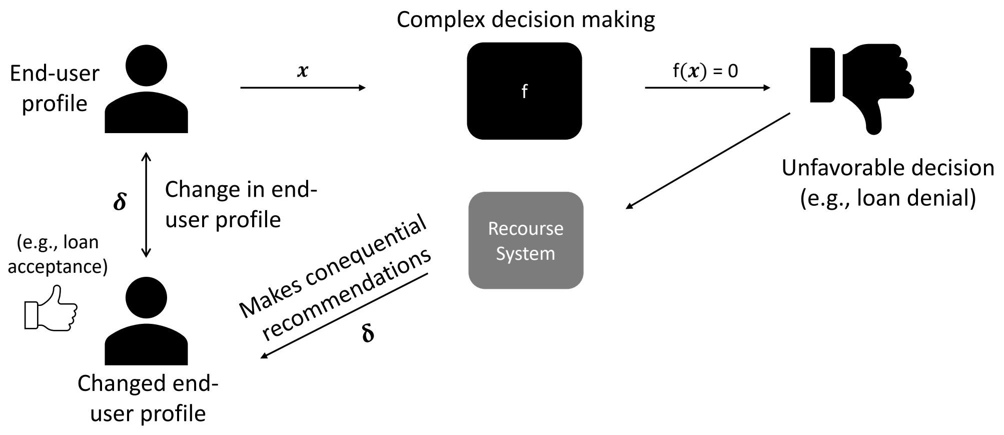


Figure 1.1: **A schematic depiction of an algorithmic decision support-recourse system**. End users, who are faced with an unfavorable decision (e.g., loan denial), receive counterfactual explanations δ from the recourse system. These counterfactual explanations provide the user with an instruction on how to change their end user profile (e.g., decrease amount of outstanding credit card debt by 500 Euros) to receive the favorable outcome (e.g., loan acceptance).

f' would have been returned.

Consistent with the view of Wachter et al. [166] we adapt a non-causal perspective on these explanations. Although misleading due to the term "counterfactual" being rooted in causal literature [123, 126], we follow the convention of the explainability literature and call these explanations *counterfactual explanations*.

We end this section by providing real-world examples that highlight the shortcomings of automated decision-making systems. At the same time, these examples also demonstrate how counterfactual explanations can be used to improve transparency and provide a means of recourse in algorithmic decision-making (see also Figure 1.1 for a schematic depiction of a recourse system):

(i) **Human resource allocation:** Consider the specific use-case of college admissions in the US. In 2014, the students for fair admissions (SFFA) group filed a lawsuit against Harvard College claiming that Harvard's admission process would violate Title VI of the American Civil Rights Act by using racial quotas that disadvantage Asian-Americans. In this context, Arcidiacono [14] used data on applicants to Harvard college to analyze whether the process of accepting applicants was biased towards admitting "white" applicants relative to African-American or Asian-American applicants. Arcidiacono's report used a statistical model to predict the applicants' acceptance probabilities based on the their available information such as test scores, academic achievements, extra curricular activities and demographic information. Using a counterfactual explanation, the report concluded that an Asian-American applicant with a 25% chance of being admitted would increase their chance by 11 percentage points if the applicant changed their race attribute to "White" holding all other feature values in the analysis constant. While this is not the only way to measure whether a prediction system is fair on not, such counterfactual explanations can give rise to further scrutinize the admission process and current practices.

(ii) **Health care:** A patient could be observed more closely if an algorithmic decision support model suggests a high probability of an adverse outcome (e.g., cario-vascular disease (CVD)) [88]. In this setup, we could be interested in identifying actions that reduce or even minimize the probability of CVD. Suppose an automated decision making model is based on patient records including information on medications, lab measurements (e.g., blood pressure, heart rate), lifestyle choices (e.g., diet, sports) and demographic information (e.g., age, sex). While doctors often give handwavy suggestions or suggestions that apply to the "average patient", a recourse system can make exact recommendation on how to change a patient's profile to improve health outcomes (e.g., by suggesting a particular kind of medication).

1.2 Outline and Contribution

In this thesis, we present solutions to three major challenges in the field of counterfactual explanations for algorithmic recourse. These challenges are discussed in more detail in Chapter 2. Throughout this thesis, we illustrate these problems through real-world examples and provide new methodological, theoretical, and empirical insights to address them. In Chapter 3, we introduce novel frameworks and implementations that overcome the limitations of previous works. In Chapter 4, we explore the connections between adversarial examples and algorithmic recourse. With our findings, we develop more robust algorithms for generating algorithmic recourse in Chapter 5. Our final contribution is the development of a benchmarking library in Chapter 6, which allows practitioners and researchers to track the progress of new recourse algorithms in a central repository and compare the performance of different algorithmic recourse methods. Through these contributions, we aim to enhance the reliability of automated machine learning systems and address known issues in the recourse process (as discussed in [80, 163]).

This thesis is based on six publications, each contributing to one of the three issues mentioned above. In the following, we briefly summarize our contributions.

1.2.1 Algorithms for Realistic Recourse

Generating recourses that are likely to occur. Previous algorithmic recourse methods did not consider whether the suggested explanations were likely to occur or meet requirements such as counterfactual faithfulness (i.e., proximity to correctly classified observations and connection to regions with substantial data density). In order to address this issue, we propose a framework called `CCHVAE` (Counterfactual Conditional Heterogenous Variational Autoencoder) that generates counterfactual explanations that are faithful to the data, drawing on ideas from learning generative models. Additionally, we suggest the use of a criterion to evaluate the difficulty of a particular counterfactual suggestion as a complement to existing quality measures. Our experiments show that generating faithful counterfactual explanations can come at a higher cost of recourse.

Generating recourses under feature dependencies. Most algorithmic recourse methods rely on the assumption of independently manipulable features (IMF) in order to generate low-cost recourse. However, this assumption is not always realistic. To address the feature dependency issue the recourse problem is posed through the causal recourse paradigm. However, it is well known that strong assumptions, as encoded in causal models and structural equations, hinder the applicability of these methods in complex domains where causal dependency structures are ambiguous. To mitigate both issues, we introduce `DEAR` (DisEntangling Algorithmic Recourse), a recourse framework that disentangles latent features that covary with a subset of promising recourse features.

1.2.2 On the Connections between Algorithmic Recourse and Adversarial Examples

This part of the thesis explores the relationship between two distinct but related areas of research: adversarial examples and counterfactual explanations. These examples are small, often imperceptible perturbations to inputs (most often images) that flip the prediction of the classifier in an often unpredictable way. For example, the prediction could flip from “Human” to “Frog”, while the image would still be perceived as a human. Research in this area has focused on understanding why these examples exist, how to generate them, and how to prevent them. This literature has primarily framed the problem of the existence of adversarial examples as a nuisance that should be avoided when possible. Meanwhile, counterfactual explanations involve identifying minimal changes that need to be made to an input to change the model's prediction towards a more favorable outcome for the individual (e.g., loan acceptance). This research has primarily focused on providing explanations for model predictions. Here we show that these two fields are closely related and that understanding the relationship between them can inform the design of counterfactual explanations.

1.2.3 Algorithms for Robust Recourse

Tradeoffs between actionable explanations and the right to be forgotten. There is currently a lack of understanding on whether the principles of data protection, such as the "right to be forgotten" and the alleged "right to an explanation" (algorithmic recourse), can be implemented simultaneously in machine learning (ML) systems. To address this issue, we investigate the problem of recourse invalidation in the context of data deletion requests. We analyze the behavior of state-of-the-art algorithms and find that recourses generated by these algorithms are likely to be invalidated if a small number of data deletion requests result in updates to the predictive model. We propose a framework to identify a minimal set of critical training points that, when removed, maximize the fraction of invalidated recourses. Across data sets used in our experiments, we demonstrate that the removal of as few as two data instances from the training set can invalidate up to 95% of recourses output by popular algorithms. This part of our thesis raises questions about the compatibility of algorithmic recourse and the right to be forgotten, and provides insights into the factors that determine recourse robustness.

Robust recourses under noisy human responses. The existing algorithmic recourse methods face challenges in delivering low-cost solutions while maintaining robustness in the face of real-world noise, which can happen when users do not follow the prescribed recourse precisely. These methods also lack the option for users to balance the cost and robustness. To address these issues, we present the Probabilistically ROBust rEcourse (`PROBE`) framework that enables users to set the probability of recourse invalidation if there is some deviation in real-world responses from the prescribed recourse. We propose a novel objective function that simultaneously encourages robust recourses, minimizes recourse costs, and ensures that the resulting recourse has a favorable prediction from the model. We also provide theoretical results that characterize the invalidation rates for different types of models and use these results to optimize the proposed objective. Our experiments on real-world datasets demonstrate the effectiveness of PROBE.

1.2.4 Standardizing the Evaluation of Recourse Methods

This part of the thesis establishes a standardized approach for evaluating algorithmic recourse methods by providing a set of meaningful metrics for evaluating counterfactual explanations and introducing a novel evaluation framework, CARLA (Counterfactual And Recourse LibrAry). The lack of a commonly used code-base for counterfactual explanation and algorithmic recourse algorithms, and the vast array of evaluation metrics in literature make it difficult to compare the performance of different methods. CARLA streamlines the evaluation process and can be used for benchmarking, rapidly developing models, and setting up new experiments. It is highly customizable and can be integrated with custom code. The framework is open-source and accessible on Github and Pypi.

1.3 Publications

This thesis is based on 6 publications, 5 of which have been accepted at internationally highly respected conferences and 1 of which was under review at the time of thesis submission.

Contribution 1

Learning Model-Agnostic Counterfactual Explanations for Tabular Data. Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. 2020. In Proceedings of The Web Conference (**WWW**), [115].

Contribution 2

Decomposing Counterfactual Explanations for Consequential Decision Making. **Martin Pawelczyk**, Lea Tiyavorabun, and Gjergji Kasneci. 2022. arXiv:2211.02151, [119]. Under review at UAI 2023.

Contribution 3

Exploring Counterfactual Explanations Through the Lens of Adversarial Examples: A Theoretical and Empirical Analysis. **Martin Pawelczyk**, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay and Himabindu Lakkaraju. 2022. In International Conference on Artificial Intelligence and Statistics (**AISTATS**), [118].

Contribution 4

On the Trade-Off between Actionable Explanations and the Right to be Forgotten. **Martin Pawelczyk**, Tobias Leemann, Asia Biega*, Gjergji Kasneci*. 2022. In International Conference on Learning Representations (**ICLR**), [122].

Contribution 5

Probabilistically Robust Recourse: Navigating the Tradeoffs between Costs and Robustness in Algorithmic Recourse. **Martin Pawelczyk**, Teressa Datta, Johannes van-den-Heuvel, Gjergji Kasneci, and Himabindu Lakkaraju. 2023. In International Conference on Learning Representations (**ICLR**), [120].

Contribution 6

CARLA: A Python Library to Benchmark Algorithmic Recourse and Counterfactual Explanation Algorithms. **Martin Pawelczyk**, Sascha Bielawski., Johannes Van den Heuvel, Tobias Richter* and Gjergji Kasneci*. 2021. In Advances in Neural Information Processing Systems (**NeurIPS**), [117].

During my time as a PhD student I also contributed to numerous other papers, which will not be described in this thesis:

Leveraging Model Inherent Variable Importance for Stable Online Feature Selection. Johannes Haug, **Martin Pawelczyk**, Klaus Broelemann, and Gjergji Kasneci. 2020. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (**KDD**), [71].

On Counterfactual Explanations under Predictive Multiplicity. **Martin Pawelczyk**, Klaus Broelemann and Gjergji Kasneci. 2020. In 36th Conference on Uncertainty in Artificial Intelligence (**UAI**), [116].

Model Selection in Local Approximation Gaussian Processes: A Markov Random Fields Approach. Hamed Jalali, **Martin Pawelczyk**, and Gjergji Kasneci. 2021. In IEEE International Conference on Big Data (**Big Data**), [76].

OpenXAI: Towards a Transparent Evaluation of Model Explanations. Chirag Agarwal, Eshika Saxena, Satyapriya Krishna, **Martin Pawelczyk**, Nari Johnson, Isha Puri, Markinka Zitnik and Himabindu Lakkaraju. 2022. In Advances in Neural Information Processing Systems (**NeurIPS**), [4].

Deep Neural Networks and Tabular Data: A Survey. Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, **Martin Pawelczyk**, and Gjergji Kasneci. 2022. In Transactions on Neural Networks and Learnings Systems (**TNNLS**), [29].

On the Privacy Risks of Algorithmic Recourse. **Martin Pawelczyk**, Himabindu Lakkaraju, and Seth Neel. 2023. In International Conference on Artificial Intelligence and Statistics (**AISTATS**), [121].

Large Language Models are Realistic Tabular Data Generators. Vadim Borisov, Kathrin Seßler, Tobias Leemann, **Martin Pawelczyk**, and Gjergji Kasneci. 2023. In International Conference on Learning Representations (**ICLR**), [30].

I Prefer not to Say: Are Users Penalized for Protecting Sensitive Data?. Tobias Leemann, **Martin Pawelczyk**, Christian Thomas Eberle, and Gjergji Kasneci. 2022. arXiv:2210.13954, [92]. Under review at ICML 2023.

On the (In)compatibility of Short and Long-Term Fairness in Sequential Decision Making. Danilo Brajovic*, Tobias Leemann*, **Martin Pawelczyk***, Niki Kilbertus, and Gjergji Kasneci. 2023. Unpublished. Under review at FAccT 2023.

2 Background

In this Chapter we provide background on the fundamental challenges around generating algorithmic recourse. We will take a top-down approach and start our discussion where the motivation of numerous research papers on algorithmic recourse stops: with the argument that algorithmic recourse through counterfactual explanations is well justified by the European Union's (EU) General Data Protection Regulation (GDPR) [51]. We then discuss to what extent research on algorithmic recourse can be motivated by modern data protection regulation, and whether we can use these regulations to identify implicitly encoded desiderata for algorithmic recourse from these regulations. We then formulate an extended catalogue of desiderata on how recourses should best be designed to serve a variety of possible end user needs. With the basic requirements in place, we present the basic mathematical background on the workhorse recourse model which we will be using throughout. Finally, we will introduce the three main themes of this thesis: the *generation of **realistic** and **robust** counterfactual explanations for algorithmic recourse and their reliable **evaluation***. Motivated by these themes we will describe the particular problems within these main themes in more detail before we conclude.

2.1 The Legal Basis of Algorithmic Recourse

The majority of works (e.g., [79, 115, 122, 159]) that focus on building technical recourse algorithms to explain complex ML models derive the motivation for the use and development of recourse methods from data protection regulations such as the GDPR which came into effect in April 2018. As a European Union regulation, the GDPR does not require its member states to enable state specific legislations and has effectively replaced the EU's Data Protection Directive (DPD). In contrast to the GDPR, the DPD required member states to pass individual state specific regulations implementing the directives as they deemed appropriate. Hence, as of 2018 the GDPR is effectively the EU's data processing and data protection law. In this Section, we will take a closer look at key laws from the GDPR which supposedly motivate the need for counterfactual explanations for algorithmic recourse and analyze what exactly the GDPR and other data regulations say about the provision of algorithmic explanations when automated decision making algorithms are a part of a data controller's decision making process.

2.1.1 Algorithmic Recourse through Counterfactual Explanations: A Clear Mandate through the GDPR?

A closer look at European data protection regulation. It has often been argued that the General Data Protection Regulation (GDPR) gives individuals the "right to explanations" from the controller (the organization or individual that processes their personal data) regarding the processing of their personal data (e.g., [66, 136]). This is usually understood as an ideal procedure to enhance the transparency of automated decision-making systems. Arguably, this right is provided under Articles 13 and 14 of the GDPR, which state that individuals have the right to obtain [51, Articles 13 and 14]:

> "[...] information necessary to ensure fair and transparent processing in respect of the data subject [...]".

Specific information that must be provided includes information on [51, Articles 13 and 14]:

> "[...] the existence of automated decision-making, including profiling, referred to in Article 22(1) and (4) and, at least in those cases, meaningful information about the logic involved, as well as the significance and the envisaged consequences of such processing for the data subject."

These articles grant users to obtain "meaningful information about the logic involved" in automated decision making processes. At first glance, such formulations seem to suggest that something like a right to explanations could be derived from here. Additionally, Article 22(3) gives individuals the right to obtain human intervention to express their point of view, and to contest the algorithmic and automated decision [51, Article 22, italics added]:

> "[...] the data controller shall implement suitable measures to safeguard the data subject's rights and freedoms and legitimate interests, *at least the right to obtain human intervention on the part of the controller, to express his or her point of view and to contest the decision*."

Thus, together the articles 13, 14 and 22 of the GDPR establish the data controllers notification duties for individuals to understand how their personal data are being processed. However, do these articles actually allow to derive a "right to explanations" for individuals affected by automated decision-making carried out on their personal data?

No right to decision specific explanations? While a right to an explanation is neither explicitly mentioned in articles 13, 14 and 22 nor in any other article of the GDPR [165], such a right could be derived from recital 71 of the GDPR, which states:

> "[...] should be subject to suitable safeguards, which should include specific information to the data subject and the right to obtain human intervention, to express his or her point of view, to obtain an explanation of the decision reached after such assessment and to challenge the decision."

In contrast to articles 13, 14 and 22, which do not explicitly state a right to explanations, recital 71 does state such a right by granting individuals a "right to obtain human intervention, to express his or her point of view, to obtain an explanation of the decision reached". To better understand whether end users have a right to individual explanations or not it is instrumental to understand the relation between articles within the GDPR and recitals. Thus, a short digression is in place: as opposed to articles within the GDPR recitals form the preamble of the GDPR, which explain the rationale behind groups of or individual articles, but rectícals are not legally binding. Rectícals help to explain the purpose of the law [86] and cannot be used for "legitimate expectations to arise" [18]. This begs the question of what kind of information end users can realistically receive from the controller when automated decision making is applied on their data. Wachter et al. [165] address this question by categorizing explanation types along the dimensions *explanation style* and *timing* to understand what kind of explanations the GDPR requires data controllers to provide to end users:

- A *system functionality* explanation illustrates the logic, the envisaged consequences and the general functionality of automated decision-making to the individual (e.g., the system uses a decision trees to forecast credit risk);
- A *specific decision* explains the logic and the underlying reasons that led to the individual outcome of the automated decision making process given

the very specific circumstances of the individual (e.g., individual specific rules list);

- An *ex ante* explanation is issued before the automated decision making takes place (note that ex ante explanations will mostly be system functionality explanations);
- An *ex post* explanation is issued after the automated decision making took place (e.g., individual specific rule list is issued after the decision took place to explain the decision to the end user).

Based on this categorization Wachter et al. [165] further argue that the GDPR's specific use of language suggests that data controllers are specifically required to provide ex ante system functionality explanations as opposed to ex post specific decision explanations (e.g., the phrasing "envisaged consquenes" suggests that articles 13 and 14 are aiming at ex ante explanations).

In summary, under the GDPR, it is not clear whether individuals do have the right to an actionable explanation as it appears to lack well-defined rights to make such claims. Individuals appear to have a "right to be notified" by the controller prior to the automated decision making process regarding the processing of their personal data, including the purposes of the processing, the categories of personal data concerned, and the recipients to whom the personal data have been or will be disclosed. This information must be provided to individuals ex ante, i.e., *at the time their personal data is collected*, and individuals also have the right to request additional information about the processing of their personal data at any time. A right to an ex post explanation of a specific decision can likely not be derived from the GDPR although recital 71 suggests that the law makers want data controllers to provide such decision specific explanations.

In the next section, we turn our focus to US legislation and see whether we can identify a more clearly articulated footing for decision specific explanations of automated decision making.

2.1.2 Adverse Action Notices and Counterfactual Explanations

Adverse action notices are required in certain situations where a decision or action has been taken that may negatively impact an individual's credit, employment, insurance, or other important financial or personal matters. These notices are typically required by laws such as the Equal Credit Opportunity Act (ECOA) in the United States, which prohibits discrimination in credit transactions based on certain factors such as race, color, religion, national origin, sex, age, or marital status. Adverse action notices should state "the principal reason for the denial", must be provided in a clear and concise manner, and must include information about the specific decision or action that has been taken, the reasons for the decision or action, and any rights that the individual has to challenge or contest the decision or action [39].

Counterfactual explanations and adverse action notices are both types of explanations that can be used to provide information about decisions or actions that have been taken or may be taken in relation to individuals. As these two can be used in different contexts, they may not necessarily be considered to provide "algorithmic recourse" in the same sense as it is well-known that adverse action notices do not provide actionable information on the features that should be changed to receive the desirable outcome (see Taylor [153] for a critique).

Proposal	Feature Subset	Current Value		Required
1	`# delays elsewhere / year`	5	→	0
2	`current income`	\$1000	→	\$2500
3	`tenure w/ current job`	4 months	→	12 months
	`credit file`	NaN	→	True

(a) **Algorithmic Recourse Example**. Stylised example for one individual who was denied credit by a machine learning classifier. There exists an entire set of features representing this individual (not shown). The rows show *detailed prescriptions* on which input subsets would need to change for the individual to be awarded the loan. The difference between the *current values* and the *required values* are the costs of counterfactual explanations.

Proposal	Key Factors
1	`poor credit performance elsewhere`
2	`insufficient income`
3	`length of tenure`
	`no credit file`

(b) **Adverse Action Notice Example**. Stylised example for one individual, who was denied credit under the Equal Credit Opportunity Act (ECOA) regulation. The rows show which *high–level factors* led to the loan rejection decision.

Table 2.1: **Comparing counterfactual explanations to adverse action notices**.

One example of an adverse action notice is a notice that is provided to an individual who has been denied a credit application (see Table 2.1b). In this case, the adverse action notice must explain the reasons for the credit denial, such as a low credit score or a high debt-to-income ratio, and must provide information about any rights the individual has to challenge or contest the decision.

Counterfactual explanations, on the other hand, are not required by US or European law (see the previous Section) in the same way that adverse action notices are. As opposed to adverse action notices, counterfactual explanations can be used to provide information about what might have happened if a different decision or action had been taken (see Table 2.1a). Thus, counterfactual explanations can help to provide transparency and accountability for automated decisions or actions by explaining how the decision or action would have been different if certain factors had been changed. By doing so, counterfactual explanations could provide end users with more granular explanations.

2.1.3 Key Takeaways

Both adverse action notices and counterfactual explanations are used to provide information about decisions that have been taken or may be taken, but they serve different purposes. Adverse action notices are required by law in certain situations and must be provided to individuals in a clear and concise manner, while counterfactual explanations are not required by law, but do provide an elevated level of transparency and accountability for automated decisions by explaining how the decision would have been different under different circumstances.

There is no clearly articulated law on the European level that motivates the use of counterfactual explanations to provide end users with decision specific explanations. However, the GDPR's preamble suggests that the law makers envisioned a right to a decision specific explanation when writing articles 13, 14 and 22. Regarding the US, law makers have passed the ECOA which gives users the right to obtain an adverse action notice. So far, these notices are being constructed in a rather vague way (see Table 2.1b). Thus, counterfactual explanations could provide an algorithmic tool to beyond the basic legal requirement and enhance algorithmic decision making. To the best of our knowledge, there does not seem to be regulation that would prevent to replace adverse action notices by more fine grained counterfactual explanations.

Companies in regulated industries may have a financial incentive to implement algorithmic recourse methods in order to identify and address issues with their decision-making algorithms. By using counterfactual explanations, they can identify and address ethical concerns, reduce the risk of legal action, improve transparency and accountability in their decision-making processes, and increase the overall effectiveness and efficiency of their algorithms [19, 20, 112, 162, 166].

Additionally, there are other complex and nuanced reasons for the adoption of algorithmic recourse. These reasons include:

(i) **Fitting more complex models:** Using algorithmic recourse methods holds the promise of relaxing machine learning model constraints. The law often puts a constraint on what types of predictive machine learning models can be deployed in practice because of the lack of interpretability of black-box models. Additionally, industries such as credit lending or insurance are highly regularized industries exactly due to the low interpretability of opaque black–box predictive models, constraining institutions on what can realistically be deployed in practice.

(ii) **Protect business models from fraud:** This reason is tightly linked to the one in (i) as Wachter et al. [166] argue that the use of (complex) black–box models enables companies to better shield their models from fraudulent behaviours and thus enable companies to more effectively protect their business model.[1] Such fraudulent behaviors include the theft of proprietary information including the accurate identification or reconstruction of the deployed predictive model or the training data set.

(iii) **Identifying data issues:** Algorithmic recourse can also be useful because it can help people identify and address potential issues with the data that the algorithm is trained on. If the data used to train the algorithm is biased or incomplete in some way, this can lead to biased or inaccurate predictions. By understanding how the algorithm arrived at a particular decision, people can identify any issues with the data and take steps to correct them.

(iv) **Compliance with (shaky) regulatory requirements:** Algorithmic recourse seems to comply with legal requirements [60]. Counterfactual explanations offer a fix to this looming problem by revealing the key factors in favour of the decision to the end–user, which enhances the decision maker's accountability [166], and may improve procedural fairness [67].

(v) **Increasing user satisfaction and profits:** Algorithmic recourse can enhance the perception of fairness and trust in the system by providing users with meaningful options to achieve more favorable outcomes. Furthermore, recourse can be mutually beneficial for both end users and companies - end users can attain their desired outcome while companies may increase profits by expanding their customer base.

In conclusion, as there is no set standard for the generation of decision specific explanations for algorithmic recourse, current regulations do not provide direc-

[1]We will come back to this aspect in Chapter 7. There, we evaluate this motivation in light of new evidence, and what this implies for future work on reliable algorithmic recourse.

tions on how to create them. However, despite this lack of guidance, experts in the field have begun to consider the ideal characteristics of algorithmic recourse. In the following section, we summarize the commonly agreed-upon desiderata that have been established in the literature on algorithmic recourse.

2.2 From Regulations to Desiderata: The Machine Learning Challenges

In a concerted effort over the past few years, the research community has agreed on a catalog of desirable properties for algorithmic recourse (e.g., [20, 80, 116, 117, 162, 163, 166]). This is to ensure that the prescribed recourses are actually implementable and useful to end users. In the following, we describe key properties for generating realistic, robust, and actionable counterfactual explanations for algorithmic recourse:

(a) Feasibility. As the goal is that recourses should be reliably used by end users we have to define what we mean by a recourse to be feasible to the end user. The following criteria capture such a high level definition:

   (i) *Complexity.* It is commonly stated that parsimonious explanations are the easiest for humans to parse and understand [103, 154, 157]. Therefore the counterfactual should prescribe as few changes to the factual input as possible.

   (ii) *Costs.* The concept of costs reflects the idea that individuals have to pay a “price” to change from the current outcome to the desired outcome [159, 166]. Ideally, the recourses should be easy to reach at low costs to the individual. For example, it could be more desirable for an individual to change three inputs by a little instead of changing one input by a lot. Complexity and costs are thus two sides of the same coin. Ideally, a recourse should have both low recourse costs and low complexity.

   (iii) *Meaningfulness.* Recourses need to adhere to actionability constraints as certain input feature cannot be changed by individuals [159]. For example, it is unethical to ask users to change protected attributes such as sex or race as these attributes are not under the user’s control.

   (iv) *Density.* Recourses should lie in regions of sufficient data density as we do not want the counterfactuals to be outliers. Additionally, the recourses should also be close to correctly and positively classified points from the desired class [90]. This captures the idea that a particular area of the feature space is generally reachable by humans.

  (v) *Dependency.* Recourses should adhere to data dependencies [82, 119]. This requirement aims to ensure that logically correct counterfactual explanations should be generated: e.g., (1) age can only increase or (2) when education level increases, we expect the feature age to increase as well.

(b) Robustness. When implementing prescribed recourse, a change in circumstances to both the decision maker and the end user could render the recourse invalid or impossible to achieve. To minimize the decision maker's liability problem and maximize the end user's ability to plan ahead and take appropriate action, recourse robustness is a key characteristic. Since there are different sources of unpredictability, we divide three scenarios to be robust against:

  (i) *Noisy responses.* Recourses should be robust to small implementation inaccuracies by the end users [120] as the phenomenon of noisy responses to prescribed recourses appears common in the real world: Björkegren et al. [26] conducted a field experiment in Kenya by mimicking the "digital loan" setting to study algorithmic recourse in real-world scenarios, and found that individuals respond in a noisy fashion.

  (ii) *Noisy input data.* Another requirement is to ensure that the recourse is robust to very small perturbations in the factual input data [15]. Such perturbations could result from rounding errors of the input data or erroneous transcription of an individual's data record into a database.

  (iii) *Model parameter changes.* An update of the underlying predictive model due to shifts in the data distribution will lead to an update of the model parameters. Therefore the recourse are at risk of being invalidated under the new model [116, 132]. Hence, we require that counterfactual explanations should be made robust to small changes in the model parameters.

It is generally difficult to address all requirements simultaneously. In fact, there are various trade-offs that need to be considered some of which we will see in Chapter 5. In the following Section, we will review the basic model to generate counterfactuals for algorithmic recourse which set out to provide recourses with (i) low complexity and (ii) low costs while (iii) adhering to feasibility constraints. Based on this model, we will then discuss some of the fundamental open problems from this literature that we will tackle in this thesis. Using the coarse categorization developed in this section (feasibility versus robustness) the individual Chapters 3 - 5 then provide a review of state of the art (counterfactual) explanation methods suited to provide algorithmic recourse.

2.3 From Desiderata to Implementations

The previous section has established several desiderata for the generation of reliable counterfactual explanations. In order to transfer these desiderata to an algorithmic output we require a mathematical model, which is presented in Section 2.3.1. In its most basic form, this model does not incorporate all of the desiderata from the previous Section. However, throughout this thesis we will use this model as a workhorse and a point of departure in order to build additional elements into the model to capture some of the high level desiderata such as robustness to noisy responses or the generation of counterfactual explanations subject to data density constraints. Thereby, we address some of the open algorithmic problems in the field of algorithmic recourse, which are presented in Section 2.3.2, and further studied in Chapters 3 - 5 in more detail.

2.3.1 Formal Background

We consider prediction problems from some input space $\mathbb{R}^d$ to an output space $\mathcal{Y}$, where d is the number of input dimensions. We denote a sample by $\mathbf{z} = (\mathbf{x}, y)$, and denote the training data set by $\mathcal{D} = \{\mathbf{z}_1, \dots, \mathbf{z}_n\}$. Consider the weighted empirical risk minimization problem (ERM), which gives rise to the optimal model parameters:

$$\mathbf{w}_{\boldsymbol{\omega}} \in \arg\min_{\mathbf{w}'} \sum_{i=1}^{n} \omega_i \cdot \ell\big(y_i, f_{\mathbf{w}'}(\mathbf{x}_i)\big), \tag{2.1}$$

where $\ell(\cdot, \cdot)$ is an instance-wise loss function (e.g., binary cross-entropy, mean-squared-error (MSE) loss, etc.) and $\boldsymbol{\omega} \in \{0, 1\}^n$ are data weights that *are fixed at training time*. If $\omega_i = 1$, then the point $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ is part of the training data set, otherwise it is not. During model training, we set $\omega_i = 1 \; \forall i$, that is, the decision maker uses all available training instances at training time. In the optimization expressed in (2.1), the model parameters $\mathbf{w}$ are usually an implicit function of the data weight vector $\boldsymbol{\omega}$ and we write $\mathbf{w}_{\boldsymbol{\omega}}$ to highlight this fact; in particular, when all training instances are used we write $\mathbf{w}_{\mathbf{1}}$, where $\mathbf{1} \in \mathbb{R}^n$ is a vector of 1s.

In summary, we have introduced the *weighted* ERM problem since it allows us to understand the impact of arbitrary data deletion patterns on actionable explanations in Chapter 5 as we allow users to withdraw their entire input $\mathbf{z}_i = (y_i, \mathbf{x}_i)$ from the training set used to train the model $f_{\mathbf{w}_1}$. When it is clear from the context, we will drop the dependence of f on its model parameters $\mathbf{w}$ in the following Chapters for the purpose of better readability. Next, we present the recourse model we consider.

We follow an established definition of counterfactual explanations originally proposed by [166]. For a given model $f_{\mathbf{w}_{\boldsymbol{\omega}}} : \mathbb{R}^d \rightarrow \mathbb{R}$ parameterized by $\mathbf{w}_{\boldsymbol{\omega}}$ and a cost function $c(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$, the problem of finding a recourse $\check{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$ for a factual instance $\mathbf{x}$ is given by:

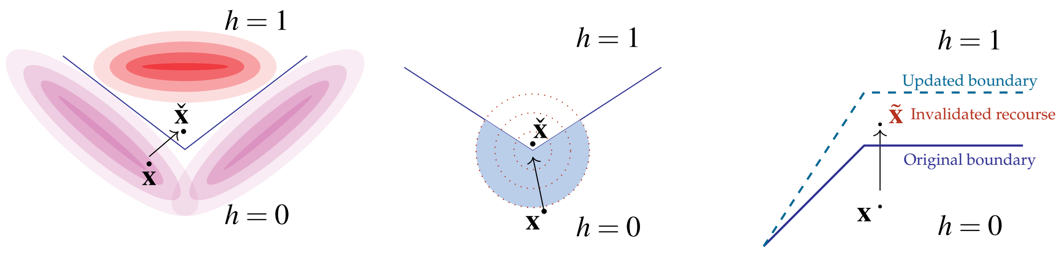
$$\boldsymbol{\delta}_{\boldsymbol{\omega},\mathbf{x}} \in \arg\min_{\boldsymbol{\delta}' \in \mathcal{A}_d} (f_{\mathbf{w}_{\boldsymbol{\omega}}}(\mathbf{x} + \boldsymbol{\delta}') - s)^2 + \lambda \cdot c(\mathbf{x}, \mathbf{x} + \boldsymbol{\delta}'), \tag{2.2}$$

where $\lambda \geq 0$ is a scalar tradeoff parameter and s denotes the target score. In the optimization from equation (2.2), the optimal recourse action $\boldsymbol{\delta}$ usually depends on the model parameters and since the model parameters themselves depend on the exact data weights configuration we write $\boldsymbol{\delta}_{\boldsymbol{\omega},\mathbf{x}}$ to highlight this fact. The first term in the objective on the right-hand-side of equation (2.2) encourages the outcome $f_{\mathbf{w}_{\boldsymbol{\omega}}}(\check{\mathbf{x}})$ to become close to the user-defined target score s, while the second term encourages the distance (e.g., ℓ_2 distance) between the factual instance $\mathbf{x}$ and the recourse $\check{\mathbf{x}}_{\boldsymbol{\omega},\mathbf{x}} := \mathbf{x} + \boldsymbol{\delta}_{\boldsymbol{\omega},\mathbf{x}}$ to be low. When it is clear from the context, we will drop the dependence of $\check{\mathbf{x}}$ on the data weights $\boldsymbol{\omega}$ and the factual input $\mathbf{x}$ in the following Chapters for the purpose of better readability. Finally, the set of constraints $\mathcal{A}_d$ ensures that only admissible changes are made to the factual $\mathbf{x}$. For example, changes to an individual's protected attributes such as *sex* or *race* should not be allowed. This is to make sure that the recourses are actionable and that an end user can realistically act upon the suggestions.

2.3.2 Objects of Study

Algorithmic problems. Here we use typical counterfactuals generated by the workhorse recourse model from the previous Section to develop an intuition for the *algorithmic* issues that arise when the goal is to generate reliable recourse through counterfactual explanations. It is well known that approaches that primarily rely on the generation of low cost recourse [166] produce counterfactuals $\check{\mathbf{x}}$ that typically lie extremely close to the decision boundary which we have depicted in Figures 2.1a - 2.1c.

Under this basic model from (2.2), the following concrete problems can occur



(a) Generating realistic recourses (see Contributions 1 and 2).

(b) Recourse robustness deteriorates in the presence of noisy responses (see Contribution 5).

(c) Recourse robustness deteriorates in the presence of deletion requests (see Contribution 4).

Figure 2.1: **Illustrating the fundamental problems for the generation of reliable algorithmic recourse**. (a): The closest point (e.g., measured in ℓ_2 norm) on the opposite side of the decision boundary has no data support. This scenario shows a typical recourse output by approaches such as Wachter et al. [166] which neglect whether counterfactual points are atypical. Methods tackling this problem are described in Sections 3.2 and 3.3 of Chapter 3. (b): The shaded area corresponds to the regions of recourse invalidation. The larger the shaded area the less robust is a recourse to noisy responses. This scenario shows the recourse output by approaches such as Wachter et al. [166] that are typically non-robust. Methods tackling this problem are described in Section 5.2 of Chapter 5. (c): This scenario describes the effect of applying "outdated" recourses to a model that was updated due to data deletion requests. When the model parameters are being updated while the prescribed recourses remain unchanged the recourses likely become invalidated. We develop methods which expose this problem and describe them in Section 5.3 of Chapter 5.

since the objective in (2.2) typically encourages counterfactuals to be as close as possible to the input point to keep the recourse costs low while generating a counterfactual on the opposite side of the decision boundary:

(i) **Generating unrealistic recourses**. There is no mechanism that ensures that the so generated counterfactuals are not atypical or, even worse, outliers. This is due to the implicit assumption underlying (2.2) which asserts that each feature can be independently modified without considering dependencies on other features. This assumption is called independently manipulable feature (IMF) assumption.

(ii) **Noisy human responses can invalidate recourses**. There is no mechanism that ensures that the generated counterfactuals will likely lead to the desired outcome (e.g., loan acceptance) if humans nosily respond to the prescribed recourse (i.e., they do not exactly implement $\check{\mathbf{x}}$ in practice but $\check{\mathbf{x}} + \varepsilon$

instead).

(iii) **Data deletion requests can invalidate recourses.** There is no evaluation method that can evaluate the risk of recourse invalidation as a consequence of other users sending requests that their personal data should be deleted from the predictive model.

In Chapter 3, we introduce algorithms that tackle problem (i) by introducing a data model g which approximates the fundamentally unknown data generating process of the input $\mathbf{x}$. This model g ideally captures how likely input points are under g, and in combination with the scoring function f it generates recourses that are likely to occur under this data model. In Chapter 5, we study the problems (ii) and (iii). To address (ii), we develop a novel robustness notion and show how this notion can be leveraged to generate robust algorithmic recourse in the presence of noisy human responses. To address (iii), we will develop effective algorithms to identify the most critical data points, which, when removed from the training set, would lead to a maximum number of recourse invalidations.

Evaluation problems. The various desiderata shown in Section 2.2 have led to the rapid development of numerous recourse algorithms. However, there is no central repository that keeps track of this development; neither exist standardized ways of evaluating algorithmic recourses. In Chapter 6, we take a step towards this goal by developing a benchmarking library called `CARLA`.

3 On the Generation of Realistic Counterfactual Explanations

In this chapter, we present algorithmic approaches for generating realistic counterfactual explanations for algorithmic recourse. We first overview the existing literature on generating feasible algorithmic recourse and place our work in relation to this specific literature and the broader context of explainability literature. The previous section highlighted the importance of various desiderata, including (a) the generation of counterfactuals that are likely to occur and (b) the consideration of data dependency constraints, in the creation of feasible recourses. Building on the workhorse recourse model introduced earlier, we propose two methods for generating realistic counterfactual explanations for algorithmic recourse that address points (a) and (b). Then we discuss potential directions for future research in this area. This Chapter summarizes Contributions 1 and 2.

3.1 Related Work

Our work builds on the vast literature in the field of explainable ML research. Therefore we discuss how algorithmic recourse fits into the broader realm of explainable ML research before we dive into relevant prior works and their connections to this thesis. If the reader is familiar with the explainable ML literature, we suggest to skip to the next subsection.

Generating decision specific model explanations. One usually distinguishes between *counterfactual explanations* (e.g., [77, 79, 115, 159, 166]) (for algorithmic recourse) and *feature attributions*. The former concerns the ability of people, who are negatively affected by model predictions, to obtain desired outcomes from a fixed prediction model. Although this thesis focuses on counterfactual explanations for algorithmic recourse, we will briefly review work on feature attribution techniques. These feature attribution techniques further fan out into *direct* and *indirect* feature attributions. In *indirect* feature attribution techniques, the primary goal is to identify whether the black-box model uses certain attributes even when they are not directly used to train the model [2, 100]. In *direct* attribution techniques, the main goal is to understand how all inputs available to the model are being used to arrive at a certain prediction. A plethora of direct techniques were recently suggested (e.g., [16, 36, 97, 133, 143, 170, 171], among many others). Direct feature attribution techniques can be further divided into *gradient-based* techniques [16, 143, 170] and *sampling-based* techniques [97, 133, 143, 171]. Relative to sampling-based methods, gradient-based methods compute the partial derivatives of the black-box model at an instance of interest. Sampling-based techniques, on the other hand, (often implicitly) approximate this partial derivative using samples [5, 58, 59].

Algorithmic approaches to recourse. Below, we discuss relevant prior works and their connections to this part of the thesis. Several approaches have been proposed in literature to provide recourse to individuals who have been negatively impacted by model predictions. These approaches can be roughly categorized along the following dimensions [163]:

- **Model type:** What is the underlying *type of predictive model*? Are the developed methods based on tree architectures (e.g., [96, 120, 155]), differentiable classifiers (e.g., [128, 161, 166]) or model agnostic (e.g., [89, 115, 128, 131, 137]), which means that they work across all kinds of classifiers.

- **Sparsity:** Do the search algorithms encourage *sparsity*? This criterion makes sure that only a small number of features should be changed [79, 88, 159, 166].

- **Causality:** Are *causal relationships* considered when generating counterfactual explanations [45, 77, 81, 82, 99]?

- **Diversity:** Is the method *diversity* promoting? Will the produced output by a given method be one counterfactual (e.g., [79, 166]) or multiple counterfactuals [106, 137, 159]?

- **Task:** Is the underlying *task* posed as a regression (e.g., [41, 147]) or classification problem (e.g. [115])?

Contributions. To bridge the gap between the strong IMF assumption and the strong causal assumption, the main idea in Contribution 1 is to change the geometry of the intervention space to a lower dimensional latent space, which encodes different latent factors of variation of the underlying data for which latent recourse actions can be found (see Section 3.2). Relative to our work from Contribution 1, Wachter et al. [166] does not use generative models to capture the likelihood of a counterfactual occurring, while Mahajan et al. [99] and Joshi et al. [77] focus on encoding causal constraints. Poyiadzi et al. [128] suggested `FACE`, which uses a shortest path algorithm on graphs to find counterfactual explanations. In contrast, Kanamori et al. [78] use integer programming techniques to generate realistic recourses.

In Contribution 2 (see Section 3.3), we shift the intervention space from the latent space to the input space, while still maintaining the generative model. Keeping the generative model allows to disentangle recourse actions into direct and indirect action, which is akin to the benefits provided by causal recourse approaches [81, 82], without requiring the strong causal assumptions. Our approach is based on the idea of disentangled representation learning, which aims to identify and separate out the independent factors that contribute to the overall variation in the data [22, 139]. This approach has been effective in achieving fairness in machine learning models while maintaining high accuracy, performing local model audits, and generating realistic data [50, 94, 98, 100, 125]. In contrast to the aforementioned works we demonstrate that disentangled representations can also be helpful to generate counterfactual explanations for algorithmic recourse, even in the presence of dependent data, by identifying indirect actions from direct actions.

3.2 Generating recourses with high occurrence probability

Wachter et al. [166] argued that counterfactual explanations should come from a "possible world" that is "close" to the user's starting point. Laugel et al. [90] further refined this idea, referred to as the "close world desideratum", into two measurable criteria: "proximity" and "connectedness". Proximity indicates that counterfactuals should not be isolated outliers, and connectedness measures whether counterfactuals are close to correctly classified observations. Laugel et al. [90] call counterfactuals that fulfill these criteria *faithful*.

We informally say that a counterfactual explanation is *attainable* if it satisfies three conditions: it is a "close" suggestion that is not an isolated outlier, it is similar to correctly classified observations, and it is associated with low recourses costs. Essentially, attainability combines the characteristics of faithful counterfactuals (proximity and connectedness) with the requirement that they are not overly difficult to achieve. To illustrate this concept, consider a client applying for a loan at a bank that uses a recourse algorithm. The algorithm should not make suggestions that are unrealistic for the client, such as those that are not typically observed in the data or not typical for the client's subgroup, or those that are extremely difficult to attain based on a measure of cost of the input features.

3.2.1 The generative model

We propose using a generative model as a means of finding counterfactual explanations that are both proximate and connected to the input data. The main idea is to change the geometry of the intervention space to a lower dimensional latent space, which encodes different factors of variation of the underlying data. The premise is the following: we aim at generating recourses that are likely to occur under specific distributional assumptions; however, the objective in (2.2) makes the IMF assumption, and can generate recourses that are outliers in the worst case. To mitigate this problem, we postulate a data generating process for the inputs $\mathbf{x}$, which will be governed by a generative model; this generative model transforms latent codes $\mathbf{z}$ into inputs $\mathbf{x}$.

To successfully map an input point into the latent space, we will also require an encoder $e_z : \mathbb{R}^d \to \mathbb{R}^k$ which maps the input data to a lower-dimensional representation. In summary, we assume the factual input $\mathbf{x} \in \mathcal{X} = \mathbb{R}^d$ is generated by a generative model $g_z : \mathbb{R}^k \to \mathbb{R}^d$ such that:

$$\mathbf{x} = g_z(\mathbf{z}), \tag{3.1}$$

where $\mathbf{z} \in \mathbb{R}^k$ is the latent code. Recall that we denote the counterfactual explanation in input space by $\check{\mathbf{x}} = \mathbf{x} + \delta_x$. The counterfactual code in latent space is denoted $\check{\mathbf{z}} = \mathbf{z} + \boldsymbol{\delta}_z$. Thus, we have $\check{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}_x = g_z(\check{\mathbf{z}}) = g_z(\mathbf{z} + \boldsymbol{\delta}_z)$.

3.2.2 The objective function

Given the data generating process from (3.1), we can now rewrite the recourse problem from (2.2) to faithfully capture data dependencies using the generative model g_z:

$$\boldsymbol{\delta}_z \in \underset{\boldsymbol{\delta}_z, \check{\mathbf{x}}' \in \mathcal{A}_d}{\arg\min} \, (f(\check{\mathbf{x}}') - s)^2 + \lambda \cdot c(\mathbf{x}, \check{\mathbf{x}}') \;\; \text{s.t.} \;\; \check{\mathbf{x}}' = g_z(\mathbf{z} + \boldsymbol{\delta}_z), \tag{3.2}$$

where $\lambda \geq 0$ is a scalar tradeoff parameter and s denotes the target score in logit space. Again, the first term in the objective on the right-hand-side of equation (3.2) encourages the outcome $f(\check{\mathbf{x}})$ to become close to the user-defined target score s, while the second term encourages the distance between the factual instance $\mathbf{x}$ and the recourse $\check{\mathbf{x}}$ to be low. The primary difference relative to (2.2) is that the search for counterfactual explanations is conducted in latent space as opposed to the input space. Finally, note that the counterfactual explanation in input space is then given by:

$$\check{\mathbf{x}}^* = g_z(\mathbf{z} + \boldsymbol{\delta}_z) \text{ where } \mathbf{z} = e_z(\mathbf{x}). \tag{3.3}$$

The problem in (3.2) is an abstraction from how the problem is solved in practice: we first train a type of autoencoder model (described in the next Section), and then use the model's trained decoder as a deterministic function g_z to find counterfactual explanations. In fact, we encode an input $\mathbf{x}$ into its corresponding dense representation, $\mathbf{z}$, which serves as the starting point for our stochastic counterfactual search (see Algorithm 1 in Contribution 1). The representation is then stochastically perturbed, $\mathbf{z} + \boldsymbol{\delta}_z$, and the perturbed representation is fed through the model's decoder g_z to generate a potential counterfactual. The classifier is then used to determine whether the prediction was changed. We illustrate the search procedure in Figure 3.1b where we have abstracted away the classifier.

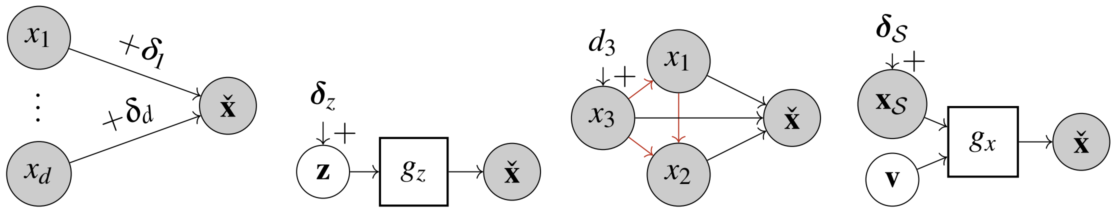
3.2.3 Training the generative model

We propose using (Heterogeneous) Variational Autoencoders (HVAE) [85, 108] as generative models within our recourse framework to approximate the data density of counterfactual explanations for algorithmic recourse. The simple VAE is typically accompanied by an isotropic Gaussian prior $p(z) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. The goal is to optimize the Evidence Lower Bound (ELBO) objective, which is given by:

$$\mathbf{L}_{\text{VAE}}(p, q) = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z})] - \text{D}_{\text{KL}}[q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z})]. \tag{3.4}$$

This objective lower bounds the log of the evidence, $\log p(\mathbf{x})$. In this model, the decoder g_z and the encoder e_z are chosen to be Gaussian likelihood functions with distributional parameters that are estimated by neural networks.

For our purpose in Contribution 1, we use a heterogeneous variational autoencoder designed for modelling tabular data [108]. For this autoencoder, the decoder is factorized as a composition of various likelihood functions; usually one likelihood function per input which enables modelling tabular data that usually contains a variety of feature modalities such as real-valued, positive real valued,



(a) Wachter et al. [166]: IMF recourse (b) Contribution 1: C-CCHVAE (c) Karimi et al. [82]: MINT (d) Contribution 2: DEAR

Figure 3.1: **The spectrum of recourse frameworks from Contribution 2**. Illustrating the different assumptions underlying each recourse framework. (a): Recourses are found by neglecting input dependencies (e.g., [166]). (b): Actions made to the latent code $\mathbf{z}$ generate recourse using a generative model h and neglect control over feature costs (e.g., [115]). (c): Recourses are found by having the decision maker come up with a causal model between the input features (illustrated by the red directed edges) (e.g., [82]). (d): Our framework bridges this gap by allowing a generative model g to be influenced by a subset of inputs $\mathbf{x}_{\mathcal{S}}$. This enables (i) generation of counterfactuals in dense regions of the input space, and (ii) modeling of feature dependencies (iii) without the burden to specify complex causal models.

count, categorical and ordinal distributed features at the same time. Additionally, the modelling framework lets us specify a variety of interval constraints by choosing likelihoods appropriately (e.g., truncated normal distribution or Beta distribution for interval data). This can be useful in order to ensure that a feature such a wage income only takes positive values.

3.3 Generating recourse under data dependency

The recourse model proposed in the previous section has two limitations since it uses a generative model of the form $\mathbf{x} = g_z(\mathbf{z})$ to determine recourses: For one, an important desideratum is to allow for an interface where end-users can insert their search preferences over attributes they wish to take actions on [82, 137, 159, 166]. This is crucial since individuals often know best which feature can realistically be changed given their circumstances [20]. Second, recourse is determined by finding latent actions δ_z, but these actions carry little direct meaning on tabular data or are subject to personal interpretation, which likely varies across decision makers.

3.3.1 The generative model

On a high level, our goal is to design a recourse model which separates the latent code of a generative model into 1) observable features $\mathbf{x}_{\mathcal{S}}$ – that we wish to perform direct recourse actions on – and 2) latent space features $\mathbf{v}$ that have been trained to become disentangled of the observable features. A direct recourse action then ideally has two effects: a direct effect on the input features that have to be changed, and an indirect effect on other, dependent features. The strength of the indirect effect is then determined by the generative model (see Figure 1 in Contribution 2).

To formalize this intuition, we make an adjustment to the generative model from (3.1) and let the input $\mathbf{x}$ be instead generated by the following model:

$$\mathbf{x} = g_{\mathbf{x}}(\mathbf{v}, \mathbf{x}_{\mathcal{S}}) = [\mathbf{x}_{\mathcal{S}}, \mathbf{x}_{\mathcal{S}^c}], \tag{3.5}$$

where $g_{\mathbf{x}} : \mathbb{R}^k \to \mathbb{R}^d$, $\mathbf{v} \in \mathbb{R}^{k-|\mathcal{S}|}$ refers to the latent code where the number of features in $\mathcal{S}$ is smaller than k, and $\mathbf{x}_{\mathcal{S}}$ corresponds to a subset of the input features where $\mathcal{S} \subset \{1, \ldots, d\}$, and the complement set is $\mathcal{S}^c = \{1, \ldots, d\} \setminus \mathcal{S}$. Finally, to successfully map an input point into the latent space as in the previous section, we will also require an encoder $e_{\mathbf{x}} : \mathbb{R}^d \to \mathbb{R}^k$ which maps the input data $\mathbf{x}$ to the lower-dimensional representation $\mathbf{v}$.

3.3.2 The objective function

Given the data generating process from (3.5), we can now rewrite the recourse problem from (2.2) to faithfully capture data dependencies using the generative model $g_{\mathbf{x}}$ while allowing for an interface to directly conduct recourse actions on interpretable input features:

$$\boldsymbol{\delta}_{\mathcal{S}} \in \underset{\boldsymbol{\delta}'_{\mathcal{S}},\, \check{\mathbf{x}}' \in \mathcal{A}_d}{\arg\min} \, (f(\check{\mathbf{x}}') - s)^2 + \lambda \cdot c(\mathbf{x}, \check{\mathbf{x}}') \; s.t. \; \check{\mathbf{x}}' = g_{\mathbf{x}}(\mathbf{v}, \mathbf{x}_{\mathcal{S}} + \boldsymbol{\delta}'_{\mathcal{S}}), \tag{3.6}$$

where $\lambda \geq 0$ is a scalar tradeoff parameter and s denotes the target score in logit space. Again, the first term in the objective on the right-hand-side of Equation (3.6) encourages the outcome $f(\check{\mathbf{x}})$ to become close to the user-defined target score s, while the second term encourages the distance between the factual instance $\mathbf{x}$ and the recourse $\check{\mathbf{x}}$ to be low. The primary difference relative to (3.2) is that the search for counterfactual explanations is now conducted in input space as opposed to latent space.

Our reformulation has several advantages compared to existing recourse methods from the literature: i) relative to the approach presented in the previous Section the actions are applied to input space variables as opposed to latent space variables, and thus they are inherently interpretable; ii) relative to the approach presented in the previous Section and recourse methods which use the IMF assumption (e.g., Section 2.3.1), we can clearly separate the direct effect, which $\boldsymbol{\delta}_{\mathcal{S}}$ has on $\check{\mathbf{x}}$ via $\mathbf{x}_{\mathcal{S}}$, from its indirect effect, which $\boldsymbol{\delta}_{\mathcal{S}}$ has on $\check{\mathbf{x}}$ determined by the generative model when it is dependent of $\mathbf{x}_{\mathcal{S}^c}$ (Proposition 1 in Contribution 2); iii) relative to causal recourse methods (e.g., [81, 82]), we neither assumed causal graphical models nor did we assume structural equation models to incorporate input dependencies. Finally, note that the counterfactual explanation in input space is then given by:

$$\check{\mathbf{x}}^* = g_{\mathbf{x}}(\mathbf{v}, \mathbf{x}_{\mathcal{S}} + \boldsymbol{\delta}_{\mathcal{S}}) \text{ where } \mathbf{v} = e_{\mathbf{x}}(\mathbf{x}). \tag{3.7}$$

3.3.3 Training the generative model

Motivated by the insights from Proposition 1 (see Contribution 2), we train autoencoder models that encourage disentanglement of $\mathbf{x}_{\mathcal{S}}$ and $\mathbf{v}$ in order to keep the *entanglement costs* low. To do that we have to train a generative model, in which $\mathbf{x}_{\mathcal{S}}$ is encouraged to be independent of the latent variable $\mathbf{v}$, while providing high-quality reconstruction of the input $\mathbf{x}$. Thus, the training loss for the generative model consists of two components. First, it consists of both an encoder network e_x and decoder network g_x, for which the reconstruction loss,

$$\mathcal{L}_R(g_x, e_x; \mathbf{x}_{\mathcal{S}}) = \|g_x(e_x(\mathbf{x}), \mathbf{x}_{\mathcal{S}}) - \mathbf{x}\|_2^2, \tag{3.8}$$

guides both networks towards a good reconstruction of $\mathbf{x}$. Second, we want to drive the *entanglement costs* to 0, for which we need the decoder g_x to be disentangled with respect to the latent space, i.e., each component of $\mathbf{z} = [\mathbf{v}, \mathbf{x}_{\mathcal{S}}]$ should ideally control a single factor of variation in the output of g_x. To formalize this intuition, recall that $g(\mathbf{x}_{\mathcal{S}}, \mathbf{v}) = \mathbf{x} \in \mathbb{R}^d$, where each output $g_j = x_j$ for $1 \leq j \leq d$ has its own $|\mathbf{x}_{\mathcal{S}}| \times |\mathbf{v}|$ Hessian matrix $\mathbf{H}^{(j)}$. We refer to the collections of the d Hessian matrices as $\mathbf{H}$. Thus, the second loss we seek to minimize is given by:

$$\mathcal{L}_H(g_x; \mathbf{x}_{\mathcal{S}}) = \sum_{j=1}^{d} \sum_{k=1}^{|\mathbf{v}|} \sum_{l=1}^{|\mathbf{x}|} H_{kl}^{(j)}, \tag{3.9}$$

which is also known as the Hessian penalty [125]. We illustrate the intuition of this objective on the j-th output x_j: we regularize the Hessian matrix $\mathbf{H}^{(j)} = \frac{\partial}{\partial \mathbf{v}} \frac{\partial g_j}{\partial \mathbf{x}_{\mathcal{S}}}$ and encourage its off-diagonal terms to become 0. Driving the off-diagonal terms to 0 implies that $\frac{\partial g_j}{\partial \mathbf{x}_{\mathcal{S}}}$ is not a function of $\mathbf{v}$ and thus $\mathbf{v}$ plays no role for the output of g_j when searching for minimum cost actions using $\mathbf{x}_{\mathcal{S}}$.

3.4 Discussion

In this Chapter, we considered the problem of generating algorithmic recourse in the presence of feature dependencies – a problem previously only studied through the lens of causality. We developed two methods called C-CHVAE and DEAR which used the generative models to capture some of the main practical desiderata: (i) recourses should adhere to feature dependencies without the reliance on hand-crafted causal graphical models and (ii) recourses should lie in dense regions of the feature space, while providing (iii) low recourse costs.

Some of the limitation of the approach in Contribution 1 could already be alleviated by Contribution 2. One main limitation of both approaches is that the autoencoders' reconstructions are usually not perfect; this introduces additional recourse costs and inhibits the sparsity of the identified solutions. Thus, directly operating on the input space without using a generative model is usually more effective at keeping recourse costs down.

In Contribution 2, every recourse action for a *fixed set of features* requires training a separate generative model; thus, the approach is ineffective in scaling to large search spaces. To improve upon this limitation, two different approaches come to mind: First, instead of using one autoencoder per feature set $\mathcal{S}$ one can introduce a generative model that captures multiple conditionals by using autoregressive generative models [30]. Second, if one wanted to get rid of the generative model altogether while still making sure to capture feature dependencies when searching for counterfactual explanations, this could also be achieved by adjusting the cost penalty for feature dependencies. For example, one could solve the following optimization problem:

$$\boldsymbol{\delta}_x \in \arg\min_{\boldsymbol{\delta}' \in \mathcal{A}_d} (f(\mathbf{x} + \boldsymbol{\delta}') - s)^2 + \lambda \cdot \boldsymbol{\delta}'^{\top} \hat{\Sigma}^{-1} \boldsymbol{\delta}', \tag{3.10}$$

where $\hat{\Sigma}$ is the data's positive-definite correlation matrix. Again, the first term encourages the outcome $f(\check{\mathbf{x}})$ to become close to the user-defined target score s, while the second term encourages the Mahalanobis-distance between the factual instance $\mathbf{x}$ and the recourse $\check{\mathbf{x}}$ to be low. To keep costs low, the Mahalanobis-distance would encourage to change features jointly which are conditionally dependent; i.e., recourse actions on dependent features are encouraged to be executed jointly, as desired. To see this, recall that the precision matrix $\hat{\Sigma}^{-1}$ encodes conditional independencies between features, if the data generating process is multivariate Gaussian [56, 91, 101] – generalizations to arbitrary data distributions also exist and allow to capture more general conditional dependency structures (see [52, 91]). Compared to our formalization from (3.6), the downside of this approach is that $\hat{\Sigma}^{-1}$ remains fixed across all inputs, and thus only allows to capture *global data dependencies*.

4 The Connections Between Algorithmic Recourse & Adversarial Attacks

In this chapter, we investigate the relationship between counterfactual explanations for algorithmic recourse and adversarial attacks that are created to trick a classifier. We provide a summary of related research and connect it to Contribution 3. From a theoretical perspective, we examine the similarities and differences between popular adversarial attack and counterfactual explanation algorithms, and consider the implications of our findings for designing counterfactual explanations for algorithmic recourse. This chapter summarizes Contribution 3.

4.1 Related Work

Since this chapter lies at the intersection of counterfactual explanations and adversarial examples in machine learning we discuss related work for each of these topics and their connection to our research below.

Adversarial examples. Adversarial examples are artificially constructed input instances that have been modified in a way that forces a machine learning model to produce an output desired by an adversary. The process of generating these examples is called an adversarial attack [25, 65, 150]. There are various types of adversarial attacks that have been proposed in recent literature, which vary depending on the level of knowledge or access to the model, training data, and optimization techniques. While gradient-based methods [65, 87, 105] are commonly used to find the smallest perturbations to generate adversarial examples, other methods have been proposed to generate adversarial examples in non-differentiable and non-decomposable domains such as speech recognition and image segmentation [38]. To get a more detailed understanding of adversarial examples, readers can refer to a well-established survey [9].

Counterfactual explanations. Counterfactual explanation methods aim to explain a model's predictions by identifying the minimal changes that need to be made to an input instance to change the original prediction (e.g., [160, 166]). These methods can be classified [80, 163] based on their access to the model or its gradients, the level of sparsity in the generated explanation, and whether the explanations are constrained to the data distribution. For instance, Wachter et al. [166] proposed a gradient-based method to find counterfactual explanations by minimizing a distance-based penalty. Additionally, many methods impose constraints on attributes such as *race*, *age*, and *gender* to ensure that the generated explanations are realistic and actionable by users. Manifold-based constraints are also commonly used to ensure that the explanations are consistent with the data distribution (e.g., see Contributions 1 and 2). More recently, there are also causal-based methods proposed for generating counterfactual explanations that adhere to causal constraints [82]. For more detailed literature overviews on counterfactual explanations, refer to Sections 3.1 and 5.1.

Contribution. Previous research has established connections between adversarial examples and counterfactual explanations [31, 55]. While Freiesleben [55] highlights the conceptual differences in goals, formulation, and use-cases, suggesting that counterfactual explanations encompass a broader category of examples, of which adversarial examples are a subset; Browne and Swift [31] examine the differences in the semantics of hidden layer representations in deep neural networks. In contrast, in Contribution 3, our objective is to systematically investigate and compare these two fields, using a theoretical and empirical approach.

4.2 Exploring the Connections between Adversarial Examples and Counterfactual Explanations

Counterfactual explanations. Counterfactual explanations provide recourses by identifying which attributes to change for reversing a models' adverse outcome. As we have discussed in Chapter 2 in more detail, methods designed to output counterfactual explanations find a counterfactual $\check{\mathbf{x}}$ that is "closest" to the original instance $\mathbf{x}$ and changes the model's class prediction $h(\check{\mathbf{x}})$ to the desired label.

Adversarial attacks. Similar to counterfactual explanation methods, most methods generating adversarial examples also solve a constrained optimization problem to find perturbations in the input manifold that cause models to misclassify. These methods are broadly categorized into *poisoning* (e.g., Shafahi et al. [141]) and *exploratory* (e.g., Goodfellow et al. [65]) methods. While *poisoning* methods attack the model during training and attempt to learn, influence, and corrupt the underlying training data or model, *exploratory* methods do not tamper with the underlying model but instead generate specific examples that cause the model to produce the desired output. Like counterfactual explanation methods, evasion methods also use gradient-based optimization to generate adversarial examples.

For a given input $\mathbf{x}$ and classifier h, Carlini and Wagner [32] formulate the problem of finding an adversarial example $\mathbf{x}'=\mathbf{x}+\delta$ such that $h(\mathbf{x}') \neq h(\mathbf{x})$ as:

$$\arg\min_{\mathbf{x}'} \gamma \cdot \ell(\mathbf{x}') + c(\mathbf{x}, \mathbf{x}') \;\text{ s.t. }\; \mathbf{x}' \in [0,1]^d, \tag{4.1}$$

where γ is a hyperparameter and $\ell(\cdot)$ is a loss function such that $h(\mathbf{x}')=y$ if and only if $\ell(\mathbf{x}') \leq 0$. The constraint $\mathbf{x}' \in [0,1]^d$ is applied so that the resulting $\mathbf{x}'$ is within a given data range.

Connections. In Contribution 3, we establish theoretical connections between counterfactual explanation and adversarial attack methods by examining similarities in their objective functions and optimization procedures. Specifically, we compare different methods such as `SCFE` [166], `C&W` [32], `DeepFool` [105], `C-CHVAE` [115], and `NAE` [172] based on their similarity in objectives and constraints imposed during the optimization process – we refer to Figure 4.1 for a visual illustration of the differences in outputs by these models. Our focus will lie on linear models as they provide a foundation for understanding the behavior of nonlinear models through local linearization [58, 69, 135, 160]. We give a more detailed comparison of these methods for specific loss functions and solutions based on classification models in Contribution 3; here we will briefly summarize our approach to theoretically compare the results output by counterfactual explanation and adversarial attack algorithms using a concrete example.

Two popular gradient-based methods for generating adversarial and counterfactual samples are the Carlini and Wagner (`C&W`) attack and score based counterfactual explanations (`SCFE`), respectively. For these two approaches, we first derive closed-form solutions for the minimum perturbation required by `C&W` and `SCFE` to generate adversarial examples and counterfactuals. We then leverage these solutions to derive upper bounds; in particular, using the loss function recommended by Carlini and Wagner [32], we derive an upper bound for the distance between the counterfactuals and adversarial examples generated using `SCFE` and `C&W`. Using this general approach, we derive the following key takeaways form our theoretical analysis:

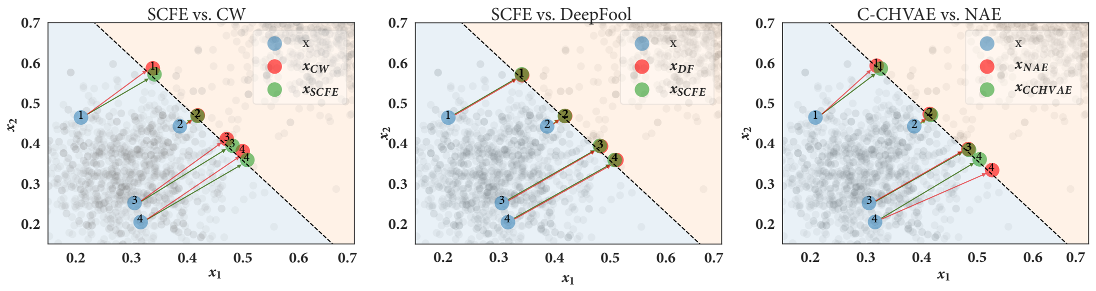


Figure 4.1: **Similarity comparison of adversarial example and counterfactual explanation methods from Contribution 3.** Based on synthetic data, we generate adversarial examples (in red) and counterfactual explanations (in green) for some randomly chosen test set points (in blue). (**Left**) Both `SCFE` (in green) and `C&W` (in red) samples are close to each other, indicating strong similarity between these methods. (**Middle**) `SCFE` (in green) and `DeepFool` (in red) samples exactly coincide, indicating equivalence. (**Right**) `C-CCHVAE` (in green) and `NAE` (in red) samples are closer if the blue factual points are closer to the boundary.

- The upper bounds are smaller when the original score $f(\mathbf{x})$ is close to the target score s, suggesting that `SCFE` and `C&W` are more similar when $\mathbf{x}$ is closer to the decision boundary.
- Intuitively, the adversarial example and counterfactual explanation generated by the methods are bounded depending on the data manifold properties (captured by the Lipschitzness of the generative model) and the radius hyperparameters used by the respective search algorithms (`C-CHVAE` and `NAE`).

In the next Section, we will discuss what these findings mean for the design of more reliable counterfactual explanations for algorithmic recourse.

4.3 Discussion

In this chapter, we examined the similarities between state-of-the-art methods for adversarial attacks and counterfactual explanations. In contribution 3, we compared the objective functions, optimization algorithms and constraints used in these methods to analyze the conditions for equivalence between counterfactual explanations and adversarial attacks. Using linear models, we demonstrated how we can bound the distance between the solutions obtained by the methods proposed by Carlini and Wagner [32] and Wachter et al. [166] using the loss functions preferred in their respective works. This helps to provide a theoretical foundation for understanding the relationship between these two fields of research.

Our research raises important questions about the design and development of counterfactual explanation algorithms by demonstrating, through both theoretical and empirical evidence, that several commonly used algorithms produce results that are similar to those produced by well-known adversarial attack algorithms:

(i) Is it desirable for counterfactual explanations to closely resemble adversarial examples, as indicated in Contribution 3?

(ii) How can a decision maker differentiate between adversarial attacks and counterfactual explanations?

(iii) Does this mean that decision makers are fooling their own models by using counterfactual explanations similar to those presented by Wachter et al. [166]?

(iv) How can counterfactual explanations be designed to be less similar to adversarial attacks?

Moreover, by establishing connections between popular counterfactual explanation and adversarial example algorithms, our work opens up the possibility of using insights from adversarial robustness literature to improve the design and development of counterfactual explanation algorithms. Motivated by the results of this work, in the next Chapter, we develop techniques to evaluate how reliable counterfactual explanations for algorithmic recourse are in the presence of data deletion request which may lead to small updates of the model parameters. Additionally, we will also develop a technique aimed to generate more robust counterfactual explanations for algorithmic recourse.

5 On the Generation of Robust Counterfactual Explanations

In this chapter, we present algorithmic approaches for assessing the robustness of counterfactual explanations to data deletion requests and the required recourse recalibrations (Contribution 4). Additionally, we discuss the generation of robust counterfactual explanations for algorithmic recourse (Contribution 5). To do that, we first overview the existing literature on the generation of robust algorithmic recourse and place our work in relation to this specific literature and the broader context of explainability literature. Chapter 2 highlighted the importance of various desiderata, including the generation of robust counterfactuals. Building on the workhorse recourse model introduced there, this Chapter introduces one method for generating robust counterfactual explanations for algorithmic recourse and another method for assessing recourse robustness in the presence of deletion requests. Finally, we discuss potential directions for future research in this area. This Chapter summarizes Contributions 4 and 5.

5.1 Related Work

Fragility of decision specific model explanations. For the class of *gradient based* techniques, recent work shows that a small change to the original instance can alter neural network (NN) explanations while keeping the network output intact [43, 44, 61]. A slightly different line of work demonstrates empirically that an adversary can train a NN model that arbitrarily controls model explanations without changing the original input [72, 152]. Based on this observation, Anders et al. [12] prove that an adversary can arbitrarily manipulate NN model explanations. For the class of *sampling based* techniques, Slack et al. [144] suggest an algorithmic scaffolding procedure which allows an adversary to construct a classifier whose explanations are manipulated by the adversary. Other related lines of work devise devise manipulations of statistical fairness notions [6, 8, 57], or study the robustness of `LIME` explanations to hyperparameter choices [17, 58].

Fragility of algorithmic recourse. Prior works have focused on determining the extent to which recourses remain robust to the choice of the underlying model [27, 116], shifts or changes in the underlying models [132, 158], or small perturbations to the input instances [15, 45, 146]. To address these problems, these works have primarily proposed adversarial minmax objectives to minimize the worst-case loss over a plausible set of instance perturbations for linear models to generate robust recourses [45, 158], which are known to generate overly costly recourse suggestions. Pawelczyk et al. [116] provided an analysis of the extra cost associated with algorithmic recourse under *model multiplicity*. For this setting, Black et al. [27] suggested a sampling procedure to find recourses that can handle model multiplicity. Rawal et al. [132], on the other hand, demonstrated theoretically and empirically that recourses generated by state-of-the-art approaches become invalid when the underlying model is updated. Artelt et al. [15], Dominguez-Olmedo et al. [45] consider the setting in which the input instance for which recourse is being computed may itself be noisy. Both works derive upper bounds on the recourse costs, while the latter work focuses on the causal recourse setting. More recently, Slack et al. [146] demonstrate how adversaries can manipulate the recourse generation process by designing an attack to generate fundamentally different recourses based on slightly different initial conditions. One notable exception from these works is by Karimi et al. [81], where the authors consider uncertainty with respect to the choice of the structural causal model, which has to be taken into account for reliable recourses. We also refer to Mishra et al. [104] for a brief survey on that topic.

Contributions. In Contribution 4, we address the problem of the fragility of algorithmic recourses in the presence of data deletion requests, which has not been previously studied. To reveal this fragility, we propose algorithms to identify the minimal subset of critical training points to delete in order to maximize the fraction of invalidated recourses when the model needs to be updated. While previous research in data deletion has primarily focused on strategies for effectively removing data from predictive models [34, 62, 63, 64, 75, 167], there has been no examination of how data deletion requests affect the output of state-of-the-art recourse methods. Our work is the first to address these issues and sets the stage for recourse providers to evaluate and reconsider their recourse strategies in the context of the right to be forgotten.

In Contribution 5, we present a user-driven framework for dealing with the trade-off between the cost of recourse and its robustness to noise in human

Reference	Assumption	Source of Issue	Insight	Method	DGP
[116]	Classifier choice is uncertain	Decision maker	Classifier multiplicity makes CEs fragile	TA	NC
[81]	SCM is uncertain	Decision maker	SCM uncertainity cause fragility	Infer SCM	C
[132]	Shift in data distribution	Exogenous	Data shifts cause fragility	TA	NC
[158]	Shift in data distribution	Exogenous	Data shifts make CEs fragile	Adversarial objective	NC
[27]	Classifier choice is uncertain	Decision maker	Classifier multiplicity makes CEs fragile	Neighbor Search	NC
[45]	Uncertainty in recourse process	End-user	Response inaccuracies make CEs fragile	Adversarial heuristic	C
Cont. 4	Model update after deletion request	Users	Deletions make CEs fragile	Maximize IR	NC
Cont. 5	'Precision landing' on prescribed recourse	End-user	Response inaccuracies make CEs fragile	Minimize IR	NC

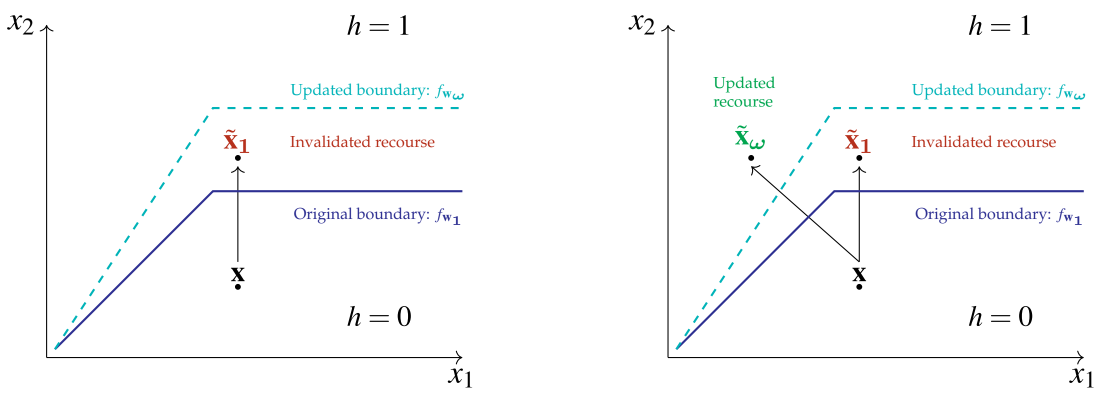
Table 5.1: **Comparison of approaches for generating and analyzing robust algorithmic recourse**. We compare our contributions across a variety of dimensions: what are the underlying assumptions, where is the source of the issue, what are the high level insights, what are the suggested methods (TA: theoretical analysis), to which models are the methods applicable and assumptions regarding the data generating process (DGP) are made; C: causal, SCM: structural causal model, NC: non-causal.

implementations of prescribed recourses. We introduce a novel probabilistic recourse approach that allows users to choose the probability of invalidation for a recourse when it is implemented under uncertainty with some noise. We also propose algorithms that can be used effectively with both linear and non-linear models (such as deep neural networks and tree-based models), resulting in more favorable cost/invalidation rate trade-offs compared to previous methods [45, 158].

5.2 Assessing Robustness of Algorithmic Recourse under Data Deletion

Laws such as the GDPR [60] and CCPA [111] aim to protect users by regulating the usage of personal data and ML model deployments. These laws also grant users greater control over their personal data, including the right to withdraw consent for the use of their data at any time [24]. Therefore, it is important for technology platforms to consider these regulations when training ML models on personal user data, as the continued use of models relying on deleted data instances could potentially be considered illegal [164].

In the following, we will explore the effects of data deletion on model explanations, specifically examining how it impacts recourse algorithms. We aim to understand the limitations of recourse methods when data instances may need to be removed from trained machine learning models. More concretely, we study how data deletion requests can affect whether a prescribed recourse can be successfully implemented by unsuspecting users. This is a particularly important


x_2
$h=1$
Updated boundary: $f_{\mathbf{w}_\omega}$
$\tilde{\mathbf{x}}_1$
Invalidated recourse
Original boundary: $f_{\mathbf{w}_1}$
$\mathbf{x}$
$h=0$
x_1
x_2
$h=1$
Updated recourse
$\tilde{\mathbf{x}}_\omega$
$\tilde{\mathbf{x}}_1$
Updated boundary: $f_{\mathbf{w}_\omega}$
Invalidated recourse
Original boundary: $f_{\mathbf{w}_1}$
$\mathbf{x}$
$h=0$
x_1

(a) Recourse Outcome Instability (b) Recourse Action Instability

Figure 5.1: **Pictorial representation of the two key robustness notions from Contribution 4**. In Fig. 5.1a, recourse $\tilde{\mathbf{x}}_1$ for an input $\mathbf{x}$ is invalidated due to a model update. In Fig. 5.1b, recourse is additionally recomputed (i.e., $\tilde{\mathbf{x}}_\omega$) to avoid recourse invalidation.

issue to consider as users may have already started taking costly actions based on previously received recourses.

5.2.1 Robustness Through the Lens of the Right to be Forgotten

We consider the problem of data deletion requests and their impact on algorithmic recourse. In this context, some individuals may choose to withdraw their data, leading to the need for model updates. Throughout this section, the model is updated once a deletion request is issued to minimize the empirical risk on the remaining training data points; while this procedure is an extreme operationalization of the GDPR's "right to be forgotten", it is also consistent with approaches taken in related work (see for example [62]). We define two key terms, *prescribed recourses* and *recourse outcomes*. A prescribed recourse $\check{\mathbf{x}}$ represents a recourse provided to a user by a recourse method (e.g., increasing salary by \$500). The recourse outcome $f(\check{\mathbf{x}})$ is the prediction made by the model when applied to the recourse. Based on these definitions, we propose two definitions of recourse instability.

Definition 1 *(Recourse outcome instability (Contribution 4)) The recourse outcome instability with respect to a factual instance* $\mathbf{x}$*, where at least one data weight is set to* 0*, is defined as follows:*

$$\Delta_{\mathbf{x}}(\omega) = \left| f_{\mathbf{w}_1}(\check{x}_1) - f_{\mathbf{w}_\omega}(\check{x}_1) \right|, \tag{5.1}$$

where $f_{\mathbf{w}_1}(\check{x}_1)$ is the prediction at the prescribed recourse $\check{x}_1$ based on the model that uses the full training set (i.e., $f_{\mathbf{w}_1}$) and $f_{\mathbf{w}_\omega}(\check{x}_1)$ is the prediction at the prescribed recourse for an updated model and data deletion requests have been incorporated into the predictive model (i.e., $f_{\mathbf{w}_\omega}$).

In this definition, we consider how applying "outdated" recourses to an updated model will affect the validity of the recourses. We assume that only the model parameters are being updated, while the recourses remain unchanged. For a model with binary output, Definition 1 captures whether the recourses will become invalid after training data is deleted. We can also apply this definition to a continuous-score model with a target value of s by using a discretized version of the model's predictions. This allows us to determine the invalidation rate of the recourses.

In Definition 2 below, the cost function c is specified to be a p-norm and the recourse is allowed to change due to model parameter updates, consistent with related work (e.g., [166]).

Definition 2 *(Recourse action instability (Contribution 4)) The Recourse action instability with respect to a factual input* $\mathbf{x}$*, where at least one data weight is set to* 0*, is defined as follows:*

$$\Phi_{\mathbf{x}}^{(p)}(\boldsymbol{\omega}) = \left\|\check{x}_1 - \check{x}_\omega\right\|_p, \tag{5.2}$$

where $p \in [1, \infty)$, and $\check{x}_\omega$ is the recourse obtained for the model trained on the data instances that remain present in the data set after the deletion request.

Definition 2 assesses the extent to which prescribed recourses need to be modified to still achieve the desired recourse outcome after data deletion requests (i.e., $\check{\mathbf{x}}_\omega$, see Figure 5.1b). The lowest cost recourse is of particular importance as it is easiest to implement. Therefore, the main goal in the field of recourse is to identify the recourse with the lowest costs. However, we are also interested in how the optimal low-cost recourse changes, even if the outdated recourse would still be valid.

Using the invalidation measures introduced above, the next section discusses how we can use these to identify the set of points which would lead to maximum recourse invalidation according to these two measures.

5.2.2 Identifying the Most Critical Training Points

We aim to identify the minimum number of data deletion requests that will have the greatest impact on the instability of our chosen measure, m, which can be either Δ or $\Phi^{(2)}$. This is done by summing the instability over the entire dataset, e.g., resulting in $\Delta = \sum_{\mathbf{x} \in \mathcal{D}_{test}} \Delta_{\mathbf{x}}$. We then formulate an optimization objective to find the smallest number of deletion requests that leads to a maximum impact on the instability measure m. To operationalize this, we define the set of possible data weight configurations over n data points:

$$\Gamma_\alpha := \{\boldsymbol{\omega} : \text{Maximally } \lfloor \alpha \cdot n \rfloor \text{ entries of } \boldsymbol{\omega} \text{ are 0 and the remainder is 1.}\}. \quad (5.3)$$

In (5.3), the parameter α controls the fraction of instances that are being removed from the training set. For a fixed fraction α, our problem of interest becomes:

$$\boldsymbol{\omega}^* \in \arg\max_{\boldsymbol{\omega} \in \Gamma_\alpha} m(\boldsymbol{\omega}). \quad (5.4)$$

In optimizing our objective, we encounter two main challenges: (i) evaluating $m(\boldsymbol{\omega})$ for various weight configurations $\boldsymbol{\omega}$ can be costly as the objective is defined through the solutions of multiple non-linear optimization problems, such as model fitting and finding recourses. Additionally, (ii) even if $m(\boldsymbol{\omega})$ can be computed rapidly, optimizing this objective can still be NP-hard (see Appendix in Contribution 4).

5.2.3 Comparison with the State-of-the-art

We note that the problem stated in objective (5.4) has previously not been considered in recourse literature before. The closest work to our's is due to Slack et al. [146], who identify small and adversarial perturbations to feature dimensions of the original input to entirely change the counterfactual explanation. Our objective is different as our goal is to identify and remove entire training points adversarially to maximally impact or robustness notions.

We present new methods for addressing the problem of optimizing the objective in equation (5.4). Specifically, we propose greedy and a gradient descent style algorithms that approach the problem from different perspectives. Both algorithms efficiently evaluate $m(\boldsymbol{\omega})$ by either utilizing a closed-form expression that shows the dependency of m on $\boldsymbol{\omega}$ or by applying an approximation of m that can be differentiated with respect to $\boldsymbol{\omega}$. Then we can optimize $m(\boldsymbol{\omega})$ by either using a greedy method (e.g., gradient descent). Together, these methods contribute to the field of recourse by providing new solutions to the problem of computing and optimizing the objective in equation (5.4).

Not only do we present practical algorithms, but we also present theoretical analyses to identify the factors that determine the instability of recourses when users whose data is part of the training set submit deletion requests. In particular, we provide upper bounds for the recourse instability notions defined in the previous section when the underlying models are linear or overparameterized neural networks (see Propositions 1 and 2 of Contribution 4).

In summary, the following are the key novelties of Contribution 4:

(i) **Novel recourse robustness problem:** We introduce the problem of *recourse invalidation under the right to be forgotten* by defining two new recourse instability measures.

(ii) **Novel objective and analysis:** Using our instability measures, we present an optimization framework to identify a small set of critical training data points which, when removed, invalidates most of the issued recourses. Additionally, through theoretical analysis, we identify the factors that determine the instability of recourses when users whose data is part of the training set submit deletion requests.

5.3 Algorithmic Recourse in the Presence of Noisy Human Responses

In previous research, it has been shown that the approach presented in Chapter 2 produces recourses that are low-cost as the corresponding counterfactuals are similar to the original instances [166]. However, these recourses have been found to lack robustness in studies [116, 132]. For example, the recourses generated by this approach may not remain valid (i.e., result in a positive model prediction) if small changes are made to them (see Figure 5.2a). It is also known that recourses are often noisily implemented in real-world settings, as noted in previous research [26]. For instance, an individual who was asked to change their credit card spending habits may end up spending a slightly different amount than what was prescribed. To address this issue, some researchers [45, 158] have proposed using adversarial training in the recourse objective to consider worst-case perturbations of the input data or model parameters (see Figure 5.2c). However, it is well-known that such adversarial objectives can be conservative [130, 156], resulting in recourses that are more costly and harder to implement. The existing approaches generate robust recourses that are often high in cost and therefore difficult to implement, without considering the preferences of individual users. In practice, users may have different preferences for balancing the trade-offs between recourse costs and robustness - for example, some may be willing to accept higher costs for increased robustness to noisy responses, while others may not. In the next Section we present an end-user driven framework for the generation of robust algorithmic recourse.

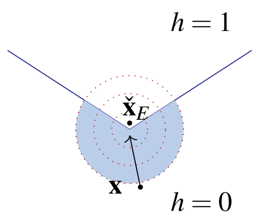
5.3.1 Defining the Recourse Invalidation Rate

In order to enable end users to effectively navigate the trade-offs between recourse costs and robustness, we let them choose the probability with which a prescribed recourse could get invalidated (recourse invalidation rate) if small changes are made to the prescribed recourse, i.e., the prescribed recourse is implemented somewhat noisily. To this end, we formally define the notion of Recourse Invalidation Rate (IR) in this section. We first introduce two key terms, namely, *prescribed recourses* and *implemented recourses*. A prescribed recourse is a recourse that was provided to an end user by some recourse method (e.g., increase salary by \$500). An implemented recourse corresponds to the recourse that the end user finally implemented (e.g., salary increment of \$505) upon being provided with the prescribed recourse. With this basic terminology in place, we now proceed to formally define the Recourse Invalidation Rate (IR) below.

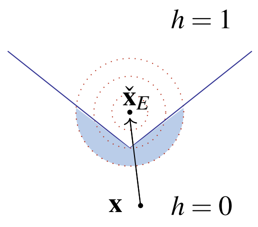
Definition 3 (Recourse Invalidation Rate (Contribution 5)) *For a given classifier* $h : \mathbb{R}^d \rightarrow \{0,1\}$*, the recourse invalidation rate corresponding to the counterfactual* $\check{\mathbf{x}}_E = \mathbf{x} + \boldsymbol{\delta}_E$ *output by a recourse method E is given by:*

$$\Delta(\check{\mathbf{x}}_E) = \mathbb{E}_{\varepsilon}\big[\underbrace{h(\check{\mathbf{x}}_E)}_{\textit{CF class}} - \underbrace{h(\check{\mathbf{x}}_E + \varepsilon)}_{\textit{class after response}}\big]. \tag{5.5}$$

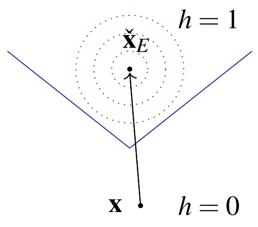
Since the implemented recourses do not typically match the prescribed recourses $\check{\mathbf{x}}_E$ [26], we add ε to model the noise in human responses. As we primarily compute recourses for individuals $\mathbf{x}$ such that $h(\mathbf{x}) = 0$, the label corresponding to the counterfactual is given by $h(\check{\mathbf{x}}_E)$=1 and therefore $\Delta \in [0,1]$. For example, the following cases help understand our recourse invalidation rate better: When Δ=0, then the prescribed recourse and the recourse implemented by the user agree all the time; when Δ=0.5, the prescribed recourse and the implemented recourse agree half of the time, and finally, when Δ=1 then the prescribed recourse and the recourse implemented by the user never agree. To illustrate our ideas, we will use our IR measure with a Gaussian probability distribution (i.e., $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$) to model the noise in human responses.



(a) Low recourse cost and low robustness



(b) Medium recourse cost and medium robustness



(c) High recourse cost and high robustness

Figure 5.2: **Pictorial representation of the recourses output by various state-of-the-art methods and our framework from Contribution 5**. The blue line is the decision boundary, and the shaded areas correspond to the regions of recourse invalidation. Fig. 5.2a shows the recourse output by approaches such as Wachter et al. [166] where both the recourse cost as well as robustness are low. Fig. 5.2c shows the recourse output by approaches such as Dominguez-Olmedo et al. [45] where both the recourse cost and robustness are high. Fig. 5.2b shows the recourse output by our framework `PROBE` in response to user input requesting an intermediate level of recourse robustness.

5.3.2 The Probabilistically Robust Recourse Objective

The core idea is to find a recourse $\check{\mathbf{x}}$ whose prediction at any point y within some set around $\check{\mathbf{x}}$ belongs to the favorable class with probability $1 - r$. Hence, our goal is to devise an algorithm that reliably guides the recourse search towards regions of low invalidation probability while maintaining low cost recourse. For a fixed scoring function $f : \mathbb{R}^d \to \mathbb{R}$ corresponding to the classifier h, our objective reads:

$$\mathcal{L} = R(\mathbf{x}'; \sigma^2 \mathbf{I}) + \lambda_1 \cdot \left(f(\mathbf{x}') - s\right)^2 + \lambda_2 \cdot c(\mathbf{x}', \mathbf{x}), \tag{5.6}$$

where s is the target score for the input $\mathbf{x}$, $R(\mathbf{x}'; r, \sigma^2 \mathbf{I}) = \max(0, \Delta(\mathbf{x}'; \sigma^2 \mathbf{I}) - r)$ with r being the target IR, $\Delta(\mathbf{x}'; \sigma^2 \mathbf{I})$ is the recourse invalidation rate from (5.5), λ_1 and λ_2 are the balance parameters, and c quantifies the cost between the input and the prescribed recourse. To arrive at an output probability of 0.5, the target score for $f(\mathbf{x})$ for a sigmoid function is $s = 0$, where the score corresponds to a 0.5 probability for $y = 1$.

The new component R is a Hinge loss encouraging that the prescribed recourse has a low probability of invalidation, and the parameter σ^2 is the uncertainty magnitude and controls the size of the neighbourhood in which the recourse has to be robust. The middle term encourages the score at the prescribed recourse $f(\check{\mathbf{x}})$ to be close to the target score s, while the last term promotes the distance between the input $\mathbf{x}$ and the recourse $\check{\mathbf{x}}$ to be small.

In practice, the choice of r depends on the risk-aversion of the end-user. If the end-user is not confident about achieving a 'precision landing', then a rather low invalidation target should be chosen (i.e., $r < 0.5$).

5.3.3 Comparison with the State-of-the-art

Prior works by Upadhyay et al. [158] and Dominguez-Olmedo et al. [45] attribute robustness to recourses in different ways. While the former constructs recourses that are robust to small shifts in the underlying model, the latter constructs recourses that are robust to small input perturbations. These approaches adapt the classic minimax objective functions commonly employed in adversarial robustness and robust optimization literature to the setting of algorithmic recourse, and use gradient descent style approaches to optimize these functions. In an attempt to generate recourses that are robust to either small shifts in the model ([158]) or to small input perturbations ([45]), the above approaches find recourses that are farther away from the underlying model's decision boundaries, thereby increasing the distance between the counterfactuals (recourses) and the original instances (i.e., the recourse costs). Higher cost recourses are harder to implement for end users as they are farther away from the original instance vectors (current user profiles). Putting it all together, the aforementioned approaches generate robust recourses that are often high in cost and are therefore harder to implement, without providing end users with any say in the matter.

In contrast, our work puts forth a paradigm shifting idea of enabling users to control the recourse robustness-cost tradeoffs by letting them choose the probability with which a recourse could get invalidated (recourse invalidation rate) if small changes are made to the recourse i.e., the recourse is implemented somewhat noisily. Given this problem formulation, we can no longer use the minimax objectives outlined by prior works as we need to ensure that the resulting recourse invalidation rates match desired invalidation rates input by end users. To this end, we propose a objective (5.6) which simultaneously minimizes the gap between the achieved (resulting) and desired recourse invalidation rates, minimizes recourse costs, and also ensures that the counterfactual (recourse) achieves a positive model prediction.

To optimize the proposed objective, we outline a gradient descent style approach (Algorithm 1, Contribution 5). Note that we need to compute the achieved recourse invalidation rate at each step of the gradient descent algorithm, and computing this empirically can be computationally prohibitive as demonstrated by prior work (e.g., [145]) since it involves generating perturbations of each candidate counterfactual and querying the underlying model for labels of all the perturbations. To this end, we develop theoretical expressions for the recourse invalidation rates (Theorems 1 and 2 in main body and Appendix of Contribution 5) corresponding to any given instance w.r.t. different classes of underlying models (e.g., linear models, non-linear models such as deep neural networks and tree based models), and then use these estimates to efficiently optimize the proposed objective.

Note that the approaches put forth by prior works only handle gradient-based linear models or locally linear model approximations. In contrast, the approaches and theoretical derivation of the invalidation rate we propose (Section 4 and Appendix in Contribution 5) enable us to handle both linear and non-linear models (e.g., logistic regression, neural networks, decision tree models) effectively. Our empirical results (Figure 4 in Contribution 5) also demonstrate that our approach achieves better recourse cost/invalidation rate tradeoffs compared to both Upadhyay et al. [158] and Dominguez-Olmedo et al. [45].

In summary, the following are the key novelties of Contribution 5:

(i) **Novel problem formulation.** We propose a novel problem formulation which enables end users to manage the recourse robustness-cost tradeoffs;

(ii) **Novel objective function.** We introduce a novel objective function along with theoretical insights to effectively optimize this objective for various classes of underlying models (e.g., linear models, non-linear models such as neural networks, single decision trees) to encourage that the resulting invalidation rates match the user preferred invalidation rates.

5.4 Discussion

In Contribution 4, we have demonstrated that the robustness of algorithmic recourse depends on the robustness of the predictive model to data deletion requests. Therefore, deletion robust predictive models will generally lead to more robust recourse. Our work is the first to establish this fundamental link between data point influence and the robustness of counterfactual explanations. As a consequence, our theory also suggests a paradigm shift in the way that the recourse literature thinks about robustness of algorithmic recourse: most of the literature follows the paradigm of developing robust recourse methods taking the predictive model as fixed ([27, 45], Contribution 5). This usually leads to recourses that are more difficult to act upon as they are less parsimonious and have higher recourse costs at the benefit of higher robustness (see Section 5.3). In contrast to this paradigm, our theoretical analysis suggests that one can achieve increased levels of robust recourse by training the models with restricted empirical influence functions. Thus, we expect that our theory can be operationalized with some approaches to minimize influence but which have been proposed in a different context, for instance private or deletion-robust models. In summary, our work builds the bridge between these two fields, which we see as a valuable contribution, while some of our current theoretical results offer room for improvement as they are limited to linear models and neural tangent kernels. Future work, would generalize these results to more general function classes.

In Contribution 5, we have tackled the problem of generating robust recourses while anticipating that humans will react nosily to the prescribed recourses. For tree-based methods our currently suggested algorithm works very well on single, axis-aligned decision tree classifiers. However, there is room for improvement of the algorithm for more complex tree-based classifiers as the current algorithm requires a model distillation step for tree-based ensemble classifiers, which can lead to too coarse approximations of the decision regions, resulting in mismatches between the targeted invalidation rate r and the empirical invalidation rate obtained from the prescribed recourse.

6 Standardizing the Evaluation of Recourse Methods

This chapter presents guidelines and a benchmarking framework for assessing the effectiveness of recourse methods. It gives a summary of relevant past research and connects it to Contribution 6. We also introduce various evaluation metrics and a Python toolkit that facilitates transparent and consistent evaluation of recourse methods. This chapter summarizes Contribution 6.

6.1 Related Work

Benchmarks and evaluation frameworks. There are many different methods for evaluating explanations generated by black-box models [3, 23, 53, 107, 109]. Doshi-Velez and Kim [46] divide these methods into two categories: human-grounded metrics, which rely on human judgment, and functional-grounded metrics, which do not require a human-generated reference point. The latter category often involves changing the most important part of an input and observing the effect on the model's output probability. Examples of this approach include the Sensitivity-n measure by Ancona et al. [11] and the infidelity and max-sensitivity metrics by Yeh et al. [169]. Samek et al. [138] and Petsiuk et al. [127] also propose altering the input pixels according to importance scores, but Hooker et al. [73] show that this perturbation can introduce artifacts and cause a distribution shift, calling into question the validity of these ”no-retraining” approaches. As a consequence, Hooker et al. [73] developed the Remove and Retrain (`ROAR`) framework which addresses the distribution shift issue by including an extensive model retraining step. As a result, we can distinguish between evaluation methods that use a “retrainig” approach, like `ROAR`, and those that use a “no-retraining” approach. `ROAR` has been widely used in recent research

[70, 74, 102, 140, 148], and there are ongoing efforts to develop variations of it [142] and improve upon this evaluation technique [134].

Contribution. While there exist several benchmarks in machine learning literature (e.g., [4, 40, 93]), they focus on evaluating the quality of feature attributions [4, 93] or other key model properties such as adversarial robustness [40]. Therefore, these benchmarks cannot be readily used to evaluate the reliability of counterfactual explanations for algorithmic recourse as it requires an entirely new set of evaluation measures, and experimental set ups. While some prior works have constructed synthetic datasets with ground truth explanations to benchmark feature attribution methods [4, 84, 93], these evaluations are limited to feature attribution techniques and do not apply to recourse methods. To the best of our knowledge, Contribution 6 is the only benchmark that provides a complete pipeline to evaluate and compare counterfactual explanations for algorithmic recourse on many key evaluation measures to promote transparency and collaboration around evaluations of counterfactual explanations.

6.2 Evaluating Recourse Methods

The usefulness of the counterfactual explanations that are provided to individuals is heavily dependent on the approach used to compute the recourse suggestions. Therefore, there is a significant need for a standardized benchmarking platform that allows for transparent and meaningful comparisons between different recourse methods. This is important for researchers, who need to be able to evaluate their proposed methods against the wide range of existing methods, as well as for practitioners, who need to be able to choose the right recourse mechanism for their specific problem. In short, a standardized framework for comparison and quality assurance is an essential requirement.

6.2.1 Unifying Evaluation Standards

We introduce a concise and up-to-date summary of evaluation practices for recourse methods (Contribution 6). Our work helps to compare existing techniques and thus achieve a more comparable and standardized evaluation of available recourse methods. Below, we briefly present and discuss general evaluation strategies. As algorithmic recourse is a multi–modal problem we use a variety of measures to evaluate the methods' performances:

- **Costs.** When answering the question of generating the nearest counterfactual explanation, it is essential to define the distance of the factual $\mathbf{x}$ to the nearest counterfactual $\check{\mathbf{x}}$. The literature has formed a consensus to use either the normalized ℓ_0 or ℓ_1 norm or any convex combination thereof (see for example [79, 106, 116, 131, 159, 166]). The ℓ_0 encourages the number of total required feature changes between factual and counterfactual instance to be small, while the ℓ_1 norm requires the average change to be small while also encouraging to change few features only.

- **Constraint violation.** This measure counts the number of times a recourse method violates user-defined constraints. Depending on the data set, we fixed a list of features which should not be changed by the used method (e.g., *sex*, *age* or *race*).

- **yNN.** This measure evaluates how much data support recourses have from positively classified instances. Ideally, recourses should be close to positively classified individuals which is a desideratum formulated by [90].

- **Redundancy.** We evaluate how many of the proposed feature changes were not necessary. This is a particularly important criterion for methods that generate counterfactual under the IMF assumption. We measure this by *successively* flipping one value of $\check{\mathbf{x}}$ after another back to $\mathbf{x}$, and then we inspect whether the label flipped from 1 back to 0: e.g., we check whether flipping the value for the second dimension would change the counterfactual outcome 1 back to the predicted factual outcome of 0. If the predicted outcome does not change, we increase the redundancy counter, concluding that a sparser counterfactual explanation could have been found. We iterate this process over all dimensions of the input vector.

- **Success rate.** Some generated counterfactual explanations do not alter the predicted label of the instance as anticipated. To keep track how often the generated CE does hold its promise, the success rate shows the fraction of respective models' correctly determined counterfactuals.

- **Average time.** By measuring the average time a CE method needs to generate its result, we evaluate the effectiveness and feasibility for real–time prediction settings. We have included the run time measure to give users a rough estimate on what run times to expect when executing the respective algorithms.

6.2.2 Software Architecture

Here we introduce our open-source benchmarking software. We describe the architecture in more detail and provide examples of different use-cases and their implementation. The purpose of this Python library is to provide a simple and standardized framework to allow users to apply different state-of-the-art recourse methods to arbitrary data sets and black-box-models. `CARLA` also provides an implementation interface to integrate new recourse methods in an easy-to-use way, which allows to compare newly developed methods to already existing methods.

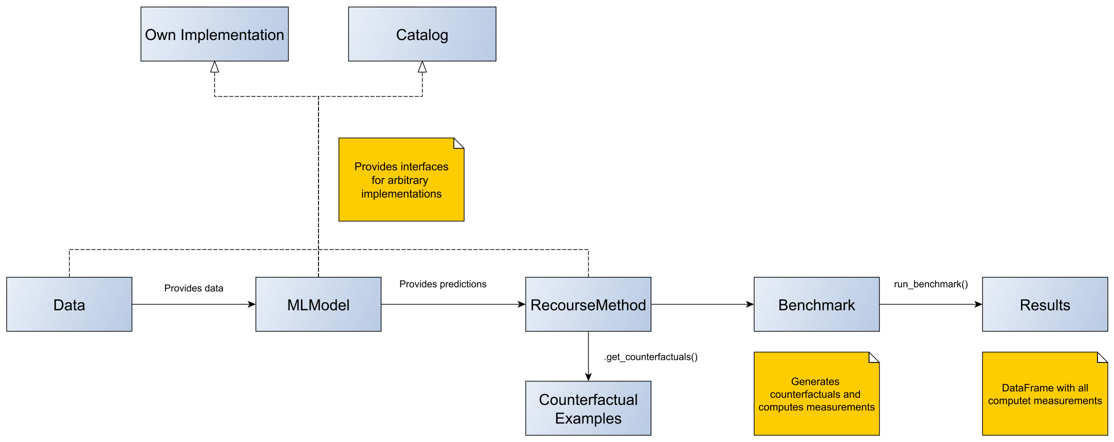


Figure 6.1: **Architecture of our `CARLA` library from Contribution 6**. The silver boxes show the individual objects that will be created to generate counterfactual explanations and evaluate recourse methods. Useful explanations to specific processes are illustrated as yellow notes. The dashed arrows are showing the different implementation possibilities; either use pre-defined catalog objects or provide custom implementation. All dependencies between these objects are visualised by solid arrows with an additional description.

In Figure 6.1, we depict a visualization of the software architecture. For every of the three components `Data`, `MLModel`, and `RecourseMethod` the user can either utilize existing methods from our catalog, or extend the library by adding custom methods and implementations. The components represent an interface to the key parts in the process of generating counterfactual explanations. In particular, `Data` provides a common way to access the data across the software and maintains information about the data's features. `MLModel` wraps each black-box model and stores details on the encoding, scaling and feature order specific to the model. The primary purpose of `RecourseMethod` is to provide a common interface to easily generate counterfactual examples.

```

from carla import RecourseMethod

# Custom recourse implementations need to
# inherit from the RecourseMethod interface
class MyRecourseMethod(RecourseMethod):
    def __init__(self, mlmodel):
        super().__init__(mlmodel)

    # Generate and return encoded and
    # scaled counterfactual examples
    def get_counterfactuals(self, factuals: pd.DataFrame):
    [...]
    return counterfactual_examples

```

Figure 6.2: **Pseudo-implementation of the recourse method wrapper from Contribution 6.**

Besides the option to use pretrained predictive models and preprocessed data, `CARLA` also provides users with an easy way to load and define data sets and model structures independent of their framework (e.g., `Pytorch`, `sklearn`, `Tensorflow`). In the following, we will give an overview and provide example implementations of different use cases:

(i) **Predefined methods:** A common usage of the library is to generate counterfactual explanations. This can be done by loading black-box-models and data sets from our provided catalogs, or by user-defined models and data sets via integration with the defined interfaces. Figure 6.3 shows an implementation example of a simple use-case, applying a recourse method to a predefined data set and model from our catalog. After importing both catalogs, the only necessary step is to describe the data set name (e.g., adult, give me some credit, or compas) and the model type (e.g., ann, or linear) the user wants to load. Every recourse method contains the same properties to generate counterfactual explanations.

(ii) **Benchmarking recourse methods.** Besides the generation of counterfactual explanations, the focus of `CARLA` lies on benchmarking recourse methods. Users are able to compute evaluation measures to make qualitative statements about usability and applicability. All evaluation measures, which have been described in the previous Section, are implemented in the *Benchmarking* class of `CARLA` and can be used for every wrapped recourse method.

(iii) **Exentsibility.** Our libraray also features a *RecourseMethod*-wrapper which

```

from carla import DataCatalog, MLModelCatalog
from carla.recourse_methods import GrowingSpheres

# 1. Load data set from the DataCatalog
data_name = "adult"
dataset = DataCatalog(data_name)

# 2. Load pre-trained black-box model from the MLModelCatalog
model = MLModelCatalog(dataset, "ann")

# 3. Load recourse model with model specific hyperparameters
gs = GrowingSpheres(model)

# 4. Generate counterfactual examples
factuals = dataset.raw.sample(10)
counterfactuals = gs.get_counterfactuals(factuals)

```

Figure 6.3: **Example usage of the data and model catalogs from Contribution 6.**

users to implement their own method to generate counterfactual explanations. This opens up a way of standardized and consistent comparisons between different recourse methods; thereby, advantages and disadvantages of new newly developed methods can be swiftly analysed by benchmarking against the state of the art. In Figure 6.2, we show how an implementation of a custom recourse method can be structured. This aspect of our work is described in more detail in the Appendix of Contribution 6. Additionally, users can either utilize our provided catalogs of data sets, recourse methods and black-box models (Figure 6.3), or they can contribute custom implementations.

6.3 Discussion

In this Chapter, we introduced `CARLA`, a Python library that serves as a competitive benchmarking platform for counterfactual explanation and recourse methods. The library has been designed to enable standardized and transparent comparisons of recourse methods on different data sets across a variety predictive models. In summary, it provides the following contribution:

(i) **Competetive baselines:** Our library provides competitive baselines to compare new methods to existing ones.

(ii) **Common evaluation framework:** Our library serves as a common framework that includes over 10 counterfactual explanation methods and allows for easy integration of new methods. It also includes built-in evaluation measures to compare the performance of different recourse mechanisms across various data sets and models. This makes it a useful tool to benchmark existing counterfactual methods on popular data sets and models.

(iii) **Interface:** `CARLA` supports popular ML frameworks such as `PyTorch` [114], `Tensorflow` [1] or `Sklearn` [124], as well as providing a generic abstraction layer that allows for custom implementations of recourse methods and predictive models. It also allows users to define problem-specific data set characteristics and hyperparameters for the chosen counterfactual explanation method.

Despite the numerous advantages, our library also has limitations: it is currently limited to tabular data modalities where we support categorical features with only two categories – this is to provide a fair comparison between a variety of different recourse methods of which some provide support for categorical features and others do not (e.g., the algorithm by Wachter et al. [166] does not, while the method by Pawelczyk et al. [115] does). Including new evaluation measures and including the most recent recourse algorithms is an avenue for future work.

7 Conclusions and Outlook

In this thesis, we developed techniques and tools to generate more trustworthy counterfactual explanations for consequential decision-making scenarios. By doing so, we expanded the usefulness of recourse systems and made them more suitable for real-world applications. This chapter summarizes our findings, provides final thoughts, and suggests areas for further research

7.1 Summary of Contributions

In Chapter 3, we developed methodology and tools to bridge the gap between methods that generate recourse under the independently manipulable feature assumption (see Chapter 2) and the strong causal assumption. To this end, we introduced generative models to capture the intuition that features may be dependent and should move jointly when perturbed and the data's dependence structure allows for it. To this end, we had to change the geometry of the intervention space to a lower dimensional latent space, which encodes different latent factors of variation of the underlying data for which latent recourse actions were found (see Section 3.2). In Contribution 2 (see Section 3.3), we shifted the intervention space from the latent space back to the input space, while still maintaining the generative model. Keeping the generative model had two advantages: first, it allowed for an interface where end-users could insert their search preferences over attributes they would like to do actions on and second, we could disentangle recourse actions into direct and indirect action. The latter aspect is akin to the benefits provided by causal recourse works [81, 82], without requiring the strong causal assumptions.

In Chapter 4, we investigated the connections between various well-known counterfactual explanation algorithms used for algorithmic recourse and adversarial attack generating algorithms. Our analysis revealed that there are significant similarities in the solutions produced by these methods. These findings prompted us to question the design of current algorithmic recourse algorithms and how to create more robust and reliable recourses that do not resemble adversarial attacks.

Motivated by our findings in Chapter 4, we studied the problem of generating robust recourses 5. We first introduced the problem of *recourse invalidation under the right to be forgotten* and considered various recourse instability notions. Based on our instability notions, we then presented an optimization framework to identify a small set of critical training data points which, when removed, invalidates most of the issued recourses. Additionally, we demonstrated that the robustness of algorithmic recourse depends on the robustness of the predictive model to data deletion requests. Therefore, we expect deletion robust predictive models will generally lead to more robust recourse. In Contribution 5 of the same Chapter, we tackled the problem of generating robust recourses while keeping the underlying model fixed and while anticipating that humans will react nosily to the prescribed recourses. Our suggested solutions enabled users to control the recourse robustness-cost tradeoffs by letting them choose the probability with which a recourse could get invalidated (recourse invalidation rate) if small changes are made to the prescribed recourse.

Finally, in Chapter 6, we took a step towards standardizing the evaluation of recourse methods by introducing a benchmarking library called `CARLA`. This library aims to facilitate the comparison of different recourse methods by providing a suite of established evaluation measures. It comes with predefined datasets, pretrained models, and a variety of recourse methods out of the box. Additionally, `CARLA` is designed in a modular way, which allows users to go beyond its existing resources and easily implement their own methods according to their specific needs.

7.2 Future Work

In the future, several intriguing research questions should be addressed. First, handing out recourses to the public poses a potential security threat to the decision maker, which calls for studying security aspects of counterfactual explanations in more detail; Second, there is little understanding of how offering algorithmic recourse to the public will affect population dynamics and the distribution of predictive outcomes over time; Third, recourse is fundamentally an applied concept, and therefore we need to make sure that the research agenda is heading in the right direction.

7.2.1 Goodheart's Law and Algorithmic Recourse

Goodheart's law states that "[w]hen a measure becomes a target, it ceases to be a good measure" [149]. Applying this logic to algorithmic recourse can lead to interesting insights. This law suggests that as individuals keep receiving counterfactual explanations over time, they will start to challenge the predictive outcomes produced by the machine learning algorithms, causing changes in the feature distribution and possibly the distribution of true labels. As a result, the original statistical regularities that the machine learning algorithm has been based on may no longer hold, potentially leading to shifts in the relative importance of certain features for algorithmic recourse, and in the worst case to features that might become increasingly uninformative over time. This highlights the importance of continuously monitoring and re-evaluating algorithmic recourse systems to ensure that they continue to produce valid and fair results. Developing this idea into a formal mathematical framework seems an interesting avenue for future research.

7.2.2 Validating the Utility of Research on Algorithmic Recourse

Despite the growing number of contributions to the body of counterfactual explainability research, *empirically validated* desiderata lack behind the numerous postulated desiderata and their algorithmic and theoretical contributions in the field (see Chapters 2 - 5).

One way to address this issue is through the use of systematic user studies. For instance, recent research in the field of explainability [83] has utilized methods such as semi-structured interviews and surveys to understand how data scientists utilize local feature attribution tools [97, 171] to identify problems in the machine learning pipeline. These studies have shown that data scientists often place too much trust in these methods. Similarly, it would be valuable to investigate how companies use counterfactual explanations for algorithmic recourse, and whether end-users find them to be useful in achieving their desired outcomes. The following are some intriguing, yet unanswered questions that could be explored through such research:

- **The corporate perspective**: From this perspective, it would helpful to know whether there is indeed a need for precise instructions as the algorithmic research literature seems to suggest? To date, there is a lack of work addressing this and related elementary questions:

(i) Do institutions consider using these explanation techniques to provide customers with recourse?

(ii) If they have not used these techniques yet, how do they plan to use them in the future? Do they issue *adverse action notices* (recall Chapter 2) or precise counterfactual explanations?

(iii) What challenges do these institutions face when deploying these explanations in practice?

- **The end-user perspective**: The key requirement of counterfactual explanations is that they should be low in cost and easy to achieve by an end-user. This requirement relies on the premise that a low-complexity counterfactual explanation would be easiest to put into practice (recall Chapter 2). The lowest cost counterfactual explanations are usually generated under the independently manipulable feature assumption, which is prone to generating atypical recourses, where the term "atypical" refers to how likely the generated instance would occur under the (estimated) data distribution (recall Chapter 3). Thus, it would be interesting to understand whether users find IMF recourses more or less atypical compared to recourses generated by the methods we have proposed in Chapter 3.

  Another central desideratum is to ascertain that an issued counterfactual explanation is *feasible*, i.e., the recourse should be attainable by the individual. In the causal recourse literature, feasibility constraints can easily be encoded into a postulated causal graphical model that represents the underlying data generating process. However, identifying the causal relations and evaluating their correctness is a cognitively demanding task. Therefore, it would be fascinating to understand whether supporting participants with a postulated causal model leads to better performance in spotting feasibility violations within the counterfactual explanations.

7.2.3 Can Recourse be Provided without Opening the Black-box?

In their seminal work, Wachter et al. [166] argue that counterfactual explanations allow end users access to useful explanations while avoiding that model owners have to disclose proprietary information in terms of model internals (e.g., model parameters) or information regarding the training data. In contrast to this initial motivation, recent research has demonstrated the privacy risks that arise when attackers have access to counterfactual model explanations that highlight the rationale behind one or more model predictions [7, 121]. These works have shown that malicious attackers could exploit these recourses to extract information about the underlying models and their training data, thus leaking sensitive information (e.g., a bank's customer data) and enabling fraudulent activities. In particular, the generated recourses can be used for;

(i) **Privacy attacks**: infer whether an end-user's private data was part of the model's training data set [121] or;

(ii) **Model extraction:** reverse-engineer the underlying predictive model [7].

Therefore, there is an urgent need to investigate how recourses can be made more secure against such fraudulent activities. We envision the notion of Differential Privacy (DP) [35, 48, 49] to play a central part when devising strategies to mitigate such privacy attacks. Therefore, we conjecture that the reliability of algorithmic recourse will likely benefit from underlying predictive models that have been trained under DP constraints.

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX}*, 2016.

[2] Philip Adler, Casey Falk, Sorelle A. Friedler, Gabriel Rybeck, Carlos Scheidegger, Brandon Smith, and Suresh Venkatasubramanian. Auditing black-box models for indirect influence. *Knowledge and Information Systems*, 2018.

[3] Darius Afchar, Vincent Guigue, and Romain Hennequin. Towards rigorous interpretations: a formalisation of feature attribution. In *International Conference on Machine Learning*. PMLR, 2021.

[4] Chirag Agarwal, Satyapriya Krishna, Eshika Saxena, Martin Pawelczyk, Nari Johnson, Isha Puri, Marinka Zitnik, and Himabindu Lakkaraju. Openxai: Towards a transparent evaluation of post-hoc model explanations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[5] Sushant Agarwal, Shahin Jabbari, Chirag Agarwal, Sohini Upadhyay, Zhiwei Steven Wu, and Himabindu Lakkaraju. Towards the unification and robustness of perturbation and gradient based explanations. In *International Conference on Machine Learning (ICML)*, 2021.

[6] Ulrich Aïvodji, Hiromi Arai, Olivier Fortineau, Sébastien Gambs, Satoshi Hara, and Alain Tapp. Fairwashing: the risk of rationalization. In *International Conference on Machine Learning (ICML)*. PMLR, 2019.

[7] Ulrich Aïvodji, Alexandre Bolot, and Sébastien Gambs. Model extraction from counterfactual explanations. *arXiv preprint arXiv:2009.01884*, 2020.

[8] Ulrich Aïvodji, Hiromi Arai, Sébastien Gambs, and Satoshi Hara. Characterizing the Risk of Fairwashing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[9] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep

learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

[10] Maher Ala'raj, Maysam F Abbod, Munir Majdalawieh, and Luay Jum'a. A deep learning model for behavioural credit scoring in banks. *Neural Computing and Applications*, 34(8):5839–5866, 2022.

[11] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

[12] Christopher J Anders, Plamen Pasliev, Ann-Kathrin Dombrowski, Klaus-Robert Müller, and Pan Kessel. Fairwashing explanations with off-manifold detergent. *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[13] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. And it's biased against blacks, 2016.

[14] Peter S. Arcidiacono. Expert report by Peter S. Arcidiacono, Students for Fair Admissions, 2019. URL https://tinyurl.com/2p9cw7pp.

[15] André Artelt, Valerie Vaquet, Riza Velioglu, Fabian Hinder, Johannes Brinkrolf, Malte Schilling, and Barbara Hammer. Evaluating robustness of counterfactual explanations. *arXiv:2103.02354*, 2021.

[16] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. How to explain individual classification decisions. *The Journal of Machine Learning Research (JMLR)*, 11: 1803–1831, 2010.

[17] Naman Bansal, Chirag Agarwal, and Anh Nguyen. Sam: The sensitivity of attribution methods to hyperparameters. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition (CVPR)*, pages 8673–8683, 2020.

[18] Roberto Baratta. Complexity of EU law in the domestic implementing process. *The Theory and Practice of Legislation*, 2014.

[19] Solon Barocas and Andrew D Selbst. Big data's disparate impact. *California law review*, 2016.

[20] Solon Barocas, Andrew D Selbst, and Manish Raghavan. The hidden assumptions behind counterfactual explanations and principal reasons. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2020.

[21] BBC. Amazon scrapped 'sexist ai' tool. *BBC.com*, 2018. URL `https://www.bbc.com/news/technology-45809919`.

[22] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 2013.

[23] Umang Bhatt, Adrian Weller, and José MF Moura. Evaluating and aggregating feature-based model explanations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 3016 – 3022, 2020.

[24] Asia J. Biega and Michèle Finck. Reviving purpose limitation and data minimisation in data-driven systems. *Technology and Regulation*, 2021.

[25] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

[26] Daniel Björkegren, Joshua E. Blumenstock, and Samsun Knight. Manipulation-proof machine learning. *arXiv preprint: arXiv:2004.03865*, 2020.

[27] Emily Black, Zifan Wang, Matt Fredrikson, and Anupam Datta. Consistent counterfactuals for deep models. *arXiv:2110.03109*, 2021.

[28] Miranda Bogen and Aaron Rieke. Help wanted: An examination of hiring algorithms, equity, and bias. *Upturn, December*, 7, 2018.

[29] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep Neural Networks and Tabular Data: A Survey. *Transactions on Neural Networks and Learning Systems (TNNLS)*, 2022.

[30] Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. In *International Conference on Learning Representations (ICLR)*, 2023.

[31] Kieran Browne and Ben Swift. Semantics and explanation: why counterfactual explanations produce adversarial examples in deep neural networks. *arXiv preprint arXiv:2012.10076*, 2020.

[32] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[33] Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8 (8):832, jul 2019. doi: 10.3390/electronics8080832.

[34] Gavin C Cawley and Nicola LC Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural networks*, 17(10):1467–1475, 2004.

[35] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[36] Jianbo Chen, Le Song, Martin J Wainwright, and Michael I Jordan. Learning to explain: An information-theoretic perspective on model interpretation. *Proceedings of the 35 th International Conference on Machine Learning (ICML)*, 2018.

[37] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.

[38] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373*, 2017.

[39] United States Congress. The Fair and Accurate Credit Transactions Act, 2003.

[40] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. In *NeurIPS Datasets and Benchmarks Track*, 2021.

[41] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*. Springer, 2020.

[42] Xolani Dastile, Turgay Celik, and Moshe Potsane. Statistical and machine learning models in credit scoring: A systematic literature survey. *Applied Soft Computing*, 91:106263, 2020.

[43] Botty Dimanov, Umang Bhatt, Mateja Jamnik, and Adrian Weller. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. In *SafeAI@ AAAI*, pages 63–73, 2020.

[44] Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can

be manipulated and geometry is to blame. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, pages 13589–13600. Curran Associates, Inc., 2019.

[45] Ricardo Dominguez-Olmedo, Amir-Hossein Karimi, and Bernhard Schölkopf. On the adversarial robustness of causal algorithmic recourse. In *International Conference on Machine Learning (ICML)*, 2022.

[46] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[47] Julia Dressel and Hany Farid. The accuracy, fairness, and limits of predicting recidivism. *Science advances*, 4(1):eaao5580, 2018.

[48] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[49] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

[50] Harrison Edwards and Amos Storkey. Censoring representations with an adversary. *International Conference on Learning Representations*, 2016.

[51] Council of the European Union European Parliament. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016. *Official Journal of the European Union*, 2016.

[52] Jianqing Fan, Yang Feng, and Lucy Xia. A conditional dependence measure with applications to undirected graphical models. *arXiv preprint arXiv:1501.01617*, 2015.

[53] Thomas Fel, David Vigouroux, Rémi Cadène, and Thomas Serre. How good is your explanation? algorithmic stability measures to assess the quality of explanations for deep neural networks. *arXiv:2009.04521*, 2021.

[54] Michele Finck and Asia J Biega. Reviving Purpose Limitation and Data Minimisation in Data-Driven Systems. *Technology and Regulation*, 2021, 2021.

[55] Timo Freiesleben. Counterfactual explanations & adversarial examples–common grounds, essential differences, and potential transfers. *arXiv preprint arXiv:2009.05487*, 2020.

[56] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441,

2008.

[57] Kazuto Fukuchi, Satoshi Hara, and Takanori Maehara. Faking fairness via stealthily biased sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[58] Damien Garreau and Ulrike Luxburg. Explaining the explainer: A first theoretical analysis of lime. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1287–1296. PMLR, 2020.

[59] Damien Garreau and Ulrike von Luxburg. Looking deeper into lime. *arXiv 2008.11092*, 2020.

[60] GDPR. Regulation (EU) 2016/679 of the European Parliament and of the Council. *Official Journal of the European Union*, 2016.

[61] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 3681–3688, 2019.

[62] Antonio Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. Making ai forget you: Data deletion in machine learning. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada*, 2019.

[63] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[64] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. *arXiv:2003.02960*, 2020.

[65] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[66] Bryce Goodman and Seth Flaxman. European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation". *AI magazine*, 2017.

[67] Nina Grgic-Hlaca, Muhammad Bilal Zafar, Krishna P Gummadi, and Adrian Weller. The case for process fairness in learning: Feature selection for fair decision making. In *NIPS Symposium on Machine Learning and the Law*, 2016.

[68] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM Computing Surveys*, 51(5):1–42, jan 2019. doi: 10.1145/3236009.

[69] Moritz Hardt and Tengyu Ma. Identity matters in deep learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[70] Thomas Hartley, Kirill Sidorov, Christopher Willis, and David Marshall. Explaining failure: Investigation of surprise and expectation in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 12–13, 2020.

[71] Johannes Haug, Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Leveraging model inherent variable importance for stable online feature selection. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[72] Juyeon Heo, Sunghwan Joo, and Taesup Moon. Fooling neural network interpretations via adversarial model manipulation. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, pages 2925–2936. Curran Associates, Inc., 2019.

[73] Sara Hooker, Dumitru Erhan, Pieter Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[74] Cosimo Izzo, Aldo Lipani, Ramin Okhrati, and Francesca Medda. A baseline for shapely values in mlps: from missingness to neutrality. *arXiv preprint arXiv:2006.04896*, 2020.

[75] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130. PMLR, 2021.

[76] Hamed Jalali, Martin Pawelczyk, and Gjergji Kasneci. Model selection in local approximation gaussian processes: A markov random fields approach. In *IEEE International Conference on Big Data (Big Data)*, 2021.

[77] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv:1907.09615*, 2019.

[78] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Hiroki Arimura. Dace: Distribution-aware counterfactual explanation by mixed-integer linear optimization. In *IJCAI*, 2020.

[79] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[80] Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv:2010.04050*, 2020.

[81] Amir-Hossein Karimi, Julius von Kügelgen, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[82] Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021.

[83] Harmanpreet Kaur, Harsha Nori, Samuel Jenkins, Rich Caruana, Hanna Wallach, and Jennifer Wortman Vaughan. Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning. In *Proceedings of the 2020 Conference on Human Factors in Computing Systems (CHI)*, 2020.

[84] Joon Sik Kim, Gregory Plumb, and Ameet Talwalkar. Sanity simulations for saliency methods. *arXiv*, 2021.

[85] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[86] Tadas Klimas and Jurate Vaiciukaite. The law of recitals in European Community legislation. *ILSA J. Int'l & Comp. L.*, 2008.

[87] Alexey Kurakin, Ian Goodfellow, Samy Bengio, et al. Adversarial examples in the physical world, 2016.

[88] Michael T Lash, Qihang Lin, Nick Street, Jennifer G Robinson, and Jeffrey Ohlmann. Generalized inverse classification. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017.

[89] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Inverse classification for comparison-based inter-

pretability in machine learning. *arXiv preprint arXiv:1712.08443*, 2017.

[90] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, and Marcin Detyniecki. Issues with post-hoc counterfactual explanations: a discussion. *ICML Workshop on Human in the Loop Learning*, 2019.

[91] Steffen L Lauritzen. *Graphical models*, volume 17. Clarendon Press, 1996.

[92] Tobias Leemann, Martin Pawelczyk, Christian Thomas Eberle, and Gjergji Kasneci. I prefer not to say: Operationalizing fair and user-guided data minimization. *arXiv preprint: arxiv.2210.13954*, 2022.

[93] Yang Liu, Sujay Khandagale, Colin White, and Willie Neiswanger. Synthetic benchmarks for scientific research in explainable machine learning. In *NeurIPS Datasets and Benchmarks Track*, 2021.

[94] Francesco Locatello, Gabriele Abbati, Tom Rainforth, Stefan Bauer, Bernhard Schölkopf, and Olivier Bachem. On the fairness of disentangled representations. In *Advances in neural information processing systems (NeurIPS)*, 2019.

[95] Javier Louro, Margarita Posso, Michele Hilton Boon, Marta Román, Laia Domingo, Xavier Castells, and María Sala. A systematic review and quality assessment of individualised breast cancer risk prediction models. *British journal of cancer*, 121(1):76–85, 2019.

[96] Ana Lucic, Harrie Oosterhuis, Hinda Haned, and Maarten de Rijke. Focus: Flexible optimizable counterfactual explanations for tree ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[97] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS) 30*, 2017.

[98] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Learning adversarially fair and transferable representations, 2018.

[99] Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*, 2019.

[100] Charles Marx, Richard Phillips, Sorelle Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. Disentangling influence: Using disentangled representations to audit model predictions. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, 2019.

[101] Nicolai Meinshausen and Peter Bühlmann. High-dimensional graphs and variable selection with the lasso. *The annals of statistics*, pages 1436–1462, 2006.

[102] Chuizheng Meng, Loc Trinh, Nan Xu, and Yan Liu. Mimic-if: Interpretability and fairness evaluation of deep learning models on mimic-iv dataset. *arXiv preprint arXiv:2102.06761*, 2021.

[103] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[104] Saumitra Mishra, Sanghamitra Dutta, Jason Long, and Daniele Magazzeni. A survey on the robustness of feature importance and counterfactual explanations. *arXiv:2111.00358*, 2021.

[105] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[106] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2020.

[107] Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *arXiv preprint arXiv:2201.08164*, 2022.

[108] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. *arXiv preprint arXiv:1807.03653*, 2018.

[109] An Phi Nguyen and María Rodríguez Martínez. On quantitative aspects of model interpretability. *arXiv preprint arXiv:2007.07584*, 2020.

[110] Eirini Ntoutsi, Pavlos Fafalios, Ujwal Gadiraju, Vasileios Iosifidis, Wolfgang Nejdl, Maria-Esther Vidal, Salvatore Ruggieri, Franco Turini, Symeon Papadopoulos, Emmanouil Krasanakis, Ioannis Kompatsiaris, Katharina Kinder-Kurlanda, Claudia Wagner, Fariba Karimi, Miriam Fernandez, Harith Alani, Bettina Berendt, Tina Kruegel, Christian Heinze, Klaus Broelemann, Gjergji Kasneci, Thanassis Tiropanis, and Steffen Staab. Bias in data-driven artificial intelligence systems—an introductory sur-

vey. *WIREs Data Mining and Knowledge Discovery*, 10(3), feb 2020. doi: 10.1002/widm.1356.

[111] CA OAG. Ccpa regulations: Final regulation text. *Office of the Attorney General, California Department of Justice*, 2021.

[112] Cathy O'Neil. *Weapons of Math Destruction: How big data increases inequality and threatens democracy*. Crown Publishing House / Penguin Random House, 2016.

[113] Seong Ho Park and Kyunghwa Han. Methodologic guide for evaluating clinical performance and effect of artificial intelligence technology for medical diagnosis and prediction. *Radiology*, 286(3):800–809, 2018.

[114] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv:1912.01703*, 2019.

[115] Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning Model-Agnostic Counterfactual Explanations for Tabular Data. In *Proceedings of The Web Conference 2020 (WWW)*. ACM, 2020.

[116] Martin Pawelczyk, Klaus Broelemann, and Gjergji. Kasneci. On Counterfactual Explanations under Predictive Multiplicity. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2020.

[117] Martin Pawelczyk, Sascha Bielawski, Johan Van den Heuvel, Tobias Richter, and Gjergji Kasneci. CARLA: A Python Library to Benchmark Algorithmic Recourse and Counterfactual Explanation Algorithms. In *Advances in Neural Information Processing Systems (NeurIPS) (Benchmark and Datasets Track)*, volume 34, 2021.

[118] Martin Pawelczyk, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. Exploring Counterfactual Explanations Through the Lens ofAdversarial Examples: A Theoretical and Empirical Analysis. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

[119] Martin Pawelczyk, Lea Tiyavorabun, and Gjergji Kasneci. Decomposing Counterfactual Explanations for Consequential Decision Making. *arXiv preprint arxiv.2211.02151*, 2022.

[120] Martin Pawelczyk, Teresa Datta, Johannes van-den Heuvel, Gjergji Kasneci, and Himabindu Lakkaraju. Probabilistically Robust Recourse: Nav-

igating the Tradeoffs Between Costs and Robustness in Algorithmic Recourse. In *International Conference on Learning Representations (ICLR)*, 2023.

[121] Martin Pawelczyk, Himabindu Lakkaraju, and Seth Neel. On the Privacy Risks of Algorithmic Recourse. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2023.

[122] Martin Pawelczyk, Tobias Leemann, Asia Biega, and Gjergji Kasneci. On the Trade-Off between Actionable Explanations and the Right to be Forgotten. In *International Conference on Learning Representations (ICLR)*, 2023.

[123] Judea Pearl. *Causality*. Cambridge university press, 2009.

[124] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 2011.

[125] William Peebles, John Peebles, Jun-Yan Zhu, Alexei A. Efros, and Antonio Torralba. The hessian penalty: A weak prior for unsupervised disentanglement. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[126] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[127] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

[128] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. Face: Feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES)*, 2020.

[129] Manish Raghavan, Solon Barocas, Jon Kleinberg, and Karen Levy. Mitigating bias in algorithmic hiring: Evaluating claims and practices. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 469–481, 2020.

[130] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C Duchi, and Percy Liang. Adversarial training can hurt generalization. *arXiv preprint arXiv:1906.06032*, 2019.

[131] Kaivalya Rawal and Himabindu Lakkaraju. Interpretable and interactive summaries ofactionable recourses. In *Advances in Neural Information Pro-*

cessing Systems (NeurIPS), volume 33, 2020.

[132] Kaivalya Rawal, Ece Kamar, and Himabindu Lakkaraju. Algorithmic recourse in the wild: Understanding the impact of data and model shifts. *arXiv:2012.11788*, 2021.

[133] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, 2016.

[134] Yao Rong, Tobias Leemann, Vadim Borisov, Gjergji Kasneci, and Enkelejda Kasneci. A consistent and efficient evaluation strategy for attribution methods. In *International Conference on Machine Learning (ICML)*, 2022.

[135] Elan Rosenfeld, Ezra Winston, Pradeep Ravikumar, and Zico Kolter. Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*, pages 8230–8241. PMLR, 2020.

[136] Francesca Rossi. Artificial intelligence: Potential benefits and ethical considerations. *EPRS: European Parliamentary Research Service*, 2016.

[137] Christopher Russell. Efficient search for diverse coherent explanations. In *Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency (FAT*)*. ACM, 2019.

[138] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11):2660–2673, 2016.

[139] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 2021.

[140] Patrick Schramowski, Wolfgang Stammer, Stefano Teso, Anna Brugger, Franziska Herbert, Xiaoting Shao, Hans-Georg Luigs, Anne-Katrin Mahlein, and Kristian Kersting. Making deep neural networks right for the right scientific reasons by interacting with their explanations. *Nature Machine Intelligence*, 2(8):476–486, 2020.

[141] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *NeurIPS*, 2018.

[142] Harshay Shah, Prateek Jain, and Praneeth Netrapalli. Do input gradients

highlight discriminative features?, 2021.

[143] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 35 th International Conference on Machine Learning (ICML)*, 2018.

[144] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. How can we fool lime and shap? adversarial attacks on post hoc explanation methods. *arXiv preprint arXiv:1911.02508*, 2019.

[145] Dylan Slack, Anna Hilgard, Sameer Singh, and Himabindu Lakkaraju. Reliable post hoc explanations: Modeling uncertainty in explainability. In *Advances in Neural Information Processing Systems*, 2021.

[146] Dylan Slack, Sophie Hilgard, Himabindu Lakkaraju, and Sameer Singh. Counterfactual explanations can be manipulated. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[147] Thomas Spooner, Danial Dervovic, Jason Long, Jon Shepard, Jiahao Chen, and Daniele Magazzeni. Counterfactual explanations for arbitrary regression models. *arXiv:2106.15212*, 2021.

[148] Suraj Srinivas and François Fleuret. Full-gradient representation for neural network visualization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[149] Marilyn Strathern. 'improving ratings': audit in the british university system. *European review*, 5(3):305–321, 1997.

[150] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[151] Prasanna Tambe, Peter Cappelli, and Valery Yakubovich. Artificial intelligence in human resources management: Challenges and a path forward. *California Management Review*, 61(4):15–42, 2019.

[152] Ruixiang Tang, Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. An embarrassingly simple approach for trojan attack in deep neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2020.

[153] Winnie F Taylor. Meeting the Equal Credit Opportunity Act's Specificity Requirement: Judgmental and Statistical Scoring Systems. *Buff. L. Rev.*, 1980.

[154] Paul Thagard. Explanatory coherence. *Behavioral and brain sciences*, 12(3): 435–467, 1989.

[155] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. ACM, 2017.

[156] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[157] Amos Tversky and Daniel Kahneman. Extensional versus intuitive reasoning: The conjunction fallacy in probability judgment. *Psychological review*, 90(4):293, 1983.

[158] Sohini Upadhyay, Shalmali Joshi, and Himabindu Lakkaraju. Towards robust and reliable algorithmic recourse. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

[159] Berk Ustun, Alexander Spangher, and Y. Liu. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2019.

[160] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, Jan 2019.

[161] Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2019.

[162] Suresh Venkatasubramanian and Mark Alfano. The philosophical basis of algorithmic recourse. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, New York, NY, USA, 2020. ACM.

[163] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv:2010.10596*, 2020.

[164] Eduard Fosch Villaronga, Peter Kieseberg, and Tiffany Li. Humans forget, machines remember: Artificial intelligence and the right to be forgotten. *Computer Law & Security Review*, 34(2):304–313, 2018.

[165] Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 2017.

[166] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 2018.

[167] Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning (ICML)*. PMLR, 2020.

[168] Yufei Xia, Chuanzhe Liu, YuYing Li, and Nana Liu. A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring. *Expert systems with applications*, 78:225–241, 2017.

[169] Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Suggala, David I Inouye, and Pradeep K Ravikumar. On the (in) fidelity and sensitivity of explanations. *Advances in Neural Information Processing Systems*, 32:10967–10978, 2019.

[170] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision (ECCV)*, pages 818–833. Springer, 2014.

[171] Xuezhou Zhang, Sarah Tan, Paul Koch, Yin Lou, Urszula Chajewska, and Rich Caruana. Axiomatic interpretability for multiclass additive models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2019.

[172] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2018.

A Publications

This thesis is based on six papers. All papers are publicly available. The papers provided here are identical to the versions published online. I am the main contributor and first author of each publication. A more detailed statement about the contributions of each author can be found in the following sections.

A.1 Learning Model-Agnostic Counterfactual Explanations for Tabular Data

Publication: Published in the proceedings of the Web Conference 2020 (**WWW**).

Contribution: I developed the method, wrote the implementation and performed the experiments. Moreover, I wrote all parts of the paper. Klaus Broelemann, and Gjergji Kasneci contributed to the paper by improving concepts, ideas and the presentation, and by revising the manuscript.

Learning Model-Agnostic Counterfactual Explanations for Tabular Data

Martin Pawelczyk
University of Tuebingen
Tuebingen, Germany
martin.pawelczyk@uni-tuebingen.de

Klaus Broelemann
Schufa Holding AG
Wiesbaden, Germany
klaus.broelemann@schufa.de

Gjergji Kasneci
University of Tuebingen
Tuebingen, Germany
gjergji.kasneci@uni-tuebinge.de

ABSTRACT

Counterfactual explanations can be obtained by identifying the smallest change made to an input vector to influence a prediction in a positive way from a user's viewpoint; for example, from 'loan rejected' to 'awarded' or from 'high risk of cardiovascular disease' to 'low risk'. Previous approaches would not ensure that the produced counterfactuals be *proximate* (i.e., not local outliers) and *connected* to regions with substantial data density (i.e., close to correctly classified observations), two requirements known as *counterfactual faithfulness*. Our contribution is twofold. First, drawing ideas from the manifold learning literature, we develop a framework, called C-CHVAE, that generates *faithful counterfactuals*. Second, we suggest to complement the catalog of counterfactual quality measures using a criterion to quantify the *degree of difficulty* for a certain counterfactual suggestion. Our real world experiments suggest that *faithful counterfactuals* come at the cost of higher *degrees of difficulty*.

KEYWORDS

Transparency, Counterfactual explanations, Interpretability

ACM Reference Format:
Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. 2020. Learning Model-Agnostic Counterfactual Explanations for Tabular Data. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3366423.3380087

1 INTRODUCTORY REMARKS

Machine learning models are increasingly being deployed to automate high-stake decisions in industrial applications, e.g., financial, employment, medical or public services. Wachter et al. [22] discuss to establish a legally binding right to request explanations on any prediction that is made based on personal data of an individual. In fact, the EU General Data Protection Regulation (GDPR) includes a right to request "meaningful information about the logic involved, as well as the significance and the envisaged consequences" [22] of automated decisions.

As people are increasingly being affected by these automated decisions, it is natural to ask how those affected can be empowered to receive desired results in the future. To this end, Wachter et al. [22] suggest using counterfactual explanations. In this context, a counterfactual is defined as a small change made to the input vector to influence a classifier's decision in favor of the person represented by the input vector.

1.1 A step towards user empowerment

The "close world" desideratum. At a high level, Wachter et al. [22] formulated the desideratum that counterfactuals should come from a 'possible world' which is 'close' to the user's starting point. Laugel et al. [11] formalized the *close world* desideratum and split it into two measurable criteria, *proximity* and *connectedness*. Proximity describes that counterfactuals should not be local outliers and connectedness quantifies whether counterfactuals are close to correctly classified observations. We shortly review both criteria in section 5. To these two criteria, we add a third one based on percentile shifts of the cumulative distribution function (CDF) of the inputs, as a measure for the *degree of difficulty*. Intuitively, all criteria help quantify how *attainable* suggested counterfactuals are.

The C-CHVAE. In this work, our main contribution is a general-purpose framework, the *Counterfactual Conditional Heterogeneous Autoencoder*, C-CHVAE, which allows finding (multiple) counterfactual feature sets while generating counterfactuals with high occurrence probability. This is a fundamental requirement towards *attainability* of counterfactuals. In particular, our framework is compatible with a multitude of autoencoder (AE) architectures as long as the AE allows both modelling of heterogeneous data and approximating the conditional log likelihood of the the mutable/free inputs given the immutable/protected ones. Moreover, the C-CHVAE does not require access to a distance function (for the input space) and is classifier agnostic. Part of this work was previously published as a *NeurIPS HCML workshop paper* [15]. Our source code can be found at: https://github.com/MartinPawel/c-chvae.

1.2 Challenges for counterfactuals

Attainability. Intuitively, a counterfactual is attainable, if it is jointly (1) a 'close' suggestion that is not a local outlier, (2) similar to correctly classified observations and (3) associated with low total CDF percentile shifts. Hence, in our point of view, *attainability* is a composition of faithful counterfactuals ((1) and (2)) which are at the same time not too difficult to attain (3). To reach a better understanding, let us translate conditions (1), (2) and (3) into the following synthetic bank loan setting: a client applies for a loan and a bank employs a counterfactual empowerment tool. Under these circumstances, we focus on one problematic aspect. The tool could make suggestions that '*lie outside of a client's wheelhouse*', that is to say, it is not reasonable to suggest counterfactuals that (a) one would typically not observe in the data, (b) that are not typical for the subgroup of users the client belongs to, and that (c) are extremely difficult to attain, where difficulty is measured in terms of the percentiles of the CDF of the given inputs. For example, in table 1, the suggestion made by the second method is likely not attainable given her age and education level.

Similarity via latent distance. Additionally, in health, banking or credit scoring contexts we often face continuous, ordinal and nominal inputs concurrently. This is also known as *heterogeneous* or *tabular* data. For this type of data, it can sometimes be difficult to measure distance in a meaningful way (e. g. measuring distance between different occupations). Furthermore, existing methods leave the elicitation of appropriate distance/cost functions up to (expert) opinions [4, 10, 12, 21, 22], which can vary considerably across individuals [5]. Therefore, we suggest measuring similarity between the input feature $\boldsymbol{x}_i$ and a potential counterfactual $\tilde{\boldsymbol{x}}_i$ as follows.

Definition 1 (Latent distance). *Let $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^n$ be two observations in input space with corresponding lower dimensional representations $\boldsymbol{z}_i, \boldsymbol{z}_j \in \mathbb{R}^k$ with $k < n$, in latent space. Then the distance $d_L(\boldsymbol{x}_i, \boldsymbol{x}_j) := \|z_i - z_j\|_p$ is called the latent distance of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.*

1.3 Overview

Learning faithful counterfactuals via the C-CHVAE. We suggest embedding counterfactual search into a data density approximator, here a variational autoencoder (VAE) [9]. The idea is to use the VAE as a *search device* to find counterfactuals that are *proximate* and *connected* to the input data. The intuition of this approach becomes apparent by considering each part of the VAE in turn. As opposed to classical generative model contexts, the encoder part is not discarded at *test time/generation time.* Indeed, it is the *trained encoder* that plays a crucial role: given the original heterogeneous data, the encoder specifies a lower dimensional, real-valued and dense representation of that data, $\boldsymbol{z}$, Therefore, it is the encoder that determines which low-dimensional neighbourhood we should look to for potential counterfactuals. Next, we perturb the low dimensional data representation, $\boldsymbol{z} + \boldsymbol{\delta}$, and feed the perturbed representation into the decoder. For small perturbations the decoder gives a potential counterfactual by reconstructing the input data from the perturbed representation. This counterfactual is likely to occur. Next, the potential counterfactual is passed to the pretrained classifier, which we ask whether the prediction was altered. Figure 1 represents this mechanism.

Consistent search for heterogeneous data. While we aim to avoid altering immutable inputs, such as *age* or *education*, it is reasonable to believe that the immutable inputs can have an impact on what is attainable to the individual. Thus, the immutable inputs should influence the neighbourhood search for counterfactuals. For example, certain drugs can have different treatment effects, depending on whether a patient is male or female [16]. Hence, we wish to generate *conditionally consistent* counterfactuals.

Again, consider Figure 1 for an intuition of counterfactual search in the presence of immutable inputs. Unlike in vanilla VAEs, we assume a Gaussian mixture prior on the latent variables where each mixture component is also estimated by the immutable inputs. This helps cluster the latent space and has the advantage that we look for counterfactuals among *semantically similar* alternatives.

Contribution. The C-CHVAE is a general-purpose framework that generates counterfactuals. Its main merits are:

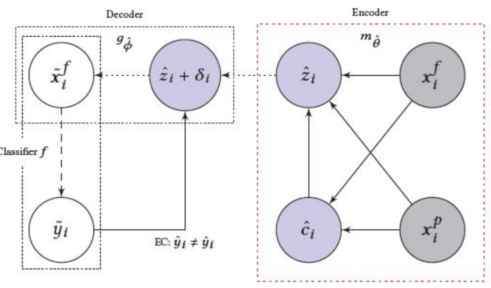


Figure 1: Autoencoding Counterfactual search. The learned encoder, $m_{\hat{\theta}}$, maps heterogeneous protected and free features, $\boldsymbol{x}^p$ and $\boldsymbol{x}^f$, and latent mixture components, $\hat{\boldsymbol{c}}$, into a latent representation, $\hat{\boldsymbol{z}}$. The learned decoder, $g_{\hat{\phi}}$, reconstructs the free inputs $\boldsymbol{x}^f$ from the perturbed representation, providing a potential counterfactual, $\tilde{\boldsymbol{x}} = (\boldsymbol{x}^p, \tilde{\boldsymbol{x}}^f)$. The counterfactual acts like a typical observation from the data distribution. Next, we feed the potential counterfactual $\tilde{\boldsymbol{x}}$ to the classifier, f. We stop the search, if the EC condition is met.

- **Faithful counterfactuals.** The generated counterfactuals are *proximate* and *connected* to regions of high data density and therefore likely attainable, addressing the most important desiderata in the literature on counterfactuals [11, 22];
- **Suitable for tabular data and classifier agnostic.** The data distribution is modelled by an autoencoder that handles heterogeneous data and interval constraints by choosing appropriate likelihood models. It can also be combined with a multitude of autoencoder architectures [7, 9, 13, 14, 18, 20];
- **No ad-hoc distance measures for *input* data.** The C-CHVAE does not require ad-hoc predefined distance measures for input data to generate counterfactuals. This is can be an advantage over existing work, since it can be difficult to devise meaningful distance measures for tabular data.

2 RELATED LITERATURE

Explainability through counterfactuals. At a meta level, the major difference separating our work from previous approaches is that we *learn* a separate model to learn similarity in latent space and use this model to generate counterfactual recommendations. Doing this allows us to generate counterfactuals that lie on the data manifold.

Approaches dealing with heterogeneous data rely on integer programming optimization [17, 21]. To produce counterfactuals that take on reasonable values (e. g. non negative values for wage income) one directly specifies the set of features and their respective support subject to change. The C-CHVAE also allows for such constraints by choosing the likelihood functions for each feature appropriately (see Section 4.2 and our github repo.).

A closely related collection of approaches assumes that distances or costs between any two points can be measured in a meaningful way [4, 10, 12, 21, 22]. The C-CHVAE, however, does not rely on

Method	ID	Input subset	Current	Percentile	Counterfactual	Percentile	Shift	Tot. shift	L. Outlier	Connected
I	1	`credit card debt`	5000	**55**	3500	**75**	**20**	40	No	Yes
		`saving account`	200	**45**	600	**65**	**20**			
II	1	`monthly income ($)`	2500	**40**	10000	**95**	**55**	75	Yes	No
		`# loans elsewhere`	5	**85**	2	**65**	**20**			

Table 1: Hypothetical counterfactuals for the same 22 year old individual without a college degree, who was denied credit. Suggestions were made by two different methods for a given classifier f. The rows suggest how a subset of free inputs would need to change to obtain credit, i. e. from $\hat{y} = 0$ to $\hat{y} = 1$. For the first empowerment technology, the suggestion might be reasonable whereas for the second one, the suggestion looks atypical and could be difficult to attain measured in terms of connectedness to existing knowledge, an outlier measure and the total percentile shift.

Method	Train	Classifier agnostic	Classifier	Tabular data
AR [21]	No	No	Lin. Models	No
HCLS [10]	No	No	SVM	No
GS [12]	No	Yes	All	No
FT [19]	No	No	Trees	No
C-CHVAE (ours)	Yes	Yes	All	Yes

Table 2: Overview of existing counterfactual generation methods. 'Train' indicates that a method requires training. 'Classifier agnostic' means whether a method can be combined with any black-box classifier.

task-specific, predefined similarity functions between the inputs and counterfactuals. For a *given autoencoder architecture*, we learn similarity between inputs and counterfactuals from the data.

Other approaches strongly rely on the pretrained classifier f and make use of restrictive assumptions, e. g. that f stems from a certain hypothesis class. For example, Ustun et al. [21] and Tolomei et al. [19] assume the pretrained classifiers to be linear or tree based respectively, which can restrict usefulness.

In independent work from our's, Joshi et al. [8] suggest a similar explanation model, however, they focus on causal models and are less concerned with the issue of evaluating counterfactual explanations.

Adversarial perturbations. Since counterfactuals are often generated independently of the underlying classification model, they are related to universal adversarial attacks (see for example Brown et al. [3]). While adversarial examples aim to alter the prediction a deep neural network makes on a data point via small and *imperceptible* changes, counterfactuals aim to alter data points to suggest *impactful* changes to individuals. Notice that counterfactuals do not fool a classifier in a classical sense, since individuals need to *exert real-world effort* to achieve the desired prediction. Since a review of the entire literature on adversarial attacks goes beyond the scope of this work, we refer the reader to the survey by Akhtar and Mian [2]. For an overview of counterfactual generation methods consider table 2.

Notation. In the remainder of this work, we denote the D dimensional feature space as $\mathcal{X} = \mathbb{R}^D$ and the feature vector for observation i by $\boldsymbol{x}_i \in \mathcal{X}$. We split the feature space into two disjoint feature subspaces of immutable (i. e. protected) and free features denoted by $\mathcal{X}_p = \mathbb{R}^{D_p}$ and $\mathcal{X}_f = \mathbb{R}^{D_f}$ respectively such that w.l.o.g. $\mathcal{X} = \mathcal{X}_p \times \mathcal{X}_f$ and $\boldsymbol{x}_i = (\boldsymbol{x}_i^p, \boldsymbol{x}_i^f)$. This means in particular that the d-th free feature of x_i is given by $x_{d,i}^f = x_{d,i}$ and the d-th protected feature is given by $x_{d,i}^p = x_{d+D_f,i}$. Let $\boldsymbol{z} \in \mathcal{Z} = \mathbb{R}^k$ denote the latent space representation of $\boldsymbol{x}$. The labels corresponding to the i'th observation are denoted by $y_i \in \mathcal{Y} = \{0, 1\}$. Moreover, we assume a given pretrained classifier $f : \mathcal{X} \to \mathcal{Y}$. Further, we introduce the following sets: $H^- = \{\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x}) = 0\}, H^+ = \{\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x}) = 1\}, D^+ = \{\boldsymbol{x}_i \in \mathcal{X} : y_i = 1\}$. We attempt to find an explainer $E : \mathcal{X} \to \mathcal{X}$, generating counterfactuals $E(\boldsymbol{x}) = \tilde{\boldsymbol{x}}$, such that $f(\boldsymbol{x}) \neq f(E(\boldsymbol{x}))$. Finally, values with $\hat{\cdot}$ usually denote estimated quantities, values carrying $\tilde{\cdot}$ denote candidate values and values with $\cdot^*$ denote the best value among a number of candidate values.

3 BACKGROUND

3.1 (Conditional) Variational Autoencoder

The simple VAE is often accompanied by an isotropic Gaussian prior $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. We then aim to optimize the following objective known as the Evidence Lower Bound (ELBO),

$$\mathrm{L}_{VAE}(p, q) = \mathbb{E}_{q(z|\boldsymbol{x}^f)}[\log p(\boldsymbol{x}^f|z)] - \mathrm{D}_{KL}[q(z|\boldsymbol{x}^f)||p(z)].$$

This objective bounds the data log likelihood, log $p(\boldsymbol{x}^f)$, from below. In the simple model, the decoder and the encoder are chosen to be Gaussians, that is, $q(z|\boldsymbol{x}^f) = \mathcal{N}(z|\boldsymbol{\mu}_q, \Sigma_q)$ and $p(\boldsymbol{x}^f|z) = \mathcal{N}(\boldsymbol{x}^f|\boldsymbol{\mu}_p, \Sigma_p)$, where the distributional parameters $\boldsymbol{\mu}(\cdot)$ and $\Sigma(\cdot)$ are estimated by neural networks. If all inputs were binary instead, one could use a Bernoulli decoder, $p(\boldsymbol{x}^f|z) = Ber(\boldsymbol{x}^f|\varrho_p(z))$.

Conditioning on a set of inputs, say $\boldsymbol{x}^p$, the objective that bounds the conditional log likelihood, log $p(\boldsymbol{x}^f|\boldsymbol{x}^p)$, can be written as [18],

$$\begin{aligned}\mathrm{L}_{CVAE}(p, q) = &\mathbb{E}_{q(z|\boldsymbol{x}^f, \boldsymbol{x}^p)}[\log p(\boldsymbol{x}^f|z, \boldsymbol{x}^p)] - \\ &\mathrm{D}_{KL}[q(z|\boldsymbol{x}^f, \boldsymbol{x}^p)||p(z|\boldsymbol{x}^p)],\end{aligned} \tag{1}$$

where one assumes that the prior $p(z|\boldsymbol{x}^p)$ is still an isotropic Gaussian, i.e. $z|\boldsymbol{x}^p \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. We will refer to this model as the CVAE.

4 C-CHVAE

In this part, we present both our objective function and the CHVAE architecture in Sections 4.1 and 4.2, respectively.

4.1 The C- in C-CHVAE

We take the pretrained, potentially non-linear, classifier $f(\cdot)$ as given, which can also be a training time fairness constraint classifier [1, 23]. Let us denote the encoder function, parameterized by θ, by $m_\theta(\cdot;\boldsymbol{x}^p)$), taking arguments $\boldsymbol{x}^f$. The decoder function, parametrized by ϕ, is denoted by $g_\phi(\cdot)$. It has inputs $m_\theta(\boldsymbol{x}^f;\boldsymbol{x}^p) = z \in \mathbb{R}^k$. Then our objective reads as follows,

$$\min_{\delta\in\mathcal{Z}} ||\boldsymbol{\delta}|| \text{ subject to} \tag{2}$$

$$f(g_{\hat{\phi}}(m_{\hat{\theta}}(\boldsymbol{x}) + \boldsymbol{\delta}), \boldsymbol{x}^p) \neq f(\boldsymbol{x}^f, \boldsymbol{x}^p) \ \& \tag{3}$$

$$\min_{\phi,\theta} \ell(\boldsymbol{x}, g_\phi(m_\theta(\boldsymbol{x}^f;\boldsymbol{x}^p))) - \Omega(m_\theta(\boldsymbol{x}^f;\boldsymbol{x}^p)), \tag{4}$$

where $\Omega(\cdot)$ is a regularizer on the latent space and $||\cdot||$ denotes the p-norm. The idea behind the objective is as follows. First, (4) approximates the conditional log likelihood, $p(\boldsymbol{x}^f|\boldsymbol{x}^p)$, while learning a lower dimensional latent representation. Subsequently, we use this latent representation, $\hat{\boldsymbol{z}} = m_{\hat{\theta}}(\boldsymbol{x}^f;\boldsymbol{x}^p)$, to search for counterfactuals ((2) and (3)). If the perturbation on $\hat{\boldsymbol{z}}$ is small enough, the trained decoder $g_{\hat{\phi}}$ gives a reconstruction $\tilde{\boldsymbol{x}}^{f*}$ that (a) is similar to $\boldsymbol{x}^f$, (b) satisfies the empowerment condition (3), and (c) lies in regions where we would usually expect data. Also, notice that this regularizer effectively plays the role of the distance function. It determines the neighbourhood of $\boldsymbol{x}$ in which we search for counterfactuals.

4.2 CHVAE

To solve the above optimization problem defined in (2)-(4), it is crucial to elicit an appropriate autoencoder architecture. We adjust the HVAE [14] so that it approximates conditional densities.

Factorized decoder. We suggest using the following hierarchical model to accommodate the generation of counterfactuals conditional on some immutable attributes. The factorized decoder with a conditional uniform Gaussian mixture prior ((5) and (6)) with parameters $\pi_l = 1/L$ for all mixture components l reads:

$$p(\boldsymbol{c}_i|\boldsymbol{x}_i^p) = Cat(\boldsymbol{\pi}) \tag{5}$$

$$p(\boldsymbol{z}_i|\boldsymbol{c}_i, \boldsymbol{x}_i^p) = \mathcal{N}(\boldsymbol{\mu}_p(\boldsymbol{c}_i), \boldsymbol{I}_K) \tag{6}$$

$$\begin{aligned} p(z_i, \boldsymbol{x}_i^f, \boldsymbol{c}_i|\boldsymbol{x}_i^p) &= p(z_i, \boldsymbol{c}_i|\boldsymbol{x}_i^p) \prod_{d=1}^{D_f} p(\boldsymbol{x}_{d,i}^f|z_i, \boldsymbol{c}_i, \boldsymbol{x}_i^p) \\ &= p(z_i|\boldsymbol{c}_i, \boldsymbol{x}_i^p)p(\boldsymbol{c}_i|\boldsymbol{x}_i^p) \cdot \prod_{d=1}^{D_f} p(\boldsymbol{x}_{d,i}^f|z_i, \boldsymbol{c}_i, \boldsymbol{x}_i^p), \end{aligned} \tag{7}$$

where $\boldsymbol{z}_i \in \mathbb{R}^k$ is the continuous latent vector and $\boldsymbol{c}_i \in \mathbb{R}^C$ is a vector indicating mixture components, generating the instance $\boldsymbol{x}_i^f \in \mathbb{R}^{D_f}$. Note that (5) and (6), where we assume independence between $\boldsymbol{c}_i$ and $\boldsymbol{x}_i^p$, are analogous to the prior on $\boldsymbol{z}$ in the CVAE above, (1). Moreover, the intuition behind the mixture prior is to facilitate clustering of the latent space in a meaningful way.

Since the factorized decoder in (7) is a composition of various likelihood models, we can use one likelihood function per input, giving rise to modelling data with real-valued, positive real valued, count, categorical and ordinal values, concurrently. Additionally, the modelling framework lets us specify a variety of interval constraints by choosing likelihoods appropriately (e.g. truncated normal distribution or Beta distribution for interval data).

Factorized encoder. Then the factorized encoder is given by:

$$\begin{aligned} q(\boldsymbol{c}_i|\boldsymbol{x}_i^p, \boldsymbol{x}_i^f) &= Cat(\boldsymbol{\pi}(\boldsymbol{x}_i^p, \boldsymbol{x}_i^f)) \\ q(z_i|\boldsymbol{x}_i^f, \boldsymbol{x}_i^p, \boldsymbol{c}_i) &= \mathcal{N}(\boldsymbol{\mu}_q(\boldsymbol{x}_i^f, \boldsymbol{x}_i^p, \boldsymbol{c}_i), \Sigma_q(\boldsymbol{x}_i^f, \boldsymbol{x}_i^p, \boldsymbol{c}_i)) \\ q(z_i, \boldsymbol{x}_i^f, \boldsymbol{c}_i|\boldsymbol{x}_i^p) &= q(z_i, \boldsymbol{c}_i|\boldsymbol{x}_i^p) \prod_{d=1}^{D_f} p(\boldsymbol{x}_{d,i}^f|z_i, \boldsymbol{c}_i, \boldsymbol{x}_i^p) \\ &= q(z_i|\boldsymbol{c}_i, \boldsymbol{x}_i^p)q(\boldsymbol{c}_i|\boldsymbol{x}_i^p) \cdot \prod_{d=1}^{D_f} p(\boldsymbol{x}_{d,i}^f|z_i, \boldsymbol{c}_i, \boldsymbol{x}_i^p). \end{aligned} \tag{8}$$

Parameter sharing and likelihood models. Unlike in the vanilla CVAE in (1), which is only suitable for one data type at the time, the decoder was factorized into multiple likelihood models. In practice, one needs to carefully specify one likelihood model per input dimension $p(\boldsymbol{x}_{d,i}^f|z_i, \boldsymbol{c}_i)$. In our github repository, we describe more details of the model architecture and which likelihood models we have chosen.

ELBO. The evidence lower bound (ELBO) can be derived as:

$$\begin{aligned} \log p(\boldsymbol{x}^f|\boldsymbol{x}^p) \geq\ & \mathbb{E}_{q(\boldsymbol{c}_i, z_i|\boldsymbol{x}_i^f, \boldsymbol{x}_i^p)} \sum_{d=1}^{D_f} \log p(\boldsymbol{x}_{d,i}^f|z_i, \boldsymbol{c}_i, \boldsymbol{x}_i^p) \\ & - \sum_i \mathbb{E}_{q(\boldsymbol{c}_i|\boldsymbol{x}_i^f, \boldsymbol{x}_i^p)} \mathrm{D}_{KL}[q(z_i|\boldsymbol{c}_i, \boldsymbol{x}_i^f, \boldsymbol{x}_i^p)||p(z_i|\boldsymbol{c}_i, \boldsymbol{x}_i^p)] \\ & - \sum_i \mathrm{D}_{KL}[q(\boldsymbol{c}_i|\boldsymbol{x}_i^f, \boldsymbol{x}_i^p)||p(\boldsymbol{c}_i|\boldsymbol{x}_i^p)] \end{aligned}$$

where we recognize the influence of the factorized decoder in the first line, effectively allowing us to model complex, heterogeneous data distributions.

4.3 Counterfactual search algorithm

As inputs, our algorithm requires any pretrained classifier f and the trained decoder and encoder from the CHVAE. It returns the closest $E(\boldsymbol{x})$ due to a nearest neighbour style search in the latent space. The details can be found in our github repository, but it uses a standard procedure to generate random numbers distributed uniformly over a sphere [6, 12] around the latent observation $\hat{\boldsymbol{z}}$. Thus, we sample observations $\tilde{z}$ in l_p-spheres around the point $\hat{\boldsymbol{z}}$ until we find a counterfactual explanation $\tilde{\boldsymbol{x}}^*$.

5 EVALUATING ATTAINABILITY OF COUNTERFACTUALS

To quantify faithfulness, [11] suggest two measures, which we shortly review here since they do not belong to the catalog of commonly used evaluation measures (such as for example accuracy). Their two suggested measures quantify *proximity* (i.e. whether $E(\boldsymbol{x})$ is a local outlier) and *connectedness* (i.e. whether $E(\boldsymbol{x})$ is connected to other correctly classified observations from the same class). However, these measures do not indicate the *degree of difficulty* for the individual to attain a certain counterfactual given the current state. We suggest two appropriate measures in 5.2.

5.1 Counterfactual faithfulness

Proximity. Ideally, the distance between a counterfactual explanation $E(\boldsymbol{x})$ and its closest, non-counterfactual neighbour $\boldsymbol{a}_0 \in H^+ \cap D^+$ should be small:

$$\boldsymbol{a}_0 = \underset{\boldsymbol{x} \in H^+ \cap D^+}{\arg\min}\; d(E(\boldsymbol{x}), \boldsymbol{x}).$$

Moreover, it is required that the observation resembling our counterfactual, $\boldsymbol{a}_0$, be close to the rest of the data, which gives rise to the following relative metric:

$$P(E(\boldsymbol{x})) = \frac{d(E(\boldsymbol{x}), \boldsymbol{a}_0)}{\min\limits_{\boldsymbol{x} \neq \boldsymbol{a}_0 \in H^+ \cap D^+} d(\boldsymbol{a}_0, \boldsymbol{x})}.$$

The intuition behind this measure is to help evaluate whether counterfactuals are outliers relative to correctly classified observations.

Connectedness. We say that that a counterfactual $\boldsymbol{e}$ and an observation $\boldsymbol{a}$ are ϵ-chained, with $\epsilon > 0$, if there exists a sequence $\boldsymbol{e}_0, \boldsymbol{e}_1, ..., \boldsymbol{e}_N \in \mathcal{X}$ such that $\boldsymbol{e}_0 = \boldsymbol{e}, \boldsymbol{e}_N = \boldsymbol{a}$ and $\forall i < N, d(\boldsymbol{e}_i, \boldsymbol{e}_{i+1}) < \epsilon$ and $f(\boldsymbol{e}) = f(\boldsymbol{a})$. Now, given an appropriate value for ϵ, we can evaluate the connectedness of a counterfactual $E(\boldsymbol{x})$ using a binary score: $C(E(\boldsymbol{x})) = 1$, if $E(\boldsymbol{x})$ is ϵ-connected to $\boldsymbol{a} \in H^+ \cap D^+$ and $C(E(\boldsymbol{x})) = 0$, otherwise.

5.2 Degree of difficulty

Individual costs of counterfactuals. We suggest to measure the degree of difficulty of a certain counterfactual suggestion $\tilde{\boldsymbol{x}}$ in terms of the percentiles of $x_d^f = \{x_{i,d}\}_{i=1}^N$ and $\tilde{x}_d^{f*}$: $Q_j(\tilde{x}_d^{f*})$ and $Q_d(x_d^f)$ where $Q_d(\cdot)$ is the cumulative density function of x_d^f. As an example, a cost of p suggests changing a free feature by at least p percentiles to receive a desired result.

We suggest two measures with the following properties: (a) $cost(x_d^f; x_d^f) = \mathbf{0}_{N\times 1}$, implying that staying at the current state is costless and (b) $cost(\tilde{x}^f + \nu \mathbf{1}_{N\times 1}; x^f) \geq cost(\tilde{x}^f; x^f)$ with $\nu \geq 0$, that is, the further from the current state, the more difficulties we have to incur to achieve the suggestion. The *difficulty measures* then read as follows:

$$cost_1(\tilde{x}^{f*}; x^f) = \sum_{d=1}^{D_f} |(Q_d(\tilde{x}_d^{f*}) - Q_d(x_d^f)|, \tag{9}$$

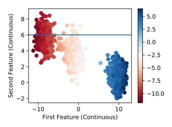
$$cost_2(\tilde{x}^{f*}; x^f) = \max_d |Q_d(\tilde{x}_d^{f*}) - Q_d(x_d^f)|. \tag{10}$$

The total percentile shift (TS) in (9) can be thought of as a baseline measure for how attainable a certain counterfactual suggestion might be. The maximum percentile shift (MS) in (10) across all free features reflects the maximum difficulty across all mutable features.

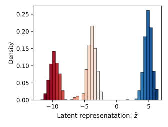
6 EXPERIMENTS

6.1 Synthetic experiments

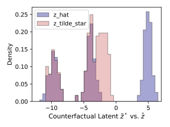
Homogeneous features. We begin by describing a data generating processes (DGP) for which it can be difficult to identify faithful counterfactuals. Example 1 corresponds to the case when all features are numerical. We generate 10000 observations from this DGP. We assume that the constant classifier $I(x_2 > 6)$ is given to us and our goal is to find counterfactuals for observations with 0-labels.



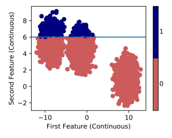
(a) Reconstructed train data generated by different $\hat{z}$ (coloured).



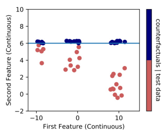
(b) Density of $\hat{\boldsymbol{z}}$. Colours aligned so that left red $\hat{z}$ generates left $\hat{x}$ in 2(a).



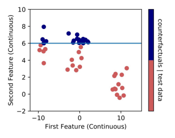
(c) Density of $\hat{z}$ from 2(b) in blue. Density of $\tilde{z}^*$ (red) belongs to E(x) from 2(f) .



(d) True DGP.



(e) Test data and $E(\boldsymbol{x})$ by GS/AR (not shown). Upper right $E(\boldsymbol{x})$ lie where no data is expected.



(f) Test data and $E(\boldsymbol{x})$ by our cchvae. Most $E(\boldsymbol{x})$ lie in high-density areas and are connected.

Figure 2: Example 1. Homogeneous features. Figure 2(c) shows that generating close and meaningful counterfactuals amounts to finding the closest latent code from only 2 of the 3 modes of the latent distribution.

Figure 2(a) shows the reconstructed training data. The true DGP is shown in figure 2(d).

EXAMPLE 1 (***MAKE BLOBS***). *We generate* $\boldsymbol{x} = [x_1, x_2]$ *from a mixture of 3 Gaussians with* $\boldsymbol{\mu} = [\mu_1, \mu_2, \mu_3]$ *and* $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \sigma_3] = [1, 1, 1]$ *with a fixed seed. The response y is generated from* $Pr(y = 1|X) = I(x_2 > 6)$, *where* $I(\cdot)$ *denotes the indicator function.*

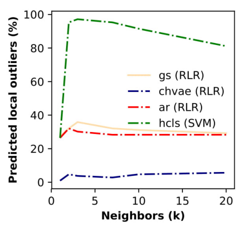
Figure 2(e) shows test data and their generated counterfactuals from AR and GS. For values from the lower right (blue) cluster in figure 2(a), both AR and GS suggest $E(\boldsymbol{x})$ that lie in the top right corner (figure 2(e)). Since both GS and AR generate almost identical values, we report the results for GS only. AR and GS favour sparse $E(\boldsymbol{x})$, meaning they only require changes along the second feature axis. However, it is apparent that the upper right corner $E(\boldsymbol{x})$ are not attainable – according to the DGP no data lives in this region. In contrast, our C-CHVAE suggests $E(\boldsymbol{x})$ that lie in regions of high data density, figure 2(f).

To gain a better understanding of our method consider figure 2(b). It shows the density of the estimated latent variable $\hat{z}$. The colours correspond to the clusters in the reconstructed data of figure 2(a). In figure 2(c), the counterfactual latent density $\tilde{z}^*$, i.e. the density of the latent variables from the counterfactuals $\tilde{x}^*$, is depicted on top of the density of $\hat{z}$. It shows that the density of $\tilde{z}^*$ is concentrated on the two modes which generate data that lies close to the decision boundary of the DGP.

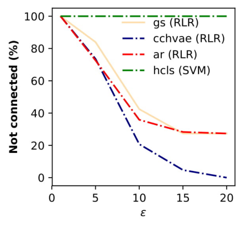
6.2 Real world data sets

For our real world experiments we choose 2 credit data sets; a processed version of the "`Give me some credit`" data set and the *Home Equity Line of Credit* (`HELOC`) data set.[1][2] For the former, the

[1] https://www.kaggle.com/brycecf/give-me-some-credit-dataset.
[2] https://community.fico.com/s/explainable-machine-learning-challenge.

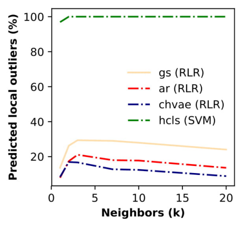


(a) Local outlier factor score

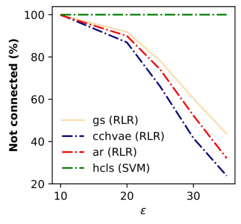


(b) Connectedness score

Figure 3: Faithfulnes relative to $x \in H^+ \cap D^+$ for GMSC.

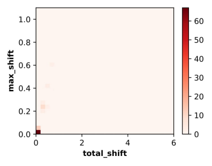


(a) Local outlier factor score

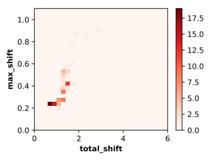


(b) Connectedness score

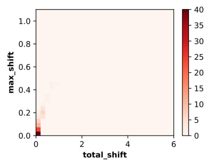
Figure 4: Faithfulness relative to $x \in H^+ \cap D^+$ for HELOC data.



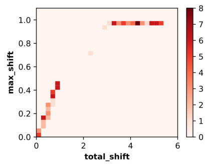
(a) AR



(b) C-CHVAE

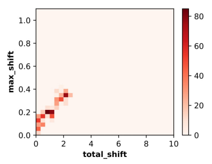


(c) GS

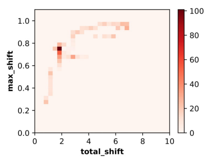


(d) HCLS

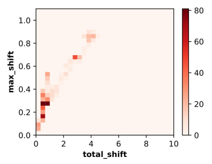
Figure 5: Total shift vs. max. shift for $E(x)$ on GMSC data.



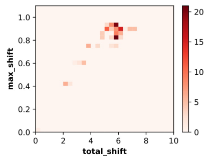
(a) AR



(b) C-CHVAE



(c) GS



(d) HCLS

Figure 6: Total shift vs. max. shift for $E(x)$ on HELOC data.

target variable records whether individuals experience financial distress within a period of two years, in the latter case one uses the applicants' information from credit reports to predict whether they will repay the HELOC account within a fixed time window. Both data sets are standard in the literature [4, 17, 21] and are described in more detail in our github repository.

While GS works for different classifiers, the AR and HCLS algorithms do not. To also compare our results with AR we follow Ustun et al. [21] and choose an ℓ_2-penalized logistic regression model. For HCLS, we use SVM with a linear kernel when possible.

***"Give Me Some Credit"** (GMSC).* For this data set, GS and AR produce very similar results in terms of faithfulness. HCLS performs worst and C-CHVAE (our's) outperforms all other methods. In terms of the local outlier score, the difference gets as high as 20 percentage points (figure 3(a)). With respect to the connectedness score the difference grows larger for large ϵ (figure 3(b)). In terms of *difficulty*, it the C-CHVAE's faithfully generated counterfactuals come at the cost of greater TS and MS (figure 5).

HELOC. With respect to *counterfactual faithfulness*, the C-CHVAE outperforms all other methods for both measures and all parameter choices (figure 4). Again, HCLS is not performing well; one reason could lie in the fact that one needs to specify the directions in which all free features are allowed to change. This seems to require very careful choices. Moreover, it is likely to restrict the counterfactual suggestions, leading to counterfactuals that might look less typical, which is what *faithfulness* measures. In terms of *difficulty*, the pattern is similar to the one above (see figure 6). The C-CHVAE tends to make suggestions with higher MS. This time, to obtain *faithful counterfactuals* we are paying a price in terms of higher MS.

7 CONCLUSION AND FUTURE WORK

We have introduced a general-purpose framework for generating counterfactuals; in particular, the fact that our method works for tabular data without the specification of distance or cost functions in the input space allows practitioners and researchers to adapt this work to a wide variety of applications. To do so, several avenues for future work open up. First, all existing methods make recommendations of how features would need to be altered to receive a desired result, but none of these methods give associated input importance. And second, it would be desirable to formalize the tradeoff between the autoencoder capacity and counterfactual faithulness.

ACKNOWLEDGMENTS

We would like to thank Lars Holdijk and Michael Lohaus for insightful discussions and Alfredo Nazabal for his assistance in running the HVAE.

REFERENCES

[1] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. 2019. A reductions approach to fair classification. In *ICML*.

[2] Naveed Akhtar and Ajmal Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* 6 (2018), 14410–14430.

[3] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665* (2017).

[4] Rory Mc Grath, Luca Costabello, Chan Le Van, Paul Sweeney, Farbod Kamiab, Zhao Shen, and Freddy Lecue. 2018. Interpretable Credit Application Predictions With Counterfactual Explanations. *NeurIPS workshop: Challenges and Opportunities for AI in Financial Services: the Impact of Fairness, Explainability, Accuracy, and Privacy* (2018).

[5] Nina Grgic-Hlaca, Elissa M Redmiles, Krishna P Gummadi, and Adrian Weller. 2018. Human perceptions of fairness in algorithmic decision making: A case study of criminal risk prediction. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 903–912.

[6] Radoslav Harman and Vladimír Lacko. 2010. On decompositional algorithms for uniform sampling from n-spheres and n-balls. *Journal of Multivariate Analysis* 101, 10 (2010), 2297–2304.

[7] Oleg Ivanov, Michael Figurnov, and Dmitry Vetrov. 2018. Variational Autoencoder with Arbitrary Conditioning. *arXiv preprint arXiv:1806.02382* (2018).

[8] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. 2019. Towards Realistic Individual Recourse and Actionable Explanations in Black-Box Decision Making Systems. *arXiv preprint arXiv:1907.09615* (2019).

[9] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *Proceedings of the 2nd International Conference on Learning Representations (ICLR)* (2013).

[10] Michael T Lash, Qihang Lin, Nick Street, Jennifer G Robinson, and Jeffrey Ohlmann. 2017. Generalized inverse classification. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 162–170.

[11] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, and Marcin Detyniecki. 2019. Issues with post-hoc counterfactual explanations: a discussion. *ICML Workshop on Human in the Loop Learning* (2019).

[12] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. 2017. Inverse Classification for Comparison-based Interpretability in Machine Learning. *arXiv preprint arXiv:1712.08443* (2017).

[13] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* (2015).

[14] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. 2018. Handling incomplete heterogeneous data using VAEs. *arXiv preprint arXiv:1807.03653* (2018).

[15] Martin Pawelczyk, Johannes Haug, Klaus Broelemann, and Gjergji Kasneci. 2019. Towards User Empowerment. *NeurIPS Workshop on Human-Centric Machine Learning* (2019).

[16] Vera Regitz-Zagrosek. 2012. Sex and gender differences in health. *EMBO reports* 13, 7 (2012), 596–603.

[17] Christopher Russell. 2019. Efficient Search for Diverse Coherent Explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. ACM FAT, 20–28.

[18] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*. 3483–3491.

[19] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. 2017. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 465–474.

[20] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. 2017. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558* (2017).

[21] Berk Ustun, Alexander Spangher, and Yang Liu. 2019. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. ACM, 10–19.

[22] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: automated decisions and the GDPR. *Harvard Journal of Law & Technology* 31, 2 (2017), 2018.

[23] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. 2017. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1171–1180.

A COUNTERFACTUAL SEARCH – ALGORITHMS

A.1 Counterfactual search algorithm

As inputs, the algorithm requires any pretrained classifier f and the trained decoder and encoder from the CHVAE. It returns the closest counterfactual. We note algorithm 1 uses a standard procedure to generate random numbers distributed uniformly over a sphere. Laugel et al. [12] use a similar algorithm, but relative to their work, we look for the smallest change in the *latent representation z* (not in input space) that would lead to a change in the predicted label. Thus, we sample observations $\tilde{z}$ in l_p-spheres around the point $\hat{z}$ until we find a counterfactual $\tilde{x}^*$. For positive numbers r_1 and r_2, we define a (r_1, r_2)-sphere around $\hat{z}$:

$$S(\hat{z}, r_1, r_2) = \{\tilde{z} \in \mathcal{Z} : r_1 \leq \|\hat{z} - \tilde{z}\| \leq r_2\}. \tag{11}$$

In order to generate uniform random numbers over a sphere, we also use the YPHL algorithm [6]. Their algorithm allows us to generate observations uniformly distributed over the unit-sphere. Next, one draws observations uniformly from $U[r_1, r_2]$, which are in turn used to rescale the distance between uniform sphere values and $\hat{z}$. Eventually, we arrive at observations $\tilde{z}$ that are uniformly distributed over $S(\hat{z}, r_1, r_2)$.

Algorithm 1 shows a counterfactual search procedure when the latent variable has a dense distribution. It is straightforward to adjust the algorithm to scenarios when one desires to generate multiple counterfactual examples, also known as *flip sets* [17, 21]. The idea is that the user can choose one counterfactual from a menu of different counterfactuals, which fits her preferences best.

B COMMON LIKELIHOOD MODELS

For the sake of completeness we enumerate a list of commonly used likelihood models for numerical and nominal features [14]:

- **Real-valued data**. For real valued data, one usually assumes a Gaussian likelihood model such as,
$$p(\boldsymbol{x}_{i,d}|\boldsymbol{\gamma}_{i,d}) = \mathcal{N}(\mu_d(z_i), \sigma_d^2(z_i)),$$
where $\boldsymbol{\gamma}_{i,d} = \{\mu_d(z_i), \sigma_d^2(z_i)\}$ are modelled by the outputs of a DNN with inputs z.
- **Positive real-valued data**. For positive real valued data, one can assume a log normal likelihood model such as,
$$p(\boldsymbol{x}_{i,d}|\boldsymbol{\gamma}_{i,d}) = \log \mathcal{N}(\mu_d(z_i), \sigma_d^2(z_i)),$$
where $\boldsymbol{\gamma}_{i,d} = \{\mu_d(z_i), \sigma_d^2(z_i)\}$.
- **Count data**. For count data, one can assume a Poisson likelihood model such as,
$$p(\boldsymbol{x}_{i,d}|\boldsymbol{\gamma}_{i,d}) = Poisson(\lambda_d(z_i)),$$
where $\boldsymbol{\gamma}_{i,d} = \{\lambda_d(z_i)\}$.
- **Ordinal data**. For ordinal valued data, we use the same procedure as in [14].
- **Categorical data.** For categorical data one can assume a multinomial logit model, where the probability of every category r is given by
$$p(\boldsymbol{x}_{i,d} = r|\boldsymbol{\gamma}_{i,d}) = \frac{\exp^{(h_{d_r}(z_i))}}{\sum_{r=1}^{R} \exp^{(h_{d_r}(z_i))}},$$
with parameters $\boldsymbol{\gamma}_{i,d} = \{h_{d_0}(\boldsymbol{z}_i), h_{d_1}(\boldsymbol{z}_i), ..., h_{d_{R-1}}(\boldsymbol{z}_i)\}$ and $h_{d_0}(z_i) = 0$ to ensure identifiablity.

Algorithm 1 Stochastic Counterfactual Search For Latent Space

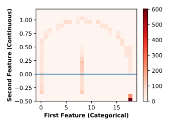
```
Input: $X_{train}$: training data; $x_{i,test}$: test observation; $f$: classifier trained on $X_{train}$; $m_{\hat{\theta}}, g_{\hat{\phi}}$: CHVAE encoder and decoder trained on $X_{train}$; $S$: number search samples; $\Delta r$: search radius.
Initialize: $f(x_{i,test}) = \hat{y}_{i,test}$; $m_{\hat{\theta}}(x_{i,test}) = \hat{z}_{i,test}$; $r = 0$; $C = \varnothing$; $\hat{z}_{train,min} = \min_i \hat{z}_{i,train}$; $\hat{z}_{train,max} = \max_i \hat{z}_{i,train}$.

while $C = \varnothing \ \wedge \tilde{z}_{i,test} \in [\hat{z}_{train,min}, \hat{z}_{train,max}]$ do
  for $j = 1$ to $J$ do
    sample $\tilde{z}^j_{i,test}$ from $S(\hat{z}_{i,test}, r, \Delta r)$ in (11) {Perturbed representation}
    $\tilde{x}^j_{i,test} = g_{\hat{\phi}}(\tilde{z}^j_{i,test})$ {Potential counterfactual}
    $\tilde{y}^j_{i,test} = f(\tilde{x}^j_{i,test})$
    if $\tilde{y}^j_{i,test} \neq \hat{y}_{i,test}$ then
      $C \leftarrow (\tilde{z}^j_{i,test}, \tilde{x}^j_{i,test}, \tilde{y}^j_{i,test})$
    end if
  end for
  if $C = \varnothing$ then
    $r = r + \Delta r$ {Push search range outward}
  else if $\tilde{z}_{i,test} \notin [\hat{z}_{train,min}, \hat{z}_{train,max}]$ then
    Return: {No counterfactual consistent with data distribution}
    $C = \varnothing$
  else
    Return: {Find 'closest' counterfactual}
    $\tilde{z}^*_{i,test} = \text{argmin}_{\tilde{z}_{test} \in C} ||\tilde{z}_{test} - \hat{z}_{test}||$
    $\tilde{x}^*_{i,test} = g_{\hat{\phi}}(\tilde{z}^*_{i,test})$, $\tilde{y}^*_{i,test} = f(\tilde{x}^*_{i,test})$
  end if
end while
```

C SYNTHETIC EXAMPLE

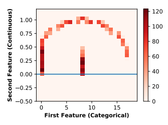
Example 2 (***Discretized Make Moons***). *We generate the upper have circle $[x_1, x_2]$-pairs by $cos(i)$ for $i \in (0, \pi)$, which we round to the closest decimal. The lower have circle $[x_1, x_2]$-pairs are then generated by 1 - $sin(i)$ for $i \in (0, \pi)$, where we round $sin(i)$ to the closest integer. Both the upper and the lower half contain half of the observations each and x_2 is treated as categorical with 19 categories. The response y is then generated from $Pr(y = 1|X) = I(x_1 > 0)$.*

Heterogeneous features. Figures 7(g), 7(a) and 7(b) depict the true data generating process with corresponding distribution of labels, corresponding 2d-histogram and test observations from the 0-class. Figure 7(c) depicts the 2d-histogram from the reconstructed test data. Despite the simplistic class assignment, finding attainable counterfactuals might not be trivial in this case since the data density is very fragmented.

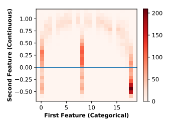
Next, figures 7(d)-7(f) depict 25 counterfactuals generated using AR, HCLS and our C-CHVAE, respectively. For AR, counterfactuals appear in the 1st and the 9th category. For GS, counterfactuals appear in all categories. For our C-CHVAE, counterfactuals appear



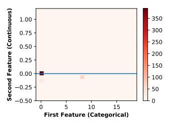
(a) 2-d histogram of data generating process



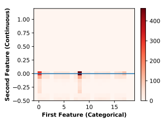
(b) Test data 2d-histogram from 0-class.



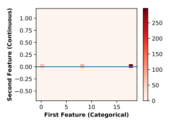
(c) Reconstructed training data.



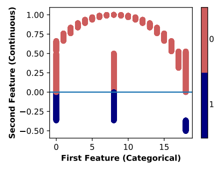
(d) $E(\boldsymbol{x})$ by ar.



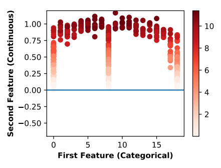
(e) $E(\boldsymbol{x})$ by gs.



(f) $E(\boldsymbol{x})$ by cchvae.



(g) Data generating process.



(h) Test data cond. on Δz.

Figure 7: Example 2. Heterogeneous features. AR generates counterfactuals $E(\boldsymbol{x})$ from 1st and 9th category. GS generates counterfactuals from all categories. CCHVAE generates counterfactuals from 1st, 9th and 19th category. Figure 7(h) shows test data conditional on $\Delta z = \tilde{z}^* - \hat{z}$, which indicates how much $\hat{z}$ needs to change to alter the prediction.

in the 1st, 9th and 19th category. The model correctly produces counterfactuals for all categories.

D DATA

D.1 Real world example: "Give Me Some Credit"

In the following, we list the specified pretrained classification models as well as the parameter specification used for the experiments. We use 80 percent of the data as our training set and the remaining part is used as the holdout test set. Additionally, we allow f access to all features, i.e. $f(\boldsymbol{x}^f, \boldsymbol{x}^p)$. The state of features can be found in table 3.

AR *[21].* The AR algorithm requires to choose both an action set and free and immutable features. The implementation can be found here: https://github.com/ustunb/actionable-recourse. We specify that the *DebtRatio* feature can only move downward [21]. The AR implementation has a default decision boundary at 0 and therefore one needs to shift the boundary. We choose $p_{AR} = 0.50$, adjusting the boundary appropriately. Finally, we set the linear programming optimizer to *cbc*, which is based on an open-access python implementation. As f, we choose the l_2-regularized logistic regression model.

Feature	Free	Model	Dir. (HCLS)
Revolving Utilization Of Unsecured Lines	Y	log Normal	↓
Age	N	Poisson	
Number Of Times 30-59 Days Past Due Not Worse	Y	Poisson	↓
Debt Ratio	Y	log Normal	↓
Monthly Income	Y	log Normal	↑
Number Open Credit Lines And Loans	Y	Poisson	↓
Number Of Times 90 days Late	Y	Poisson	indirect
Number Real Estate Loans Or Lines	Y	Poisson	↓
Number Of Times 60-89 Days Past Due Not Worse	Y	Poisson	↓
Number Of Dependents	N	Poisson	

Table 3: "Give Me Some Credit": State of features and likelihood models.

GS *[12].* GS is based on a version of the YPHL algorithm described above. As such we have to choose appropriate step sizes in our implementation to generate new observations from the sphere around $\boldsymbol{x}$. We choose a step size of 0.1. As f, we choose the l_2-regularized logistic regression model.

HCLS *[10].* In our experiment we used their baseline MATLAB implementation, which can be found here: github.com/michael-lash/BCIC. HCLS requires us to choose a budget, which we set to 10. It also requires to choose a cost associated with changing each feature. We set it equal to 1 for all features. As f, we choose SVM with the Gaussian kernel, which delivered good results in reasonable time. Initially, we tried to choose the linear kernel, but after training for several hours with no convergence, we decided against it. We also experimented with different standardization forms (minmax standardization, z-score standardization), which did not help. For the evaluation metric, we choose accuracy and we used a *balance* option that weighs each individual sample inversely proportional to class frequencies in the training data. We had to specify an indirectly changeable feature, which we set to *NumberOfTimes90daysLate*. Finally, we had to choose the direction (Dir.(HCLS) in table 3) in which *every* free feature is allowed to move.

C-CHVAE *(ours).* For our algorithm we made the following choices. We set the latent space dimension of both $\boldsymbol{s}$ and $\boldsymbol{z}$ to 5 and 6, respectively. For training, we used 50 epochs. Table 3 gives details about the chosen likelihood model for each feature. For count features, we use the Poisson likelihood model, while for features with a support on the positive part of the real line we choose log normal distributions. As f, we choose the l_2-regularized logistic regression model.

D.2 Real world example: HELOC

The *Home Equity Line of Credit (HELOC)* data set consists of credit applications made by homeowners in the US, which can be obtained from the FICO community.[3] The task is to use the applicant's information within the credit report to predict whether they will repay the HELOC account within 2 years. Table 4 gives an overview of the available features and the corresponding assumed likelihood models.

[3]https://community.fico.com/s/explainable-machine-learning-challenge?tabset-3158a=2.

Feature	Free	Model	Dir. (HCLS)
MSinceOldestTradeOpen	N	Poisson	
AverageMInFile	N	log Normal	
NumSatisfactoryTrades	Y	Poissonl	↑
NumTrades60Ever/DerogPubRec	Y	log Normal	↓
NumTrades90Ever/DerogPubRec	Y	log Normal	indirect
NumTotalTrades	Y	Poisson	↓
PercentInstallTrades	Y	log Normal	↑
MSinceMostRecentInqexcl7days	Y	Poisson	↓
NumInqLast6M	Y	Poisson	↓
NetFractionRevolvingBurden	Y	log Normal	↓
NumRevolvingTradesWBalance	Y	Poisson	↑
NumBank/NatlTradesWHighUtilization	Y	log Normal	↑
ExternalRiskEstimate	N	log Normal	
MPercentTradesNeverDelq	Y	log Normal	↓
MaxDelq2PublicRecLast12M	Y	Poisson	↓
MaxDelqEver	Y	Poisson	↓
NumTradesOpeninLast12M	Y	Poisson	↓
NumInqLast6Mexcl7days	Y	Poisson	↓
NetFractionRevolvingBurden	Y	Poisson	↓
NumInstallTradesWBalance	Y	Poisson	↑
NumBank2NatlTradesWHighUtilization	Y	Poisson	↓
PercentTradesWBalance	Y	log Normal	↑

Table 4: HELOC: State of features and likelihood models.

AR and GS. As before. Additionally, we do not specify how features have to move.

HCLS. As f, we choose SVM with the linear kernel. We specified *NumTrades90Ever/DerogPubRec* as the indirect feature. Again, we had to specfiy which directions features move, which we indicated in the 'Direction' column of table 4.

***C-CHAVE** (ours).* For our algorithm we made the following choices. We set the latent space dimension of both $\boldsymbol{s}$ and $\boldsymbol{z}$ to 1 and 10, respectively. For training, we used 60 epochs. Table 4 gives details about the chosen likelihood model for each feature. The rest remains as before.

A.2 Decomposing Counterfactual Explanations for Consequential Recommendations

Publication: Published at the ICLR 2022 workshop on socially responsible Machine Learning (SRML). Under review at UAI 2023.

Contribution: I developed the method, wrote the implementation, performed the experiments and developed the theory. Moreover, I wrote all parts of the paper. Lea Tiyavorabun developed a model agnostic version of our algorithm which is not featured in the main paper. Gjergji Kasneci made valuable suggestions by challenging and improving concepts, ideas and the presentation as well as by revising the manuscript.

Decomposing Counterfactual Explanations for Consequential Decision Making

Martin Pawelczyk
University of Tübingen

Lea Tiyavorabun
University of Amsterdam

Gjergji Kasneci
University of Tübingen

Abstract

The goal of algorithmic recourse is to reverse unfavorable decisions (e.g., from loan denial to approval) under automated decision making by suggesting actionable feature changes (e.g., reduce the number of credit cards). To generate low-cost recourse the majority of methods work under the assumption that the features are independently manipulable (IMF). To address the feature dependency issue the recourse problem is usually studied through the causal recourse paradigm. However, it is well known that strong assumptions, as encoded in causal models and structural equations, hinder the applicability of these methods in complex domains where causal dependency structures are ambiguous. In this work, we develop `DEAR` (DisEntangling Algorithmic Recourse), a novel and practical recourse framework that bridges the gap between the IMF and the strong causal assumptions. `DEAR` generates recourses by disentangling the latent representation of co-varying features from a subset of promising recourse features to capture the main practical recourse desiderata. Our experiments on real-world data corroborate our theoretically motivated recourse model and highlight our framework's ability to provide reliable, low-cost recourse in the presence of feature dependencies.

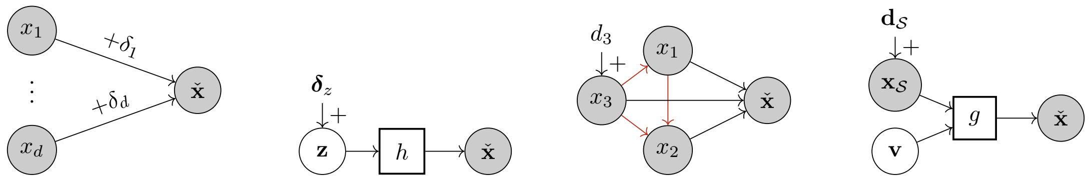
1 Introduction

Counterfactual explanations provide a means for actionable model explanations at feature level. Such explanations, which have become popular among legal and technical communities, provide both an explanation and an instruction: the former emphasizes why a certain machine learning (ML) prediction was produced; the latter gives an instruction on how to act to arrive at a desirable outcome.

Several approaches in recent literature tackled the problem of providing recourses by generating counterfactual explanations [41, 37]. For instance, Wachter et al. [41] proposed a gradient based approach which finds the nearest counterfactual resulting in the desired prediction. Pawelczyk et al. [27] proposed a method which uses a generative model to find recourses in dense regions of the input space. More recently, Karimi et al. [15] advocated for considering causal structure of the underlying data when generating recourses to avoid spurious explanations. Yet, despite their popularity these works are not without drawbacks: (i) Wachter et al. [41] implicitly assume that the input features can be independently manipulated, (ii) Pawelczyk et al. [27] narrowly focuse on generating recourse in dense regions of the input space, and (iii) Karimi et al. [15] require a correct specification of the causal graph and structural equation models.

For many practical use cases the strong causal assumptions constitute the limiting factor when it comes to the deployment of causal recourse methods. On the other hand, most of the practical approaches implicitly make the *independently manipulable feature* (IMF) assumption ignoring feature dependencies. Therefore critiques of counterfactual explanations and algorithmic recourse have highlighted the feature dependency issue [2, 39]: in a nutshell, *changing one feature will likely change others*. For instance, a recourse system might ask to increase the feature 'income' for a loan approval. However, there might be several ways of achieving the same desired outcome of loan approval: either one could increase 'income' through a promotion or one could find a new role in a different company. In the former case, the value of the variable reflecting 'time on job' would go up, which would likely amplify the model's output towards the desirable outcome. In the latter case, however, the model's output would likely swing towards a loan rejection, since the short 'time on job' opposes the positive influence of the 'income' increase.

The fundamental drawbacks of these recourse paradigms motivate the need for a new recourse framework (see Figure 1): (i) The framework should allow recourses to adhere to feature dependencies without relying on causal models. (ii) It should also enable recourses to lie in dense regions of the data distribution. (iii) Finally, it should ensure that recourses



(a) IMF recourse (b) Manifold-based recourse (c) Causal recourse (d) Our recourse framework

Figure 1: **The spectrum of recourse frameworks**. Illustrating the different assumptions underlying each recourse framework. (a): Recourses are found by neglecting input dependencies (e.g., [41]). (b): Actions made to the latent code $\mathbf{z}$ generate recourse using a generative model h and neglect control over feature costs (e.g., [27]). (c): Recourses are found by having the decision maker come up with a causal model between the input features (illustrated by the red directed edges) (e.g., [15]). (d): Our framework bridges this gap by allowing a generative model g to be influenced by a subset of inputs $\mathbf{x}_{\mathcal{S}}$. This enables (i) generation of counterfactuals in dense regions of the input space, and (ii) modeling of feature dependencies (iii) without the reliance on causal graphical models.

are attainable at low and controllable cost by the individual. Combining these requirements in one recourse system poses a severe challenge to making algorithmic recourse practicable in the real world. In this work, we address this critical problem in the face of these three challenges by formulating the problem of algorithmic recourse using a new framework called `DEAR` (DisEntangling Algorithmic Recourse). Our framework exploits a generative model and uses techniques from the disentanglement literature to capture the main practical desiderata (i) – (iii). Our key contributions can be summarized as follows:

- **Novel recourse framework.** Our framework generates recourses by disentangling the latent representation of co-varying features with indirect impact on the recourse from a subset of promising recourse features.
- **Interpretable recourse costs.** As a byproduct of our framework, we show that recourse actions can be divided into two types of actions: *direct* and *indirect actions*, which can be exploited to lower the cost of algorithmic recourse.
- **Constructive theoretical insights.** We develop theoretical expressions for the costs of recourse which guide the design of our generative model and contribute to reliably find low cost algorithmic recourse.
- **Extensive experiments.** Our experimental evaluations on real-world data sets demonstrate that `DEAR` generates significantly less costly and at the same time more realistic recourses than previous manifold based approaches [1, 31, 27, 12].

2 Related Work

Our work builds on a rich literature in the field of algorithmic recourse. We discuss prior works and the connections to this research.

Algorithmic approaches to recourse. As discussed earlier, several approaches have been proposed in literature to provide recourse to individuals who have been negatively impacted by model predictions, e.g., [36, 18, 5, 41, 37, 12, 38, 27, 23, 25, 13, 32, 14, 4, 1, 35, 31]. These approaches can be roughly categorized along the following dimensions [40]: *type of the underlying predictive model* (e.g., tree based [36, 21, 26] vs. differentiable classifier [41]), *type of access* they require to the underlying predictive model (e.g., black box [18, 9] vs. gradients [1]), whether they encourage *sparsity* in counterfactuals (i.e., only a small number of features should be changed [16, 13, 34]), whether counterfactuals should lie on the *data manifold* [12, 27, 23, 1, 8, 17, 42], whether the underlying *causal relationships* should be accounted for when generating counterfactuals [15, 14], whether the output produced by the method should be *multiple diverse counterfactuals* (e.g., [33, 25]) or a single counterfactual, and whether the underlying *task* is posed as a regression (e.g., [4, 35]) or classification problem.

While there have been few recent works that consider input dependencies in algorithmic recourse problems, these works require strong causal assumptions [15, 14]. For practical use cases, such strong causal assumptions constitute the limiting factor when it comes to the deployment of these models. In contrast, our work makes the first attempt at tackling the problem of generating recourses in the presence of feature dependencies while not relying on structural causal models.

Disentangled representations. The techniques that we leverage in this work are inspired by the representation learning literature. The core principle underlying disentangled representation learning is to learn independent factors of variation that capture well most of the variation underlying the unknown data generating process [3]. For example, the idea of using disentangled representations has been successfully leveraged to ensure that classifiers are fair while ensuring high classification accuracy downstream [7, 22, 20], to conduct local model audits [24], or to generate highly realistic data [30]. In contrast, our main insight is that disentangled representations can be used to generate recourses in the presence of dependent data by deriving indirect actions from direct actions.

3 Preliminaries

Notation. Before we introduce our framework, we note $||\cdot||$ refers to the 2-norm of a vector, $h(f(\mathbf{x}))$ denotes the probabilistic output of the trained classifier, where $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a differentiable scoring function (e.g., logit scoring function) and $h : \mathbb{R} \rightarrow [0, 1]$ is an activation function (e.g., sigmoid) that maps scores to continuous probability outputs. We denote the set of outcomes by $y \in \{0, 1\}$, where $y = 0$ is the undesirable outcome (e.g., loan rejection) and $y = 1$ indicates the desirable outcome (e.g., loan approval). Moreover, $\hat{y} = \mathbb{I}[h(f(\mathbf{x})) > \theta] = \mathbb{I}[f(\mathbf{x}) > s]$ is the predicted class, where $\mathbb{I}[\cdot]$ denotes the indicator function and θ is a threshold rule in probability space (e.g., $\theta = 0.5$), with corresponding threshold rule s in scoring space (e.g., $s = 0$ when a sigmoid activation is used).

The recourse objective. Counterfactual explanation methods provide recourses by identifying which attributes to change for reversing an unfavorable model prediction. We now describe the generic formulation leveraged by several state-of-the-art recourse methods. The goal is to find a set of actionable changes in order to improve the outcomes of instances $\mathbf{x}$ which are assigned an undesirable prediction under f. Moreover, one typically defines a cost measure in input space $c : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_+$. Typical choices are the ℓ_1 or ℓ_2 norms. Then the recourse problem is set up as follows:

$$\begin{aligned} \boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}} c(\mathbf{x}, \check{\mathbf{x}}) \text{ s.t. } \check{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}, \\ \check{\mathbf{x}} \in \mathcal{A}_d, \; f(\check{\mathbf{x}}) = s. \end{aligned} \tag{1}$$

The objective in eqn. (1) seeks to minimize the recourse costs $c(\mathbf{x}, \check{\mathbf{x}})$ subject to the constraint that the predicted label $\hat{y}$ flips from 0 (i.e., $f(\check{\mathbf{x}}) < s$) to 1 (i.e., $f(\check{\mathbf{x}}) \geq s$), and $\mathcal{A}_d$ represents a set of constraints ensuring that only admissible changes are made to the factual input x. For example, $\mathcal{A}_d$ could specify that no changes to protected attributes such as 'sex' can be made. The assumption underlying (1) is that each feature can be independently manipulated regardless of existing feature dependencies. Under this so-called independently manipulable feature (IMF) assumption, existing popular approaches use gradient based optimization techniques [41, 29], random search [18], or integer programming [37, 13, 32] to find recourses.

4 Our Framework: DEAR

The discussion in the previous sections identified three desiderata for a new recourse framework:

(i) **Feature dependencies.** The framework should capture feature dependencies while not relying on causal graphical models and structural equations.

(ii) **Realistic recourse.** The so identified recourses should lie in dense regions of the input space.

(iii) **Low costs.** The framework should allow to find recourses with controllable and low recourse costs.

With requirements (i) – (iii) in mind we present our novel recourse framework, DisEntangling Algorithmic Recourse (`DEAR`), which satisfies these fundamental requirements. More specifically: First, we introduce the generative model required to generate recourses under input dependencies that lie in dense regions of the input space. Second, using our model we then show that disentangled representations need to be learned to yield accurate recourse cost estimates. Third, we present our objective function to generate recourses under input dependencies and suggest a constructive explanation for why our framework finds recourses more reliably than existing manifold-based approaches. Finally, we provide a detailed discussion on how to operationalize and optimize our objective effectively.

4.1 The Generative Model

On a high level, our framework consists of separating the latent code of a generative model into 1) observable features $\mathbf{x}_{\mathcal{S}}$ – that we wish to perform direct recourse actions on – and 2) latent space features $\mathbf{v}$ that have been trained to become disentangled of the observable features. A direct recourse action has two effects: a direct effect on the input features that have to be changed, and an indirect effect on other, dependent features. The strength of the indirect effect is then determined by a generative model (see Figure 2). To formalize this intuition, let the input $\mathbf{x}$ be produced by the following generative model:

$$\begin{aligned} \mathbf{x} &= [\mathbf{x}_{\mathcal{S}}, \mathbf{x}_{\mathcal{S}^c}] \\ &= [g_{\mathbf{x}_{\mathcal{S}}}(\mathbf{v}, \mathbf{x}_{\mathcal{S}}), g_{\mathbf{x}_{\mathcal{S}^c}}(\mathbf{v}, \mathbf{x}_{\mathcal{S}})] = g(\mathbf{v}, \mathbf{x}_{\mathcal{S}}), \end{aligned} \tag{2}$$

where $g : \mathbb{R}^k \rightarrow \mathbb{R}^d$, $\mathbf{v} \in \mathbb{R}^{k-|\mathcal{S}|}$ refers to the latent code and $\mathbf{x}_{\mathcal{S}}$ corresponds to a subset of the input features where $S \subset \{1, \ldots, d\}$, and the complement set is $S^c = \{1, \ldots, d\} \setminus S$.

4.2 Disentangled Representations Promote Low Costs

Since one of the key considerations in recourse literature are recourse costs we use our generative model from eqn. (2) and analyze the recourse cost estimates under this model. Using the following Proposition, we obtain an intuition on how the generative model required for our framework has to be trained to yield low recourse costs.

Proposition 1 (Recourse costs)**.** *Under the generative model in* (2)*, the cost of recourse* $\|\boldsymbol{\delta}_x\|^2 = \|\mathbf{x} - \check{\mathbf{x}}\|^2$ *is given by:*

$$\|\boldsymbol{\delta}_x\|^2 \approx \underbrace{\mathbf{d}_{\mathcal{S}}^\top \big(\mathbf{J}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x}_\mathcal{S})\top} \mathbf{J}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x}_\mathcal{S})}\big)\mathbf{d}_\mathcal{S}}_{\textit{Direct Costs}} + \underbrace{\mathbf{d}_{\mathcal{S}}^\top \big(\mathbf{J}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x}_{\mathcal{S}^c})\top} \mathbf{J}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x}_{\mathcal{S}^c})}\big)\mathbf{d}_\mathcal{S}}_{\textit{Indirect Costs}},$$

$$where:$$

$$\mathbf{J}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x}_\mathcal{S})} = \underbrace{\frac{\partial g_{\mathbf{x}_\mathcal{S}}(\mathbf{v}, \mathbf{x}_\mathcal{S})}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{x}_\mathcal{S}}}_{\textit{Entanglement costs}} + \underbrace{\frac{\partial g_{\mathbf{x}_\mathcal{S}}(\mathbf{v}, \mathbf{x}_\mathcal{S})}{\partial \mathbf{x}_\mathcal{S}}}_{\textit{Identity Mapping}}$$

$$\mathbf{J}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x}_{\mathcal{S}^c})} = \underbrace{\frac{\partial g_{\mathbf{x}_{\mathcal{S}^c}}(\mathbf{v}, \mathbf{x}_\mathcal{S})}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{x}_\mathcal{S}}}_{\textit{Entanglement costs}} + \underbrace{\frac{\partial g_{\mathbf{x}_{\mathcal{S}^c}}(\mathbf{v}, \mathbf{x}_\mathcal{S})}{\partial \mathbf{x}_\mathcal{S}}}_{\textit{Elasticity of } g_{\mathbf{x}_{\mathcal{S}^c}} \textit{ w.r.t } \mathbf{x}_\mathcal{S}}.$$

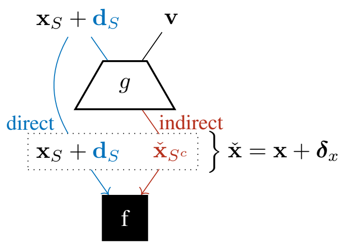
The result of Proposition 1 provides constructive insights for the implementation of the generative model g. It says that we can control the recourse costs using the actions $\mathbf{d}_\mathcal{S}$. First, the result reveals that the costs have to be partitioned into direct and indirect costs. The direct costs correspond to the costs that one would have obtained from algorithms that use the IMF assumption when searching for recourses (e.g., [41, 29, 37]). The indirect costs are due to feature dependencies of $\mathbf{x}_\mathcal{S}$ with $\mathbf{x}_{\mathcal{S}^c}$. If $\mathbf{x}_\mathcal{S}$ is independent of $\mathbf{x}_{\mathcal{S}^c}$ (i.e., the elasticity of $g_{\mathbf{x}_{\mathcal{S}^c}}$ w.r.t $\mathbf{x}_\mathcal{S}$ is $\mathbf{0}$), then a change in $\mathbf{x}_\mathcal{S}$ will not alter $\mathbf{x}_{\mathcal{S}^c}$ and the only cost remaining is the direct cost (we refer to Figure 2 for a schematic overview of the mechanism). Second, we observe that the costs can be *inflated*, if the latent space variables $\mathbf{v}$ depend on $\mathbf{x}_\mathcal{S}$. This is expressed through the *entanglement cost* terms in Proposition 1 and suggests that the model g should be trained such that $\mathbf{x}_\mathcal{S} \perp\!\!\!\perp \mathbf{v}$ to keep the recourse costs low.

4.3 Our Recourse Objective

So far the predictive model f has played no role in our considerations. Now, we introduce the predictive model to rewrite the recourse problem from (1) as follows:

$$\begin{aligned} \mathbf{d}_\mathcal{S}^* &= \arg\min_{\mathbf{d}_\mathcal{S}} c(\mathbf{x}, \check{\mathbf{x}}) \text{ s.t. } \check{\mathbf{x}} = g(\mathbf{v}, \mathbf{x}_\mathcal{S} + \mathbf{d}_\mathcal{S}), \\ &\check{\mathbf{x}} \in \mathcal{A}_d,\ \mathbf{x}_\mathcal{S} \perp\!\!\!\perp \mathbf{v},\ f(\check{\mathbf{x}}) = s, \end{aligned} \tag{3}$$

where we have used the insight that $\mathbf{x}_\mathcal{S} \perp\!\!\!\perp \mathbf{v}$ derived from Proposition 1. Relative to the objective from eqn. (1), our objective in eqn. (3) uses our generative model to capture input dependencies. Instead of finding recourse actions across the whole input space, we find recourse actions for the inputs in $\mathcal{S}$. We make recommendation on the choice of $\mathcal{S}$ in Section 4.4. Our reformulation has several advantages compared to existing recourse methods from the literature: i) relative to manifold-based recourse methods [12, 27, 1] the actions are applied to input space variables as opposed to latent space variables, and thus they are inherently interpretable; ii) relative to manifold-based recourse methods and recourse methods which use the IMF assumption [41, 18], we can sharply separate the direct effect, which $\mathbf{d}_\mathcal{S}$ has on $\check{\mathbf{x}}$ via $\mathbf{x}_\mathcal{S}$, from its indirect effect, which $\mathbf{d}_\mathcal{S}$ has on $\check{\mathbf{x}}$ determined by the generative model when it is dependent of $\mathbf{x}_{\mathcal{S}^c}$ (recall Proposition 1); iii) relative to causal recourse methods [15, 14], we neither assumed causal graphical models nor did we assume structural equation models to incorporate input dependencies.



$\mathbf{x}_\mathcal{S} + \mathbf{d}_\mathcal{S}$
$\mathbf{v}$
g
direct
indirect
$\mathbf{x}_\mathcal{S} + \mathbf{d}_\mathcal{S}$
$\check{\mathbf{x}}_{\mathcal{S}^c}$
$\check{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}_x$
f

Figure 2: Finding recourse for input $\mathbf{x}$ with `DEAR`. For an input $\mathbf{x}$ with $f(\mathbf{x}) < s$, we encode $[\mathbf{x}_\mathcal{S}, \mathbf{v}]{=}e(\mathbf{x})$. Then, we find direct actions $\mathbf{d}_\mathcal{S}$, which generate recourse, i.e. we find a $\check{\mathbf{x}}$ such that $f(\check{\mathbf{x}}) = s$, where $\check{\mathbf{x}} = [\mathbf{d}_\mathcal{S} + \mathbf{x}_\mathcal{S}, g_{\mathbf{x}_{\mathcal{S}^c}}(\mathbf{v}, \mathbf{d}_\mathcal{S} + \mathbf{x}_\mathcal{S})]$. The direct action $\mathbf{d}_\mathcal{S}$ has two effects: 1) it changes the features in $\mathcal{S}$ directly (i.e., $\mathbf{d}_\mathcal{S} + \mathbf{x}_\mathcal{S}$), and 2) it changes the features in $\mathcal{S}^c$ indirectly (i.e., $\check{\mathbf{x}}_{\mathcal{S}^c} = g_{\mathbf{x}_{\mathcal{S}^c}}(\mathbf{v}, \mathbf{d}_\mathcal{S} + \mathbf{x}_\mathcal{S})$). The strength of the indirect change $\check{\mathbf{x}}_{\mathcal{S}^c}$ is determined by the elasticity of $g_{\mathbf{x}_{\mathcal{S}^c}}$ w.r.t. $\mathbf{x}_\mathcal{S}$.

4.4 Aligned Generative Models Promote Finding Recourses Reliably

So far we have learned how disentangled represenations help reduce the recourse costs. Related work [1] has reported that manifold-based methods, which search for recourse in latent space (e.g., [27, 12, 1]), sometimes get stuck before they find a recourse. In this section, we develop a theoretical expression that will inform the choice of the set $\mathcal{S}$, and we will see that choosing this set appropriately promotes finding recourses more reliably. To this end, we derive an approximate closed-form solution for the objective in (3) which uses our insights from Proposition 1 (i.e., $\mathbf{v} \perp\!\!\!\perp \mathbf{x}_\mathcal{S}$).

Proposition 2 (Direct action)**.** *Given* $\mathbf{v} \perp\!\!\!\perp \mathbf{x}_\mathcal{S}$ *and* $c = \|\mathbf{x} - \check{\mathbf{x}}\|^2$*, a first-order approximation* $\tilde{\mathbf{d}}_\mathcal{S}^*$ *to the optimal direct action* $\mathbf{d}_\mathcal{S}^*$ *from the objective in* (3) *is given by:*

$$\tilde{\mathbf{d}}_\mathcal{S}^* = \frac{m}{\lambda + \|\mathbf{w}\|_2^2} \cdot \mathbf{w}, \tag{4}$$

where $\mathbf{Y}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x})} {:=} \frac{\partial g(\mathbf{v}, \mathbf{x}_\mathcal{S})}{\partial \mathbf{x}_\mathcal{S}}\Big|_{\mathbf{v}=\mathbf{v}, \mathbf{x}_\mathcal{S}=\mathbf{x}_\mathcal{S}}$*, s is the target score,* $m = s - f(\mathbf{x})$*,* $\mathbf{w} {=} \mathbf{Y}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x})\top} \nabla f(\mathbf{x})$ *and λ is the trade-off parameter.*

Corollary 1 (Recourse action)**.** *Under the same conditions* *stated in Proposition 2, a first-order approximation to the recourse action is given by:*

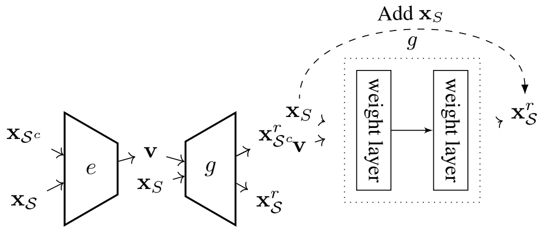
$$\boldsymbol{\delta}_x^* \approx \mathbf{Y}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x})} \tilde{\mathbf{d}}_\mathcal{S}^* = \frac{m}{\lambda + \|\mathbf{w}\|_2^2} \cdot \mathbf{Y}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x})} \mathbf{w}. \tag{5}$$

The above result is intuitive. The optimal recourse action $\tilde{\mathbf{d}}_\mathcal{S}^*$ applied to the inputs $\mathbf{x}_\mathcal{S}$ is being transformed by the generator Jacobian $\mathbf{Y}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x})}$ to yield the optimal action in input space $\boldsymbol{\delta}_x^*$. The generator Jacobian, in turn, measures the influence that the features in $\mathcal{S}$ have on the input $\mathbf{x}$.

We suggest to use *singletons* for the set of variables $\mathcal{S}$ where the direct action should be performed on as these are easiest to interpret (see Appendix B) and reliably lead to low cost recourses. Using singletons the insight from Proposition 2 becomes more clear. Then $\mathbf{w}$ from eqn. (4) is a scalar. Therefore, to make most progress towards the desired outcome at first order, the generator Jaocobian $\mathbf{Y}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x})}$ should be aligned with the model gradient $\nabla f(\mathbf{x})$, i.e., the two vectors should have a high similarity in the dot-product sense. To see this, consider a first-order approximation of $f(\mathbf{x}+\boldsymbol{\delta}_x^*) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \boldsymbol{\delta}_x^* = f(\mathbf{x}) + \frac{m \cdot w}{\lambda + \|\mathbf{w}\|_2^2} \cdot \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x})}$. To push the score $f(\mathbf{x})$ towards the target score $s \geq 0$, $\mathcal{S}$ should be chosen such that the dot product $\nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_\mathcal{S}}^{(\mathbf{x})}$ is high.

4.5 Optimizing our Objective

Motivated by the insights from Propositions 1 and 2, we present an algorithmic procedure to compute minimal cost recourses under feature dependencies using a penalty term during autoencoder training that encourages disentanglement of $\mathbf{x}_\mathcal{S}$ and $\mathbf{v}$ in order to keep the *entanglement costs* low. In summary, DEAR requires two steps: first, we need to obtain a latent space representation $\mathbf{v}$, which is independent of $\mathbf{x}_\mathcal{S}$. Second, we require an optimization procedure to identify the nearest counterfactual.


Add $\mathbf{x}_\mathcal{S}$
g
$\mathbf{x}_{\mathcal{S}^c}$
$\mathbf{x}_\mathcal{S}$
e
$\mathbf{v}$
$\mathbf{x}_\mathcal{S}$
g
$\mathbf{x}_{\mathcal{S}^c}^r \mathbf{v}$
$\mathbf{x}_\mathcal{S}^r$
$\mathbf{x}_\mathcal{S}$
weight layer
weight layer
$\mathbf{x}_\mathcal{S}^r$

(a) Conditional autoencoder (b) ResNet component within the decoder

Figure 3: DEAR's autoencoder architecture. (a): The conditional autoencoder architecture is trained subject to the Hessian penalty described in Section 4.5. (b): We achieve the identity mapping between $\mathbf{x}_\mathcal{S}$ and the reconstructed $\mathbf{x}_\mathcal{S}^r$ using a ResNet component [10, 11]: we add $\mathbf{x}_\mathcal{S}$ to $\mathbf{x}_\mathcal{S}^r$ before passing the arguments to the loss $\mathcal{L}_R$ from eqn. (6).

Step 1: Training the Generative Model. The main idea is to train a generative model, in which $\mathbf{x}_\mathcal{S}$ is independent of the latent variable $\mathbf{v}$, while providing high-quality reconstruction of the input $\mathbf{x}$. Thus, the training loss for the generative model consists of two components. First, it consists of both an encoder network e and decoder network g, for which the reconstruction loss,

$$\mathcal{L}_R(g, e; \mathbf{x}_\mathcal{S}) = \|g(e(\mathbf{x}), \mathbf{x}_\mathcal{S}) - \mathbf{x}\|_2^2, \tag{6}$$

guides both networks towards a good reconstruction of $\mathbf{x}$. Second, we want to drive the *entanglement costs* to 0, for which we need the decoder g to be disentangled with respect to the latent space, i.e., each component of $\mathbf{z} = [\mathbf{v}, \mathbf{x}_\mathcal{S}]$ should ideally control a single factor of variation in the output of g. To formalize this intuition, recall that $g(\mathbf{x}_\mathcal{S}, \mathbf{v}) = \mathbf{x} \in \mathbb{R}^d$, where each output $g_j = x_j$ for $1 \leq j \leq d$ has its own $|\mathbf{x}_\mathcal{S}| \times |\mathbf{v}|$ Hessian matrix $\mathbf{H}^{(j)}$. We refer to the collections of the d Hessian matrices as $\mathbf{H}$. Thus, the second loss we seek to minimize is given by:

$$\mathcal{L}_\mathbf{H}(g; \mathbf{x}_\mathcal{S}) = \sum_{j=1}^{d} \left(\sum_{k=1}^{|\mathbf{v}|} \sum_{l=1}^{|\mathbf{x}_\mathcal{S}|} H_{kl}^{(j)} \right), \tag{7}$$

which is also known as the Hessian penalty [30]. We illustrate the intuition of this objective on the j-th output x_j: we regularize the Hessian matrix $\mathbf{H}^{(j)} = \frac{\partial}{\partial \mathbf{v}} \frac{\partial g_j}{\partial \mathbf{x}_\mathcal{S}}$ and encourage its off-diagonal terms to become 0. Driving the off-diagonal terms to 0 implies that $\frac{\partial g_j}{\partial \mathbf{x}_\mathcal{S}}$ is not a function of $\mathbf{v}$ and thus $\mathbf{v}$ plays no role for the output of g_j when searching for minimum cost actions using $\mathbf{x}_\mathcal{S}$. We use the Hessian penalty from [30] in our implementation. Finally, Proposition 1 requires the identity mapping between the latent space $\mathbf{x}_\mathcal{S}$ and the reconstructed $\mathbf{x}_\mathcal{S}^r$. We encourage our generator g to learn this mapping by using a ResNet architecture [10, 11] as shown in Figure 3.

Step 2: Finding Minimal Cost Actions $\mathbf{d}_\mathcal{S}$. Given our trained generative model from step 1, we rewrite the problem in eqn. (3) using a Lagrangian with trade-off parameter λ. For a given encoded input instance $e(\mathbf{x}) = [\mathbf{v}, \mathbf{x}_\mathcal{S}]$, our objective function reads:

$$\begin{aligned} \mathbf{d}_\mathcal{S}^* &= \operatorname*{arg\,min}_{\mathbf{d}_\mathcal{S},\, \check{\mathbf{x}} \in \mathcal{A}_d} \mathcal{L} \\ &= \operatorname*{arg\,min}_{\mathbf{d}_\mathcal{S},\, \check{\mathbf{x}} \in \mathcal{A}_d} \ell\big(f(\check{\mathbf{x}}), s\big) + \lambda \|\mathbf{x} - \check{\mathbf{x}})\|_1, \end{aligned} \tag{8}$$

where $\check{\mathbf{x}} = g(\mathbf{v}, \mathbf{x}_\mathcal{S} + \mathbf{d}_\mathcal{S})$ is a potential counterfactual in input space, $\ell(\cdot, \cdot)$ denotes the MSE loss, and $s \geq 0$ is the target score in logit space. The term on the right side encourages the counterfactual $g(\mathbf{v}, \mathbf{x}_\mathcal{S} + \mathbf{d}_\mathcal{S}) = \check{\mathbf{x}}$ to be close to the given input point $\mathbf{x}$, while the left hand side encourages the predictions to be pushed from the factual output $f(\mathbf{x})$ towards s. We do gradient descent iteratively on the loss function in eqn. (8) until the class label changes from $y = 0$ to $y = 1$. Algorithm 1 summarizes our optimization

Algorithm 1 DEAR

```
Input: $f$, $\mathbf{x}$ s.t. $f(\mathbf{x})<0$, $g$, $e$, $\lambda > 0$, Learning rate: $\alpha > 0$, $s \geq 0$, $\mathcal{S}$
Initialize: $\mathbf{d}_{\mathcal{S}} = \mathbf{0}$, $e(\mathbf{x}) = [\mathbf{v}, \mathbf{x}_{\mathcal{S}}]$, $\check{\mathbf{x}} = g(\mathbf{v}, \mathbf{x}_{\mathcal{S}} + \mathbf{d}_{\mathcal{S}})$

while $f(\check{\mathbf{x}}) < s$ do
    $\mathbf{d}_{\mathcal{S}} = \mathbf{d}_{\mathcal{S}} - \alpha \cdot \nabla_{\mathbf{d}_{\mathcal{S}}} \mathcal{L}(\check{\mathbf{x}}; f, s, \lambda)$ {Optimize (8)}
    $\check{\mathbf{x}} = g(\mathbf{v}, \mathbf{x}_{\mathcal{S}} + \mathbf{d}_{\mathcal{S}})$
end while {Class changed, i.e., $f(\check{\mathbf{x}}) \geq s$}
Return: $\check{\mathbf{x}}^* = \check{\mathbf{x}}$
```

procedure. Finally, in Appendix C we further discuss how monotonicity constraints and how categorical variables are included in our objective.

5 Experimental Results

In this Section, we conduct extensive quantitative and qualitative evaluations to analyze DEAR's performance using our conceptual insights from the previous section. *Quantitatively*, we conduct a baseline comparison contrasting our framework DEAR with state-of-the-art recourse methods, which use generative models, using common evaluation measures [28] from the recourse literature such as recourse costs and reliability measures. *Qualitatively*, we consider three aspects: (i) the *entanglement costs*, (ii) the structure of the *cost splits* (i.e., direct vs. indirect costs) and (iii) a *case study* to showcase the advantages of our new framework which we relegated to Appendix B.

5.1 Details on Experiments

Real–world Data. Our first data set is the *Adult* data set taken from the UCI repository. This data set consists of approximately $48K$ samples with demographic (e.g., race, sex), education and employment (e.g., degree, occupation, hours-per-week), personal (e.g., marital status, relationship), financial (capital gain/loss) features where the label predicts whether an individual's income exceeds $50K$\$ per year ($y = 1$). Our second data set, *COMPAS* (Correctional Offender Management Profiling for Alternative Sanctions) consists of defendants' criminal history, jail and prison time, demographics and the goal is to predict recidivism risk for defendants from Broward County, Florida. Our third data set is the *Give Me Credit* data set from 2011 Kaggle competition. It is is a credit scoring data set, consisting of 150,000 observations and 11 features. The classification task consists of deciding whether an instance will experience financial distress within the next two years (*SeriousDlqin2yrs* is 0).

Prediction Models. For all our experiments, we obtain counterfactual explanations for two classification models, for which we provide additional details in Appendix C: We use is a binary logistic classifier that was trained without regularization, and an artificial neural network with a two-layer architecture that was trained with ReLU activation functions.

Recourse Methods. For all data sets, recourses are generated in order to flip the prediction label from the unfavorable class ($y = 0$) to the favorable class ($y = 1$). We partition the data set into 80-20 train-test splits, and do the model training and testing on these splits. We use the following six methods as our baselines for comparison:

M: CLUE [1]: This model suggests feasible counterfactual explanations that are likely to occur under the data distribution. Using the VAE's decoder, CLUE uses an objective that guides the search of CEs towards instances that have low uncertainty measured in terms of the classifier's entropy.

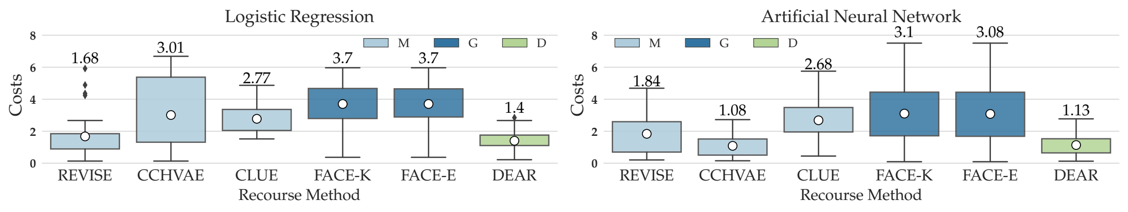
M: REVISE [12]: To find recourses that lie on the *data manifold*, this method utilizes a trained autoencoder to transform the input space into a latent embedding space. REVISE then uses gradient descent in latent space to find recourses that lie on the data manifold.

M: CCHVAE [27]: This is a method to find recourses that lie on the *data manifold*. CCHVAE also uses a trained autoencoder to transform the input space into a latent embedding space. The latent representation is then randomly perturbed to find recourses.

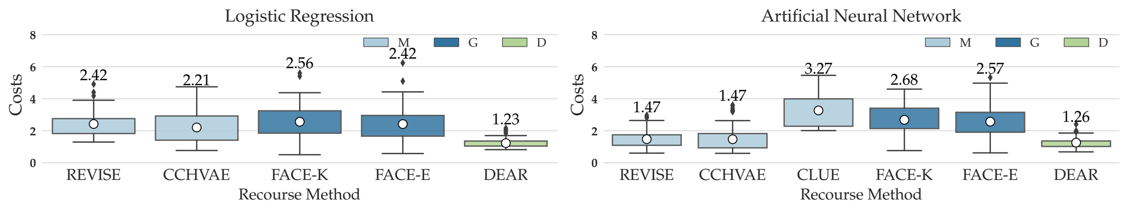
G: FACE-K & FACE-E [31]: This is classifier-agnostic method that finds recourses that lie on paths along dense regions. These methods construct *neighbourhood graphs* to find paths through dense regions. The graph is either an ϵ-graph (FACE-E) or a k-nearest neighbour graph (FACE-K).

Note that 'M' abbreviates methods which generate recourses that lie on the data manifold, and 'G' abbreviates methods, which use a graphical model to generate recourses that lead through dense paths. 'D' refers to our method (i.e., DEAR), which takes input dependencies into account. To allow for a fair comparison across the explanation models, which use autoencoders, we use similar base architectures for DEAR. Appendix C provides implementation details for the recourse methods and of all used autoencoders. We compute evaluation measures by using the min-max normalized inputs used for training the classification and generative models. Below we describe the evaluation measures.

Evaluation Measures. Since we are interested in generating small cost recourses, we define a notion of distance from the counterfactual explanation to the input point. As all methods under consideration minimize the ℓ_1 norm, we use this measure and compare the ℓ_1-costs across the methods. Further, we count the constraint violations (CV). We set the protected attributes 'sex' and 'race' to be immutable for the Adult and COMPAS data sets, and count how often



(a) Adult



(b) COMPAS

Figure 4: Measuring the cost of recourse (ℓ_1) across recourse methods, demonstrating that DEAR reduces recourse costs by up to 49% (bottom left). We use boxplots to show the distribution of recourse costs across all individuals from the test set who require algorithmic recourse. The numbers above the maximum values correspond to the white dots (= median recourse costs). The color of the boxplots represent the type of the recourse method: The methods with 'M' purely focus on the data manifold constraints, and methods with 'G' use a graphical model to generate recourses through dense paths. 'D' refers to our method which takes both manifold and input dependencies into account. Results for GMC are relegated to Appendix B.

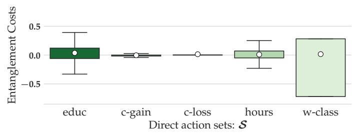
		Adult						COMPAS						GMC					
		LR			ANN			LR			ANN			LR			ANN		
Method		SR (↑)	CV (↓)	YNN (↑)	SR (↑)	CV (↓)	YNN (↑)	SR (↑)	CV (↓)	YNN (↑)	SR (↑)	CV (↓)	YNN (↑)	SR (↑)	CV (↓)	YNN (↑)	SR (↑)	CV (↓)	YNN (↑)
M	REVISE	0.35	0.00*	0.37	**1.00**	0.12	**0.72**	0.63	0.11	**1.00**	0.99	0.12	0.98	0.99	NA	**1.00**	**1.00**	NA	0.95
	CCHVAE	0.54	0.17	0.53	**1.00**	0.07	0.61	**1.00**	0.32	**1.00**	**1.00**	0.17	0.96	**1.00**	NA	0.24	**1.00**	NA	0.80
	CLUE	0.38*	0.00*	1.00*	**1.00**	0.00	0.25	0.00	–	–	0.21	0.00	1.00	**1.00**	NA	**1.00**	**1.00**	NA	0.92
G	FACE-K	0.99	0.36	0.71	**1.00**	0.29	0.57	0.99	0.38	**1.00**	0.60	0.40	**1.00**	**1.00**	NA	0.95	**1.00**	NA	0.96
	FACE-E	0.74	0.38	0.70	0.99	0.30	0.58	0.99	0.41	**1.00**	0.39	0.40	**1.00**	**1.00**	NA	0.94	**1.00**	NA	**0.97**
D	DEAR	**1.00**	**0.00**	**0.84**	**1.00**	**0.00**	0.70	**1.00**	**0.00**	**1.00**	**1.00**	0.01	**1.00**	**1.00**	NA	0.91	**1.00**	NA	0.94

Table 1: Measuring the reliability of algorithmic recourse for the ANN and LR models on all data sets. The *success rate* (SR), *constraint violation* (CV) and *y-nearest neighbors* (YNN) measures are described in Section 5. For GMC, there were no immutable features and therefore we are reporting NA. Our method (i.e., DEAR) usually performs on par or better relative to other recourse methods. *: Methods with success rates below 50% are excluded from the evaluation.

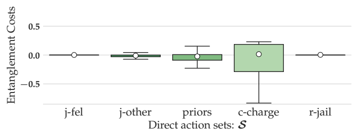
each of the explanation models suggests changes to these protected features. GMC has no protected attribute. We use the label yeighborhood (YNN) and measure how much data support recourses have from positively classified instances [28]. Ideally, recourses should be close to correct positively classified individuals, which is a desideratum formulated by the authors of [19]. Values of YNN close to 1 imply that the neighbourhoods around the recourses consists of points with the same predicted label, indicating that the neighborhoods around these points have already been reached by positively classified instances. Note that some generated recoures do not alter the predicted label of the instance as anticipated. Therefore we keep track of the success rate, i.e., how often do the suggested counterfactuals yield successful recourse. We so by counting the fraction of the respective methods' correctly determined counterfactuals. Finally, we report the entanglement costs. For a fixed set $\mathcal{S}$, for every instance at the end of training, we obtain d Hessian matrices $\mathbf{H}^{(j)} = \frac{\partial^2 g}{\partial \mathbf{v} \partial x_S}$ for $1 \leq j \leq d$. We then average the Hessian off-diagonal elements across all j and plot their distribution across all training instances. We can only do this for our recourse method DEAR.

5.2 Experimental Results

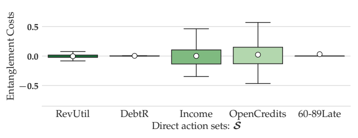
Recourse Costs. The baseline comparisons regarding the cost of recourse are shown in Figure 4. Relative to manifold-based recourse methods (i.e, REVISE and CCHVAE), DEAR usually performs more favourably ensuring up to 50 percent less costly median recourse costs. This is due to the fact that DEAR can use the most discriminative features in input space – as opposed to latent space – to search for



(a) Adult

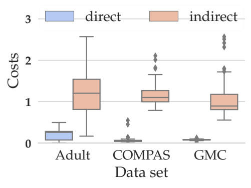


(b) COMPAS

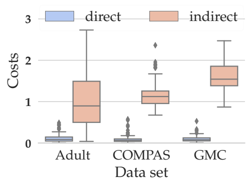


(c) GMC

Figure 5: Evaluating DEAR's entanglement costs on all data sets. At the end of autoencoder training, we compute the Hessians' off-diagonal elements of the decoder and average them (see Section 5.1 for more details). The feature names indicate the sets $\mathcal{S}$, that we perform direct recourse actions on. The white dots indicate the median entanglement costs, and the box indicates the interquartile range. In line with Proposition 1, the costs are pushed to 0.



(a) Logistic Regression



(b) Artificial Neural Network

Figure 6: Cost splits as suggested by Proposition 1 on both classifiers across all data sets. The direct costs corresponds to the direct action $\mathbf{d}_{\mathcal{S}}$ and are measured as $\|\mathbf{d}_{\mathcal{S}}\|_1$. The indirect costs are measured as $\|\mathbf{x}_{\mathcal{S}^c} - g_{\mathbf{x}_{\mathcal{S}^c}}(\mathbf{v}, \mathbf{d}_{\mathcal{S}} + \mathbf{x}_{\mathcal{S}})\|_1$.

recourses. Relative to the graph-based methods (i.e., FACE) our method performs significantly better. Since FACE has to ensure connected paths their costs are usually highest.

Reliability of Recourse. We measure the reliability of recourse using SR, CV and YNN presented in the previous Section. The results across all methods, data sets and classifiers are shown in Table 1. We see that DEAR has the highest SRs across all data sets and classifiers, among the highest YNN scores, and one of the lowest CV rates. Compared to the manifold-based recourse methods, DEAR's SR is up to 45 percentage points higher. This is due to the fact that the lower dimensional data manifold can end before the decision boundary is reached and thus the manifold-based methods, which search for recourse in latent space, sometimes get stuck before they find a counterfactual instance (see [6] for a detailed analysis of this phenomenon). Antorán et al. [1, Appendix] report a similar finding. A similar reason likely prevents FACE from reaching high SRs. Our method does not suffer from this shortcoming since it primarily uses the most discriminative features in input space (Proposition 2) to search for recourses, resulting in SRs of 1.

Qualitative Analysis. Finally, we analyze our recourse model qualitatively. We start by analyzing the *entanglement costs*. As required by Proposition 1, we require these costs to be pushed to 0. We plot the distribution of the averaged off-diagonal terms in Figure 5. The results show that the entanglement cost is consistently pushed to 0 (most medians are at 0). These results indicate that our mechanism is very well aligned with Proposition 1's requirement of disentangled $\mathbf{v}$ and $\mathbf{x}_{\mathcal{S}}$. Next, we analyze the cost splits. According to Proposition 1, we can split the costs of recourse into a direct and an indirect component. We show these cost splits in Figure 6 verifying that the elasticity of $g_{\mathbf{x}_{\mathcal{S}^c}}$ w.r.t. $\mathbf{x}_{\mathcal{S}}$ is non-zero, i.e., we observe a strong presence of feature dependencies. In App. B we provide a case study and further analyze the semantic meaning of these cost splits on GMC.

6 Conclusion

In this work, we considered the problem of generating algorithmic recourse in the presence of feature dependencies – a problem previously only studied through the lens of causality. We developed DEAR (DisEntangling Algorithmic Recourse), a novel recourse method that generates recourses by disentangling the latent representation of co-varying features from a subset of promising recourse features to capture some of the main practical desiderata: (i) recourses should adhere to feature dependencies without the reliance on handcrafted causal graphical models and (ii) recourses should lie in dense regions of the feature space, while providing (iii) low recourse costs. Quantitative as well as qualitative experiments on real-world data corroborate our theoretically motivated recourse model, highlighting our method's ability to provide reliable and low-cost recourse in the presence of feature dependencies.

We see several avenues for future work. From an end-user perspective, comparing the practical usefulness across various different recourse methods running user-studies with human participants is an important direction for future work. Further, our framework showcases the importance of feature dependencies for reliable algorithmic recourse by highlighting which individual features contributed directly and indirectly to the recourse. While this is reminiscent of recourses output by causal methods the recourses output by our framework should not be mistaken for causal recourses. Therefore, from a theoretical perspective, it would be interesting to find (local) conditions for both the classifier and the generative model under which our recourse framework would generate recourses with a causal interpretation.

References

[1] Javier Antorán, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. Getting a clue: A method for explaining uncertainty estimates. *International Conference on Learning Representations (ICLR)*, 2021.

[2] Solon Barocas, Andrew D. Selbst, and Manish Raghavan. The hidden assumptions behind counterfactual explanations and principal reasons. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2020.

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 2013.

[4] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*. Springer, 2020.

[5] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[6] Michael Downs, Jonathan L. Chu, Yaniv Yacoby, Finale Doshi-Velez, and Weiwei Pan. Cruds: Counterfactual recourse using disentangled subspaces. *ICML Workshop on Human Interpretability in Machine Learning (WHI 2020)*, 2020.

[7] Harrison Edwards and Amos Storkey. Censoring representations with an adversary. *arXiv preprint arXiv:1511.05897*, 2015.

[8] Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, Salvatore Ruggieri, and Franco Turini. Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems*, 2019.

[9] Riccardo Guidotti, Anna Monreale, Stan Matwin, and Dino Pedreschi. Black box explanation by learning image exemplars in the latent feature space. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision (ECCV)*, 2016.

[12] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615*, 2019.

[13] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 2020.

[14] Amir-Hossein Karimi, Julius von Kügelgen, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

[15] Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings Conference on Fairness, Accountability, and Transparency (FAccT)*, 2021.

[16] Mark T Keane and Barry Smyth. Good counterfactuals and where to find them: A case-based technique for generating counterfactuals for explainable ai (xai). In *International Conference on Case-Based Reasoning*, 2020.

[17] Eoin M Kenny and Mark T Keane. On generating plausible counterfactual and semi-factual explanations for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[18] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Inverse classification for comparison-based interpretability in machine learning. *arXiv preprint arXiv:1712.08443*, 2017.

[19] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

[20] Francesco Locatello, Gabriele Abbati, Tom Rainforth, Stefan Bauer, Bernhard Schölkopf, and Olivier Bachem. On the fairness of disentangled representations. In *Advances in neural information processing systems (NeurIPS)*, 2019.

[21] Ana Lucic, Harrie Oosterhuis, Hinda Haned, and Maarten de Rijke. Focus: Flexible optimizable counterfactual explanations for tree ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[22] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Learning adversarially fair and trans-

ferable representations. In *International Conference on Machine Learning (ICML)*, 2018.

[23] Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*, 2019.

[24] Charles Marx, Richard Phillips, Sorelle Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. Disentangling influence: Using disentangled representations to audit model predictions. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, 2019.

[25] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020.

[26] Axel Parmentier and Thibaut Vidal. Optimal counterfactual explanations in tree ensembles. In *International Conference on Machine Learning*, 2021.

[27] Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference 2020 (WWW)*, New York, NY, USA, 2020. ACM.

[28] Martin Pawelczyk, Sascha Bielawski, Johannes van den Heuvel, Tobias Richter, and Gjergji Kasneci. Carla: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms. In *Advances in Neural Information Processing Systems 34 (NeurIPS) Datasets and Benchmark Track*, 2021.

[29] Martin Pawelczyk, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. Exploring counterfactual explanations through the lens of adversarial examples: A theoretical and empirical analysis. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

[30] William Peebles, John Peebles, Jun-Yan Zhu, Alexei A. Efros, and Antonio Torralba. The hessian penalty: A weak prior for unsupervised disentanglement. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2020.

[31] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. Face: Feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES)*. ACM, 2020.

[32] Kaivalya Rawal and Himabindu Lakkaraju. Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

[33] Christopher Russell. Efficient search for diverse coherent explanations. In *Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency*, 2019.

[34] Lisa Schut, Oscar Key, Rory Mc Grath, Luca Costabello, Bogdan Sacaleanu, Yarin Gal, et al. Generating interpretable counterfactual explanations by implicit minimisation of epistemic and aleatoric uncertainties. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

[35] Thomas Spooner, Danial Dervovic, Jason Long, Jon Shepard, Jiahao Chen, and Daniele Magazzeni. Counterfactual explanations for arbitrary regression models. *arXiv:2106.15212*, 2021.

[36] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2017.

[37] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency (FAT*)*. ACM, 2019.

[38] Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2019.

[39] Suresh Venkatasubramanian and Mark Alfano. The philosophical basis of algorithmic recourse. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2020.

[40] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

[41] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31(2), 2018.

[42] Fan Yang, Sahan Suresh Alva, Jiahao Chen, and Xia Hu. Model-based counterfactual synthesizer for interpretation. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2021.

Supplementary Materials: Decomposing Counterfactual Explanations for Consequential Decision Making

A Theoretical Analysis

A.1 Proof of Proposition 1

Proof (Recourse Costs by DEAR) First, we note that $\mathbf{v}$ is usually obtained via some kind of training procedure, and thus it could be a function of $\mathbf{x}_S$. Next, we partition both $\mathbf{z} = [\mathbf{v} \quad \mathbf{x}_S]^\top$ and $\mathbf{d}_z = [\mathbf{d}_v \ \mathbf{d}_S]^\top$. Moreover, we partition $g(\mathbf{v}, \mathbf{x}_S) = [g_{\mathbf{x}_S}(\mathbf{v}, \mathbf{x}_S) \ \ g_{\mathbf{x}_{S^c}}(\mathbf{v}, \mathbf{x}_S)]^\top = [\mathbf{x}_S \ \mathbf{x}_{S^c}]^\top$. Then, the matrix of derivatives can be partitioned as follows:

$$\mathbf{J}_z^{(\mathbf{x})} = \begin{bmatrix} \mathbf{J}_{\mathbf{v}}^{(\mathbf{x}_{S^c})} & \mathbf{J}_{x_S}^{(\mathbf{x}_{S^c})} \\ \mathbf{J}_{\mathbf{v}}^{(\mathbf{x}_S)} & \mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_S)} \end{bmatrix} := \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}.$$

Since we are not interested in applied changes to $\mathbf{v}$ we set $\mathbf{d}_v{:=}\mathbf{0}$. By Lemma 1 we know $\|\mathbf{d}_x\|^2{=}\mathbf{d}_z^\top \big(\mathbf{J}_z^{(\mathbf{x})^\top} \mathbf{J}_{\mathbf{z}}^{(\mathbf{x})}\big)\mathbf{d}_z$. A direct computation with $\mathbf{d}_z{:=} \begin{bmatrix}\mathbf{0} & \mathbf{d}_S\end{bmatrix}^\top$ yields:

$$\begin{aligned} \|\mathbf{d}_x\|^2 &\approx \begin{bmatrix}\mathbf{0} & \mathbf{d}_S\end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{B} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{d}_S \end{bmatrix} \\ &= \mathbf{d}_S^\top \mathbf{B}^\top \mathbf{B} \mathbf{d}_S + \mathbf{d}_S^\top \mathbf{D}^\top \mathbf{D} \mathbf{d}_S \\ &= \underbrace{\mathbf{d}_S^\top \big(\mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_{S^c})^\top} \mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_{S^c})}\big)\mathbf{d}_S}_{\text{Indirect Costs}} + \underbrace{\mathbf{d}_S^\top \big(\mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_S)^\top} \mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_S)}\big)\mathbf{d}_S}_{\text{Direct Costs}}. \end{aligned}$$

By the chain rule of multivariate calculus (recall that $\mathbf{v}$ and $\mathbf{x}_S$ need not be independent), note that we can write out the above terms as follows:

$$\begin{aligned} \mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_{S^c})} &= \frac{\partial g_{\mathbf{x}_{S^c}}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{x}_S} + \frac{\partial g_{\mathbf{x}_{S^c}}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{x}_S} \frac{\partial \mathbf{x}_S}{\partial \mathbf{x}_S} \\ &= \underbrace{\frac{\partial g_{\mathbf{x}_{S^c}}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{x}_S}}_{\text{Entanglement costs}} + \underbrace{\frac{\partial g_{\mathbf{x}_{S^c}}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{x}_S}}_{\text{Elasticity of } g(\mathbf{x}_{S^c}) \text{ w.r.t to } \mathbf{x}_S} \\ \mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_S)} &= \frac{\partial g_{\mathbf{x}_S}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{x}_S} + \frac{\partial g_{\mathbf{x}_S}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{x}_S} \frac{\partial \mathbf{x}_S}{\partial \mathbf{x}_S} \\ &= \underbrace{\frac{\partial g_{\mathbf{x}_S}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{x}_S}}_{\text{Entanglement costs}} + \underbrace{\frac{\partial g_{\mathbf{x}_S}(\mathbf{v}, \mathbf{x}_S)}{\partial \mathbf{x}_S}}_{\text{Identity Mapping}}. \end{aligned}$$

□

Let us consider what this implies intuitively. For the direct costs, notice that g would achieve the best reconstruction of $\mathbf{x}_S$ by using the identify mapping. Recall, in Section 4.5, we suggested to use a ResNet component within the decoder to enforce this identity mapping during training of our autoencoder model. Hence, under perfect disentanglement the disentanglement costs are 0, and the $\mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_S)}{=}\mathbf{1}$: thus, the direct cost would ideally be given by $\mathbf{d}_S^\top \mathbf{d}_S$. This is the squared ℓ_2 norm of $\mathbf{d}_S$.

The indirect costs, on the other hand, depend on the sensitivity of $\mathbf{x}_{S^c}$ with respect to $\mathbf{x}_S$, that is, $\mathbf{J}_{\mathbf{x}_S}^{(\mathbf{x}_{S^c})}$. Again, we consider the case of perfect disentanglement first: Suppose $\mathbf{x}_S$ was a variable that was unrelated to the remaining variables $\mathbf{x}_{S^c}$, while still being predictive of the outcome: Then $\mathbf{J}_{x_S}^{(x_{S^c})} = \mathbf{0}$, and a change $\mathbf{d}_S$ would only have a direct impact on the outcome, and thus the indirect costs would disappear. In this case, the recourse cost for independence–based and dependence–based methods coincide. On the other extreme, suppose $\mathbf{x}_{\mathcal{S}}$ was almost a copy of $\mathbf{x}_{\mathcal{S}^c}$, then $\mathbf{J}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x}_{\mathcal{S}^c})} \approx \mathbf{1}$, and changing $\mathbf{x}_{\mathcal{S}}$ clearly impacts the remaining variables $\mathbf{x}_{\mathcal{S}^c}$. In this case, an independence–based method would not reliably capture the recourse costs.

A.2 Proof of Proposition 2

Proof of Proposition 2. Recall the problem in (3):

$$\mathbf{d}_{\mathcal{S}}^* = \arg\min_{\mathbf{d}_{\mathcal{S}}} \mathcal{L} = \arg\min_{\mathbf{d}_{\mathcal{S}}} \lambda \|\mathbf{d}_{\mathcal{S}}\|^2 + \|s - f\big(g(\mathbf{v}, \mathbf{x}_{\mathcal{S}} + \mathbf{d}_{\mathcal{S}})\big)\|^2. \tag{9}$$

We use the following first-order approximation to $f\big(g(\mathbf{v}, \mathbf{x}_{\mathcal{S}} + \mathbf{d}_{\mathcal{S}})\big) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{d}_{\mathcal{S}}$, where we have substituted $g(\mathbf{v}, \mathbf{x}_{\mathcal{S}}) = \mathbf{x}$ and used that $\mathbf{v} \perp\!\!\!\perp \mathbf{x}_{\mathcal{S}}$ by design of the generative model. Then, we can derive a surrogate loss to the loss from (9):

$$\mathcal{L} \approx \tilde{\mathcal{L}} = \lambda \cdot \mathbf{d}_{\mathcal{S}}^\top \mathbf{d}_{\mathcal{S}} + \big(m - \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{d}_{\mathcal{S}}\big)^\top \big(m - \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{d}_{\mathcal{S}}\big), \tag{10}$$

where $m = s - f(\mathbf{x})$. The second term on the right in (10) can be written as:

$$m^2 - 2m \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{d}_{\mathcal{S}} + \mathbf{d}_{\mathcal{S}}^\top {\mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}}^\top \nabla f(\mathbf{x}) \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{d}_{\mathcal{S}}.$$

By solving $\arg\min_{\mathbf{d}_{\mathcal{S}}} \tilde{\mathcal{L}}$ we find the optimal change required in the features $\mathbf{x}_{\mathcal{S}}$ as follows:

$$\tilde{\mathbf{d}}_{\mathcal{S}}^* = \boldsymbol{M}^{-1} \boldsymbol{\eta}, \tag{11}$$

where

$$\boldsymbol{M} = \lambda \cdot \mathbf{I} + {\mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}}^\top \nabla f(\mathbf{x}) \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \qquad\qquad \boldsymbol{\eta} = m \cdot \nabla f(\mathbf{x})^\top \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}. \tag{12}$$

Next, we define $\mathbf{w} = {\mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})}}^\top \nabla f(\mathbf{x})$. Note that $\mathbf{w}\mathbf{w}^\top$ is a rank-1 matrix. Thus, by the well-known Sherman-Morrison-Woodbury formula, $\mathbf{M}$ can be inverted as follows:

$$\mathbf{M}^{-1} = \frac{1}{\lambda} \left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^\top}{\lambda + \|\mathbf{w}\|_2^2} \right). \tag{13}$$

As a consequence, after substituting (13) into (12) we obtain that:

$$\tilde{\mathbf{d}}_{\mathcal{S}}^* = \frac{m}{\lambda + \|\mathbf{w}\|_2^2} \mathbf{w}. \tag{14}$$

Further, note that $\boldsymbol{\delta} = g(\mathbf{z} + \mathbf{d}_{\mathcal{S}}) - g(\mathbf{z}) \approx \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{d}_{\mathcal{S}}$, where we have used that $\mathbf{v} \perp\!\!\!\perp \mathbf{x}_{\mathcal{S}}$. Therefore, we obtain a first-order approximation to the optimal recourse in input space:

$$\boldsymbol{\delta}_x^* \approx \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \tilde{\mathbf{d}}_{\mathcal{S}}^* = \frac{m}{\lambda + \|\mathbf{w}\|_2^2} \cdot \mathbf{Y}_{\mathbf{x}_{\mathcal{S}}}^{(\mathbf{x})} \mathbf{w}, \tag{15}$$

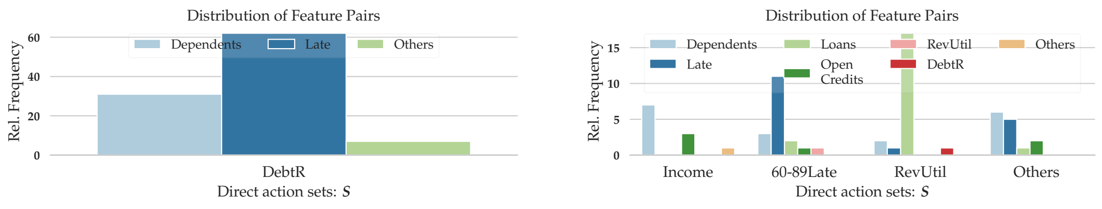
as claimed. □

A.3 Proof of Lemma 1

Lemma 1 (Recourse costs in terms latent space quantities)**.** *Given a latent representation* $\mathbf{z}$ *of a sample* $\mathbf{x} = g(\mathbf{z})$ *and a generated counterfactual* $\check{\mathbf{x}} = g(\check{\mathbf{z}})$ *with* $\check{\mathbf{z}} = \mathbf{z} + \mathbf{d}_z$*, the cost of recourse* $\|\mathbf{x} - \check{\mathbf{x}}\|^2$ *can be expressed in terms of latent space quantities:*

$$\|\boldsymbol{\delta}_x\|^2 \approx \mathbf{d}_z^\top \big({\mathbf{J}_z^{(\mathbf{x})}}^\top \mathbf{J}_z^{(x)}\big) \mathbf{d}_z,$$

where the matrix of derivatives with respect to output $\mathbf{x} = g(\mathbf{z})$ *is given by* $\mathbf{J}_z^{(\mathbf{x})} := \frac{\partial g(\mathbf{z})}{\partial \mathbf{z}}\Big|_{\mathbf{z}=\mathbf{z}}$.



(a) Logistic Regression (b) Artificial Neural Network

Figure 7: We show the rel. frequency of important feature pairs that need to be changed together. Associated with each direct minimum cost action on the x-axis (i.e., $\mathbf{d}_{\mathcal{S}}$), we plot the second most important feature (y-axis) that should change together with the direct action feature from $\mathcal{S}$. For example, in the bottom panel, for 10 percent of all instances, decreasing '60-89 days late' goes hand in hand with either a decrease in '30-60 days late' or a decrease in 'more than 90 days late'.

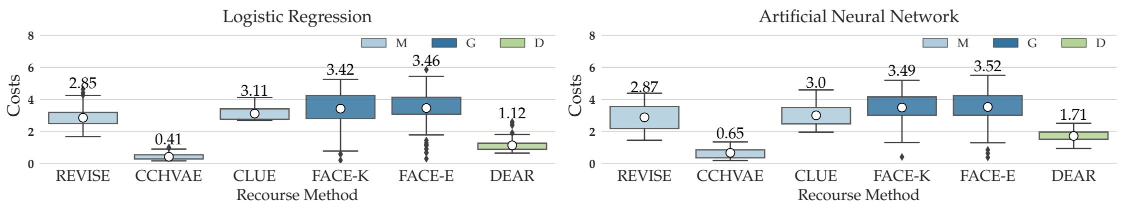


Figure 8: Give Me Credit: Plot from main text.

Proof. We use the cost of recourse, and a first-order Taylor series approximation for $g(\mathbf{z}+\mathbf{d}_z)$ at $\mathbf{z}$ to arrive at:

$$\begin{aligned}\|\boldsymbol{\delta}_x\|^2 &= \|g(\mathbf{z}) - g(\mathbf{z}+\mathbf{d}_z)\|^2 \\ &\approx \|g(\mathbf{z}) - (g(\mathbf{z}) + \mathbf{J}_{\mathbf{z}}^{(\mathbf{x})}\mathbf{d}_z)\|^2 \\ &= \mathbf{d}_z^\top \big(\mathbf{J}_z^{(\mathbf{x})^\top}\mathbf{J}_z^{(\mathbf{x})}\big)\mathbf{d}_z,\end{aligned}$$

where $\mathbf{J}_{\mathbf{z}}^{(\mathbf{x})} := \frac{\partial g(\mathbf{z})}{\partial \mathbf{z}}\Big|_{\mathbf{z}=\mathbf{z}}$. □

On an intuitive level, Lemma 1 measures how the cost – measured in input space quantities – depends on perturbations of each component of the generative latent space $\mathbf{z}$.

B Case Study: "Credit Risk"

As a practical example, we showcase additional insights that our recourse model can provide. Here we analyze the cost splits from a semantic point of view. In Figure 7, we show the distribution of important feature pairs that need to change together to lead to loan approvals for individuals from the *Give Me Credit* data set: the x-axis shows the direct actions resulting in the lowest costs, and the y-axis shows the relative frequency of the most important indirect actions. The following noteworthy patterns emerge: (i) the non-linear classifier has picked up more non-linear relations since the feature, on which the minimum cost direct actions are suggested, vary more heavily across instances for the ANN model (bottom panel) relative to the LR model (top panel); For example, a decrease in 'revolving utilization' is often followed by a decrease in the number of 'loans', which is semantically meaningful suggesting ways to reduce the 'revolving utilization'. Finally, we emphasize that our method showcases the importance of feature dependencies for reliable algorithmic recourse by highlighting how it arrived at the recourse. The recourses output by our framework should not be mistaken for causal recourses.

Factual		Recourse Method		
		`SCFE` (IMF)	`REVISE` (M)	`DEAR` (D)
`Age`	45	**58**	45	45
`Educ`	9	**7**	**11**	9
`C-gain`	0	0	**2110**	**307**
`Hours`	0	0	**80**	**24**
`C-loss`	60	**20**	**45**	**41**
`W-class`	Private: No	No	No	No
`M-status`	Married: No	No	No	No
`Occu`	Specialized: No	No	**Yes**	**Yes**
`Race`	White: No	No	No	No
`Sex`	Male: No	No	No	No
`Native`	US: Yes	Yes	Yes	Yes
ANN	$> 50K$: No	**Yes**	**Yes**	**Yes**

Table 2: Comparing recourses qualitatively across explanation methods for a factual instance. The bottom row shows the ANN classifier's predicted class outcomes. For `DEAR`, the recourse intervention was done to `C-loss`. The remaining features changed as a result of this. For `CCHVAE`, the results were similar to those by `REVISE`.

C Implementation Details

C.1 Handling Constraints

Encoding Monotonicity Constraints. In the presence of strong prior knowledge on how certain features are allowed to change (e.g., 'years of schooling' (yos) or 'age' can only go up) one can add Hinge-losses [23] to encourage monotonicity constraints. Let x_{yos} correspond to the schooling feature. Then we can add $-\min(0, \check{x}_{yos} - x_{yos})$ to the loss function in (8) to ensure that the counterfactual $\check{x}_{yos}$ should increase, where $\check{x}_{yos}$ is the corresponding entry from $g(\mathbf{v}, \mathbf{x}_S + \mathbf{d}_S)$.

Handling Categorical Variables. Using `DEAR`, one can easily handle (high-cardinality) categorical features. We can turn all categorical features into numeric features by standard one-hot encoding. For each categorical feature, we can then use a softmax-layer after the final output layer of the decoder. For the purpose of the one-hot-encoded reconstruction, we apply the argmax.

C.2 Recourse Methods

For all data sets, the features are binary-encoded and the data is scaled to lie between 0 and 1. We partition the data sets into train-test splits. The training set is used to train the classification models for which recourses are generated. Recourses are generated for all samples in the test split for the fixed classification model. In particular, we use the following algorithms to generate recourses. Specifically,

- `C-CHVAE` An autoencoder is additionally trained to model the data-manifold. The explanation model uses a counterfactual search algorithm in the latent space of the AE. Particularly, a latent sample within an ℓ_1-norm ball with search radius r_l is used until recourse is successfully obtained. The search radius of the norm ball is increased until recourse is found. The architecture of the generative model are provided in Appendix C.4.
- `REVISE` As with the recourse model of Pawelczyk et al. [27], an autoencoder is additionally trained to model the data-manifold. The explanation model uses a gradient-based search algorithm in the latent space of the AE. For a fixed weight on the distance component, we allow up to 500 gradient steps until recourse is successfully obtained. Moreover, we iteratively search for the weight leading up to minimum cost recourse. The architectures of the generative model are provided in Appendix C.4.
- `FACE` Poyiadzi et al. [31] provide `FACE`, which uses a shortest path algorithm (for graphs) to find counterfactual explanations from high–density regions. Those explanations are actual data points from either the training or test set. Immutability constraints are enforced by removing incorrect neighbors from the graph. We implemented two variants of this model: one uses an epsilon–graph (`FACE-EPS`), and a second one uses a knn–graph (`FACE-KNN`).

To determine the strongest hyperparameters for the graph size we conducted a *grid search*. We found that values of $k_{\text{FACE}} = 50$ gave rise to the best balance of success rate and costs. For the epsilon graph, a radius of 0.25 yields the strongest results to balance between high ynn and low cost.

- CLUE Antorán et al. [1] propose CLUE, a generative recourse model that takes a classifier's uncertainty into account. This model suggests feasible counterfactual explanations that are likely to occur under the data distribution. The authors use a variational autoencoder (VAE) to estimate the generative model. Using the VAE's decoder, CLUE uses an objective that guides the search of CEs towards instances that have low uncertainty measured in terms of the classifier's entropy. We use the default hyperparameters, which are set as a function of the data set dimension d. Performing hyperparameter search did not yield results that were improving distances while keeping the same success rate.

We describe architecture and training details in the following.

C.3 Supervised Classification Models

All models are implemented in PyTorch and use a $80 - 20$ train-test split for model training and evaluation. We evaluate model quality based on the model accuracy. All models are trained with the same architectures across the data sets:

	Neural Network	Logistic Regression
Units	[Input dim, 18, 9, 3, 1]	[Input dim, 1]
Type	Fully connected	Fully connected
Intermediate activations	ReLU	N/A
Last layer activations	Sigmoid	Sigmoid

Table 3: Classification Model Details

		Adult	COMPAS	Give Me Credit
Batch-size	ANN	512	32	64
	Logistic Regression	512	32	64
Epochs	ANN	50	40	30
	Logistic Regression	50	40	30
Learning rate	ANN	0.002	0.002	0.001
	Logistic Regression	0.002	0.002	0.001

Table 4: Training details

	Adult	COMPAS	Give Me Credit
Logistic Regression	0.83	0.84	0.92
Neural Network	0.84	0.85	0.93

Table 5: Performance of classification models used for generating algorithmic recourse.

C.4 Generative Model Architectures used for DEAR, CCHVAE and REVISE

For all experiments, we use the following architectures.

Additionally, for DEAR all generative models use the Hessian Penalty [30] and a residual block, which we both described in more detail in Section 4.5 of the main text.

	Adult	COMPAS	Give Me Credit
Encoder layers	[input dim, 16, 32, 10]	[input dim, 8, 10, 5]	[input dim, 8, 10, 5]
Decoder layers	[10, 16, 32, input dim]	[5, 10, 8, input dim]	[5, 10, 8, input dim]
Type	Fully connected	Fully connected	Fully connected
Loss function	MSE	MSE	MSE

Table 6: Autoencoder details

A.3 Exploring Counterfactual Through the Lens of Adversarial Examples

Publication: Published at the international conference on artificial intelligence and statistics (**AISTATS**) 2022.

Contribution: Himabindu Lakkaraju had the idea for this paper. Sohini Upadhyay and I developed the theoretical results. I conducted all experiments. Section 1 was written by Himabindu Lakkaraju. Shalmali Joshi wrote section 2. I wrote the remainder of the work; and Chirag Agarwal helped supervise all stages of the research process.

Exploring Counterfactual Explanations Through the Lens of Adversarial Examples: A Theoretical and Empirical Analysis

Martin Pawelczyk[1] **Chirag Agarwal**[2,3] **Shalmali Joshi**[3]
Sohini Upadhyay[3] **Himabindu Lakkaraju**[3]
[1]University of Tübingen [2]Adobe Research [3]Harvard University

Abstract

As machine learning (ML) models become more widely deployed in high-stakes applications, counterfactual explanations have emerged as key tools for providing actionable model explanations in practice. Despite the growing popularity of counterfactual explanations, a deeper understanding of these explanations is still lacking. In this work, we systematically analyze counterfactual explanations through the lens of adversarial examples. We do so by formalizing the similarities between popular counterfactual explanation and adversarial example generation methods identifying conditions when they are equivalent. We then derive the upper bounds on the distances between the solutions output by counterfactual explanation and adversarial example generation methods, which we validate on several real world data sets. By establishing these theoretical and empirical similarities between counterfactual explanations and adversarial examples, our work raises fundamental questions about the design and development of existing counterfactual explanation algorithms.

1 INTRODUCTION

With the increasing use of Machine learning (ML) models in critical domains, such as health care and law enforcement, it becomes important to ensure that their decisions are robust and explainable. To this end, several approaches have been proposed in recent literature to explain the complex behavior of ML models (Simonyan et al., 2013; Ribeiro et al., 2016; Lundberg and Lee, 2017; Sundararajan et al., 2017). One such popular class of explanations designed to provide recourse to individuals adversely impacted by algorithmic decisions are *counterfactual explanations* (Wachter et al., 2017; Ustun et al., 2019; Barocas et al., 2020; Venkatasubramanian and Alfano, 2020). For example, in a credit scoring model where an individual loan application is denied, a counterfactual explanation can highlight the minimal set of changes the individual can make to obtain a positive outcome (Pawelczyk et al., 2020a; Karimi et al., 2020c). Algorithms designed to output counterfactual explanations often attempt to find a closest counterfactual for which the model prediction is positive (Wachter et al., 2017; Ustun et al., 2019; Pawelczyk et al., 2020a; Karimi et al., 2020c).

Adversarial examples, on the other hand, were proposed to highlight how vulnerabilities of ML models can be exploited by (malicious) adversaries (Szegedy et al., 2013; Ballet et al., 2019; Cartella et al., 2021). These adversarial examples are usually also obtained by finding minimal perturbations to a given data instance such that the model prediction changes (Goodfellow et al., 2014; Carlini and Wagner, 2017; Moosavi-Dezfooli et al., 2016).

Conceptually, adversarial examples and counterfactual explanations solve a similar optimization problem (Freiesleben, 2020; Wachter et al., 2017; Cartella et al., 2021). Techniques generating adversarial examples and counterfactual explanations use distance or norm constraints in the objective function to enforce the notion of minimal perturbations. While adversarial methods generate instances that are semantically indistinguishable from the original instance, counterfactual explanations or algorithmic recourse[1], encourage minimal changes to an input so that so that a

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

[1]Note that counterfactual explanations, contrastive explanations, and recourses are used interchangeably in prior literature. Counterfactual/contrastive explanations serve as a means to provide recourse to individuals with unfavorable algorithmic decisions. We use these terms interchangeably as introduced and defined by Wachter et al. (2017).

user can readily act upon these changes to obtain the desired outcome. In addition, some methods in both lines of work use manifold-based constraints to find natural adversarial examples (Zhao et al., 2018) or realistic counterfactual explanations by restricting them to lie on the data manifold (Joshi et al., 2019; Pawelczyk et al., 2020a,b).

While the rationale of producing a counterfactual close to the original instance is motivated by the desideratum that counterfactuals should be actionable and easily understandable, producing close instances on the other side of the decision boundary could just as easily indicate adversarial activity. This begs the question to what extent do counterfactual explanation algorithms return solutions that resemble adversarial examples. However, there has been little to no work on systematically analyzing the aforementioned connections between the literature on counterfactual explanations and adversarial examples.

Present Work. In this work, we approach the study of similarities between counterfactual explanations and adversarial examples from the perspective of counterfactual explanations for algorithmic recourse. Therefore, we consider consequential decision problems (e.g., loan applications) commonly employed in recourse literature and our choices of data modalities (i.e., tabular data) and algorithms are predominantly motivated by this literature. In particular, we make one of the first attempts at establishing theoretical and empirical connections between state-of-the-art counterfactual explanation and adversarial example generation methods.

More specifically, we analyze these similarities by bounding the distances between the solutions of salient counterfactual explanation and popular adversarial example methods for locally linear approximations. Our analysis demonstrates that several popular counterfactual explanation and adversarial example generation methods such as the ones proposed by Wachter et al. (2017) and Carlini and Wagner (2017); Moosavi-Dezfooli et al. (2016), are equivalent for certain hyperparameter choices. Moreover, we demonstrate that C-CHVAE and the natural adversarial attack (NAE) (Zhao et al., 2018) provide similar solutions for certain generative model choices.

Finally, we carry out extensive experimentation with multiple synthetic and real-world data sets from diverse domains such as financial lending and criminal justice to validate our theoretical findings. We further probe these methods empirically to validate the similarity between the counterfactuals and adversarial examples output by several state-of-the-art methods. Our results indicate that counterfactuals and adversarial examples output by manifold-based methods such as NAE and C-CHVAE are more similar compared to those generated by other techniques. By establishing these and other theoretical and empirical similarities, our work raises fundamental questions about the design and development of existing counterfactual explanation algorithms.

2 RELATED WORK

This work lies at the intersection of counterfactual explanations and adversarial examples in machine learning. Below we discuss related work for each of these topics and their connection.

Adversarial examples. Adversarial examples are obtained by making infinitesimal perturbations to input instances such that they force a ML model to generate adversary-selected outputs. Algorithms designed to successfully generate these examples are called Adversarial attacks (Szegedy et al., 2013; Goodfellow et al., 2014). Several attacks have been proposed in recent literature depending on the degree of knowledge/access of the model, training data, and optimization techniques. While gradient-based methods (Goodfellow et al., 2014; Kurakin et al., 2016; Moosavi-Dezfooli et al., 2016) find the minimum ℓ_p-norm perturbations to generate adversarial examples, generative methods (Zhao et al., 2018) constrain the search for adversarial examples to the training data-manifold. Finally, some methods (Cisse et al., 2017) generate adversarial examples for non-differentiable and non-decomposable measures in complex domains such as speech recognition and image segmentation. We refer to a well-established survey for a more comprehensive overview of adversarial examples (Akhtar and Mian, 2018).

Counterfactual explanations. Counterfactual explanation methods aim to provide explanations for a model prediction in the form of minimal changes to an input instance that changes the original prediction (Wachter et al., 2017; Ustun et al., 2019; Van Looveren and Klaise, 2019; Karimi et al., 2020b). These methods are categorized based on the access to the model (or gradients), sparsity of the generated explanation and whether the generated explanations are constrained to the manifold (Verma et al., 2020; Karimi et al., 2020b). To this end, Wachter et al. (2017) proposed a gradient-based method to obtain counterfactual explanations for models using a distance-based penalty and finding the nearest counterfactual explanation. Further, restrictions on attributes such as race, age, and gender are generally enforced to ensure that the output counterfactual explanations are realistic for users to act on them. In addition, manifold-based constraints are imposed in many methods (Pawelczyk et al., 2020a; Joshi et al., 2019) so that the counterfactual explanations are faithful to the data distribution. Finally, causal approaches have recently been proposed to generate counterfactual explanations that adhere to causal constraints (Karimi et al., 2020a; Barocas et al., 2020; Karimi et al., 2021, 2020c).

Connections between adversarial examples and counterfactual explanations. Conceptual connections between adversarial examples and counterfactual explanations have been previously identified in the literature (Freiesleben, 2020; Browne and Swift, 2020). While Freiesleben (2020) highlight conceptual differences in aims, formulation and use-cases between both sub-fields suggesting that counterfactual explanations represent a broader class of examples of which adversarial examples represent a subclass, Browne and Swift (2020) focus on discussing the differences *w.r.t* semantics hidden layer representations of DNNs. Our goal, on the other hand, is to theoretically formalize and empirically analyze the (dis)similarity between these fields.

3 PRELIMINARIES

Notation. We denote a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$ mapping features $\mathbf{x} \in \mathcal{X}$ to labels $\mathcal{Y}$. Further, we define $h(\mathbf{x}) = g(f(\mathbf{x}))$, where $f : \mathcal{X} \rightarrow \mathbb{R}$ is a scoring function (*e.g.,* logits) and $g : \mathbb{R} \rightarrow \mathcal{Y}$ an activation function that maps output logit scores to discrete labels. Below we describe some representative methods used in this work to generate counterfactual explanations and adversarial examples.

3.1 Counterfactual explanation methods

Counterfactual explanations provide recourses by identifying which attributes to change for reversing a models' adverse outcome. Methods designed to output counterfactual explanations find a counterfactual $\mathbf{x}'$ that is "closest" to the original instance $\mathbf{x}$ and changes the models' prediction $h(\mathbf{x}')$ to the desired label. While several of these methods incorporate distance metrics (*e.g.,* ℓ_p-norm) or user preferences (Rawal and Lakkaraju, 2020) to find the desired counterfactuals, some efforts also impose causal (Karimi et al., 2020c) or data manifold constraints (Joshi et al., 2019; Pawelczyk et al., 2020a,b) to find realistic counterfactuals. We now describe counterfactual explanation methods from two broad categories: 1) Gradient- (Wachter et al., 2017) and 2) search-based (Pawelczyk et al., 2020a).

Score CounterFactual Explanations (SCFE). For a given classifier h and the corresponding scoring function f, and a distance function $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$, Wachter et al. (2017) formulate the problem of finding a counterfactual $\mathbf{x}'$ for $\mathbf{x}$ as:

$$\arg\min_{\mathbf{x}'} \ (f(\mathbf{x}') - s)^2 + \lambda\, d(\mathbf{x}, \mathbf{x}'), \tag{1}$$

where s is the target score for $\mathbf{x}$ and λ is set to iteratively increase until $f(\mathbf{x}') = s$. More specifically, to arrive at a counterfactual probability of 0.5, the target score for $g(\mathbf{x})$ for a sigmoid function is $s = 0$, where the logit corresponds to a 0.5 probability for $y = 1$.

C-CHVAE. Let I_γ and $\mathcal{G}_\theta$ denote the encoder and decoder of the VAE model used by C-CHVAE (Pawelczyk et al., 2020a) to generate realistic counterfactuals. Note that the counterfactuals for $\mathbf{x}$ are generated in the latent space of the encoder $\mathcal{Z}$, where $I_\gamma : \mathcal{X} \rightarrow \mathcal{Z}$. Let $\mathbf{z}$ and $\tilde{\mathbf{z}} = \mathbf{z} + \delta$ denote the latent representation and generated counterfactuals for the original instance $\mathbf{x}$. Intuitively, $\mathcal{G}_\theta$ is a generative model that projects the latent counterfactuals to the feature space and I_γ allows to search for counterfactuals in the data manifold. Thus, the objective function is defined as follows:

$$\begin{aligned} \delta^* &= \arg\min_{\delta \in \mathcal{Z}} \|\delta\| \\ &\text{s.t.} \ \ h(\mathcal{G}_\theta(I_\gamma(\mathbf{x}^f) + \delta), \mathbf{x}^p) \neq h(\mathbf{x}^f, \mathbf{x}^p), \end{aligned} \tag{2}$$

where $\mathbf{x}^p$ and $\mathbf{x}^f$ indicate the protected and non-protected features of $\mathbf{x}$ and Eqn. 2 finds the minimal perturbation δ by changing the non-protected features $\mathbf{x}^m$ constrained to the data-manifold.

3.2 Adversarial example generation methods

Similar to counterfactual explanation methods, most methods generating adversarial examples also solve a constrained optimization problem to find perturbations in the input manifold that cause models to misclassify. These methods are broadly categorized into *poisoning* (e.g., Shafahi et al. (2018)) and *exploratory* (e.g., Goodfellow et al. (2014)) methods. While *poisoning* methods attack the model during training and attempts to learn, influence, and corrupt the underlying training data or model, *Exploratory* methods do not tamper with the underlying model but instead generate specific examples that cause the model to produce the desired output. Like counterfactual explanation methods, evasion methods also use gradient-based optimization to generate adversarial examples. Below, we briefly outline three evasion techniques considered in this work.

C&W Attack. For a given input $\mathbf{x}$ and classifier $h(\cdot)$, Carlini and Wagner (2017) formulate the problem of finding an adversarial example $\mathbf{x}' = \mathbf{x} + \delta$ such that $h(\mathbf{x}') \neq h(\mathbf{x})$ as:

$$\arg\min_{\mathbf{x}'} c \cdot \ell(\mathbf{x}') + d(\mathbf{x}, \mathbf{x}') \quad \text{s.t.} \quad \mathbf{x}' \in [0, 1]^d \tag{3}$$

where c is a hyperparameter and $\ell(\cdot)$ is a loss function such that $h(\mathbf{x}')=y$ if and only if $\ell(\mathbf{x}') \leq 0$. The constraint $\mathbf{x}' \in [0,1]^d$ is applied so that the resulting $\mathbf{x}'$ is within a given data range.

DeepFool. For a given instance $\mathbf{x}$, DeepFool (Moosavi-Dezfooli et al., 2016) perturbs $\mathbf{x}$ by adding small perturbation δ_{DF} at each iteration of the algorithm. The perturbations from each iterations are finally aggregated to generate the final perturbation once the output label changes. The minimal perturbation to change the classification model's prediction is the solution to the following objective:

$$\begin{aligned} \delta^*_{\text{DF}}(\mathbf{x}) &= \arg\min_{\delta} ||\delta||_2 \\ &\text{s.t. } \operatorname{sign}(f(\mathbf{x}+\delta)) \neq \operatorname{sign}(f(\mathbf{x})), \end{aligned} \tag{4}$$

where $\mathbf{x}$ is the input sample. The closed-form step for each iteration is: $\delta^*_{\text{DF}}= -(f(\mathbf{x})/||\nabla f(\mathbf{x})||_2^2)\nabla f(\mathbf{x})$.

Natural Adversarial Example (NAE). Similar to C-CHVAE, Zhao et al. (2018) proposes NAE to search for adversarial examples using a generative model $\mathcal{G}_\theta$ where the similarity is measured in the latent space of $\mathcal{G}_\theta$. Thus, the objective is given by:

$$\mathbf{z}^* = \arg\min_{\tilde{\mathbf{z}} \in \mathcal{Z}} \|\tilde{\mathbf{z}} - I_\gamma(\mathbf{x})\| \text{ s.t. } h(\mathcal{G}_\theta(\tilde{\mathbf{z}})) \neq h(\mathbf{x}), \tag{5}$$

where $I_\gamma(\mathbf{x})$ corresponds to the latent representation of $\mathbf{x}$ and $\mathcal{G}_\theta(\tilde{\mathbf{z}})$ maps the latent sample to the feature space. NAE separately trains an inverter function from $\mathcal{G}_\theta$ by enforcing the latent representation to be normally distributed (*i.e.,* corresponding to the noise model of the generator) while minimizing the reconstruction error of the feature space.

4 THEORETICAL ANALYSIS

In this section, we provide theoretical connections between counterfactual explanation and adversarial example methods by leveraging similarities in the objective functions and optimization procedures. In particular, we compare: 1) SCFE and C&W (Sec. 4.1), 2) SCFE and DeepFool (Sec. 4.2), and 3) C-CHVAE and NAE (Sec. 4.3) due to their similarity in the objective functions. We do these comparisons either for a specific loss, solutions based on the classification model, or constraints imposed during optimization. We focus on locally linear model approximations as these are often studied as a first step (Hardt and Ma, 2017; Ustun et al., 2019; Rosenfeld et al., 2020; Garreau and Luxburg, 2020) towards understanding nonlinear model behaviour.

4.1 SCFE and C&W

Two popular gradient-based methods for generating adversarial and counterfactual samples are the C&W Attack and SCFE, respectively. Here, we first show the closed-form solutions for the minimum perturbation required by C&W (δ^*_{CW}) and SCFE (δ^*_{SCFE}) to generate adversarial examples and counterfactuals. We then leverage these solutions to derive an upper bound for the distance between the adversarial and counterfactual samples. Using the loss function $\ell^*(\cdot)=\max(0, \max_i(f(\mathbf{x})_i) - f(\mathbf{x})_y)$ recommended by Carlini and Wagner (2017), we derive an upper bound for the distance between the counterfactuals and adversarial examples generated using SCFE and C&W. For the upper bound, we first state a lemma that derives the closed-form solution for δ^*_{SCFE}.

Lemma 1. *(Optimal Counterfactual Perturbation) For a scoring function with weights $\mathbf{w}$ the SCFE method generates a counterfactual $\mathbf{x}_{SCFE}$ for an input $\mathbf{x}$ using the counterfactual perturbation δ^*_{SCFE} such that:*

$$\delta^*_{SCFE} = m \cdot (\mathbf{w}\mathbf{w}^{\mathbf{T}} + \lambda \mathbf{I})^{-1}\mathbf{w}, \tag{6}$$

where s is the target score for $\mathbf{x}$, $m{=}s{-}f(\mathbf{x})$ is the target residual, $f(x){=}\mathbf{w}^\top\mathbf{x} + b$ is a local linear score approximation, and λ is a given hyperparameter.

Proof Sketch. We derive the closed-form solution for δ^*_{SCFE} by formulating the SCFE objective in its vector quadratic form. See Appendix B.1 for the complete proof. □

Using Lemma 1, we now formally state and derive the upper bound for the distance between the counterfactuals and adversarial examples.

Theorem 1. *(Difference between SCFE and C&W) Under the same conditions as stated in Lemma 1, the normed difference between the SCFE counterfactual $\mathbf{x}_{SCFE}$ and C&W adversarial example $\mathbf{x}_{CW}$ using the loss function $\ell^*(\cdot)$ is upper bounded by:*

$$\begin{aligned} &\|\mathbf{x}_{SCFE} - \mathbf{x}_{CW}\|_p \\ &\leq \left\| \frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \|\mathbf{w}\|_2^2}\right)(s - f(\mathbf{x})) - c\mathbf{I} \right\|_p ||\mathbf{w}||_p. \end{aligned} \tag{7}$$

Proof Sketch. We first derive the closed-form solution for the perturbation used by C&W. Intuitively, this solution is equivalent to shifting $\mathbf{x}$ in the direction of the models' decision boundary scaled by c. The upper bound follows by applying Lemma 1 and Cauchy-Schwartz inequality. Moreover, choosing the hyperparameter such that $\lambda \to 0$ and setting $c{=}m/\|\mathbf{w}\|_2^2$ yields equivalence, *i.e.,* $||\mathbf{x}_{\text{SCFE}} - \mathbf{x}_{\text{CW}}||_p \to 0$. See Appendix B.3 for the complete proof. □

We note that the upper bound is smaller when the original score $f(\mathbf{x})$ is close to the target score s, suggesting that $\mathbf{x}_{\text{SCFE}}$ and $\mathbf{x}_{\text{CW}}$ are more similar when $\mathbf{x}$ is closer to the decision boundary.

4.2 SCFE and DeepFool

DeepFool is an adversarial attack that uses an iterative gradient-based optimization approach to generate adversarial examples. Despite the differences in the formulations of SCFE and DeepFool, our theoretical analysis reveals a striking similarity between the two methods. In particular, we provide an upper bound for the distance between the solutions output by counterfactuals and adversarial examples generated using SCFE and DeepFool, respectively.

Theorem 2. *(Difference between SCFE and DeepFool) Under the same conditions as stated in Lemma 1, the normed difference between the SCFE counterfactual $\mathbf{x}_{SCFE}$ and the DeepFool adversarial example $\mathbf{x}_{DF}$ is upper bounded by:*

$$\|\mathbf{x}_{SCFE} - \mathbf{x}_{DF}\|_p \leq \left\| \left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \|\mathbf{w}\|_2^2}\right) \frac{(s - f(\mathbf{x}))}{\lambda} + \mathbf{I}\frac{f(\mathbf{x})}{\|\mathbf{w}\|_2^2} \right\|_p \|\mathbf{w}\|_p. \tag{8}$$

Proof Sketch. We show the similarity between SCFE and DeepFool methods by comparing their closed-form solutions for the generated counterfactual and adversarial examples. Similar to Theorem 1, the results follow from Cauchy-Schwartz inequality, (see Appendix B.4 for the complete proof). Moreover, choosing the hyperparameter such that $\lambda \to 0$ and setting s:=0 yields equivalence, *i.e.,* $||\mathbf{x}_{\text{SCFE}} - \mathbf{x}_{\text{DF}}||_p \to 0$. □

The right term in the inequality (Eqn. 8) entails that the l_p-norm of the difference between the generated samples is bounded if: 1) the predicted score is closer to the target score of a given input, and 2) the gradients with respect to the logit scores of the underlying model are bounded.

4.3 Manifold-based methods

We formalize the connection between manifold-based methods by comparing NAE to C-CHVAE as both rely on generative models. While C-CHVAE uses variational autoencoders, NAE uses GANs, specifically Wasserstein GAN (Arjovsky et al., 2017), to generate adversarial example. To allow a fair comparison, we assume that both methods use the same generator $\mathcal{G}_\theta$ and inverter I_γ networks.

Proposition 1. *Let p=∅ in C-CHVAE. Assuming that C-CHVAE and NAE use the same generator $\mathcal{G}_\theta$ and inverter functions $\mathcal{I}_\theta$. Then the proposed objectives of NAE and C-CHVAE are equivalent.*

Proof. Since p=∅, equation 2 reduces to:

$$\delta^* = \arg\min_{\delta \in \mathcal{Z}} \|\delta\| \text{ s.t. } h(\mathcal{G}_\theta(I_\gamma(\mathbf{x}^f) + \delta)) \neq h(\mathbf{x}^f) \tag{9}$$

Also, $I_\gamma(\mathbf{x})$=$\mathbf{z}$. Replacing $\tilde{\mathbf{z}} - \mathbf{z} = \delta$ in eqn. 5, we get:

$$\delta^* = \arg\min_{\delta \in \mathcal{Z}} \|\delta\| \text{ s.t. } h(\mathcal{G}_\theta(I_\gamma(\mathbf{x}) + \delta)) \neq h(\mathbf{x}) \tag{10}$$

Since $\mathbf{x}^f$=$\mathbf{x}$, we get the equivalence. □

Both C-CHVAE and NAE use search methods to generate adversarial examples or counterfactuals using the above objective function. In particular, both NAE and C-CHVAE samples $\mathbf{z}$ using an ℓ_p-norm ball of radius range $(r_{\text{NAE}}, \Delta r_{\text{NAE}}]$ and r_{C}. $\tilde{\mathbf{z}}_{\text{NAE}}$ denotes the solution returned by Zhao et al. (2018) and $\tilde{\mathbf{z}}_{\text{C}}$ the solution returned by C-CHVAE. We denote r^*_{NAE} and r^*_{C} as the corresponding radius parameters from NAE and C-CHVAE, respectively, and restrict our analysis to the class of L-Lipschitz generative models:

Definition 1. *Bora et al. (2017): A generative model $\mathcal{G}_\theta(\cdot)$ is L-Lipschitz if $\forall$ $\mathbf{z}_1$, $\mathbf{z}_2 \in \mathcal{Z}$, we have,*

$$\|\mathcal{G}_\theta(\mathbf{z}_1) - \mathcal{G}_\theta(\mathbf{z}_2)\|_p \leq L\|\mathbf{z}_1 - \mathbf{z}_2\|_p. \tag{11}$$

Note that commonly used DNN models comprise of linear, convolutional and activation layers, which satisfy Lipschitz continuity (Gouk et al., 2021).

Lemma 2. *(Difference between C-CHVAE and NAE) Let $\tilde{\mathbf{z}}_C$ and $\tilde{\mathbf{z}}_{NAE}$ be the output generated by C-CHVAE and NAE by sampling from ℓ_p-norm ball in the latent space using an L-Lipschitz generative model $\mathcal{G}_\theta(\cdot)$. Analogously, let $\mathbf{x}_{NAE}$=$\mathcal{G}_\theta(\tilde{\mathbf{z}}_{NAE})$ and $\mathbf{x}_C$=$\mathcal{G}_\theta(\tilde{\mathbf{z}}_C)$ generate perturbed samples by design of the two methods. Let r^*_{NAE} and r^*_C be the corresponding radii chosen by each algorithm such that they successfully return an adversarial example or counterfactual. Then, $\|\mathbf{x}_C - \mathbf{x}_{NAE}\|_p \leq L(r^*_C + r^*_{NAE})$.*

Proof Sketch. The proof follows from triangle inequality, L-Lipschitzness of the generative model, and the fact that the ℓ_p-norm of the method's outputs are known in the latent space. See Appendix B.5 for a detailed proof. □

Intuitively, the adversarial example and counterfactual explanation generated by the methods are bounded depending on the data manifold properties (captured by the Lipschitzness of the generative model) and the radius hyperparameters used by the search algorithms.

4.4 On the Underlying Assumptions

The analyzed counterfactual explanation objectives in our analysis slightly differ from their original implementations. Hence, the following remarks are in place. First, the original objective stated by Wachter et al. (2017) includes the median absolute deviation (MAD) for each input as a normalization term for the regularizer. However, the regularizer is independent of δ, and can therefore be incorporated into Lemma 1 and Theorem 1. Second, our analysis focuses on objective functions from SCFE and C-CHVAE explanation methods which consider generic distance functions. Hence, to facilitate a fair comparison across all counterfactual explanation and adversarial example algorithms, we use ℓ_2-norm in our analysis. Finally, immutability constraints can be readily incorporated. For instance, instead of taking the derivative of the SCFE objective function with respect to all available features, we take the derivative with respect to the mutable features only.

5 EXPERIMENTS

We now present the empirical analysis to demonstrate the similarities between counterfactual explanations and adversarial examples. More specifically, we verify the validity of our theoretical upper bounds using real-world datasets and determine the extent to which counterfactual explanations and adversarial examples similar to each other.

5.1 Experimental Setup

We first describe the synthetic and real-world datasets used to study the connections between counterfactual explanations and adversarial examples, and then we outline our experimental setup.

Synthetic Data. We generate 5000 samples from a mixture of Gaussians with pdfs $\mathcal{N}(\mu_1{=}[1.0, 1.0], \Sigma_1{=}\mathbf{I})$ and $\mathcal{N}(\mu_2{=}[-1.0, -1.0], \Sigma_2{=}\mathbf{I})$.

Real-world Data. We use three datasets in our experiments. 1) The *UCI Adult* dataset (Dua and Graff, 2017) consisting of 48842 individuals with demographic (*e.g.,* age, race, and gender), education (degree), employment (occupation, hours-per-week), personal (marital status, relationship), and financial (capital gain/loss) features. The task is to predict whether an individual's income exceeds $50K per year or not. 2) The *COMPAS* dataset (Mattu et al., 2016) comprising of 10000 individuals representing defendants released on bail. The task is to predict whether to release a defendant on bail or not using features, such as criminal history, jail, prison time, and defendant's demographics. 3) The *German Credit* dataset from the UCI repository (Dua and Graff, 2017) consisting of demographic (age, gender), personal (marital status), and financial (income, credit duration) features from 1000 credit applications. The task is to predict whether an applicant qualifies for credit or not.

Methods. Following our analysis in Sec. 4, we compare the following pair of methods: i) SCFE (Wachter et al., 2017) vs. C&W (Carlini and Wagner, 2017), ii) SCFE vs. DeepFool (Moosavi-Dezfooli et al., 2016), and iii) C-CHVAE (Pawelczyk et al., 2020a) vs. NAE (Zhao et al., 2018).

Prediction Models. For the synthetic dataset, we train a logistic regression model (LR) to learn the mixture component (samples and corresponding decision boundary shown in Fig. 1), whereas for real-world datasets, we obtain adversarial examples and counterfactuals using LR and artificial neural network (ANN) models. See Appendix C for more details.

Implementation Details For all real-world data, adversarial examples and counterfactuals are generated so as to flip the target prediction label from unfavorable ($y{=}0$) to favorable ($y{=}1$). We use ℓ_2-norm as the distance function in all our experiments. We partition the dataset into train-test splits where the training set is used to train the predictor models. Adversarial examples and counterfactuals are generated for the trained models using samples in the test splits. For counterfactual explanation methods applied to generate recourse, all features are assumed actionable for fair comparison with adversarial example generation methods. See Appendix C for more implementation details.

5.2 Results

Validating our Theoretical Upper Bounds. We empirically validate the theoretical upper bounds obtained in Sec. 4. To this end, we first estimate the bounds for each instance in the test set according to Theorems 1 and 2, and compare them with the empirical estimates of the ℓ_2-norm differences (LHS of Theorems 1 and 2). We use the same procedure to validate the bounds from Lemma 3.

SCFE vs. C&W and DeepFool. In Fig. 2, we show the empirical evaluation of our theoretical bounds for all real-world datasets. For each dataset, we show four box-plots: empirical estimates (green) and theoretical upper bounds (blue) of the distance (ℓ_2-norm) between the resulting counterfactuals and adversarial examples for SCFE and C&W (labeled as SCFE vs. CW), and SCFE and DeepFool (labeled as SCFE vs. DF). Across all three datasets, we observe that no bounds were violated for both theorems. The gap between empirical and theoretical values is relatively small for German credit dataset as compared to COMPAS and Adult datasets. From Theorems 1 and 2, we see that the bound strongly depends on

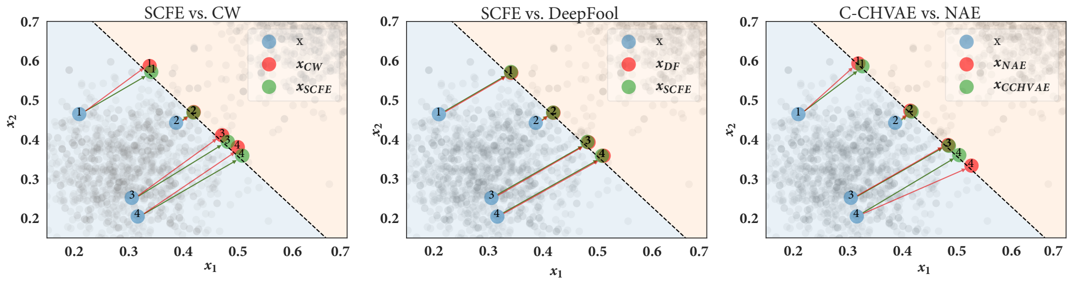
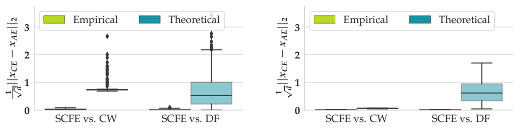


Figure 1: Similarity comparison of adversarial example and counterfactual explanation methods. Based on synthetic data, we generate adversarial examples (in red) and counterfactual explanations (in green) for some randomly chosen test set points (in blue) using methods described in Sec. 3. (**Left**) Both SCFE (in green) and C&W (in red) samples are close to each other, indicating strong similarity between these methods. (**Middle**) SCFE (in green) and DeepFool (in red) samples exactly coincide, indicating equivalence. (**Right**) C-CHVAE (in green) and NAE (in red) samples are closer if the blue factual points are closer to the boundary.


(a) COMPAS – LR (b) German Credit – LR

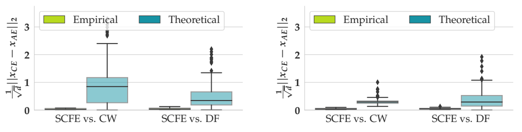
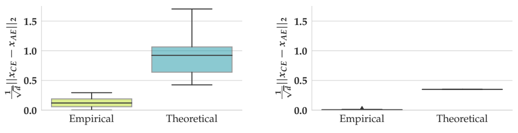

(c) COMPAS – ANN (d) German Credit – ANN

Figure 2: Verifying the theoretical bounds from Theorems 1 and 2. The green boxplots correspond to the empirical norm differences between SCFE (i.e., x_{CE}) and CW or DF (i.e., x_{AE}). The blue boxplots show the distribution of upper bounds, which we evaluated by plugging in the necessary quantities (hyperparameters, gradients, logit values) into equations 7 and 8. No bounds are violated. For ANNs, the upper bounds were computed using local linear model approximations.



(a) Adult – ANN (b) COMPAS – ANN

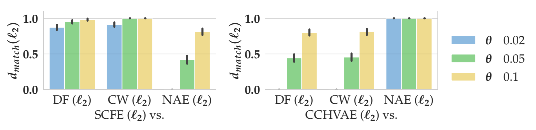
Figure 3: Verifying the theoretical bounds from Lemma 2. The green boxplots correspond to the empirical norm differences between CCHVAE (i.e., x_{CE}) and NAE (i.e., x_{AE}). The blue boxplots show the distribution of upper bounds, which we evaluated by plugging in the corresponding quantities (hyperparameters, Lipschitz constant) into the upper bound from Lemma 2. The Lipschitz constant is computed based on decoders and encoders using Lemma 4. No bounds are violated.

the norm of the logit score gradient $w{=}\nabla_x f(\mathbf{x})$, *e.g.*, for Adult dataset these norms are relatively higher leading to less tight bounds.

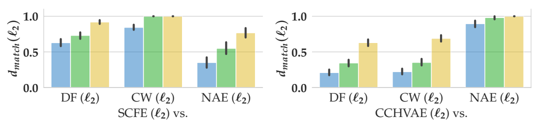
C-CHVAE vs.NAE. In Fig. 3, we validate the bounds obtained in Lemma 3 for all three datasets using an encoder-decoder framework. We observe that our upper bounds are tight, thus validating our theoretical analysis for comparing manifold-based counterfactual explanation (C-CHVAE) and adversarial example generation method (NAE).

Similarities between Counterfactuals and Adversarial examples. Here, we qualitatively and quantitatively show the similarities between counterfactuals and adversarial examples using several datasets.

Analysis with Synthetic Data. In Fig. 1, we show the similarity between counterfactual explanations and adversarial examples generated for a classifier trained on a two-dimensional mixture of Gaussian datasets. Across all cases, we observe that most output samples generated by counterfactual explanation and adversarial example methods overlap. In particular, for samples near the decision boundary, the solutions tend to be more similar. These results confirm our theoretical bounds, which depend on the difference between the logit sample prediction $f(\mathbf{x})$ and the target score s. If points are close to the decision boundary, $f(\mathbf{x})$ is closer to s, suggesting that the resulting counterfactual and adversarial example



(a) COMPAS



(b) Adult

Figure 4: Analyzing to what extent different counterfactual explanation methods and adversarial example generation methods are empirically equivalent for the logistic regression classifier. To do that, we compute d_{match} from Eqn. 12. Missing bars indicate that there was no match.

will be closer as implied by Theorems 1 and 2.

Analysis with Real Data. For real-world datasets, we define two additional metrics beyond those studied in our theoretical analysis to gain a more granular understanding about the similarities of counterfactuals and adversarial examples. First, we introduce d_{match} which quantifies the similarity between counterfactuals (i.e., $\mathbf{x}_{\text{CE}}$) and adversarial examples (i.e., $\mathbf{x}_{\text{AE}}$):

$$d_{\text{match}} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left[\frac{1}{\sqrt{d}}\|\mathbf{x}_{\text{CE}}^{(i)} - \mathbf{x}_{\text{AE}}^{(i)}\|_2 < \theta\right], \qquad (12)$$

where n is the total number of instances used in the analysis and $\theta \in \{0.02, 0.05, 0.1\}$ is a threshold determining when to consider counterfactual and adversarial examples as equivalent. d_{match} evaluates whether counterfactuals and adversarial examples are exactly the same with higher d_{match} implying higher similarity. Second, we complement d_{match} by Spearman rank ρ between δ_{CE} and δ_{AE}, which is a rank correlation coefficient measuring to what extent the perturbations' rankings agree, i.e., whether adversarial example generation methods and counterfactual explanation methods deem the same dimensions important in order to arrive at their final outputs. Here, $\rho(\delta_{\text{CE}}, \delta_{\text{AE}})=1$ implies that the rankings are same, 0 suggests that the rankings are independent, and -1 indicates reversely ordered rankings.

In Fig. 4, we compare a given counterfactual explanation method to salient adversarial example generation methods (DeepFool, C&W, and NAE) using d_{match}. We show the results for Adult and COMPAS datasets using LR models and relegate results for German Credit as well as neural network classifiers to Appendix D. Our results in Fig. 4 validate that the SCFE method is similar to DeepFool and C&W (higher d_{match} for lower θ). Across all datasets, this result aligns and validates with the similarity analysis in Sec. 4. Similarly, manifold-based methods demonstrate higher d_{match} compared to non-manifold methods (right panels in Fig. 4). Additionally, we show the results from the rank correlation analysis in Table 1 and observe that the maximum rank correlations (between 0.90 and 1.00) are obtained for methods that belong to the same categories, suggesting that the considered counterfactuals and adversarial examples are close to being equivalent.

6 CONCLUSION

In this work, we formally analyzed the connections between state-of-the-art adversarial example generation methods and counterfactual explanation methods. To this end, we first highlighted salient counterfactual explanation and adversarial example methods in literature, and leveraged similarities in their objective functions, optimization algorithms and constraints utilized in these methods to theoretically analyze conditions for equivalence and bound the distance between the solutions output by counterfactual explanation and adversarial example generation methods. For locally linear models, we bound the distance between the solutions obtained by C&W and SCFE using loss functions preferred in the respective works. We obtained similar bounds for the solutions of DeepFool and SCFE. We also demonstrated equivalence between the manifold-based methods of NAE and C-CHVAE and bounded the distance between their respective solutions. Finally, we empirically evaluated our theoretical findings on simulated and real-world data sets.

By establishing theoretically and empirically that several popular counterfactual explanation algorithms are generating extremely similar solutions as those of well known adversarial example algorithms, our work raises fundamental questions about the design and development of existing counterfactual explanation algorithms. *Do we really want counterfactual explanations to resemble adversarial examples, as our work suggests they do? How can a decision maker distinguish an adversarial attack from a counterfactual explanation? Does this imply that decision makers are tricking their own models by issuing counterfactual explanations? Can we do a better job of designing counterfactual explanations?* Moreover, by establishing connections between popular counterfactual explanation and adversarial example algorithms, our work opens up the possibility of using insights from adversarial robustness literature to improve the design and development of counterfactual explanation algorithms.

We hope our formal analysis helps carve a path for more robust approaches to counterfactual explana-

Table 1: Average Spearman rank correlation between counterfactual and adversarial perturbations. For every input $\mathbf{x}$, we compute the corresponding adversarial perturbation δ_{AE} and the counterfactual perturbation δ_{CE}. We then compute Spearman's $\rho(\delta_{\mathrm{AE}}, \delta_{\mathrm{CE}})$ and report their means (gradient-based: (g); manifold-based: (m)).

	COMPAS				Adult			
	LR		ANN		LR		ANN	
Model	SCFE (g)	CCHVAE (m)	SCFE (g)	CCHVAE (m)	SCFE (g)	CCHVAE (m)	SCFE (g)	CCHVAE (m)
CW (g)	0.88 ± 0.16	0.67 ± 0.30	0.93 ± 0.10	0.67 ± 0.22	$\mathbf{0.95 \pm 0.06}$	0.86 ± 0.10	0.92 ± 0.09	0.70 ± 0.16
DF (g)	$\mathbf{0.91 \pm 0.12}$	0.68 ± 0.31	$\mathbf{0.97 \pm 0.03}$	0.65 ± 0.22	0.92 ± 0.06	0.80 ± 0.13	$\mathbf{0.93 \pm 0.08}$	0.63 ± 0.20
NAE (m)	0.57 ± 0.35	$\mathbf{0.94 \pm 0.08}$	0.71 ± 0.19	$\mathbf{1.00 \pm 0.00}$	0.83 ± 0.12	$\mathbf{0.90 \pm 0.10}$	0.74 ± 0.13	$\mathbf{0.98 \pm 0.02}$

tions, a critical aspect for calibrating trust in ML. Improving our theoretical bounds using other strategies and deriving new theoretical bounds for other approaches is an interesting future direction.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful feedback. This work is supported in part by the NSF awards #IIS-2008461 and #IIS-2040989, and research awards from the Harvard Data Science Institute, Amazon, Bayer, and Google. HL would like to thank Mohan and Sujatha Lakkaraju, and Pracheer Gupta for all their inputs and support. The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.

References

Akhtar, N. and Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. arxiv 2017. *arXiv preprint arXiv:1701.07875*, 30.

Ballet, V., Renard, X., Aigrain, J., Laugel, T., Frossard, P., and Detyniecki, M. (2019). Imperceptible adversarial attacks on tabular data. *arXiv preprint arXiv:1911.03274*.

Barocas, S., Selbst, A. D., and Raghavan, M. (2020). The hidden assumptions behind counterfactual explanations and principal reasons. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 80–89.

Biggio, B., Nelson, B., and Laskov, P. (2012). Poisoning attacks against support vector machines. In *ICML*.

Bora, A., Jalal, A., Price, E., and Dimakis, A. G. (2017). Compressed sensing using generative models. In *International Conference on Machine Learning*, pages 537–546. PMLR.

Browne, K. and Swift, B. (2020). Semantics and explanation: why counterfactual explanations produce adversarial examples in deep neural networks. *arXiv preprint arXiv:2012.10076*.

Carlini, N. and Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE.

Cartella, F., Anunciacao, O., Funabiki, Y., Yamaguchi, D., Akishita, T., and Elshocht, O. (2021). Adversarial attacks for tabular data: Application to fraud detection and imbalanced data. *arXiv preprint arXiv:2101.08030*.

Cisse, M., Adi, Y., Neverova, N., and Keshet, J. (2017). Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373*.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Freiesleben, T. (2020). Counterfactual explanations & adversarial examples–common grounds, essential differences, and potential transfers. *arXiv preprint arXiv:2009.05487*.

Garreau, D. and Luxburg, U. (2020). Explaining the explainer: A first theoretical analysis of lime. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1287–1296. PMLR.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Gouk, H., Frank, E., Pfahringer, B., and Cree, M. J. (2021). Regularisation of neural networks by enforcing lipschitz continuity. Springer.

Hardt, M. and Ma, T. (2017). Identity matters in deep learning. In *International Conference on Learning Representations (ICLR)*.

Joshi, S., Koyejo, O., Vijitbenjaronk, W., Kim, B., and Ghosh, J. (2019). Towards realistic individual recourse and actionable explanations in black-box decision making systems. *arXiv preprint arXiv:1907.09615*.

Karimi, A.-H., Barthe, G., Balle, B., and Valera, I. (2020a). Model-agnostic counterfactual explanations for consequential decisions. In *International*

Conference on Artificial Intelligence and Statistics (AISTATS), pages 895–905. PMLR.

Karimi, A.-H., Barthe, G., Schölkopf, B., and Valera, I. (2020b). A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050*.

Karimi, A.-H., Schölkopf, B., and Valera, I. (2021). Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, pages 353–362.

Karimi, A.-H., von Kügelgen, J., Schölkopf, B., and Valera, I. (2020c). Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. *arXiv preprint arXiv:2006.06831*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kurakin, A., Goodfellow, I., Bengio, S., et al. (2016). Adversarial examples in the physical world.

Lundberg, S. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

Mattu, S., Kirchner, L., and Angwin, J. (2016). How we analyzed the compas recidivism algorithm. *ProPublica*.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582.

Pawelczyk, M., Bielawski, S., van den Heuvel, J., Richter, T., and Kasneci, G. (2021). Carla: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms. *arXiv preprint arXiv:2108.00783*.

Pawelczyk, M., Broelemann, K., and Kasneci, G. (2020a). Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference 2020*, pages 3126–3132.

Pawelczyk, M., Broelemann, K., and Kasneci, G. (2020b). On counterfactual explanations under predictive multiplicity. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124.

Rawal, K. and Lakkaraju, H. (2020). Interpretable and interactive summaries ofactionable recourses. *arXiv preprint arXiv:2009.07165*.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Rosenfeld, E., Winston, E., Ravikumar, P., and Kolter, Z. (2020). Certified robustness to label-flipping attacks via randomized smoothing. In *International Conference on Machine Learning*, pages 8230–8241. PMLR.

Shafahi, A., Huang, W. R., Najibi, M., Suciu, O., Studer, C., Dumitras, T., and Goldstein, T. (2018). Poison frogs! targeted clean-label poisoning attacks on neural networks. In *NeurIPS*.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *International Conference on Machine Learning (ICML)*, pages 3319–3328. PMLR.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tavallali, P., Behzadan, V., Tavallali, P., and Singhal, M. (2021). Adversarial poisoning attacks and defense for general multi-class models based on synthetic reduced nearest neighbors. *arXiv*.

Ustun, B., Spangher, A., and Liu, Y. (2019). Actionable recourse in linear classification. *Proceedings of the Conference on Fairness, Accountability, and Transparency*.

Van Looveren, A. and Klaise, J. (2019). Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*.

Venkatasubramanian, S. and Alfano, M. (2020). The philosophical basis of algorithmic recourse. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT*)*, pages 284–293.

Verma, S., Dickerson, J., and Hines, K. (2020). Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*.

Wachter, S., Mittelstadt, B., and Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the gdpr. In *Harvard Journal & Technology*.

Zhao, Z., Dua, D., and Singh, S. (2018). Generating natural adversarial examples. In *International Conference on Learning Representations (ICLR)*.

Supplementary Material: Exploring Counterfactual Explanations Through the Lens of Adversarial Examples: A Theoretical and Empirical Analysis

APPENDIX SUMMARY

Section A provides a categorization of counterfactual explanation and adversarial example methods. In Section B, we provide detailed proofs for Lemmas 1 and 3, and Theorems 1 and 2. In Section C, we provide implementation details for all models used in our experiments including (i) the supervised learning models, (ii) the counterfactual explanation and adversarial example methods, and the (iii) generative models required to run the manifold-based methods. Finally, in Section D, we present the remaining experiments we referred to in the main text.

A TAXONOMY OF COUNTERFACTUAL AND ADVERSARIAL EXAMPLE METHODS

In order to choose methods to compare across counterfactual explanation methods and adversarial example generation methods, we surveyed existing literature. We use a taxonomy to categorize each subset of methods based on various factors. The main characteristics we use are based on type of method, based on widely accepted terminology and specific implementation details. In particular, we distinguish between i) constraints imposed for generating adversarial examples or counterfactual explanations, ii) algorithms used for generating them. For the class of adversarial example generation methods, we further distinguish between *poisoning attacks* and *evasion attacks* and note that evasion attacks are most closely related to counterfactual explanation methods. The taxonomy for counterfactual explanation methods is provided in Table 2 and that for adversarial example generation methods is provided in Table 3.

The main algorithm types used for counterfactual explanation methods are search-based, gradient-based and one method that uses integer programming (Ustun et al., 2019). The main constraints considered are actionability i.e., only certain features are allowed to change, and counterfactual explanations are encouraged to be realistic using either causal and/or manifold constraints. Similarly, for adversarial example generation methods primarily, Greedy search-based and gradient-based methods are most common. Manifold constraints are also imposed in a few cases where the goal is to generate adversaries close to the data-distribution. Based on this taxonomy, we select the appropriate pairs of counterfactual explanation method and adversarial example generation method to compare to each other for theoretical analysis. This leads us to compare gradient-based methods SCFE and C&W attack, SCFE and DeepFool and finally, manifold-based methods C-CHVAE and NAE with their search-based algorithms.

Table 2: Taxonomy of counterfactual explanation methods

Algorithm	Constraints	Method
Search-based	Causal, Actionability	**MINT** (Karimi et al., 2020c)
	Manifold, Actionability	**C-CHVAE** (Pawelczyk et al., 2020a)
Gradient-based	Actionability	**CFE, SCFE** Wachter et al. (2017)
	Manifold, Actionability	**REVISE** (Joshi et al., 2019)
Integer-programming	Actionability/Linear black-box	**AR** (Ustun et al., 2019)

Table 3: Taxonomy of adversarial example generation methods

	Algorithm	Constraints	Method
Poisoning Attacks	Greedy Search Gradient-based	Manifold Data-domain	**Adv. Data Poisoning** (Tavallali et al., 2021) **SVM-attack** (Biggio et al., 2012) **One-Shot Kill** (Shafahi et al., 2018)
Evasion Attacks	Search-based Gradient-based	Manifold Data-domain	**NAE** (Zhao et al., 2018) **DeepFool** (Moosavi-Dezfooli et al., 2016) **C&W Attack** (Carlini and Wagner, 2017)

B PROOFS OF SECTION 4

B.1 Proof of Lemma 1

Lemma 1. *For a linear score function $f(x) = \mathbf{w}^\top \mathbf{x} + b$, the* ***SCFE*** *counterfactual for* $\mathbf{x}$ *on* f *is* $\mathbf{x}' = \mathbf{x} + \delta^*$ *where*

$$\delta^* = (\mathbf{w}\mathbf{w}^\mathbf{T} + \lambda \mathbf{I})^{-1}(s - \mathbf{w}^T\mathbf{x} - b)\mathbf{w}.$$

Proof. Reformulating Equation 1 using l_2-norm as the distance metric, we get:

$$\min_{\mathbf{x}'} (\mathbf{w}^T\mathbf{x}' + b - s)^2 + \lambda ||\mathbf{x}' - \mathbf{x}||_2^2.$$

We can convert this minimization objective into finding the minimum perturbation δ by substituting $\mathbf{x}' = \mathbf{x} + \delta$, *i.e.,*

$$\min_{\delta} (\mathbf{w}^T\mathbf{x} + \mathbf{w}^T\delta + b - s)^2 + \lambda ||\mathbf{x}' - \mathbf{x}||_2^2. \tag{13}$$

Using $s - \mathbf{w}^T\mathbf{x} - b = m$ as a dummy variable and $\mathbf{x}' - \mathbf{x} = \delta$, we get:

$$\begin{aligned}
&\min_{\delta} (\mathbf{w}^T\delta - m)^2 + \lambda ||\delta||_2^2 \\
&\min_{\delta} (\mathbf{w}^T\delta - m)^T(\mathbf{w}^T\delta - m) + \lambda \delta^T\delta \\
&\min_{\delta} (\delta^T\mathbf{w} - m)(\mathbf{w}^T\delta - m) + \lambda \delta^T\delta && (m \text{ is a scalar, hence } m^T = m) \\
&\min_{\delta} \delta^T\mathbf{w}\mathbf{w}^T\delta - 2m\delta^T\mathbf{w} + m^2 + \lambda\delta^T\delta \\
&\min_{\delta} \delta^T(\mathbf{w}\mathbf{w}^T + \lambda\mathbf{I})\delta - 2m\delta^T\mathbf{w} + m^2 \\
&\min_{\delta} \delta^T\mathbf{M}\delta - 2m\mathbf{w}^T\delta + m^2 && (\text{where } \mathbf{M} = \mathbf{w}\mathbf{w}^T + \lambda\mathbf{I}) \\
&\min_{\delta} \delta^T\mathbf{M}\delta - 2\eta^T\delta + m^2 && (\text{where } m\mathbf{w} = \eta) \\
&\min_{\delta} \delta^T\mathbf{M}\delta - 2\eta^T\delta + \eta^T\mathbf{M}^{-1}\eta - \eta^T\mathbf{M}^{-1}\eta + m^2 \\
&\min_{\delta} (\delta - \mathbf{M}^{-1}\eta)^T\mathbf{M}(\delta - \mathbf{M}^{-1}\eta) - \eta^T\mathbf{M}^{-1}\eta + m^2
\end{aligned}$$

The closed form solution is given by,

$$\delta^* = \mathbf{M}^{-1}\eta, \tag{14}$$

where $M = \mathbf{w}\mathbf{w}^T + \lambda\mathbf{I}$.

The expression in equation 14 can further be simplified:

$$\begin{aligned}
\delta^* &= \frac{m}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \|\mathbf{w}\|_2^2}\right)\mathbf{w} && (\text{Sherman-Morrison Formula}) \\
&= \frac{m}{\lambda}\left(\mathbf{I}\mathbf{w} - \mathbf{w}\frac{\|\mathbf{w}\|_2^2}{\lambda + \|\mathbf{w}\|_2^2}\right) \\
&= \frac{m}{\lambda} \cdot \frac{\lambda}{\lambda + \|\mathbf{w}\|_2^2} \cdot \mathbf{w} = \frac{m}{\lambda + \|\mathbf{w}\|_2^2} \cdot \mathbf{w},
\end{aligned} \tag{15}$$

where $m := s - \mathbf{w}^T\mathbf{x} - b$. Finally, we note that as $\lambda \to 0$, we have:

$$\delta^{**} = \frac{m}{\|\mathbf{w}\|_2^2} \cdot \mathbf{w}. \tag{16}$$

□

B.2 Proof of Lemma 3

Lemma 3. *For a binary classifier $h(\mathbf{x}) = g(f(\mathbf{x}))$ such that $f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x} + b$, $g(\mathbf{x}) = \sigma(\mathbf{x})$, and $h(\mathbf{x})$ is the probability that $\mathbf{x}$ is in the class $y = 1$,*

$$\ell^*(\mathbf{x}) = \max(0, -2(\mathbf{w}^\top\mathbf{x} + b))$$

.

Proof. Given our formulation of $h(\mathbf{x})$, $f(\mathbf{x})$ is the score corresponding to class $y = 1$. By the definition of $\sigma(\mathbf{x})$,

$$f(\mathbf{x}) = \ln \frac{h(\mathbf{x})}{1 - h(\mathbf{x})} = \ln h(\mathbf{x}) - \ln(1 - h(\mathbf{x}))$$

Then the score corresponding to the class $y = 0$ is

$$\ln \frac{1 - h(\mathbf{x})}{1 - (1 - h(\mathbf{x}))} = \ln \frac{1 - h(\mathbf{x})}{h(\mathbf{x})} = \ln(1 - h(\mathbf{x})) - \ln h(\mathbf{x}) = -f(\mathbf{x})$$

Substituting back into definition of $\ell^*(\mathbf{x})$,

$$\begin{aligned}\ell^*(\mathbf{x}) &= \max(0, \max_i(f(\mathbf{x})_i) - f(\mathbf{x})_y) \\ &= \max(0, (-f(\mathbf{x}) - f(\mathbf{x})) \\ &= \max(0, (-2f(\mathbf{x})) \\ &= \max(0, -2(\mathbf{w}^\top\mathbf{x} + b)).\end{aligned}$$

□

B.3 Proof of Theorem 1

Theorem 1. *For a linear classifier $h(\mathbf{x}) = g(f(\mathbf{x}))$ such that $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, the difference between the **SCFE** counterfactual example $\mathbf{x}_{SCFE}$ and the **C&W** adversarial example $\mathbf{x}_{CW}$ using the recommended loss function $\ell^*(\cdot) = \max(0, \max_i(f(\mathbf{x})_i) - f(\mathbf{x})_y)$ is given by:*

$$\|\mathbf{x}_{SCFE} - \mathbf{x}_{CW}\|_p \leq \left\| \frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \mathbf{w}^T\mathbf{w}}\right)(s - f(\mathbf{x})) - c\mathbf{I} \right\|_p \|\mathbf{w}\|_p$$

.

Proof. Consider a binary classifier $h(\mathbf{x}) = g(f(\mathbf{x}))$ such that $f(\mathbf{x}) = \mathbf{w}^\top\mathbf{x} + b$, $g(\mathbf{x}) = \sigma(\mathbf{x})$, and $h(\mathbf{x})$ is the probability that $\mathbf{x}$ is in the class $y = 1$. Then by Lemma 3 and using ℓ_2-nrom as the distance metric, we can write the **C&W Attack** objective as

$$\arg\min_{\mathbf{x}'} c \max(0, -2(\mathbf{w}^\top\mathbf{x}' + b)) + \|\mathbf{x} - \mathbf{x}'\|_2^2$$

We can convert this minimization objective into finding the minimum perturbation δ by substituting $\mathbf{x}' = \mathbf{x} + \delta$,

$$\arg\min_{\delta} c \max(0, -2(\mathbf{w}^\top\mathbf{x} + \mathbf{w}^\top\delta + b)) + \|\delta\|_2^2$$

The subgradients of this objective are

$$\begin{cases} 2\delta & \text{when } -2(\mathbf{w}^\top\mathbf{x} + \mathbf{w}^\top\delta + b) < 0 \\ -2c\mathbf{w} + 2\delta & \text{otherwise} \end{cases}$$

By Lemma 3, $-2(\mathbf{w}^\top\mathbf{x} + \mathbf{w}^\top\delta + b) = -f(\mathbf{x}) - f(\mathbf{x}) < 0$. This implies that $f(\mathbf{x}) > -f(\mathbf{x})$, i.e that the score for class $y = 1$ is greater than the score for $y = 0$. As this indicates an adversarial example has already been found, we focus on minimizing the other subgradient. Setting this subgradient equal to 0,

$$\begin{aligned} 0 &= -2c\mathbf{w} + 2\delta \\ \delta &= c\mathbf{w} \end{aligned}$$

Thus the minimum perturbation to generate and adversarial example using the **C&W Attack** is

$$\delta^*_{\text{CW}} = c\mathbf{w}$$

Now, taking the difference between the minimum perturbation to generate a **SCFE** counterfactual (Lemma 1) and DeepFool (equation 18), we get:

$$\begin{aligned} \delta^*_{\text{SCFE}} - \delta^*_{\text{CW}} &= (\mathbf{w}\mathbf{w}^T + \lambda\mathbf{I})^{-1}(s - \mathbf{w}^T\mathbf{x} - b)\mathbf{w} - c\mathbf{w} \\ &= ((\mathbf{w}\mathbf{w}^T + \lambda\mathbf{I})^{-1}(s - f(\mathbf{x})) - c\mathbf{I})\mathbf{w} \\ &= \left(\frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \mathbf{w}^T\mathbf{w}}\right)(s - f(\mathbf{x})) - c\mathbf{I}\right)\mathbf{w} \qquad \text{(Using Sherman–Morrison formula)} \end{aligned}$$

Taking the l_p-norm on both sides, we get:

$$\begin{aligned} \|\delta^*_{\text{SCFE}} - \delta^*_{\text{CW}}\|_p &= \left\| \left(\frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \mathbf{w}^T\mathbf{w}}\right)(s - f(\mathbf{x})) - c\mathbf{I}\right)\mathbf{w} \right\|_p \\ &\leq \left\| \frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \mathbf{w}^T\mathbf{w}}\right)(s - f(\mathbf{x})) - c\mathbf{I} \right\|_p \|\mathbf{w}\|_p \qquad \text{(Using Cauchy-Schwartz)} \end{aligned}$$

Adding and subtracting the input instance $\mathbf{x}$ in the left term, we get:

$$\begin{aligned} \|\mathbf{x} + \delta^*_{\text{SCFE}} - (\mathbf{x} + \delta^*_{\text{CW}})\|_p &\leq \left\| \frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \mathbf{w}^T\mathbf{w}}\right)(s - f(\mathbf{x})) - c\mathbf{I} \right\|_p \|\mathbf{w}\|_p \\ \|\mathbf{x}_{\text{SCFE}} - \mathbf{x}_{\text{CW}}\|_p &\leq \left\| \frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \mathbf{w}^T\mathbf{w}}\right)(s - f(\mathbf{x})) - c\mathbf{I} \right\|_p \|\mathbf{w}\|_p, \end{aligned}$$

where the final equation gives an upper bound on the difference between the **SCFE** counterfactual and the **C&W** adversarial example.

Furthermore, we ask under which conditions the normed difference becomes 0. We start with:

$$\delta^*_{\text{SCFE}} - \delta^*_{\text{CW}} = \frac{m}{\lambda + \|\mathbf{w}\|_2^2} \cdot \mathbf{w} - c \cdot \mathbf{w}$$

Taking the l_p-norm on both sides, we get:

$$\|\delta^*_{\text{SCFE}} - \delta^*_{\text{CW}}\|_p = \left| \frac{m}{\lambda + \|\mathbf{w}\|_2^2} - c \right| \cdot \|\mathbf{w}\|_p$$

If we were to choose $\lambda \to 0$ we would get:

$$\|\delta^{**}_{\text{SCFE}} - \delta^*_{\text{CW}}\|_p = \left| \frac{m - c \cdot \|\mathbf{w}\|_2^2}{\|\mathbf{w}\|_2^2} \right| \cdot \|\mathbf{w}\|_p,$$

where equality holds when the hyperparameter is chosen so that $c := \frac{m}{\|\mathbf{w}\|_2^2}$. □

B.4 Proof of Theorem 2

Theorem 2. *For a linear classifier $h(\mathbf{x}) = g(f(\mathbf{x}))$ such that $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, the difference between the counterfactual example $\mathbf{x}_{SCFE}$ generated by Wachter et al. (2017) and the adversarial example $\mathbf{x}_{DF}$ generated by Moosavi-Dezfooli et al. (2016) is given by:*

$$||\mathbf{x}_{SCFE} - \mathbf{x}_{DF}||_p \leq \left\| \frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \mathbf{w}^T\mathbf{w}}\right)(s - f(\mathbf{x})) + \left(\mathbf{I}\frac{f(\mathbf{x})}{\|\mathbf{w}\|_2^2}\right) \right\|_p \cdot ||\mathbf{w}||_p, \qquad (17)$$

Proof. The minimal perturbation to change the classifier's decision for a binary model $f(\mathbf{x})$ is given by the closed-form formula (Moosavi-Dezfooli et al., 2016):

$$\delta^*_{\text{DF}} = -\frac{f(\mathbf{x})}{||\mathbf{w}||_2^2}\mathbf{w}. \tag{18}$$

Now, taking the difference between the minimum perturbation added to an input instance $\mathbf{x}$ by Wachter algorithm (Lemma 1) and DeepFool (equation 18), we get:

$$\begin{aligned}
\delta^*_{\text{SCFE}} - \delta^*_{\text{DF}} &= (\mathbf{w}\mathbf{w}^T + \lambda\mathbf{I})^{-1}(s - \mathbf{w}^T\mathbf{x} - b)\mathbf{w} - \left(-\frac{f(\mathbf{x})}{||\mathbf{w}||_2^2}\mathbf{w}\right) \\
\delta^*_{\text{SCFE}} - \delta^*_{\text{DF}} &= \left((\mathbf{w}\mathbf{w}^T + \lambda\mathbf{I})^{-1}(s - f(\mathbf{x})) + \frac{f(\mathbf{x})}{||\mathbf{w}||_2^2}\right)\mathbf{w} \\
\delta^*_{\text{SCFE}} - \delta^*_{\text{DF}} &= \left(\frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \mathbf{w}^T\mathbf{w}}\right)(s - f(\mathbf{x})) + \frac{f(\mathbf{x})}{||\mathbf{w}||_2^2}\right)\mathbf{w} && \text{(Using Sherman–Morrison formula)}
\end{aligned}$$

Taking the l_p-norm on both sides, we get:

$$\begin{aligned}
\|\delta^*_{\text{SCFE}} - \delta^*_{\text{DF}}\|_p &= \left\|\left(\frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \mathbf{w}^T\mathbf{w}}\right)(s - f(\mathbf{x})) + \mathbf{I}\frac{f(\mathbf{x})}{\|\mathbf{w}\|_2^2}\right)\mathbf{w}\right\|_p \\
&\leq \left\|\frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \mathbf{w}^T\mathbf{w}}\right)(s - f(\mathbf{x})) + \mathbf{I}\frac{f(\mathbf{x})}{\|\mathbf{w}\|_2^2}\right\|_p \|\mathbf{w}\|_p && \text{(Using Cauchy-Schwartz)}
\end{aligned}$$

Adding and subtracting the input instance $\mathbf{x}$ in the left term, we get:

$$\begin{aligned}
\|\mathbf{x} + \delta^*_{\text{SCFE}} - (\mathbf{x} + \delta^*_{\text{DF}})\|_p &\leq \left\|\frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \mathbf{w}^T\mathbf{w}}\right)(s - f(\mathbf{x})) + \mathbf{I}\frac{f(\mathbf{x})}{\|\mathbf{w}\|_2^2}\right\|_p \|\mathbf{w}\|_p \\
\|\mathbf{x}_{\text{SCFE}} - \mathbf{x}_{\text{DF}}\|_p &\leq \left\|\frac{1}{\lambda}\left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}^T}{\lambda + \mathbf{w}^T\mathbf{w}}\right)(s - f(\mathbf{x})) + \mathbf{I}\frac{f(\mathbf{x})}{\|\mathbf{w}\|_2^2}\right\|_p \|\mathbf{w}\|_p,
\end{aligned}$$

where the final equation gives an upper bound on the difference between the **SCFE** counterfactual and the adversarial example from DeepFool (Moosavi-Dezfooli et al., 2016).

Furthermore, we ask under which conditions the normed difference becomes 0. If we were to choose $\lambda \to 0$ we would get:

$$\begin{aligned}
\|\delta^{**}_{\text{SCFE}} - \delta^*_{\text{DF}}\|_p &= \left\|\frac{-f(\mathbf{x}) + s}{\|\mathbf{w}\|_2^2} \cdot \mathbf{w} + \frac{-f(\mathbf{x})}{\|\mathbf{w}\|_2^2} \cdot \mathbf{w}\right\|_p \\
&= \frac{|s|}{\|\mathbf{w}\|_2^2} \cdot \|\mathbf{w}\|_p,
\end{aligned}$$

where equality holds when the target score is chosen so that s=0, which corresponds to a probability of Y=1 of 0.5.

□

B.5 Proof of Lemma 2

Lemma 2. *Let $\tilde{\mathbf{z}}_{NAE}$ be the solution returned Zhao et al. (2018, Algorithm 1) and $\tilde{\mathbf{z}}_C$ the solution returned by the counterfactual search algorithm of Pawelczyk et al. (2020a) by sampling from ℓ_p-norm ball in the latent space using an L-Lipschitz generative model $\mathcal{G}_\theta(\cdot)$. Analogously, let $\mathbf{x}_{NAE} = \mathcal{G}_\theta(\tilde{\mathbf{z}}_{NAE})$ and $\mathbf{x}_C = \mathcal{G}_\theta(\tilde{\mathbf{z}}_C)$ by design of the two algorithms. Let r^*_{NAE} and r^*_C be the corresponding radius chosed by each algorithm respectively that successfully returns an adversarial example or counterfactual explanation. Then, $\|\mathbf{x}_{NAE} - \mathbf{x}_C\| \leq L(r^*_{NAE} + r^*_C)$.*

Proof. The proof straightforwardly follows from triangle inequality and L-Lipschitzness of the Generative model:

$$\begin{aligned}
\|\mathbf{x}_{\text{NAE}} - \mathbf{x}_C\| &= \|\mathcal{G}_\theta(\tilde{\mathbf{z}}_{\text{NAE}}) - \mathcal{G}_\theta(\tilde{\mathbf{z}}_C)\|_p && (19)\\
&\leq \|\mathcal{G}_\theta(\tilde{\mathbf{z}}_{\text{NAE}}) - \mathbf{x} + \mathbf{x} - \mathcal{G}_\theta(\tilde{\mathbf{z}}_C)\|_p && (20)\\
&\leq \|\mathcal{G}_\theta(\tilde{\mathbf{z}}_{\text{NAE}}) - \mathbf{x}\|_p + \|\mathbf{x} - \mathcal{G}_\theta(\tilde{\mathbf{z}}_C)\|_p && (21)\\
&= \|\mathcal{G}_\theta(\tilde{\mathbf{z}}_{\text{NAE}}) - \mathcal{G}_\theta(\mathbf{z})\|_p + \|\mathcal{G}_\theta(\mathbf{z}) - \mathcal{G}_\theta(\tilde{\mathbf{z}}_C)\|_p && (22)\\
&\leq L\|\tilde{\mathbf{z}}_{\text{NAE}} - \mathbf{z}\|_p + L\|\mathbf{z} - \tilde{\mathbf{z}}_C\|_p && (23)\\
&\leq L\{r^*_{\text{NAE}} + r^*_C\} && (24)
\end{aligned}$$

where equation 20 follows from triangle inequality in the ℓ_p-norm, equation 23 follows from the Lipschitzness assumption and equation 24 follows from properties of the counterfactual search algorithms. □

In the following we outline a lemma that allows us to estimate the Lipschitz constant of the generative model. This will be used for empirical validation of our theoretical claims.

Lemma 4 (Bora et al. (2017))**.** *If G is a d-layer neural network with at most c nodes per layer, all weights $\leq w_{\max}$ in absolute value, and M -Lipschitz non-linearity after each layer, then $G(\cdot)$ is L -Lipschitz with $L = (Mcw_{\max})^d$.*

C EXPERIMENTAL SETUP

C.1 Implementation Details for Counterfactual Explanation and Adversarial Example Methods

For all datasets, categorical features are one-hot encoded and data is scaled to lie between 0 and 1. We partition the dataset into train-test splits. The training set is used to train the classification models for which adversarial examples and counterfactual explanations are generated. adversarial examples and counterfactual explanations are generated for all samples in the test split for the fixed classification model. For counterfactual explanation methods applied to generate recourse examples, all features are assumed actionable for comparison with adversarial examples methods. Adversarial examples and counterfactuals are appropriately generated using the prescribed algorithm implementations in each respective method. Specifically,

i) **SCFE**: As suggested in Wachter et al. (2017), an Adam optimizer (Kingma and Ba, 2014) is used to obtain counterfactual explanations corresponding to the cost function of equation 14. We have based our implementation on the implementation provided by Pawelczyk et al. (2021).

ii) **C-CHVAE**: A (V)AE is additionally trained to model the data-manifold as prescribed in Pawelczyk et al. (2020a). As suggested in Pawelczyk et al. (2020a), a counterfactual search algorithm in the latent space of the (V)AEs. Particularly, a latent sample within an ℓ_p-norm ball with a fixed search radius is used until a counterfactual example is successfully obtained. The search radius of the norm ball is increased until a counterfactual explanation is found. The architecture of the generative model is provided in Appendix C.3. We have based our implementation on the implementation provided by Pawelczyk et al. (2021).

iv) **C&W Attack**: As prescribed in Carlini and Wagner (2017), we use gradient-based optimization to find Adversarial Examples using this attack.

v) **DeepFool**: We implement Moosavi-Dezfooli et al. (2016, Algorithm 1) to generate Adversarial Examples using DeepFool.

vi) **NAE**: This method trains a generative model and an inverter to generate Adversarial Examples. For consistency of comparison with **C-CHVAE**, we use the decoder of the same (V)AE as the generative model for this method. The inverter then corresponds to the encoder of the (V)AE. We use Zhao et al. (2018, Algorithm 1) which uses an iterative search method to find natural adversarial examples. The algorithm searches for adversarial examples in the latent space with radius between $(r, r + \Delta r]$. The search radius is iteratively increased until an Adversarial Example is successfully found.

We describe architecture and training details for real-world data sets in the following.

C.2 Supervised Classification Models

All models are implemented in PyTorch and use a $80-20$ train-test split for model training and evaluation. We evaluate model quality based on the model accuracy. All models are trained with the same architectures across the data sets:

	Neural Network	Logistic Regression
Units	[Input dim. , 18, 9, 3, 1]	[Input dim. , 1]
Type	Fully connected	Fully connected
Intermediate activations	ReLU	N/A
Last layer activations	Sigmoid	Sigmoid

Table 4: Classification model details

		Adult	COMPAS	German Credit
Batch-size	NN	512	32	64
	Logistic Regression	512	32	64
Epochs	NN	50	40	30
	Logistic Regression	50	40	30
Learning rate	NN	0.002	0.002	0.001
	Logistic Regression	0.002	0.002	0.001

Table 5: Training details

	Adult	COMPAS	German Credit
Logistic Regression	0.83	0.84	0.71
Neural Network	0.84	0.85	0.72

Table 6: Performance of models used for generating adversarial examples and counterfactual explanations

C.3 Generative model architectures used for C-CHVAE and NAE

For the results in Lemma 3, we used linear encoders and decoders. For the remaining experiments, we use the following architectures.

	Adult	COMPAS	German Credit
Encoder layers	[input dim, 16, 32, 10]	[input dim, 8, 10, 5]	[input dim, 16, 32, 10]
Decoder layers	[10, 16, 32, input dim]	[5, 10, 8, input dim]	[10, 16, 32, input dim]
Type	Fully connected	Fully connected	Fully connected
Intermediate activations	ReLU	ReLU	ReLU
Loss function	MSE	MSE	MSE

Table 7: Autoencoder details

D ADDITIONAL EMPIRICAL EVALUATION

D.1 Remaining Empirical Results from Section 5

In Table 8, we show the remaining results on the German Credit data pertaining to the Spearman rank correlation experiments, while Figure 5 depicts the remaining d_{match} results for the German Credit data set on the logistic regression classifier.

	German Credit			
	LR		ANN	
Model	SCFE	CCHVAE	SCFE	CCHVAE
CW	$\mathbf{0.92 \pm 0.04}$	0.52 ± 0.08	$\mathbf{0.98 \pm 0.02}$	0.72 ± 0.13
DF	$\mathbf{0.92 \pm 0.04}$	0.57 ± 0.08	0.97 ± 0.02	0.72 ± 0.13
NAE	0.44 ± 0.11	$\mathbf{0.99 \pm 0.01}$	0.71 ± 0.19	$\mathbf{0.99 \pm 0.01}$

Table 8: Average Spearman rank correlation between counterfactual perturbations and adversarial perturbations. For every input $\mathbf{x}$, we compute the corresponding adversarial perturbation δ_{AE} and the counterfactual perturbation δ_{CE}. We then compute the rank correlation of δ_{AE} and δ_{CE} and report their means. The maximum rank correlation is obtained for methods that belong to the same categories (gradient based vs. manifold-based).

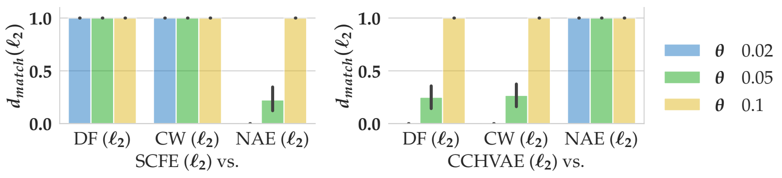
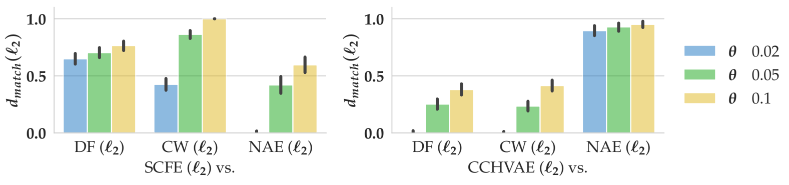


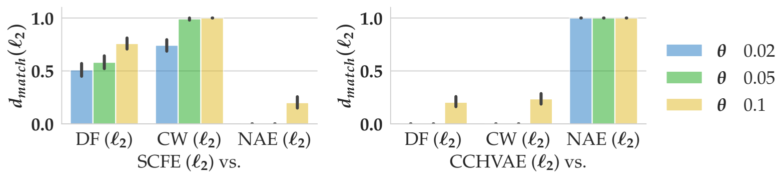
Figure 5: Analyzing to what extent different counterfactual explanation methods and adversarial example generation methods are empirically equivalent for the logistic regression classifier with German Credit data. We compute d_{match} from equation 12 with varying thresholds $\theta = \{0.02, 0.05, 0.1\}$. Missing bars indicate that there was no match.

We also include results for Neural Networks in Appendix D.2.

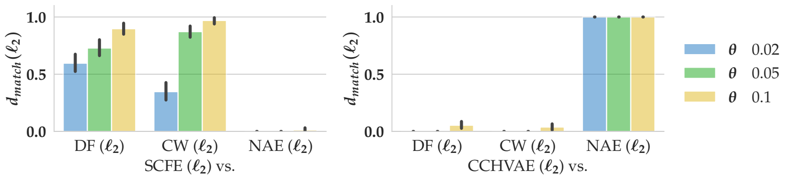
D.2 Empirical Evaluation with ANN



(a) Adult

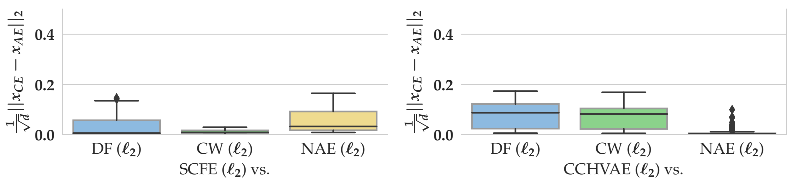


(b) COMPAS

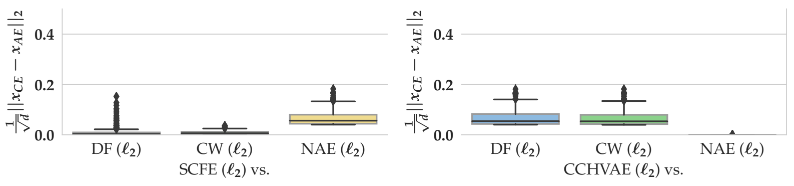


(c) German Credit

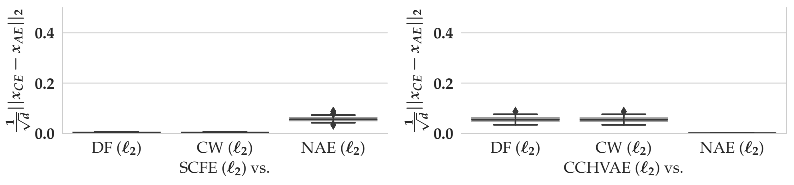
Figure 6: Analyzing to what extent different counterfactual explanations and adversarial examples are empirically equivalent for the 2-layer ANN classifier. To do that, we compute d_{match} from equation 12 with varying thresholds $\theta = \{0.02, 0.05, 0.1\}$. Missing bars indicate that there was no match.



(a) Adult

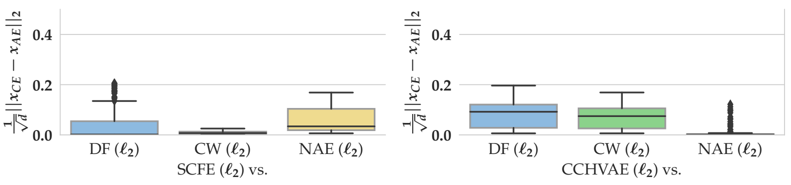


(b) COMPAS

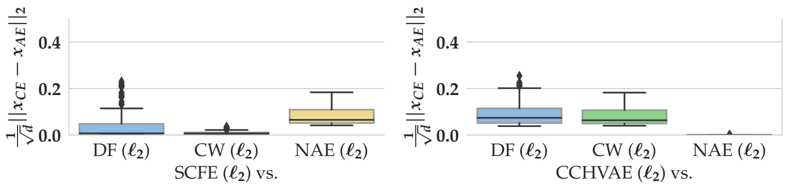


(c) German Credit

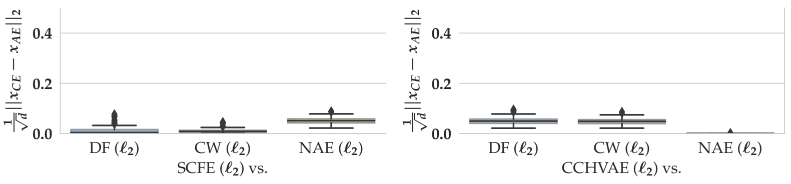
Figure 7: Distribution of instance wise norm comparisons for the logistic regression model. We show the distribution of cost comparisons across negatively predicted instances ($\hat{y} = 0$) for which we computed adversarial examples and counterfactual explanations.



(a) Adult



(b) COMPAS



(c) German Credit

Figure 8: Distribution of instance wise norm comparisons for the 2-layer ANN. We show the distribution of cost comparisons across negatively predicted instances ($\hat{y} = 0$) for which we computed adversarial examples and counterfactual explanations.

A.4 On the Trade-Off between Actionable Explanations and the Right to be Forgotten

Publication: Accepted at the 11th International Conference on Learning Representations (**ICLR**) 2023 (reviews)

Contribution: Based on discussions with Asia Biega, Jason Long and Emanuele Albini I developed the idea to study the tradeoffs between recourse and data deletion. I developed the robustness definitions, followed by obtaining first theoretical results for the linear models. Tobias Leemann extended these results for more general models. Jointly we extended the results to Neural Tangent Kernels. While I proposed the SGD based algorithm, Tobias Leemann suggested the greedy algorithm. Tobias Leemann and I jointly implemented these algorithms and we jointly conducted the experiments for the paper. Gjergji Kasneci made valuable suggestions by challenging and improving ideas. Gjergji Kasneci also suggested to study the complexity of the deletion approach. Asia Biega and Gjergji Kasneci helped sharpen the narrative and revise the manuscript.

ON THE TRADE-OFF BETWEEN ACTIONABLE EXPLANATIONS AND THE RIGHT TO BE FORGOTTEN

Martin Pawelczyk[1,*], **Tobias Leemann**[1], **Asia Biega**[2,†] **and Gjergji Kasneci**[1†]
[1]University of Tübingen, Germany
[2]Max-Planck-Institute for Security and Privacy, Germany

ABSTRACT

As machine learning (ML) models are increasingly being deployed in high-stakes applications, policymakers have suggested tighter data protection regulations (e.g., GDPR, CCPA). One key principle is the "right to be forgotten" which gives users the right to have their data deleted. Another key principle is the right to an actionable explanation, also known as algorithmic recourse, allowing users to reverse unfavorable decisions. To date, it is unknown whether these two principles can be operationalized simultaneously. Therefore, we introduce and study the problem of recourse invalidation in the context of data deletion requests. More specifically, we theoretically and empirically analyze the behavior of popular state-of-the-art algorithms and demonstrate that the recourses generated by these algorithms are likely to be invalidated if a small number of data deletion requests (e.g., 1 or 2) warrant updates of the predictive model. For the setting of differentiable models, we suggest a framework to identify a minimal subset of critical training points which, when removed, maximize the fraction of invalidated recourses. Using our framework, we empirically show that the removal of as little as 2 data instances from the training set can invalidate up to 95 percent of all recourses output by popular state-of-the-art algorithms. Thus, our work raises fundamental questions about the compatibility of "the right to an actionable explanation" in the context of the "right to be forgotten", while also providing constructive insights on the determining factors of recourse robustness.

1 INTRODUCTION

Machine learning (ML) models make a variety of consequential decisions in domains such as finance, healthcare, and policy. To protect users, laws such as the European Union's General Data Protection Regulation (GDPR) (GDPR, 2016) or the California Consumer Privacy Act (CCPA) (OAG, 2021) constrain the usage of personal data and ML model deployments. For example, individuals who have been adversely impacted by the predictions of these models have the right to *recourse* (Voigt & Von dem Bussche, 2017), i.e., a constructive instruction on how to act to arrive at a more desirable outcome (e.g., change a model prediction from "loan denied" to "approved"). Several approaches in recent literature tackled the problem of providing recourses by generating instance level counterfactual explanations (Wachter et al., 2018; Ustun et al., 2019; Karimi et al., 2020; Pawelczyk et al., 2020a).

Complementarily, data protection laws provide users with greater authority over their personal data. For instance, users are granted the right to *withdraw consent to the usage of their data* at any time (Biega & Finck, 2021). These regulations affect technology platforms that train their ML models on personal user data under the respective legal regime. Law scholars have argued that the continued use of ML models relying on deleted data instances could be deemed illegal (Villaronga et al., 2018).

Irrespective of the underlying mandate, data deletion has raised a number of algorithmic research questions. In particular, recent literature has focused on the efficiency of deletion (i.e., how to delete individual data points without retraining the model (Ginart et al., 2019; Golatkar et al., 2020a)) and model accuracy aspects of data deletion (i.e., how to remove data without compromising model

*Corresponding author: martin.pawelczyk@uni-tuebingen.de
†Equal senior author contribution.

accuracy (Biega et al., 2020; Goldsteen et al., 2021)). An aspect of data deletion which has not been examined before is *whether and how data deletion may impact model explanation frameworks*. Thus, there is a need to understand and systematically characterize the limitations of recourse algorithms when personal user data may need to be deleted from trained ML models. Indeed, deletion of certain data instances might invalidate actionable model explanations – both for the deleting user and, critically, unsuspecting other users. Such invalidations can be especially problematic in cases where users have already started to take costly actions to change their model outcomes based on previously received explanations.

In this paper, we formally examine the problem of algorithmic recourse in the context of data deletion requests. We consider the setting where a small set of individuals has decided to withdraw their data and, as a consequence of the deletion request, the model needs to be updated (Ginart et al., 2019). In particular, this work tackles the subsequent pressing question:

What is the worst impact that a deleted data instance can have on the recourse validity?

We approach this question by considering two distinct scenarios. The first setting considers to what extent the outdated recourses still lead to a desirable prediction (e.g., loan approval) on the updated model. For this scenario, we suggest a robustness measure called *recourse outcome instability* to quantify the fragility of recourse methods. Second, we consider the setting where the recourse action is being updated as a consequence of the prediction model update. In this case, we study what maximal change in recourse will be required to maintain the desirable prediction. To quantify the extent of this second problem, we suggest the notion of *recourse action instability*.

Given these robustness measures, we derive and analyze theoretical worst-case guarantees of the maximal instability induced for linear models and neural networks in the overparameterized regime, which we study through the lens of neural tangent kernels. We furthermore define an optimization problem for empirically quantifying recourse instability under data deletion. For a given trained ML model, we identify small sets of data points that maximize the proposed instability measures when deleted. Since the resulting brute-force approach (i.e., retraining models for every possible removal set) is NP-hard, we propose two relaxations for recourse instability maximization that can be optimized using (i) end-to-end gradient descent or (ii) via a greedy approximation algorithm. To summarize, in this work we make the following key contributions:

- **Novel recourse robustness problem.** We introduce the problem of *recourse invalidation under the right to be forgotten* by defining two new recourse instability measures.
- **Theoretical analysis.** Through rigorous theoretical analysis, we identify the factors that determine the instability of recourses when users whose data is part of the training set submit deletion requests.
- **Tractable algorithms.** Using our instability measures, we present an optimization framework to identify a small set of critical training data points which, when removed, invalidates most of the issued recourses.
- **Comprehensive experiments.** We conduct extensive experiments on multiple real-world data sets for both regression and classification tasks with our proposed algorithms, showing that the removal of even one point from the training set can invalidate up to 95 percent of all recourses output by state-of-the-art methods

Our results also have practical implications for system designers. First, our analysis and algorithms help identify parameters and model classes leading to higher stability when a trained ML model is subjected to deletion requests. Furthermore, our proposed methods can provide an informed way towards practical implementations of data minimization (Finck & Biega, 2021), as one could argue that data points contributing to recourse instability could be minimized out. Hence, our methods could increase designer's awareness and the compliance of their trained models.

2 Related Work

Algorithmic Approaches to Recourse. Several approaches in recent literature have been suggested to generate recourse for users who have been negatively impacted by model predictions (Tolomei et al., 2017; Laugel et al., 2017; Dhurandhar et al., 2018; Wachter et al., 2018; Ustun et al., 2019;

Van Looveren & Klaise, 2019; Pawelczyk et al., 2020a; Mahajan et al., 2019; Mothilal et al., 2020; Karimi et al., 2020; Rawal & Lakkaraju, 2020; Dandl et al., 2020; Antorán et al., 2021; Spooner et al., 2021; Albini et al., 2022). These approaches generate recourses assuming a static environment without data deletion requests, where both the model and the recourse remain stable.

A related line of work has focused on determining the extent to which recourses remain invariant to the model choice (Pawelczyk et al., 2020b; Black et al., 2021), to data distribution shifts (Rawal et al., 2021; Upadhyay et al., 2021), perturbations to the input instances (Artelt et al., 2021; Dominguez-Olmedo et al., 2022; Slack et al., 2021), or perturbations to the recourses (Pawelczyk et al., 2023).

Sample Deletion in Predictive Models. Since according to EU's GDPR individuals can request to have their data deleted, several approaches in recent literature have been focusing on updating a machine learning model without the need of retraining the entire model from scratch (Wu et al., 2020; Ginart et al., 2019; Izzo et al., 2021; Golatkar et al., 2020a;b; Cawley & Talbot, 2004). A related line of work considers the problem of data valuation (Ghorbani et al., 2020; Ghorbani & Zou, 2019). Finally, removing subsets of training data is an ingredient used for model debugging (Doshi-Velez & Kim, 2017) or the evaluation of explanation techniques (Hooker et al., 2019; Rong et al., 2022).

Contribution. While we do not suggest a new recourse algorithm, our work addresses the problem of recourse fragility in the presence of data deletion requests, which has previously not been studied. To expose this fragility, we suggest effective algorithms to delete a minimal subset of critical training points so that the fraction of invalidated recourses due to a required model update is maximized. Moreover, while prior research in the data deletion literature has primarily focused on effective data removal strategies for predictive models, there is no prior work that studies to what extent recourses output by state-of-the-art methods are affected by data deletion requests. Our work is the first to tackle these important problems and thereby paves the way for recourse providers to evaluate and rethink their recourse strategies in light of the right to be forgotten.

3 Preliminaries

The Predictive Model and the Data Deletion Mechanism. We consider prediction problems from some input space $\mathbb{R}^d$ to an output space $\mathcal{Y}$, where d is the number of input dimensions. We denote a sample by $\mathbf{z} = (\mathbf{x}, y)$, and denote the training data set by $\mathcal{D} = \{\mathbf{z}_1, \dots, \mathbf{z}_n\}$. Consider the weighted empirical risk minimization problem (ERM), which gives rise to the optimal model parameters:

$$\mathbf{w}_{\boldsymbol{\omega}} = \arg\min_{\mathbf{w}'} \sum_{i=1}^{n} \omega_i \cdot \ell\big(y_i, f_{\mathbf{w}'}(\mathbf{x}_i)\big), \tag{1}$$

where $\ell(\cdot,\cdot)$ is an instance-wise loss function (e.g., binary cross-entropy, mean-squared-error (MSE) loss, etc.) and $\boldsymbol{\omega} \in \{0,1\}^n$ are data weights that *are fixed at training time*. If $\omega_i = 1$, then the point $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ is part of the training data set, otherwise it is not. During model training, we set $\omega_i = 1 \; \forall i$, that is, the decision maker uses all available training instances at training time. In the optimization expressed in equation 1, the model parameters $\mathbf{w}$ are usually an implicit function of the data weight vector $\boldsymbol{\omega}$ and we write $\mathbf{w}_{\boldsymbol{\omega}}$ to highlight this fact; in particular, when all training instances are used we write $\mathbf{w}_{\mathbf{1}}$, where $\mathbf{1} \in \mathbb{R}^n$ is a vector of 1s. In summary, we have introduced the *weighted* ERM problem since it allows us to understand the impact of arbitrary data deletion patterns on actionable explanations as we allow users to withdraw their entire input $\mathbf{z}_i = (y_i, \mathbf{x}_i)$ from the training set used to train the model $f_{\mathbf{w}_{\mathbf{1}}}$. Next, we present the recourse model we consider.

The Recourse Problem in the Context of the Data Deletion Mechanism. We follow an established definition of counterfactual explanations originally proposed by Wachter et al. (2018). For a given model $f_{\mathbf{w}_{\boldsymbol{\omega}}} : \mathbb{R}^d \rightarrow \mathbb{R}$ parameterized by $\mathbf{w}$ and a distance function $d(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$, the problem of finding a recourse $\check{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$ for a factual instance $\mathbf{x}$ is given by:

$$\boldsymbol{\delta}_{\boldsymbol{\omega},\mathbf{x}} \in \arg\min_{\boldsymbol{\delta}' \in \mathcal{A}_d} (f_{\mathbf{w}_{\boldsymbol{\omega}}}(\mathbf{x} + \boldsymbol{\delta}') - s)^2 + \lambda \cdot d(\mathbf{x}, \mathbf{x} + \boldsymbol{\delta}'), \tag{2}$$

where $\lambda \geq 0$ is a scalar tradeoff parameter and s denotes the target score. In the optimization from equation 2, the optimal recourse action $\boldsymbol{\delta}$ usually depends on the model parameters and since the model parameters themselves depend on the exact data weights configuration we write $\boldsymbol{\delta}_{\boldsymbol{\omega},\mathbf{x}}$ to highlight this fact. The first term in the objective on the right-hand-side of equation 2 encourages the outcome $f_{\mathbf{w}_{\boldsymbol{\omega}}}(\check{\mathbf{x}})$ to become close to the user-defined target score s, while the second term encourages the distance between the factual instance $\mathbf{x}$ and the recourse $\check{\mathbf{x}}_{\omega} := \mathbf{x} + \boldsymbol{\delta}_{\omega,\mathbf{x}}$ to be low. The set of constraints $\mathcal{A}_d$ ensures that only admissible changes are made to the factual $\mathbf{x}$.

Recourse Robustness Through the Lens of the Right to be Forgotten. We first introduce several key terms, namely, *prescribed recourses* and *recourse outcomes*. A prescribed recourse $\check{\mathbf{x}}$ refers to a recourse that was provided to an end user by a recourse method (e.g., salary was increased by $500). The recourse outcome $f(\check{\mathbf{x}})$ is the model's prediction evaluated at the recourse. With these concepts in place, we develop two recourse instability definitions.

Definition 1. *(Recourse outcome instability) The recourse outcome instability with respect to a factual instance* $\mathbf{x}$*, where at least one data weight is set to* 0*, is defined as follows:*

$$\Delta_{\mathbf{x}}(\boldsymbol{\omega}) = \left| f_{\mathbf{w}_1}(\check{\mathbf{x}}_1) - f_{\mathbf{w}_{\omega}}(\check{\mathbf{x}}_1) \right|, \tag{3}$$

where $f_{\mathbf{w}_1}(\check{\mathbf{x}}_1)$ *is the prediction at the prescribed recourse* $\check{\mathbf{x}}_1$ *based on the model that uses the full training set (i.e.,* $f_{\mathbf{w}_1}$*) and* $f_{\mathbf{w}_{\omega}}(\check{\mathbf{x}}_1)$ *is the prediction at the prescribed recourse for an updated model and data deletion requests have been incorporated into the predictive model (i.e.,* $f_{\mathbf{w}_{\omega}}$*).*

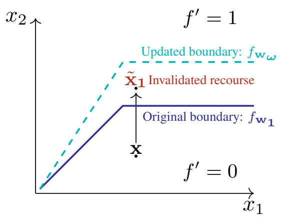
The above definition concisely describes the effect of applying "outdated" recourses to the updated model. We assume that only the model parameters are being updated while the prescribed recourses remain unchanged. For a discrete model with $\mathcal{Y} = \{0, 1\}$, Definition 1 captures whether the prescribed recourses will be invalid ($\Delta_{\mathbf{x}} = 1$) after deletion of training instances (see Fig. 1a). To obtain invalidation rates of recourses for a continuous-score model with target value s, we can also apply Definition 1 with a discretized $f'(\mathbf{x}) = \mathbb{I}\left[f(\mathbf{x}) > s\right]$, where $\mathbb{I}$ denotes the indicator function.

In Definition 2, consistent with related work (e.g., Wachter et al. (2018)), the distance function d is specified to be a p-norm and the recourse is allowed to change due to model parameter updates.

Definition 2. *(Recourse action instability) The Recourse action instability with respect to a factual input* $\mathbf{x}$*, where at least one data weight is set to* 0*, is defined as follows:*

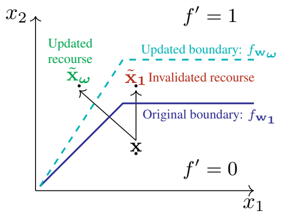
$$\Phi^{(p)}_{\mathbf{x}}(\boldsymbol{\omega}) = \left\| \check{\mathbf{x}}_1 - \check{\mathbf{x}}_{\omega} \right\|_p, \tag{4}$$

where $p \in [1, \infty)$*, and* $\check{\mathbf{x}}_{\omega}$ *is the recourse obtained for the model trained on the data instances that remain present in the data set after the deletion request.*



x_2
$f' = 1$
Updated boundary: $f_{\mathbf{w}_{\omega}}$
$\check{\mathbf{x}}_1$ Invalidated recourse
Original boundary: $f_{\mathbf{w}_1}$
$\mathbf{x}$
$f' = 0$
x_1

(a) Recourse Outcome Instability



x_2
$f' = 1$
Updated recourse
$\check{\mathbf{x}}_{\omega}$
Updated boundary: $f_{\mathbf{w}_{\omega}}$
$\check{\mathbf{x}}_1$ Invalidated recourse
Original boundary: $f_{\mathbf{w}_1}$
$\mathbf{x}$
$f' = 0$
x_1

(b) Recourse Action Instability

Figure 1: Visualizing the two key robustness notions. In Fig. 1a, recourse $\check{\mathbf{x}}_1$ for an input $\mathbf{x}$ is invalidated due to a model update. In Fig. 1b, recourse is additionally recomputed (i.e., $\check{\mathbf{x}}_{\omega}$) to avoid recourse invalidation.

Definition 2 quantifies the extent to which the prescribed recourses would have to additionally change to still achieve the desired recourse outcome *after data deletion* requests (i.e., $\check{\mathbf{x}}_{\omega}$, see Fig. 1b). Note that we are interested in how the optimal low cost recourse changes even if the outdated recourse would remain valid. Using our invalidation measures defined above, in the next section, we formally study the trade-offs between actionable explanations and the right to be forgotten. To do so, we provide data dependent upper bounds on the invalidation measures from Definitions 1 and 2, which practitioners can use to probe the worst-case vulnerability of their algorithmic recourse to data deletion requests.

4 Trade-offs Betw. Alg. Recourse and the Right to be Forgotten

Here we relate the two notions of recourse instability presented in Definitions 1 and 2 to the vulnerability of the underlying predictive model with respect to data deletion requests. We show that the introduced instability measures are directly related to data points with a high influence on the parameters after deletion.

Analyzing the Instability Measures. With the basic terminology in place, we provide upper bounds for the recourse instability notions defined in the previous section when the underlying models are linear or overparameterized neural networks. Throughout our analysis, we the term $\mathbf{d}_i := \mathbf{w_1} - \mathbf{w}_{-i}$ has an important impact. It measures the difference from the original parameters $\mathbf{w_1}$ to the parameter vector $\mathbf{w}_{-i}$ obtained after deleting the i-th instance $(\mathbf{x}_i, y_i)$ from the model. In the statistics literature, this term is also known as the empirical influence function (Cook & Weisberg, 1980). Below we provide an upper bound for recourse outcome instability in linear models.

Proposition 1 (Upper bound on recourse outcome instability for linear models)**.** *For the linear regression model $f(\mathbf{x}) = \mathbf{w}_L^\top \mathbf{x}$ with model parameters $\mathbf{w}_L = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{Y}$, an upper bound for the recourse invalidation from Definition 1 by removing an instance from the training set is given by:*

$$\Delta_{\mathbf{x}} \leq \|\check{\mathbf{x}}_\mathbf{1}\|_2 \cdot \max_{i \in [n]} \|\mathbf{d}_i^L\|_2, \tag{5}$$

where $\mathbf{d}_i^L := \mathbf{w}_L - \mathbf{w}_{L,-i} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i \cdot \frac{r_i}{1-h_{ii}}$, $r_i = y_i - \mathbf{w}_L^\top \mathbf{x}_i$ and $h_{ii} = \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{x}_i$.

The term $(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{x}_i = \frac{d\mathbf{w}_\mathrm{L}}{dy_i}$ describes how sensitive the model parameters are to y_i, while the residual r_i captures how well y_i can be fit by the model. On the contrary, the term h_{ii} from the denominator is known as the *leverage* and describes how atypical $\mathbf{x}_i$ is with respect to all training inputs $\mathbf{X}$. In summary, data instances that have influential labels or are atypical will have the highest impact when deleted. Next, we provide a generic upper bound on recourse action instability.

Proposition 2 (Upper bound on recourse action instability)**.** *For any predictive model with scoring function $f : \mathbb{R}^d \to \mathbb{R}$, an upper bound for the recourse instability from Definition 2 by removing an instance $\mathbf{z}_i = (\mathbf{x}, y)$ from the training set is given by:*

$$\Phi_{\mathbf{x}}^{(2)} \leq \|\mathbf{d}_i\|_2 \int_0^1 \left\| \frac{\mathbf{D}\boldsymbol{\delta}}{\mathbf{D}\mathbf{w}} (\tilde{\mathbf{w}}) \right\|_2 d\gamma, \tag{6}$$

where $\frac{\mathbf{D}\boldsymbol{\delta}}{\mathbf{D}\mathbf{w}}$ denotes the Jacobian of optimal recourse with the corresponding operator matrix norm, $\tilde{\mathbf{w}} := \gamma \mathbf{w} + (1-\gamma)\,\mathbf{w}_{-i}$ with $\mathbf{w}_{-i}$ being the optimal model parameters with the i-th training instance removed from the training set, and $\mathbf{d}_i = \mathbf{w} - \mathbf{w}_{-i}$.

The norm of the Jacobian of optimal recourse indicates the local sensitivity of optimal recourse with respect to changes in model parameters $\mathbf{w}$. High magnitudes indicate that a small change in the parameters may require a fundamentally different recourse action. The total change can be bounded by the integral over these local sensitivities, which means that low local sensitivities along the path will result in a low overall change. Next, we specialize this result to the case of linear models.

Corollary 1 (Upper bound on recourse action instability for linear models)**.** *For the linear model $f(\mathbf{x}) = \mathbf{w}_L^\top \mathbf{x}$ with model parameters $\mathbf{w}_L = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$, an upper bound for the recourse action instability when $s = 0, \lambda \to 0$ by removing an instance from the training set is given by:*

$$\Phi_{\mathbf{x}}^{(2)} \leq \left(\max_{i \in [n]} \|\mathbf{d}_i^L\|_2 \right) \frac{4\sqrt{2}\|\mathbf{x}\|_2}{\min(\|\mathbf{w}_L\|_2, \min_{i\in[n]} \|\mathbf{w}_{L,-i}\|_2)}, \tag{7}$$

under the condition that $\mathbf{w}_L^\top \mathbf{w}_{L,-i} \geq 0$ (no diametrical weight changes), where $\mathbf{w}_{L,-i} = \mathbf{w}_L - \mathbf{d}_i^L$ is the weight after removal of training instance i and $\mathbf{d}_i^L = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i \frac{(y_i - \mathbf{w}_L^\top \mathbf{x}_i)}{1-h_{ii}}$.

For models trained on large data sets, the absolute value of the model parameters' norm $\|\mathbf{w}_L\|$ will not change much under deletion of a single instance. Therefore we argue that the denominator $\min(\|\mathbf{w}_\mathrm{L}\|_2, \min_{i\in[n]}\|\mathbf{w}_{\mathrm{L},-i}\|_2) \approx \|\mathbf{w}_\mathrm{L}\|$. Thus, recourse action instability is mainly determined by the sensitivity of model parameters to deletion, $\max_{i\in[n]} \|\mathbf{d}_i^\mathrm{L}\|_2$, scaled by the ratio of $\frac{\|\mathbf{x}\|_2}{\|\mathbf{w}_\mathrm{L}\|_2}$.

Neural Tangent Kernels. Studying the relation between deletion requests and the robustness of algorithmic recourse for models as complex as neural networks requires recent results from computational learning theory. In particular, we also rely on insights on the behaviour of over-parameterized neural networks from the theory of Neural Tangent Kernels (NTKs), which we will now briefly introduce. Thus we study our robustness notions for neural network models in the overparameterized regime with ReLU activation functions that take the following form:

$$f_{\text{ANN}}(\mathbf{x}) = \frac{1}{\sqrt{k}} \sum_{j=1}^{k} a_j \cdot \text{relu}(\mathbf{w}_j^\top \mathbf{x}), \tag{8}$$

where $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_k] \in \mathbb{R}^{d \times k}$ and $\mathbf{a} = [a_1, \dots, a_k] \in \mathbb{R}^k$. To concretely study the impact of data deletion on recourses in non-linear models such as neural networks, we leverage ideas from the neural tangent kernel (NTK) literature (Jacot et al., 2018; Lee et al., 2019; Arora et al., 2019; Du et al., 2019). The key insight from this literature for the purpose of our work is that infinitely wide neural networks can be expressed as a kernel ridge regression problem with the NTK under appropriate parameter initialization, and gradient descent training dynamics. In particular, in the limit as the number of hidden nodes $k \to \infty$, the neural tangent kernel associated with a two-layer ReLU network has a closed-form expression (Chen & Xu, 2021; Zhang & Zhang, 2021) (see Appendix A.5):

$$K^{\infty}(\mathbf{x}_0, \mathbf{x}) = \frac{\mathbf{x}_0^\top \mathbf{x} \left(\pi - \text{arcos}\left(\frac{\mathbf{x}_0^\top \mathbf{x}}{\|\mathbf{x}_0\|\|\mathbf{x}\|}\right)\right)}{2\pi}. \tag{9}$$

Thus, the network's prediction at an input $\mathbf{x}$ can be described by:

$$f_{\text{NTK}}(\mathbf{x}) = \left(\mathbf{K}^{\infty}(\mathbf{x}, \mathbf{X})\right)^\top \mathbf{w}_{\text{NTK}}, \tag{10}$$

where $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the input data matrix, $\mathbf{K}^{\infty}(\mathbf{X}, \mathbf{X}) \in \mathbb{R}^{n \times n}$ is the NTK matrix evaluated on the training data points: $[\mathbf{K}^{\infty}(\mathbf{X}, \mathbf{X})]_{ij} = K^{\infty}(\mathbf{x}_i, \mathbf{x}_j)$ and $\mathbf{w}_{\text{NTK}} = \left(\mathbf{K}^{\infty}(\mathbf{X}, \mathbf{X}) + \beta \mathbf{I}_n\right)^{-1} \mathbf{Y}$ solves the ℓ_2 regularized minimization problem with MSE loss where $\mathbf{Y} \in \mathbb{R}^n$ are the prediction targets. With this appropriate terminology in place we provide an upper bound on recourse outcome instability of wide neural network models.

Proposition 3 (Upper bound on recourse outcome instability for wide neural networks)**.** *For the NTK model with $\mathbf{w}_{NTK} = \left(\mathbf{K}^{\infty}(\mathbf{X}, \mathbf{X}) + \beta \mathbf{I}_n\right)^{-1} \mathbf{Y}$, an upper bound for the recourse invalidation from Definition 1 by removing an instance $(\mathbf{x}, y)$ from the training set is given by:*

$$\Delta_{\mathbf{x}} \leq \|\mathbf{K}^{\infty}(\check{\mathbf{x}}_1, \mathbf{X})\|_2 \cdot \max_{i \in [n]} \|\mathbf{d}_i^{NTK}\|_2, \tag{11}$$

where $\mathbf{d}_i^{NTK} = \frac{1}{k_{ii}} \mathbf{k}_i \mathbf{k}_i^\top \mathbf{Y}$, where $\mathbf{k}_i$ is the i-th column of the matrix $\left(\mathbf{K}^{\infty}(\mathbf{X}, \mathbf{X}) + \beta \mathbf{I}_n\right)^{-1}$, and k_{ii} is its i-th diagonal element.

Intuitively, $\mathbf{d}_i^{\text{NTK}}$ is the linear model analog to $\mathbf{d}_i^{\text{L}}$ and $\mathbf{d}_i^{\text{NTK}}$ represents the importance that the point $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ has on the model parameters $\mathbf{w}_{\text{NTK}}$.

In practical use-cases, when trying to comply with both the right to data deletion and the right to actionable explanations, our results have practical implications. For example, instances with high influence captured by $\mathbf{d}_i$ should be encountered with caution during model training in order to provide reliable recourses to the individuals who seek recourse. In summary, our results suggest that the right to data deletion may be fundamentally at odds with reliable state-of-the-art actionable explanations as the removal of an influential data instance can induce a large change in the recourse robustness, the extent to which is primarily measured by the empirical influence function $\mathbf{d}_i$.

5 Finding the Set of Most Critical Data Points

The Objective Function. In this section, we present optimization procedures that can be readily used to assess recourses' vulnerability to deletion requests. On this way, we start by formulating our optimization objective. We denote by $m \in \{\Delta, \Phi^{(2)}\}$ the measure we want to optimize for. We consider the summed instability of over the data set by omitting the subscript $\mathbf{x}$, e.g., $\Delta = \sum_{\mathbf{x} \in \mathcal{D}_{test}} \Delta_{\mathbf{x}}$. Our goal is to find the smallest number of deletion requests that leads to a maximum impact on the instability measure m. To formalize this, define the set of data weight configurations:

$$\Gamma_\alpha := \{\boldsymbol{\omega} : \text{Maximally } \lfloor \alpha \cdot n \rfloor \text{ entries of } \boldsymbol{\omega} \text{ are } 0 \text{ and the remainder is } 1.\}. \tag{12}$$

In equation 12, the parameter α controls the fraction of instances that are being removed from the training set. For a fixed fraction α, our problem of interest becomes:

$$\boldsymbol{\omega}^* = \arg\max_{\boldsymbol{\omega} \in \Gamma_\alpha} m(\boldsymbol{\omega}). \tag{13}$$

Fundamental Problems. When optimizing the above objective we face two fundamental problems: (i) *evaluating* $m(\boldsymbol{\omega})$ for many weight configurations $\boldsymbol{\omega}$ can be prohibitively expensive as the objective is defined implicitly through solutions of several non-linear optimization problems (i.e., model fitting and finding recourses). Further, (ii) even for an objective $m(\boldsymbol{\omega})$ which can be computed in constant or polynomial time *optimizing* this objective can still be NP-hard (a proof is given in Appendix A.3).

Practical Algorithms. We devise two practical algorithms which approach the problem in equation 13 in different ways. As for the problem of computing $m(\boldsymbol{\omega})$ in (i), we can either solve this by (a) using a closed-form expression indicating the dependency of m on $\boldsymbol{\omega}$ or (b) by using an approximation of m that is differentiable with respect to $\boldsymbol{\omega}$. As for the optimization in (ii), once we have established the dependency of m on $\boldsymbol{\omega}$ we can either (a) use a gradient descent approach or (b) we use a greedy method. Below we explain the individual steps required for our algorithms.

5.1 Computing the Objective

In the objective $m(\boldsymbol{\omega})$, notice the dependencies $\Delta_{\mathbf{x}}(\boldsymbol{\omega}) = \Delta_{\mathbf{x}}\left(f(\mathbf{w}(\boldsymbol{\omega}), \check{\mathbf{x}})\right)$ for the recourse outcome instability, and $\Phi^{(2)}_{\mathbf{x}}(\boldsymbol{\omega}) = \Phi^{(2)}_{\mathbf{x}}\left(\boldsymbol{\delta}(\mathbf{w}(\boldsymbol{\omega}), \mathbf{x}))\right)$ for the recourse action instability. In the following, we briefly discuss how we efficiently compute each of these functions without numerical optimization.

Model parameters from data weights $\mathbf{w}(\boldsymbol{\omega})$. For the linear model, an analytical solution can be obtained, $\mathbf{w}_{\mathrm{L}}(\boldsymbol{\omega}) = \left(\mathbf{X}^\top \boldsymbol{\Omega} \mathbf{X}\right)^{-1} \mathbf{X}^\top \boldsymbol{\Omega} \mathbf{Y}$, where $\boldsymbol{\Omega} = \mathrm{diag}(\boldsymbol{\omega})$. The same goes for the NTK model where $\mathbf{w}_{\mathrm{NTK}}(\boldsymbol{\omega}) = \boldsymbol{\Omega}^{\frac{1}{2}}\left(\boldsymbol{\Omega}^{\frac{1}{2}} \mathbf{K}^\infty(\mathbf{X}, \mathbf{X}) \boldsymbol{\Omega}^{\frac{1}{2}} + \beta \mathbf{I}\right)^{-1} \boldsymbol{\Omega}^{\frac{1}{2}} \mathbf{Y}$ (Busuttil & Kalnishkan, 2007, Eqn. 3). When no closed-form expressions for the model parameters exist, we can resort to the infinitesimal jackknife (IJ) (Jaeckel, 1972; Efron, 1982; Giordano et al., 2019b;a), that can be seen as a linear approximation to this implicit function. We refer to Appendix C for additional details on this matter.

Model prediction from model parameters $f(\mathbf{w}, \check{\mathbf{x}})$. Having established the model parameters, evaluating the prediction at a given point can be quickly done even in a differentiable manner with respect to $\mathbf{w}$ for the models we consider in this work.

Recourse action from model parameters $\boldsymbol{\delta}(\mathbf{w}, \check{\mathbf{x}})$. Estimating the recourse action is more challenging as it requires solving equation 2. However, a differentiable solution exists for linear models, where the optimal recourse action is given by $\boldsymbol{\delta}_{\mathrm{L}} = \frac{s - \mathbf{w}_{\mathrm{L}}(\boldsymbol{\omega})^\top \mathbf{x}}{\lambda + \|\mathbf{w}_{\mathrm{L}}(\boldsymbol{\omega})\|_2^2} \mathbf{w}_{\mathrm{L}}(\boldsymbol{\omega})$. When the underlying predictor is a wide neural network we can approximate the recourse expression of the corresponding NTK, $\boldsymbol{\delta}_{\mathrm{NTK}} \approx \frac{s - f_{\boldsymbol{\omega},\mathrm{NTK}}(\mathbf{x})}{\lambda + \|\bar{\mathbf{w}}_{\mathrm{NTK}}(\bar{\boldsymbol{\omega}})\|_2^2} \bar{\mathbf{w}}_{\mathrm{NTK}}(\boldsymbol{\omega})$, which stems from the first-order taylor expansion $f_{\boldsymbol{\omega},\mathrm{NTK}}(\mathbf{x} + \boldsymbol{\delta}) \approx f_{\boldsymbol{\omega},\mathrm{NTK}}(\mathbf{x}) + \boldsymbol{\delta}^\top \bar{\mathbf{w}}_{\mathrm{NTK}}(\boldsymbol{\omega})$ with $\bar{\mathbf{w}}_{\mathrm{NTK}}(\boldsymbol{\omega}) = \nabla_{\mathbf{x}} K(\mathbf{x}, \mathbf{X}) \mathbf{w}_{\mathrm{NTK}}(\boldsymbol{\omega})$.

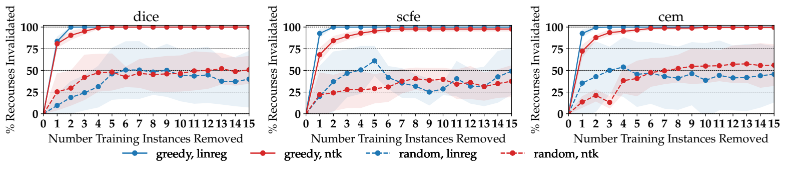
5.2 Optimizing the Objective Function

The Greedy Algorithm. We consider the model on the full data set and compute the objective function $m(\boldsymbol{\omega})$ under deletion of every instance (alone). We then select the instance that leads to the highest increase in the objective. We add this instance to the set of deleted points. Subsequently, we refit the model and compute the impact of deletion for every second instance, when deleted in combination with the first one. Again, we add the instance that results in the largest increase to the set. Iteratively repeating these steps, we identify more instances to be deleted. Computational complexity depends on the implementation of the model weight recomputation, which is required $\mathcal{O}(\alpha n^2)$ times.

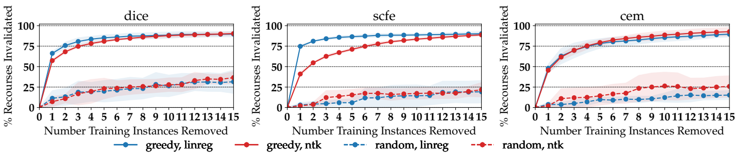
The Gradient Descent Algorithm. Because our developed computation of $m(\boldsymbol{\omega})$ can be made differentiable, we also propose a *gradient-based optimization* framework. We consider the relaxation of the problem in equation 13,

$$\boldsymbol{\omega}^* = \underset{\boldsymbol{\omega} \in \{0,1\}^n}{\arg\max}\ m(\boldsymbol{\omega}) - \|\mathbf{1} - \boldsymbol{\omega}\|_0, \tag{14}$$

where the ℓ_0 norm encourages to change as few data weights from 1 to 0 as possible while few removals of training instances should have maximum impact on the robustness measure. The problem in equation 14 can be further relaxed to a continuous and unconstrained optimization problem. To do so we use a recently suggested stochastic surrogate loss for the ℓ_0 term (Yamada et al., 2020). Using this technique, a surrogate loss for equation 14 can be optimized using stochastic gradient descent (SGD). We refer to Appendix C for more details and pseudo-code of the two algorithms.



(a) Admission



(b) Heloc

Figure 2: Measuring the tradeoff between recourse outcome instability and the number of deletion requests for both the Admission and the Heloc data sets for regression and NTK models and various recourse methods. Results were obtained by greedy optimization; see Appendix B for SGD results.

6 Experimental Evaluation

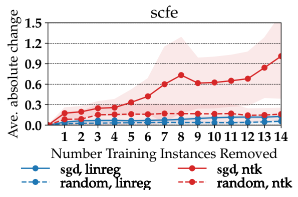
We experimentally evaluate our framework in terms of its ability to find significant recourse invalidations using the instability measures presented in Section 3.

Data Sets. For our experiments on regression tasks we use two real-world data sets. In addition, we provide results for two classification datasets in the Appendix B. First, we use law school data from the Law School Admission Council (*Admission*). The council carried out surveys across 163 law schools in the US, in which they collected information from 21,790 law students across the US (Wightman, 1998). The data contains information on the students' prior performances. The task is to predict the students' first-year law-school average grades. Second, we use the *Home Equity Line of Credit (Heloc)* data set. Here, the target variable is a score indicating whether individuals will repay the Heloc account within a fixed time window. Across both tasks we consider individuals in need of recourse if their scores lie below the median score across the data set.

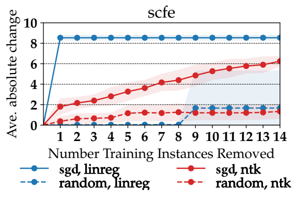
Recourse Methods. We apply our techniques to four different methods which aim to generate low-cost recourses using different principles: `SCFE` was suggested by Wachter et al. (Wachter et al., 2018) and uses a gradient-based objective to find recourses, `DICE` (Mothilal et al., 2020) uses a gradient-based objective to find recourses subject to a diversity constraint, and `CEM` (Dhurandhar et al., 2018) uses a generative model to encourage recourses to lie on the data manifold. For all methods, we used the recourse method implementations from the `CARLA` library (Pawelczyk et al., 2021) and specify the ℓ_1 cost constraint. Further details on these algorithms are provided in App. C.

Evaluation Measures. For the purpose of our evaluation, we use both the recourse outcome instability measure and the recourse action instability measure presented in Definitions 1 and 2. We evaluate the efficacy of our framework to destabilize a large fraction of recourses using a small number of deletion requests (up to 14). To find critical instances, we use the greedy and the gradient-based algorithms described in Sec. 5. After having established a set of critical points, we recompute the metrics with the refitted models and recourses to obtain a ground truth result.

For the *recourse outcome instability*, our metric Δ counts the number of invalidated recourses. We use the median as the target score s, i.e., if the recourse outcome flips back from a positive leaning prediction (above median) to a negative one (below median) it is considered invalidated. When evaluating *recourse action instability*, we identify a set of critical points, delete these points from the train set and refit the predictive model. In this case, we also have to recompute the recourses to evaluate Φ_p. We then measure the recourse instability using Definition 2 with $p = 2$. Additionally, we compare with a random baseline, which deletes points uniformly at random from the train set. We compute these measures for all individuals from the test set who require algorithmic recourse. To obtain standard errors, we split the test set into 5 folds and report averaged results over these 5 folds.



(a) Admission



(b) Heloc

Figure 3: Quantifying the tradeoff between recourse action instability as measured in Definition 2 and the number of deletion requests for both the Admission and the Heloc data sets for the `SCFE` method when the underlying model is linear or an NTK (results by SGD optimization).

Results. In Figure 2, we measure the tradeoff between *recourse outcome instability* and the number of deletion requests. We plot the number of deletion requests against the fraction of all recourses that become invalidated when up to $k \in \{1, \dots, 14\}$ training points are removed from the training set of the predictive model. When the underlying model is linear, we observe that the removal of as few as 5 training points induces invalidation rates of all recourses that are as high as 95 $\%$ percent – we observe a similar trend across all recourse methods. Note that a similar trend is present for the NTK model; however, a larger number of deletion requests (roughly 9) is required to achieve similar invalidation rates. Finally, also note that our approach is always much more effective at deleting instances than the random baseline. In Figure 3, we measure the tradeoff between *recourse action instability* and the number of deletion requests with respect to the `SCFE` recourse method when the underlying predictive model is linear or an NTK model. For this complex objective, we use the more efficient SGD optimization. Again, we observe that our optimization method significantly outperforms the random baselines at finding the most influential points to be removed.

Additional Models and Tasks. In addition to the here presented results, we provide results for classification tasks with (a) Logistic Regression, (b) Kernel-SVM and (c) ANN models on two additional data sets (Diabetes and COMPAS) in Appendix B. Across all these models, we observe that our removal algorithms outperform random guessing; often by up to 75 percentage points.

Factors of Recourse Robustness. Our empirical results shed light on which factors are influential in determining robustness of trained ML models with respect to deletion requests. In combination with results from Fig. 4 (see Appendix B), our results suggest that linear models are more susceptible to invalidation in the worst-case but are slightly more robust when it comes to random removals. Furthermore, the characteristics of the data set play a key role; in particular those of the critical points. We perform an additional experiment where we consider modified data sets without the most influential points identified by our optimization approaches. In Appendix B, initial results show that this simple technique decreases the invalidation probabilities by up to 6 percentage points.

7 Discussion and Conclusion

In this work, we made the first step towards understanding the tradeoffs between actionable model explanations and the right to be forgotten. We theoretically analyzed the robustness of state-of-the-art recourse methods under data deletion requests and suggested (i) a greedy and (ii) a gradient-based algorithm to efficiently identify a small subset of individuals, whose data, when removed, would lead to invalidation of a large number of recourses for unsuspecting other users. Our experimental evaluation with multiple real-world data sets on both regression and classification tasks demonstrates that the right to be forgotten presents a significant challenge to the reliability of actionable explanations.

Finally, our theoretical results suggest that the robustness to deletion increases when the model parameter changes under data deletion remain small. This formulation closely resembles the definition of *Differential Privacy* (DP) (Dwork et al., 2014). We therefore conjecture that the reliability of actionable recourse could benefit from models that have been trained under DP constraints. As the field of AI rapidly evolves, data protection authorities will further refine the precise interpretations of general principles in regulations such as GDPR. The present paper contributes towards this goal theoretically, algorithmically, and empirically by providing evidence of tensions between different data protection principles.

Ethics statement. Our findings raise compelling questions on the deployment of counterfactual explanations in practice. First of all, *Are the two requirements of actionable explanations and the right to be forgotten fundamentally at odds with one another?* The theoretical and empirical results in this work indicate that for many model and recourse method pairs, this might indeed be the case. This finding leads to the pressing follow-up question: *How can practitioners make sure that their recourses stay valid under deletion requests?* A first take might be to implement the principle of data minimization (Biega et al., 2020; Biega & Finck, 2021; Shanmugam et al., 2022) in the first place, i.e., exclude the k most critical data points from model training.

Acknowledgments. We would like to thank the anonymous reviewers for their insightful feedback. MP would like to thank Jason Long, Emanuele Albini, Jiāháo Chén, Daniel Dervovic and Daniele Magazzeni for insightful discussions at early stages of this work.

REFERENCES

Emanuele Albini, Jason Long, Danial Dervovic, and Daniele Magazzeni. Counterfactual shapley additive explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAccT)*, 2022.

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks, 2016.

Javier Antorán, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. Getting a clue: A method for explaining uncertainty estimates. In *International Conference on Learning Representations (ICLR)*, 2021.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

André Artelt, Valerie Vaquet, Riza Velioglu, Fabian Hinder, Johannes Brinkrolf, Malte Schilling, and Barbara Hammer. Evaluating robustness of counterfactual explanations. *arXiv:2103.02354*, 2021.

Asia J. Biega and Michèle Finck. Reviving purpose limitation and data minimisation in data-driven systems. *Technology and Regulation*, 2021.

Asia J. Biega, Peter Potash, Hal Daumé, Fernando Diaz, and Michèle Finck. Operationalizing the legal principle of data minimization for personalization. In *ACM(43) SIGIR '20*, pp. 399–408, 2020.

Emily Black, Zifan Wang, Matt Fredrikson, and Anupam Datta. Consistent counterfactuals for deep models. *arXiv:2110.03109*, 2021.

Steven Busuttil and Yuri Kalnishkan. Weighted kernel regression for predicting changing dependencies. In *European Conference on Machine Learning*, pp. 535–542. Springer, 2007.

Gavin C Cawley and Nicola LC Talbot. Fast exact leave-one-out cross-validation of sparse least-squares support vector machines. *Neural networks*, 17(10):1467–1475, 2004.

Lin Chen and Sheng Xu. Deep neural tangent kernel and laplace kernel have the same {rkhs}. In *International Conference on Learning Representations (ICLR)*, 2021.

Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. *Advances in neural information processing systems (NeurIPS)*, 22, 2009.

R Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

Lehel Csató and Manfred Opper. Sparse on-line gaussian processes. *Neural computation*, 14(3): 641–668, 2002.

Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*, pp. 448–469. Springer, 2020.

Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Ricardo Dominguez-Olmedo, Amir-Hossein Karimi, and Bernhard Schölkopf. On the adversarial robustness of causal algorithmic recourse. In *International Conference on Machine Learning (ICML)*, 2022.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. 2017.

Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407, 2014.

Bradley Efron. *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.

Michele Finck and Asia J Biega. Reviving purpose limitation and data minimisation in data-driven systems. *Technology and Regulation*, 2021:44–61, 2021.

GDPR. Regulation (eu) 2016/679 of the european parliament and of the council. *Official Journal of the European Union*, 2016.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pp. 2242–2251. PMLR, 2019.

Amirata Ghorbani, Michael Kim, and James Zou. A distributional framework for data valuation. In *International Conference on Machine Learning (ICML)*, pp. 3535–3544. PMLR, 2020.

Antonio Ginart, Melody Y. Guan, Gregory Valiant, and James Zou. Making ai forget you: Data deletion in machine learning. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada*, 2019.

Ryan Giordano, Michael I. Jordan, and Tamara Broderick. A higher-order swiss army infinitesimal jackknife. *ArXiv*, abs/1907.12116, 2019a.

Ryan Giordano, William Stephenson, Runjing Liu, Michael Jordan, and Tamara Broderick. A swiss army infinitesimal jackknife. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019b.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020a.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Forgetting outside the box: Scrubbing deep networks of information accessible from input-output observations. *arXiv:2003.02960*, 2020b.

Abigail Goldsteen, Gilad Ezov, Ron Shmelkin, Micha Moffie, and Ariel Farkash. Data minimization for gdpr compliance in machine learning models. *AI and Ethics*, pp. 1–15, 2021.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In Arindam Banerjee and Kenji Fukumizu (eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 130. PMLR, 2021.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems (NeurIPS)*, 31, 2018.

L Jaeckel. The infinitesimal jackknife. memorandum. Technical report, MM 72-1215-11, Bell Lab. Murray Hill, NJ, 1972.

Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pp. 85–103. Springer, 1972.

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Inverse classification for comparison-based interpretability in machine learning. *arXiv preprint arXiv:1712.08443*, 2017.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems (NeurIPS)*, 32, 2019.

Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*, 2019.

Rupert G Miller Jr. An unbalanced jackknife. *The Annals of Statistics*, pp. 880–891, 1974.

Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2020.

CA OAG. Ccpa regulations: Final regulation text. *Office of the Attorney General, California Department of Justice*, 2021.

Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference 2020 (WWW)*. ACM, 2020a.

Martin Pawelczyk, Klaus Broelemann, and Gjergji. Kasneci. On counterfactual explanations under predictive multiplicity. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 809–818. PMLR, 2020b.

Martin Pawelczyk, Sascha Bielawski, Johan Van den Heuvel, Tobias Richter, and Gjergji Kasneci. Carla: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms. In *Advances in Neural Information Processing Systems (NeurIPS) (Benchmark and Datasets Track)*, 2021.

Martin Pawelczyk, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. Exploring counterfactual explanations through the lens ofadversarial examples: A theoretical and empirical analysis. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

Martin Pawelczyk, Teresa Datta, Johannes van-den Heuvel, Gjergji Kasneci, and Himabindu Lakkaraju. Algorithmic recourse in the face of noisy human responses. In *International Conference on Learning Representations (ICLR)*, 2023.

Kaivalya Rawal and Himabindu Lakkaraju. Interpretable and interactive summaries ofactionable recourses. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Kaivalya Rawal, Ece Kamar, and Himabindu Lakkaraju. Algorithmic recourse in the wild: Understanding the impact of data and model shifts. *arXiv:2012.11788*, 2021.

Yao Rong, Tobias Leemann, Vadim Borisov, Gjergji Kasneci, and Enkelejda Kasneci. A consistent and efficient evaluation strategy for attribution methods. In *International Conference on Machine Learning (ICML)*, 2022.

Divya Shanmugam, Fernando Diaz, Samira Shabanian, Michèle Finck, and Asia J. Biega. Learning to limit data collection via scaling laws: A computational interpretation for the legal principle of data minimization. In *ACM FAccT '22*, 2022.

Dylan Slack, Sophie Hilgard, Himabindu Lakkaraju, and Sameer Singh. Counterfactual explanations can be manipulated. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Thomas Spooner, Danial Dervovic, Jason Long, Jon Shepard, Jiahao Chen, and Daniele Magazzeni. Counterfactual explanations for arbitrary regression models. *arXiv preprint arXiv:2106.15212*, 2021.

Beata Strack, Jonathan P DeShazo, Chris Gennings, Juan L Olmo, Sebastian Ventura, Krzysztof J Cios, and John N Clore. Impact of hba1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records. *BioMed research international*, 2014, 2014.

Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. ACM, 2017.

Sohini Upadhyay, Shalmali Joshi, and Himabindu Lakkaraju. Towards robust and reliable algorithmic recourse. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Berk Ustun, Alexander Spangher, and Y. Liu. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2019.

Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2019.

Eduard Fosch Villaronga, Peter Kieseberg, and Tiffany Li. Humans forget, machines remember: Artificial intelligence and the right to be forgotten. *Computer Law & Security Review*, 34(2): 304–313, 2018.

Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10:3152676, 2017.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31(2), 2018.

Linda F Wightman. Lsac national longitudinal bar passage study. lsac research report series. 1998.

Yinjun Wu, Edgar Dobriban, and Susan Davidson. Deltagrad: Rapid retraining of machine learning models. In *International Conference on Machine Learning (ICML)*, pp. 10355–10366. PMLR, 2020.

Bo Xie, Yingyu Liang, and Le Song. Diverse neural network learns true target functions. In *Artificial Intelligence and Statistics (AISTATS)*, pp. 1216–1224. PMLR, 2017.

Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, 13–18 Jul 2020.

Rui Zhang and Shihua Zhang. Rethinking influence functions of neural networks in the over-parameterized regime. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

APPENDIX

A THEORETICAL RESULTS

A.1 UPPER BOUNDS ON RECOURSE OUTCOME INSTABILITY

Proposition 1 (Upper Bound on Output Robustness for Linear Models). *For the linear regression model $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ with weights given by $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{Y}$, an upper bound for the output robustness by removing an instance $(\mathbf{x}, y)$ from the training set is given by:*

$$\Delta_{\mathbf{x}} \leq \max_{i\in[n]} \|\mathbf{d}_i\|_2 \cdot \|\check{\mathbf{x}}_{\mathbf{1}}\|_2, \tag{15}$$

where $\mathbf{d}_i = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{x}_i \cdot \frac{r_i}{1-h_{ii}}$, $r_i = y_i - \mathbf{w}^\top \mathbf{x}_i$ and $h_{ii} = \mathbf{x}_i^\top (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{x}_i$.

Proof. By Definition 1, we have:

$$\begin{aligned}
\Delta_{\mathbf{x}} &= \left|\mathbf{w}_{\mathbf{1}}^\top \check{\mathbf{x}}_{\mathbf{1}} - \mathbf{w}_{-i}^\top \check{\mathbf{x}}_{\mathbf{1}}\right| && (16)\\
&= \left|\left(\mathbf{w}_{\mathbf{1}} - \mathbf{w}_{-i}\right)^\top \check{\mathbf{x}}_{\mathbf{1}}\right| \\
&= \left|\left((\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i \frac{(y_i - \mathbf{w}^T\mathbf{x}_i)}{1-h_{ii}}\right)^\top \check{\mathbf{x}}_{\mathbf{1}}\right| && \text{(by Theorem 1)} \quad (17)\\
&\leq \|\mathbf{d}_i\|_2 \cdot \|\check{\mathbf{x}}_{\mathbf{1}}\|_2 && \text{(by Cauchy-Schwartz)} \quad (18)\\
&\leq \|\check{\mathbf{x}}_{\mathbf{1}}\|_2 \cdot \max_{i\in[n]} \|\mathbf{d}_i\|_2 \cdot, && (19)
\end{aligned}$$

where $\mathbf{d}_i = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i \frac{(y_i - \mathbf{w}^T\mathbf{x}_i)}{1-h_{ii}}$. This completes our proof. □

Proposition 2 (Upper Bound on Output Robustness for NTK). *For the NTK model with $\mathbf{w}_{NTK} = \left(\mathbf{K}^\infty(\mathbf{X},\mathbf{X}) + \lambda \mathbf{I}_n\right)^{-1}\mathbf{Y}$, an upper bound for the output robustness by removing an instance $(\mathbf{x}, y)$ from the training set is given by:*

$$\Delta_{\mathbf{x}} \leq \|\mathbf{K}^\infty(\check{\mathbf{x}}_{\mathbf{1}}, \mathbf{X})\|_2 \cdot \max_{i\in[n]} \|\mathbf{d}_i\|_2, \tag{20}$$

where $\mathbf{d}_i = \frac{1}{k_{ii}}\mathbf{k}_i\mathbf{k}_i^\top \mathbf{Y}$, where $\mathbf{k}_i$ is the i-th column of the matrix $\left(\mathbf{K}^\infty(\mathbf{X},\mathbf{X}) + \beta \mathbf{I}_n\right)^{-1}$, and k_{ii} is its i-th diagonal element.

Proof. By Definition 1 and the weight-update theorem by Zhang & Zhang (2021) (see Appendix A.4) and the assumption of the over-parameterized regime, we have:

$$\begin{aligned}
\Delta_{\mathbf{x}} &= \left|f_{\text{NTK}}(\check{\mathbf{x}}_{\mathbf{1}}) - f_{\text{NTK}}^{-i}(\check{\mathbf{x}}_{\mathbf{1}})\right| \\
&= \left|\left(\mathbf{K}^\infty(\check{\mathbf{x}}_{\mathbf{1}}, \mathbf{X})\right)^\top \mathbf{w}_{\text{NTK}} - \left(\mathbf{K}^\infty(\check{\mathbf{x}}_{\mathbf{1}}, \mathbf{X})\right)^\top \left(\left(\mathbf{K}^\infty(\mathbf{X},\mathbf{X}) + \beta \mathbf{I}_n\right)^{-1} - \frac{1}{k_{ii}}\mathbf{k}_i\mathbf{k}_i^\top\right)\mathbf{Y}\right| && (21)\\
&= \left|\left(\mathbf{K}^\infty(\check{\mathbf{x}}_{\mathbf{1}}, \mathbf{X})\right)^\top \frac{1}{k_{ii}}\mathbf{k}_i\mathbf{k}_i^\top \mathbf{Y}\right| && (22)\\
&\leq \|\mathbf{d}_i\|_2 \cdot \|\mathbf{K}^\infty(\check{\mathbf{x}}_{\mathbf{1}}, \mathbf{X})\|_2 \quad \text{(by Cauchy-Schwartz)} \\
&\leq \|\check{\mathbf{x}}_{\mathbf{1}}\|_2 \cdot \max_{i\in[n]} \|\mathbf{d}_i\|_2, && (23)
\end{aligned}$$

where $\mathbf{d}_i = \frac{1}{k_{ii}}\mathbf{k}_i\mathbf{k}_i^\top \mathbf{Y}$ which completes our proof. □

A.2 UPPER BOUNDS ON RECOURSE ACTION INSTABILITY

Proposition 3 (Upper Bound on Input Robustness). *For the linear regression model $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$ with weights given by $\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{Y}$, an upper bound for the input robustness in the setting $s = 0, \lambda = 0$ by removing the i-th instance $(\mathbf{x}_i, y_i)$ from the training set is given by:*

$$\Phi_{\mathbf{x}}^{(2)} \leq \|\mathbf{d}_i\|_2 \frac{4\sqrt{2}\|\mathbf{x}\|_2}{\min(\|\mathbf{w}\|_2, \|\mathbf{w}_{-i}\|_2)}, \tag{24}$$

under the condition that $\mathbf{w}^\top \mathbf{w}_{-i} \leq 0$ *(no diametrical weight changes), where* $\mathbf{w}_{-i} = \mathbf{w} - \mathbf{d}_i$ *is the weight after removal of training instance* i *and* $\mathbf{d}_i = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i \frac{(y_i - \mathbf{w}^\top \mathbf{x}_i)}{1-h_{ii}}$.

Proof. For a linear scoring function $f(\mathbf{x}) = \mathbf{w}'^\top \mathbf{x}$ with given parameters $\mathbf{w}'$, under the squared ℓ_2 norm constraint with balance parameter λ, the optimal recourse action is given by (Pawelczyk et al., 2022):

$$\boldsymbol{\delta}(\mathbf{w}') = \frac{s - \mathbf{w}'^\top \mathbf{x}}{\|\mathbf{w}'\|_2^2 + \lambda} \cdot \mathbf{w}'. \tag{25}$$

Using Definition 2, we can express the total change in $\boldsymbol{\delta}$ as a path integral over changes in $\mathbf{w}$, times the change $\frac{\mathbf{D}\boldsymbol{\delta}}{\mathbf{D}\mathbf{w}}$ they entail:

$$\Phi_{\mathbf{x}}^{(2)} = \left\|\boldsymbol{\delta}_{\mathbf{1}} - \boldsymbol{\delta}_{\boldsymbol{\omega}}\right\|_2 = \left\|\boldsymbol{\delta}(\mathbf{w}) - \boldsymbol{\delta}(\mathbf{w}_{-i})\right\|_2 \tag{26}$$

$$\leq \int_0^1 \left\|\frac{\mathbf{D}\boldsymbol{\delta}}{\mathbf{D}\mathbf{w}}(\gamma\mathbf{w} + (1-\gamma)\mathbf{w}_{-i})\right\| \|\mathbf{w} - \mathbf{w}_{-i}\|_2 d\gamma, \tag{27}$$

where $\frac{\mathbf{D}\boldsymbol{\delta}}{\mathbf{D}\mathbf{w}}$ denotes the Jacobian, with the corresponding operator matrix norm. Defining $\tilde{\mathbf{w}} := \gamma\mathbf{w} + (1-\gamma)\mathbf{w}_{-i}$ and using $\|\mathbf{w} - \mathbf{w}_{-i}\|_2 = \|\mathbf{d}_i\|_2$, we obtain

$$\Phi_{\mathbf{x}}^{(2)} \leq \|\mathbf{d}_i\|_2 \int_0^1 \left\|\frac{\mathbf{D}\boldsymbol{\delta}}{\mathbf{D}\mathbf{w}}(\tilde{\mathbf{w}})\right\|_2 d\gamma. \tag{28}$$

Because of the form $\boldsymbol{\delta}(\mathbf{w}') = f(\mathbf{w}')\mathbf{w}'$, where $f(\mathbf{w}') := \frac{s - \mathbf{w}'^\top \mathbf{x}}{\|\mathbf{w}'\|_2^2 + \lambda}$ is a scalar function, its Jacobian has the form $\frac{\mathbf{D}\boldsymbol{\delta}}{\mathbf{D}\mathbf{w}'} = \mathbf{w}'\left(\nabla f(\mathbf{w}')\right)^\top + f(\mathbf{w}')\mathbf{I}$. We will now derive a bound on the Jacobian's operator norm:

$$\left\|\frac{\mathbf{D}\boldsymbol{\delta}}{\mathbf{D}\mathbf{w}'}(\tilde{\mathbf{w}})\right\|_2 = \max_{\|\mathbf{a}\|=1} \left\|\frac{\mathbf{D}\boldsymbol{\delta}}{\mathbf{D}\mathbf{w}'}\mathbf{a}\right\|_2 = \max_{\|\mathbf{a}\|=1} \left\|\mathbf{w}'\left(\nabla f(\tilde{\mathbf{w}})\right)^\top \mathbf{a} + f(\tilde{\mathbf{w}})\mathbf{a}\right\|_2 \tag{29}$$

$$\leq \|\nabla f(\tilde{\mathbf{w}})\|_2 \|\tilde{\mathbf{w}}\|_2 + |f(\tilde{\mathbf{w}})|. \tag{30}$$

Additionally, we know that for $s = 0$, $|f(\mathbf{w}')| \leq \frac{\|\mathbf{x}\|_2 \|\tilde{\mathbf{w}}\|_2}{\|\tilde{\mathbf{w}}\|_2^2} = \frac{\|\mathbf{x}\|_2}{\|\tilde{\mathbf{w}}\|_2}$. The gradient is given by

$$\|\nabla f(\tilde{\mathbf{w}})\|_2 = \left\|\frac{-(\|\tilde{\mathbf{w}}\|_2^2 + \lambda)\mathbf{x} - 2(s - \tilde{\mathbf{w}}^\top \mathbf{x})\tilde{\mathbf{w}}}{(\|\tilde{\mathbf{w}}\|_2^2 + \lambda)^2}\right\|_2 \tag{31}$$

$$\leq \frac{(\|\tilde{\mathbf{w}}\|_2^2 + \lambda)\|\mathbf{x}\|_2 + 2(s + \|\tilde{\mathbf{w}}\|_2\|\mathbf{x}\|_2)\|_2\tilde{\mathbf{w}}\|_2}{\|\tilde{\mathbf{w}}\|_2^4} \tag{32}$$

$$= \frac{3\|\mathbf{x}\|_2}{\|\tilde{\mathbf{w}}\|_2^2} \qquad \text{(Using } \lambda \to 0,\, s = 0\text{)}. \tag{33}$$

In summary,

$$\left\|\frac{\mathbf{D}\boldsymbol{\delta}}{\mathbf{D}\mathbf{w}'}(\tilde{\mathbf{w}})\right\|_2 \leq \frac{3\|\mathbf{x}\|_2}{\|\tilde{\mathbf{w}}\|_2^2}\|\tilde{\mathbf{w}}\|_2 + \frac{\|\mathbf{x}\|_2}{\|\tilde{\mathbf{w}}\|_2} = \frac{4\|\mathbf{x}\|_2}{\|\tilde{\mathbf{w}}\|_2}. \tag{34}$$

Because $\tilde{\mathbf{w}}$ is a line between $\mathbf{w}$ and $\mathbf{w}_{-i}$, its norm is bounded from below by $\|\tilde{\mathbf{w}}\|_2 \geq \frac{1}{\sqrt{2}}\min(\|\mathbf{w}\|_2, \|\mathbf{w}_{-i}\|_2) \geq \frac{1}{\sqrt{2}}(\|\mathbf{w}\|_2 - \|\mathbf{w} - \mathbf{w}_{-i}\|_2) = \frac{1}{\sqrt{2}}(\|\mathbf{w}\|_2 - \|\mathbf{d}_i\|_2)$. We can thus uniformly bound the integral and plug in the bound because of its positivity,

$$\Phi_{\mathbf{x}}^{(2)} \leq \|\mathbf{d}_i\|_2 \int_0^1 \left\|\frac{\mathbf{D}\boldsymbol{\delta}}{\mathbf{D}\mathbf{w}}(\tilde{\mathbf{w}})\right\|_2 d\gamma \tag{35}$$

$$\leq \|\mathbf{d}_i\|_2 \int_0^1 \frac{4\sqrt{2}\|\mathbf{x}\|_2}{\min(\|\mathbf{w}\|_2, \|\mathbf{w}_{-i}\|_2)} d\gamma \tag{36}$$

$$= \|\mathbf{d}_i\|_2 \frac{4\sqrt{2}\|\mathbf{x}\|_2}{\min(\|\mathbf{w}\|_2, \|\mathbf{w}_{-i}\|_2)}, \tag{37}$$

which completes the proof. □

A.3 Calculating Recourse Outcome Instability for k Deletions is NP-hard

We can show that, for a general scoring function f, the problem defined in equation 13 is NP-hard. We make this proof by providing a function f for which solving the recourse outcome invalidity problem is as hard as solving the well-known Knapsack problem, that has been shown to be NP-hard (Karp, 1972). The knapsack problem is defined as follows:

$$\max_{q_i \in \{0,1\}} \sum_{i=1}^{n} v_i q_i \text{ s.t. } \sum_{i=1}^{n} y_i q_i \leq W, \tag{38}$$

where the problem considers n fixed items $(v_i, y_i)_{i=1\ldots n}$ with a value v_i and knapsack weight $y_i > 0$, and W is a fixed weight budget. The optimization problem consists of choosing the items that maximize the summed values but have a weight lower than W. To solve this problem through the recourse outcome invalidation problem, we suppose there is a data point for each item. We can choose any $k > \frac{W}{\min y_i}$ of points to be deleted, where this condition ensures that we can remove the number of samples maximally required to solve the corresponding knapsack problem. Note that we can always add a number of dummy points that have no effect such that the total number of data points is at least k. Suppose there is a classifier function:

$$f_{\boldsymbol{\omega}}(\mathbf{x}) := \begin{cases} \sum_{i=1}^{n} v_i(1-\omega_i), & \sum_{i=1}^{n} y_i(1-\omega_i) \leq W \\ 0, & \text{else} \end{cases} . \tag{39}$$

In this case, solving Eqn. 13 comes down to finding the set of items (i.e., removing the data points) that have maximum value, but stay under the threshold W. Thus, if we can solve Eqn. 13, the solution to the equivalent knapsack problem is given by $\mathbf{q} = (\mathbf{1} - \boldsymbol{\omega})$.

A.4 Auxiliary Theoretical Results

We state the following classic result by Miller Jr (1974) without proof.

Theorem 1. *(Leave-One-Out Estimator, Miller Jr (1974)) Define $(\mathbf{x}_i, y_i)$ as the point to be removed from the training set. Given the optimal weight vector $\mathbf{w} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$ which solves for a linear model under mean-squared-error loss, the leave-one-out estimator is given by:*

$$\mathbf{w} - \mathbf{w}_{-i} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i \frac{(y_i - \mathbf{w}^T\mathbf{x}_i)}{1 - \mathbf{x}_i^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_i \frac{(y_i - \mathbf{w}^T\mathbf{x}_i)}{1 - h_{ii}} =: \mathbf{d}_i.$$

We restate the analtical solution for the NTK weights in case of a single deletion from Zhang & Zhang (2021).

Theorem 2. *(Leave-One-Out weights for NTK models, Zhang & Zhang (2021)) Let $\mathbf{w}_{NTK} = \left(\mathbf{K}^\infty(\mathbf{X},\mathbf{X}) + \lambda\mathbf{I}_n\right)^{-1}\mathbf{Y}$ be the weight for the NTK model on the full data Kernel model, where $\mathbf{K}^\infty(\mathbf{X},\mathbf{X})$ is the NTK matrix evaluated on the training data points: $[\mathbf{K}^\infty(\mathbf{X},\mathbf{X})]_{ij} = K^\infty(\mathbf{x}_i,\mathbf{x}_j)$. Then, the NTK model that would be obtained when removing instance i, could be equivalently described by*

$$f_{NTK}^{-i}(\mathbf{x}) = \mathbf{K}^\infty(\mathbf{x},\mathbf{X})^\top \mathbf{w}_{NTK,-i} = \mathbf{K}^\infty(\mathbf{x},\mathbf{X})^\top \left(\left(\mathbf{K}^\infty(\mathbf{X},\mathbf{X}) + \lambda\mathbf{I}_n\right)^{-1} - \frac{1}{q_{-ii}} \boldsymbol{q}_{-i}\boldsymbol{q}_{-i}^\top \right) \mathbf{Y}$$

where $\boldsymbol{q}_{-i}$ is the i-th column of the matrix $\boldsymbol{Q}^{-1} = \left(\mathbf{K}^\infty(\mathbf{X},\mathbf{X}) + \beta\mathbf{I}_n\right)^{-1}$, and q_{-ii} is its i-th diagonal element of this inverse.

Proof. We begin by introducing some notation. Define:

$$\boldsymbol{Q} = \mathbf{K}^\infty(\mathbf{X},\mathbf{X}) + \beta\mathbf{I}_n \tag{40}$$

$$\boldsymbol{R} = \mathbf{K}^\infty(\mathbf{X}_{-i},\mathbf{X}_{-i}) + \beta\mathbf{I}_{n-1}, \tag{41}$$

then the analytical NTK model when considering the dataset with one instance $\mathbf{x}_i$ removed is given by

$$f_{\text{NTK}}^{-i}(\mathbf{x}) = \mathbf{K}^\infty(\mathbf{x},\mathbf{X}_{-i})^\top \boldsymbol{R}^{-1}\mathbf{Y}_{-i}, \tag{42}$$

where $\mathbf{X}_{-i}$ denotes the data matrix with row i missing and $\mathbf{Y}_{-i}$ denotes the label vector with the i-th label missing. We have to show that this expression is equivalent to that stated in the theorem.

Without loss of generality, we can assume the i is the last point in the dataset (otherwise, we just permute the data set accordingly). Therefore, we can write the matrix $\boldsymbol{Q}$ in block form:

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{R} & \mathbf{K}^{\infty}(\mathbf{x}_i, \boldsymbol{X}) \\ \mathbf{K}^{\infty\top}(\mathbf{x}_i, \boldsymbol{X}) & \mathbf{K}^{\infty}(\mathbf{x}_i, \mathbf{x}_i) \end{bmatrix} := \begin{bmatrix} \boldsymbol{R} & \boldsymbol{q}_i \\ \boldsymbol{q}_i^\top & q_{ii} \end{bmatrix} \tag{43}$$

Through the block matrix inversion formula (see for example Csató & Opper (2002) (eqn. 52)) we can write $\boldsymbol{Q}$'s inverse as

$$\boldsymbol{Q}^{-1} = \begin{bmatrix} \boldsymbol{R}^{-1} + \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{q}_i\boldsymbol{q}_i^\top\boldsymbol{R}^{-1} & \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{q}_i \\ \gamma^{-1}\boldsymbol{q}_i^\top\boldsymbol{R}^{-1} & \gamma^{-1} \end{bmatrix} \tag{44}$$

with $\gamma = q_{ii} - \boldsymbol{q}_i^\top\boldsymbol{R}^{-1}\boldsymbol{q}_i$.

We denote the i-th (and last) column of $\boldsymbol{Q}^{-1}$ as $\boldsymbol{q}_{-i} = \begin{bmatrix} \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{q}_i \\ \gamma^{-1} \end{bmatrix}$ and the i-th and last diagonal element of the inverse as $q_{-ii} = \gamma^{-1}$. We will now show, that the form of the weights given in the theorem (i.e., the weights for the points not removed) are equivalent to the weights that would have been computed by plugging in the smaller kernel matrix $\mathbf{K}^{\infty}(\mathbf{X}_{-i}, \mathbf{X}_{-i})$ in the analytical solution and the weight for the point deleted will have a value of zero, i.e.,

$$\left(\left(\mathbf{K}^{\infty}(\mathbf{X}, \mathbf{X}) + \beta\mathbf{I}_n\right)^{-1} - \frac{1}{q_{-ii}}\boldsymbol{q}_{-i}\boldsymbol{q}_{-i}^\top\right)\mathbf{Y} = \left(\boldsymbol{Q}^{-1} - \frac{1}{q_{-ii}}\boldsymbol{q}_{-i}\boldsymbol{q}_{-i}^\top\right)\mathbf{Y} \tag{45}$$

$$= \begin{bmatrix} \boldsymbol{R}^{-1}\mathbf{Y}_{-i} \\ 0 \end{bmatrix}. \tag{46}$$

To show this, we plug in the inversion formula $\boldsymbol{Q}^{-1}$ from equation 44 into equation 45 and using $q_{-ii} = \gamma^{-1}$:

$$\left(\boldsymbol{Q}^{-1} - \frac{1}{q_{-ii}}\boldsymbol{q}_{-i}\boldsymbol{q}_{-i}^\top\right)\mathbf{Y} \tag{47}$$

$$= \left(\begin{bmatrix} \boldsymbol{R}^{-1} + \boldsymbol{R}^{-1}\boldsymbol{q}_i\boldsymbol{q}_i^\top\boldsymbol{R}^{-1} & \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{q}_i \\ \gamma^{-1}\boldsymbol{q}_i^\top\boldsymbol{R}^{-1} & \gamma^{-1} \end{bmatrix} - \frac{1}{\gamma^{-1}}\begin{bmatrix} \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{q}_i \\ \gamma^{-1} \end{bmatrix}\begin{bmatrix} \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{q}_i \\ \gamma^{-1} \end{bmatrix}^\top\right)\begin{bmatrix} \mathbf{Y}_{-i} \\ Y_i \end{bmatrix} \tag{48}$$

$$= \left(\begin{bmatrix} \boldsymbol{R}^{-1} + \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{q}_i\boldsymbol{q}_i^\top\boldsymbol{R}^{-1} & \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{q}_i \\ \gamma^{-1}\boldsymbol{q}_i^\top\boldsymbol{R}^{-1} & \gamma^{-1} \end{bmatrix} - \gamma\begin{bmatrix} \gamma^{-2}\boldsymbol{R}^{-1}\boldsymbol{q}_i\boldsymbol{q}_i^\top\boldsymbol{R}^{-1} & \gamma^{-2}\boldsymbol{R}^{-1}\boldsymbol{q}_i \\ \gamma^{-2}\boldsymbol{q}_i^\top\boldsymbol{R}^{-1} & \gamma^{-2} \end{bmatrix}\right)\begin{bmatrix} \mathbf{Y}_{-i} \\ Y_i \end{bmatrix} \tag{49}$$

$$= \begin{bmatrix} \boldsymbol{R}^{-1} + \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{q}_i\boldsymbol{q}_i^\top\boldsymbol{R}^{-1} - \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{q}_i\boldsymbol{q}_i^\top\boldsymbol{R}^{-1} & \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{q}_i - \gamma^{-1}\boldsymbol{R}^{-1}\boldsymbol{q}_i \\ \gamma^{-1}\boldsymbol{q}_i^\top\boldsymbol{R}^{-1} - \gamma^{-1}\boldsymbol{q}_i^\top\boldsymbol{R}^{-1} & \gamma^{-1} - \gamma^{-1} \end{bmatrix}\begin{bmatrix} \mathbf{Y}_{-i} \\ Y_i \end{bmatrix} \tag{50}$$

$$= \begin{bmatrix} \boldsymbol{R}^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix}\begin{bmatrix} \mathbf{Y}_{-i} \\ Y_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}^{-1}\mathbf{Y}_{-i} \\ 0 \end{bmatrix}. \tag{51}$$

Therefore, we have equivalence between equation 42 and the formulation in the theorem.

□

A.5 An Analytical NTK Kernel

In this section, we provide theoretical results that allow deriving the closed form solution of the NTK for the two-layer ReLU network. First, see the paper by Jacot et al. Jacot et al. (2018) for the original derivation of the neural tangent kernel.

A closed-form solution for two-layer ReLU networks. From (Zhang & Zhang, 2021; Du et al., 2019, Assumption 3.1) we obtain the definition of the Kernel matrix $\mathbf{K}^{\infty}$ (termed Gram matrix in the paper Du et al. (2019)) for ReLU networks:

$$\mathbf{K}_{ij}^{\infty} = \mathbf{K}^{\infty}(\mathbf{x}_i, \mathbf{x}_j) = \mathbb{E}_{\mathbf{w}\sim\mathcal{N}(0,\mathbf{I})}\left[\mathbf{x}_i^\top\mathbf{x}_j\mathbb{I}\left\{\mathbf{w}^\top\mathbf{x}_i \geq 0, \mathbf{w}^\top\mathbf{x}_j \geq 0\right\}\right]$$

$$
\begin{aligned}
&= \mathbf{x}_i^\top \mathbf{x}_j \mathbb{E}_{\mathbf{w}\sim\mathcal{N}(0,\mathbf{I})} \left[\mathbb{I}\left\{\mathbf{w}^\top \mathbf{x}_i \geq 0, \mathbf{w}^\top \mathbf{x}_j \geq 0\right\}\right] \\
&= \mathbf{x}_i^\top \mathbf{x}_j \frac{\pi - \arccos\left(\frac{\mathbf{x}_i^\top \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}\right)}{2\pi}.
\end{aligned}
$$

The last reformulation uses an analytical result by Cho & Saul (2009). The derived result matches the one by Xie et al. (2017), which however does not provide a comprehensive derivation.

B Additional Experimental Results

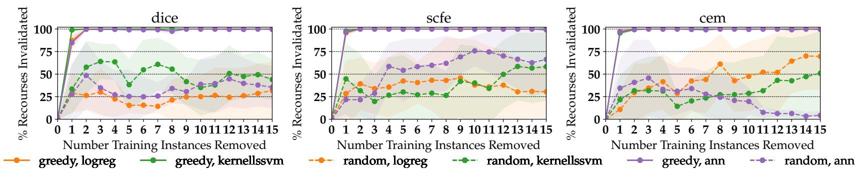
Data sets for the Classification Tasks When considering classification tasks on the *heloc* and *admission* data sets, we threshold the scores based on the median to obtain binary target labels. On the Admission data set (in the classification setting), a counterfactual is found when the predicted first-year average score switches from 'below median' to 'above median'. We then count an invalidation if, after the model update, the score of a counterfactual switches back to 'below median'. In addition to the aforementioned data sets, we use both the *Diabetes* and the *Compas* data sets. The *Diabetes* data set which contains information on diabetic patients from 130 different US hospitals (Strack et al., 2014). The patients are described using administrative (e.g., length of stay) and medical records (e.g., test results), and the prediction task is concerned with identifying whether a patient will be readmitted within the next 30 days. We sub sampled a smaller data sets of 10000 points from this dataset. 8000 points are left to train the model, while 2000 points are left for the test set. The *Compas* data set Angwin et al. (2016) contains data for more than 10,000 criminal defendants in Florida. It is used by the jurisdiction to score defendant's likelihood of reoffending. We kept a small part of the raw data as features like *name*, *id*, *casenumbers* or *date-time* were dropped. The classification task consists of classifying an instance into high risk of recidivism. Across all data sets, we dropped duplicate instances.

Discussing the Results As suggested in Section 6, here we are discussing the remaining recourse outcome invalidation results. We show these results for two settings. In Figure 4, we demonstrate the efficacy of our greedy deletion algorithm across 4 data sets on the classification tasks using different classification models (ANN, logistic regression, Kernel-SVM). For the logistic regression and the ANN model, we use the infinitesimal jackknife approximation to calculate the probitively expensive retraining step as described in Section 5. We observe that our method well outperforms random guessing. The results also highlight that while the NTK theory allows to study the deletion effects from a theoretical point of view, if one is interested in empirical worst-case approximations, the infinitesimal jackknife can be a method of choice. As we observe this pattern across all recourse methods, we hypothesize that this is related to the instability of the trained ANN models, and we leave an investigation of this interesting phenomenon for future work.

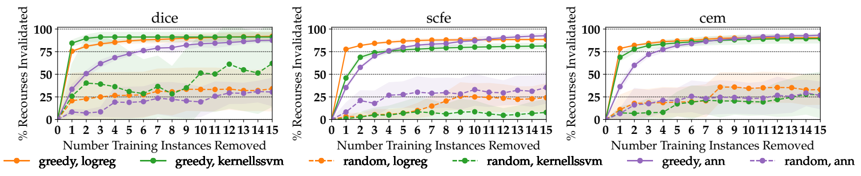
Additionally, in Figure 5, we compare our SGD-based deletion algorithm to the greedy algorithm. For the SGD-based deletion results, we observe inverse-u-shaped curves on some method-data-model combinations. The reason for this phenomenon can be explained as follows: when the ℓ_0 regularization strength (i.e., η) is not strong enough, then the importance weights for the k-th removal with $k > 5$ become more variable (i.e., SGD does not always select the most important data weight for larger k). This drop in performance can be mitigated by increasing the strength of the ℓ_0 regularizer within our SGD-based deletion algorithm.

In Figure 6, we study a simple removal strategy aimed at increasing the stability of algorithmic recourse. To this end, we identified the 15 points that lead to the highest invalidation on the NTK and linear regression models when the underlying recourse method is `SCFE`. Using our greedy method, we remove these 15 points from the training data set, and we then then rerun our proposed greedy removal algorithm. This strategy leads to an improvement of up to 6 percentage points over the initial model where the 15 most critical points were included, suggesting that the removal of these critical points can be used to alleviate the recourse instability issue. In future work, we plan to investigate strategies that increase the robustness of algorithmic recourse even further.

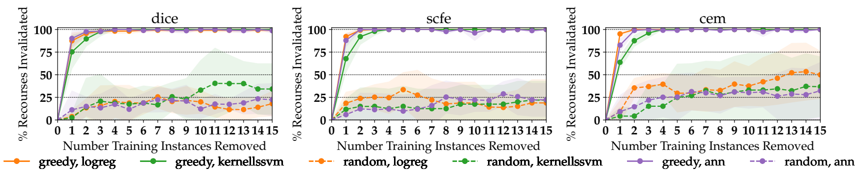
Finally, in Figure 7 we study how well the critical points identified for the NTK model would invalidate a wide 2-layer ReLU network with 10000 hidden nodes. To study that question, we identified the points that lead to the highest invalidation on the NTK using our greedy method, and we then use these identified training points to invalidate the recourses suggested by the wide ANN. As before, we are running these experiments on the full data set across 5 folds. Figure 7 demonstrates the results of this strategy for the SCFE recourse method. We see that this strategy increases the robustness of up to 30 percentage points over the random baseline, suggesting that critical points under NTK can be used to estimate recourse invalidation for wide ANN models.



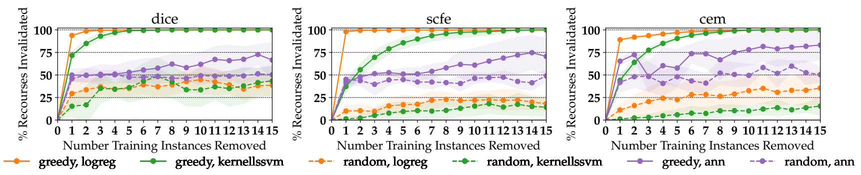
(a) Admission



(b) Heloc



(c) Compas



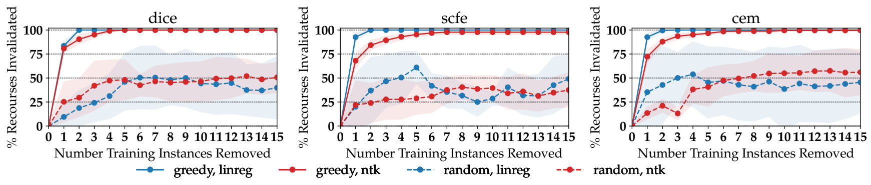
(d) Diabetes

Figure 4: Measuring the tradeoff between recourse outcome instability and the number of deletion requests for the Admission, Heloc, Diabetes and Compas data sets for logistic regression, kernel svm, and ANN models across recourse methods on classification tasks. Results were obtained by greedy optimization. The dotted lines indicate the random baselines.

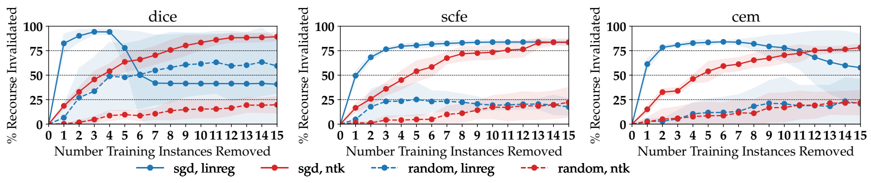
C Implementation Details

C.1 Details on Model Training

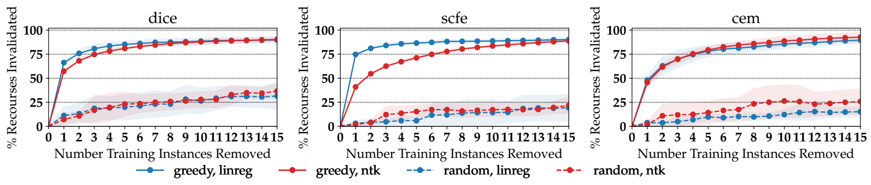
We train the classification models using the hyperparameters given in Table 1. The ANN and the Logistic regression models are fit using the quasi-newton lbgfs solver. We add L2-regularization to the ANN weights. The other methods are trained via their analytical solutions. Below, in Algorithms 1 and 2, we show pseudocodes for both our greedy and sgd-based deletion methods to invalidate the recourse outcome. In order to do the optimization with respect to the recourse action stability measure, we slightly adjust Algorithm 2 to optimize the right metric from Definition 2.



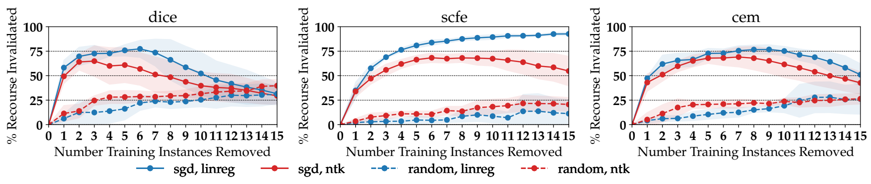
(a) Admission (Greedy)



(b) Admission (SGD)



(c) Heloc (Greedy)



(d) Heloc (SGD)

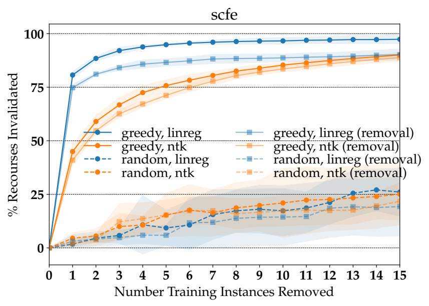
Figure 5: Measuring the tradeoff between recourse outcome instability and the number of deletion requests for the Admission and Heloc data sets for linear regression and NTK models across recourse methods on regression tasks. Results were obtained by both SGD and Greedy optimization. The dotted lines indicate the random baselines.

C.2 Details on Generating the Counterfactuals

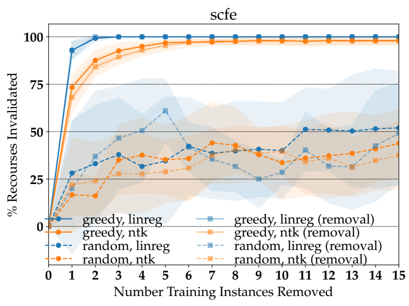
For `DICE`, for every test input, we generate two different counterfactual explanations. Then we randomly pick either the first or second counterfactual to be the counterfactual assigned to the given input. Across all recourse methods the success rates lie above 95%, i.e., for 95% of recourse seeking individuals the algorithms can identify recourses. The only exception is admission data set for the NTK model, where the success rate lies at 60%. Across all recourse methods we set $\lambda \rightarrow 0$. Note that the default implementations use early stopping once a feasible recourse has been identified.

C.3 Details on the ℓ_0 Regularizer

Since an ℓ_0 regularizer is computationally intractable for high-dimensional optimization problems, we have to resort to approximations. One such approximation approach was recently suggested by Yamada et al. (2020). The underlying idea consists of converting the combinatorial search problem to a continuous search problem over distribution parameters. To this end, recall our optimization

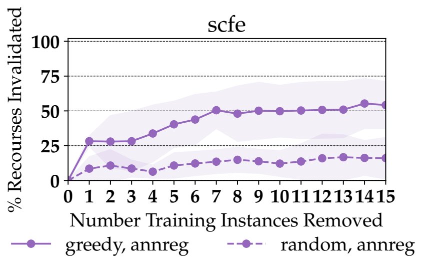


(a) Heloc (Greedy)



(b) Admission (Greedy)

Figure 6: Measuring the efficacy of a simple removal strategy on the Heloc and Admission data set for linear and NTK regression models. We removed the 15 critical points identified for the linear and NTK models when the underlying recourse method is `SCFE` and reran the removal algorithm on the remaining training set. Results were obtained by Greedy optimization. The dotted lines indicate the random baselines.



(a) Admission (Greedy)

Figure 7: Measuring the tradeoff between recourse outcome instability and the number of deletion requests for the Admission data set for a neural network regression model. We used the critical points identified for the NTK model to invalidate the recourses identified by a wide 2-layer ReLU network with 10000 hidden nodes. Results were obtained by Greedy optimization. The dotted lines indicate the random baselines.

problem from the main text:

$$\boldsymbol{\omega}^* = \underset{\boldsymbol{\omega}\in\{0,1\}^n}{\arg\max}\; m(\boldsymbol{\omega}) - \eta \cdot \|\mathbf{1} - \boldsymbol{\omega}\|_0. \tag{52}$$

We will now introduce Bernoulli random variables $Z_i \in \{0, 1\}$ with corresponding parameters π_i to model the individual ω_i. Instead of optimizing the objective above with respect to $\boldsymbol{\omega}$ we will optimize

Model	Parameters
Linear Regression	OLS, no hyperparameters.
NTK Regression	$\beta = 2$ (Admission), $\beta = 5$ (other data sets)
Logistic Regression	L2-Regularization with $C = 1.0$
Kernel-LSSVM	Gaussian Kernel with $\gamma = 1.0$ (see Cawley & Talbot (2004))
ANN	2-Layer, 30 Hidden units, Sigmoid, $\alpha = 10$ (L2-Regularization)

Table 1: Model hyperparameters used in this work

Algorithm 1 Greedy recourse outcome invalidation

```
Required: Model: $f_{\mathbf{w}(\mathbf{1})}$; Matrix of Recourses: $\check{\mathbf{X}}_\mathbf{1} \in \mathbb{R}^{q \times d}$; $d$: input dimension; $q$ number of recourse points on test set; $n$: # train points; $M$: max # deleted train points; $s$: invalidation target
$\boldsymbol{\omega}^{(0)} = \mathbf{1}_n$                                          ▷ All training instances present
for $m = 1 : M$ do
    $\boldsymbol{\omega}^{(m)} \leftarrow \boldsymbol{\omega}^{(m-1)}$
    $\tilde{\mathbf{Y}} = \mathbf{0}_{n\times q}$                                    ▷ Recourse outcomes
    $\boldsymbol{J} = \mathbf{0}_{n\times q}$                                      ▷ Invalidations present
    $S^{(m)} \leftarrow \left\{ i \,\middle|\, \boldsymbol{\omega}_i^{(m)} \neq 0 \right\}$          ▷ Set of train instances present at iteration $m$
    for $i \in S^{(m)}$ do
        $\mathbf{w}_{\text{new}}^{(i)} = \texttt{update\_w}(\boldsymbol{\omega}_{-i}^{(m)})$     ▷ $\boldsymbol{\omega}_{-i}^{(m)}$ has additionally set weight $i$ to 0.
                                                        ▷ Use analytical or IJ solution for $\mathbf{w}(\boldsymbol{\omega})$
        $\tilde{\mathbf{Y}}[i,:] = f_{\mathbf{w}_{\text{new}}^{(i)}}(\check{\mathbf{X}}_\mathbf{1})$              ▷ New recourse outcomes
        $\boldsymbol{J}[i,:] = \mathbb{I}(\tilde{\mathbf{Y}}[i,:] < s)$                 ▷ Invalidation present
    end for
    index $\leftarrow \arg\max_{i \in S^{(m)}} \|\mathbf{J}[i,:]\|_1$      ▷ Find point that leads to highest invalidation
    $\boldsymbol{\omega}^{(m)}$[index] $= 0$                             ▷ Remove training point
end for
return: $\boldsymbol{\omega}^{(M)}$                            ▷ data weights indicating $M$ removals
```

with respect to distribution parameters $\boldsymbol{\pi}$:

$$\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{\pi}} \; m(\mathbf{Z}(\boldsymbol{\pi})) - \eta \cdot \|\mathbf{1} - \mathbf{Z}(\boldsymbol{\pi})\|_0. \tag{53}$$

Since the above optimization problem is known to suffer from high-variance solutions, (Yamada et al., 2020) suggest to use a Gaussian-based continuous relaxation of the Bernoulli variables:

$$\tilde{Z}_i = \max(0, \min(1, \mu_i + \epsilon_i)), \tag{54}$$

where $\epsilon_i = \mathcal{N}(0, \sigma^2)$, resulting in the following optimization problem:

$$\boldsymbol{\mu}^* = \arg\max_{\boldsymbol{\mu}} \; m(\tilde{\mathbf{Z}}(\boldsymbol{\mu})) - \eta \cdot \|\mathbf{1} - \tilde{\mathbf{Z}}(\boldsymbol{\mu})\|_0. \tag{55}$$

At inference time, the optimal weights are then given by $\tilde{Z}_i^* = \max(0, \min(1, \mu_i^*)) \; \forall i \in [n]$. To obtain discrete weights, we take the argmax over each individual $\tilde{Z}_i$.

C.4 Details on the Jackknife Approximation

When the model parameters $\mathbf{w}$ are a function of the data weights by solving equation 1 we can approximate $\mathbf{w}(\boldsymbol{\omega})$ using the infinitesimal Jackknife (IJ) without having to optimize equation 1 repeatedly (Jaeckel, 1972; Efron, 1982; Giordano et al., 2019b;a):

$$\mathbf{w}_{\text{IJ}}(\boldsymbol{\omega}) = \mathbf{w}_\mathbf{1} - \mathbf{H}_\mathbf{1}^{-1}\mathbf{G}_{\boldsymbol{\omega}-\mathbf{1}}, \tag{56}$$

where $\mathbf{G}$ and $\mathbf{H}_\mathbf{1}$ are the Jacobian and the Hessian matrices of the loss function with respect to the data weights evaluated at the optimal model parameters $\mathbf{w}$, i.e., $\mathbf{G}_{\boldsymbol{\omega}-\mathbf{1}} = \frac{1}{n}\sum_{i=1}^{n}(\omega_i - 1) \cdot \frac{\partial \ell(f_\mathbf{w}(\mathbf{x}_i), y_i)}{\partial \mathbf{w}}$ and $\mathbf{H}_\mathbf{1} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial^2 \ell(f_\mathbf{w}(\mathbf{x}_i), y_i)}{\partial \mathbf{w} \partial \mathbf{w}^\top}$. Note that this technique computes the Hessian matrix $\mathbf{H}_\mathbf{1}$ only once. Using this Jackknife approximation, the Jacobian term $\mathbf{G}_{\boldsymbol{\omega}-1}$ becomes an explicit function of the data weights which makes the Jackknife approximation amenable to optimization.

Algorithm 2 SGD recourse outcome invalidation

```
**Required:** Model: $f_{\mathbf{w}(\mathbf{1})}$; Matrix of Recourses: $\check{\mathbf{X}}_{\mathbf{1}} \in \mathbb{R}^{q \times d}$; $d$: input dimension; $q$ number of recourse points on test set; $n$: # train points; $M$: max # deleted train points; $s$: invalidation target
$\boldsymbol{\mu}^{(1)} = \mathbf{1}_n$    ▷ Mu are soft data weights that are opimized.
**for** $m = 1$ : Step **do**    ▷ Perform $Step$ number of updates.
    $\delta-$loss=0.0
    **for** $k = 1 : K$ **do**    ▷ Use $K$ Monte-Carlo Samples for the approximation
        Sample $\boldsymbol{\epsilon}_k^{(m)} \sim \mathcal{N}\left(0, \sigma^2 \mathbf{I}_n\right)$
        $\boldsymbol{\omega}_k^{(m)} = \max\left(0, \min\left(1, \boldsymbol{\mu}^{(m)} + \boldsymbol{\epsilon}_k^{(m)}\right)\right)$    ▷ Sample (soft) data weights as in Yamada et al. (2020)
        $\mathbf{w}_{k,\text{new}}^{(m)} = \texttt{update\_w}(\boldsymbol{\omega}_k^{(m)})$    ▷ Compute model weights from data weights either analytically or with IJ
        $l_k^{(m)} = \text{sigmoid}\left(f_{\mathbf{w}_{k,\text{new}}^{(m)}}(\check{\mathbf{X}}_{\mathbf{1}}) - s\right)$    ▷ Predict with new weights and compute soft invalidation.
        $\delta-\text{loss} = \delta-\text{loss} + \|l_k^{(m)}\|_1$    ▷ Add up soft inval. loss
    **end for**
    $r^{(m)} = \sum_{i=1}^{n} \Phi\left(\frac{1-(\boldsymbol{\mu}^{(m)})_i}{\sigma}\right)$    ▷ Sparsity Regularizer from Yamada et al. (2020)
    $\boldsymbol{\mu}^{(m+1)} = \boldsymbol{\mu}^{(m)} + \gamma \nabla_{\boldsymbol{\mu}^{(m)}}\left(\frac{\delta-\text{loss}}{D} + \lambda r^{(m)}\right)$    ▷ Grad. Descent with lr. $\gamma$
**end for**
removed_ind = $\texttt{argsort}(\boldsymbol{\mu}^{(\text{Step}+1)})$    ▷ Sort indices ascendingly
$j = 0$
$\boldsymbol{\omega} = \mathbf{1}_n$
**while** $j < M$ **do**    ▷ Binarize and fulfil max number M
    $\boldsymbol{\omega}$[removed_ind[j]]= 0
    $j = j + 1$
**end while**
**return**: $\boldsymbol{\omega}$    ▷ data weights indicating $M$ removals
```

A.5 Probabilistically Robust Recourse

Publication: Accepted at the 11th International Conference on Learning Representations (**ICLR**) (2023) (reviews).

Contribution: I developed the method, wrote the implementation, performed the experiments and developed the theory. Teressa Datta added the `DICE` recourse algorithm to the `CARLA` library to have more baselines to compare against. Johannes van den Heuvel experimented with an initial idea of mine that did not work well and is not featured in the paper. Himabindu Lakkarju wrote the introduction; I wrote the remaining parts of the paper. Gjergji Kasneci made valuable suggestions by challenging and improving ideas and the presentation. Himabindu Lakkaraju and Gjergji Kasneci contributed to the paper by revising the manuscript.

PROBABILISTICALLY ROBUST RECOURSE: NAVIGATING THE TRADE-OFFS BETWEEN COSTS AND ROBUSTNESS IN ALGORITHMIC RECOURSE

Martin Pawelczyk[1]*, Teresa Datta[2], Johannes van-den-Heuvel[1]
Gjergji Kasneci[1], Himabindu Lakkaraju[2]
[1]University of Tübingen, Germany
[2]Harvard University, US

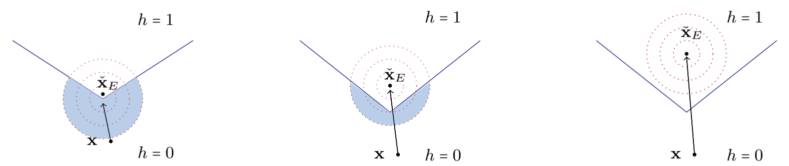
ABSTRACT

As machine learning models are increasingly being employed to make consequential decisions in real-world settings, it becomes critical to ensure that individuals who are adversely impacted (e.g., loan denied) by the predictions of these models are provided with a means for recourse. While several approaches have been proposed to construct recourses for affected individuals, the recourses output by these methods either achieve low costs (i.e., ease-of-implementation) or robustness to small perturbations (i.e., noisy implementations of recourses), but not both due to the inherent trade-offs between the recourse costs and robustness. Furthermore, prior approaches do not provide end users with any agency over navigating the aforementioned trade-offs. In this work, we address the above challenges by proposing the first algorithmic framework which enables users to effectively manage the recourse cost vs. robustness trade-offs. More specifically, our framework Probabilistically ROBust rEcourse (`PROBE`) lets users choose the probability with which a recourse could get invalidated (recourse invalidation rate) if small changes are made to the recourse i.e., the recourse is implemented somewhat noisily. To this end, we propose a novel objective function which simultaneously minimizes the gap between the achieved (resulting) and desired recourse invalidation rates, minimizes recourse costs, and also ensures that the resulting recourse achieves a positive model prediction. We develop novel theoretical results to characterize the recourse invalidation rates corresponding to any given instance w.r.t. different classes of underlying models (e.g., linear models, tree based models etc.), and leverage these results to efficiently optimize the proposed objective. Experimental evaluation with multiple real world datasets demonstrates the efficacy of the proposed framework.

1 INTRODUCTION

Machine learning (ML) models are increasingly being deployed to make a variety of consequential decisions in domains such as finance, healthcare, and policy. Consequently, there is a growing emphasis on designing tools and techniques which can provide *recourse* to individuals who have been adversely impacted by the predictions of these models (Voigt & Von dem Bussche, 2017). For example, when an individual is denied a loan by a model employed by a bank, they should be informed about the reasons for this decision and what can be done to reverse it. To this end, several approaches in recent literature tackled the problem of providing recourse by generating counterfactual explanations (Wachter et al., 2018; Ustun et al., 2019; Karimi et al., 2020a). which highlight what features need to be changed and by how much to flip a model's prediction. While the aforementioned approaches output low cost recourses that are easy to implement (i.e., the corresponding counterfactuals are close to the original instances), the resulting recourses suffer from a severe lack of robustness as demonstrated by prior works (Pawelczyk et al., 2020b; Rawal et al., 2021). For example, the aforementioned approaches generate recourses which do not remain

*Corresponding author: martin.pawelczyk@uni-tuebingen.de



(a) Low recourse cost and low recourse robustness

(b) Medium recourse cost and medium recourse robustness

(c) High recourse cost and high recourse robustness

Figure 1: Pictorial representation of the recourses (counterfactuals) output by various state-of-the-art recourse methods and our framework. The blue line is the decision boundary, and the shaded areas correspond to the regions of recourse invalidation. Fig. 1a shows the recourse output by approaches such as Wachter et al. (2018) where both the recourse cost as well as robustness are low. Fig. 1c shows the recourse output by approaches such as Dominguez-Olmedo et al. (2022) where both the recourse cost and robustness are high. Fig. 1b shows the recourse output by our framework `PROBE` in response to user input requesting an intermediate level of recourse robustness.

valid (i.e., result in a positive model prediction) if/when small changes are made to them (See Figure 1a). However, recourses are often noisily implemented in real world settings as noted by prior research (Björkegren et al., 2020). For instance, an individual who was asked to increase their salary by \$500 may get a promotion which comes with a raise of \$505 or even \$499.95.

Prior works by Upadhyay et al. (2021) and Dominguez-Olmedo et al. (2022) proposed methods to address some of the aforementioned challenges and generate robust recourses. While the former constructed recourses that are robust to small shifts in the underlying model, the latter constructed recourses that are robust to small input perturbations. These approaches adapted the classic minimax objective functions commonly employed in adversarial robustness and robust optimization literature to the setting of algorithmic recourse, and used gradient descent style approaches to optimize these functions. In an attempt to generate recourses that are robust to either small shifts in the model or to small input perturbations, the above approaches find recourses that are farther away from the underlying model's decision boundaries (Tsipras et al., 2018; Raghunathan et al., 2019), thereby increasing the recourse costs i.e., the distance between the counterfactuals (recourses) and the original instances. Higher cost recourses are harder to implement for end users as they are farther away from the original instance vectors (current user profiles). Putting it all together, the aforementioned approaches generate robust recourses that are often high in cost and are therefore harder to implement (See Figure 1c), without providing end users with any say in the matter. In practice, each individual user may have a different preference for navigating the trade-offs between recourse costs and robustness – e.g., some users may be willing to tolerate additional cost to avail more robustness to noisy responses, whereas other users may not.

In this work, we address the aforementioned challenges by proposing a novel algorithmic framework called Probabilistically ROBust rEcourse (`PROBE`) which enables end users to effectively manage the recourse cost vs. robustness trade-offs by letting users choose the probability with which a recourse could get invalidated (recourse invalidation rate) if small changes are made to the recourse i.e., the recourse is implemented somewhat noisily (See Figure 1b). To the best of our knowledge, this work is the first to formulate and address the problem of enabling users to navigate the trade-offs between recourse costs and robustness. Our framework can ensure that a resulting recourse is invalidated at most $r\%$ of the time when it is noisily implemented, where r is provided as input by the end user requesting recourse. To operationalize this, we propose a novel objective function which simultaneously minimizes the gap between the achieved (resulting) and desired recourse invalidation rates, minimizes recourse costs, and also ensures that the resulting recourse achieves a positive model prediction. We develop novel theoretical results to characterize the recourse invalidation rates corresponding to any given instance w.r.t. different classes of underlying models (e.g., linear models, tree based models etc.), and leverage these results to efficiently optimize the proposed objective.

We also carried out extensive experimentation with multiple real-world datasets. Our empirical analysis not only validated our theoretical results, but also demonstrated the efficacy of our proposed framework. More specifically, we found that our framework `PROBE` generates recourses that are not only three times less costly than the recourses output by the baseline approaches (Upadhyay et al., 2021; Dominguez-Olmedo et al., 2022), but also more robust (See Table 1). Further, our framework PROBE reliably identified low cost recourses at various target recourse invalidation rates r in case of both linear and non-linear classifiers (See Table 1 and Figure 4). On the other hand, the baseline approaches were not only ill-suited to achieve target recourse invalidation rates but also had trouble finding recourses in case of non-linear classifiers.

2 Related Work

Algorithmic Approaches to Recourse. As discussed earlier, several approaches have been proposed in literature to provide recourse to individuals who have been negatively impacted by model predictions (Tolomei et al., 2017; Laugel et al., 2017; Wachter et al., 2018; Ustun et al., 2019; Van Looveren & Klaise, 2019; Pawelczyk et al., 2020a; Mahajan et al., 2019; Mothilal et al., 2020; Karimi et al., 2020a; Rawal & Lakkaraju, 2020; Karimi et al., 2020b; Dandl et al., 2020; Antorán et al., 2021; Spooner et al., 2021). These approaches can be roughly categorized along the following dimensions (Verma et al., 2020): *type of the underlying predictive model* (e.g., tree based vs. differentiable classifier), whether they encourage *sparsity* in counterfactuals (i.e., only a small number of features should be changed), whether counterfactuals should lie on the *data manifold* and whether the underlying *causal relationships* should be accounted for when generating counterfactuals, All these approaches generate recourses assuming that the prescribed recourses will be correctly implemented by users.

Robustness of Algorithmic Recourse. Prior works have focused on determining the extent to which recourses remain robust to the choice of the underlying model (Pawelczyk et al., 2020b; Black et al., 2021; Pawelczyk et al., 2023), shifts or changes in the underlying models (Rawal et al., 2021; Upadhyay et al., 2021), or small perturbations to the input instances (Artelt et al., 2021; Dominguez-Olmedo et al., 2022; Slack et al., 2021). To address these problems, these works have primarily proposed adversarial inimax objectives to minimize the worst-case loss over a plausible set of instance perturbations for linear models to generate robust recourses (Upadhyay et al., 2021; Dominguez-Olmedo et al., 2022), which are known to generate overly costly recourse suggestions.

In contrast to the aforementioned approaches our work focuses on a user-driven framework for navigating the trade-offs between recourse costs and robustness to noisy responses by suggesting a novel probabilistic recourse framework. To this end, we present several algorithms that enable us to handle both linear and non-linear models (e.g., deep neural networks, tree based models) effectively, resulting in better recourse cost/invalidation rate tradeoffs compared to both Upadhyay et al. (2021) and Dominguez-Olmedo et al. (2022).

3 Preliminaries

Here, we first discuss the generic formulation leveraged by several state-of-the-art recourse methods including Wachter et al. (2018). We then define the notion of recourse invalidation rate formally.

3.1 Algorithmic Recourse: General Formulation

Notation Let $h : \mathcal{X} \rightarrow \mathcal{Y}$ denote a classifier which maps features $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$ to labels $\mathcal{Y}$. Let $\mathcal{Y} = \{0, 1\}$ where 0 and 1 denote an unfavorable outcome (e.g., loan denied) and a favorable outcome (e.g., loan approved), respectively. We also define $h(\mathbf{x})$=$g(f(\mathbf{x}))$, where $f : \mathcal{X} \rightarrow \mathbb{R}$ is a differentiable scoring function (e.g., logit scoring function) and $g : \mathbb{R} \rightarrow \mathcal{Y}$ an activation function that maps logit scores to binary labels. Throughout the remainder of this work we will use $g(u) = \mathbb{I}[u > \xi]$, where ξ is a decision rule in logit space. W.l.o.g. we will set $\xi = 0$.

Counterfactual (CF) explanation methods provide recourses by identifying which attributes to change for reversing an unfavorable model prediction. Since counterfactuals that propose changes to features such as gender are not actionable, we restrict the search space to ensure that only actionable changes are allowed. Let $\mathcal{A}$ denote the set of actionable counterfactuals. For a given predictive model h , and a predefined cost function $d_c : \mathbb{R}^d \longrightarrow \mathbb{R}_+$, the problem of finding a counterfactual explanation $\check{\mathbf{x}} = \mathbf{x} + \boldsymbol{\delta}$ for an instance $\mathbf{x} \in \mathbb{R}^d$ is expressed by the following optimization problem:

$$\check{\mathbf{x}} = \underset{\mathbf{x}' \in \mathcal{A}}{\arg\min}\ \ell\big(h(\mathbf{x}'), 1)\big) + \lambda \cdot d_c(\mathbf{x}, \mathbf{x}'), \tag{1}$$

where $\lambda \geq 0$ is a trade-off parameter, and $\ell(\cdot, \cdot)$ is the mean-squared-error (MSE) loss.

The first term on the right-hand-side ensures that the model prediction corresponding to the counterfactual i.e., $h(\mathbf{x}')$ is close to the favorable outcome label 1. The second term encourages low-cost recourses; for example, Wachter et al. (2018) propose ℓ_1 or ℓ_2 distances to ensure that the distance between the original instance $\mathbf{x}$ and the counterfactual $\check{\mathbf{x}}$ is small.

3.2 Defining the Recourse Invalidation Rate

In order to enable end users to effectively navigate the trade-offs between recourse costs and robustness, we let them choose the probability with which a recourse could get invalidated (recourse invalidation rate) if small changes are made to it i.e., the recourse is implemented somewhat noisily. To this end, we formally define the notion of Recourse Invalidation Rate (IR) in this section. We first introduce two key terms, namely, *prescribed recourses* and *implemented recourses*. A prescribed recourse is a recourse that was provided to an end user by some recourse method (e.g., increase salary by \$500). An implemented recourse corresponds to the recourse that the end user finally implemented (e.g., salary increment of \$505) upon being provided with the prescribed recourse. With this basic terminology in place, we now proceed to formally define the Recourse Invalidation Rate (IR) below.

Definition 1 (Recourse Invalidation Rate). *For a given classifier h, the recourse invalidation rate corresponding to the counterfactual $\check{\mathbf{x}}_E = \mathbf{x} + \boldsymbol{\delta}_E$ output by a recourse method E is given by:*

$$\Delta(\check{\mathbf{x}}_E) = \mathbb{E}_{\boldsymbol{\varepsilon}}\Big[\underbrace{h(\check{\mathbf{x}}_E)}_{\text{CF class}} - \underbrace{h(\check{\mathbf{x}}_E + \boldsymbol{\varepsilon})}_{\text{class after response}}\Big], \tag{2}$$

where the expectation is taken with respect to a random variable $\boldsymbol{\varepsilon}$ with probability distribution $p_{\boldsymbol{\varepsilon}}$ which captures the noise in human responses.

Since the implemented recourses do not typically match the prescribed recourses $\check{\mathbf{x}}_E$ (Björkegren et al., 2020), we add $\boldsymbol{\varepsilon}$ to model the noise in human responses. As we primarily compute recourses for individuals $\mathbf{x}$ such that $h(\mathbf{x}) = 0$, the label corresponding to the counterfactual is given by $h(\check{\mathbf{x}}_E)$=1 and therefore $\Delta \in [0, 1]$. For example, the following cases help understand our recourse invalidation rate metric better: When Δ=0, then the prescribed recourse and the recourse implemented by the user agree all the time; when Δ=0.5, the prescribed recourse and the implemented recourse agree half of the time, and finally, when Δ=1 then the prescribed recourse and the recourse implemented by the user never agree. To illustrate our ideas, we will use our IR measure with a Gaussian probability distribution (i.e., $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$) to model the noise in human responses.

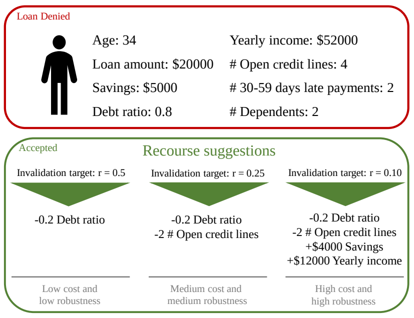


Figure 2: Practical view on navigating the cost/robustness tradeoff for a credit loan example.

4 Our Framework: Probabilistically Robust Recourse

Below we present our objective function, which is followed by a discussion on how to operationalize it efficiently.

4.1 Recourse invalidation rate aware objective

The core idea is to find a recourse $\check{\mathbf{x}}$ whose prediction at any point y within some set around $\check{\mathbf{x}}$ belongs to the positive class with probability $1 - r$. Hence, our goal is to devise an algorithm that reliably guides the recourse search towards regions of low invalidation probability while maintaining low cost recourse (see Fig. 2 for a practical example). For a fixed model, our objective reads:

$$\mathcal{L} = \lambda_1 R(\mathbf{x}'; \sigma^2\mathbf{I}) + \lambda_2 \ell\big(f(\mathbf{x}'), s)\big) + \lambda_3 d_c(\mathbf{x}', \mathbf{x}), \tag{3}$$

where s is the target score for the input $\mathbf{x}$, $R(\mathbf{x}'; r, \sigma^2\mathbf{I}) = \max(0, \Delta(\mathbf{x}'; \sigma^2\mathbf{I}) - r)$ with r being the target IR, $\Delta(\mathbf{x}'; \sigma^2\mathbf{I})$ is the recourse invalidation rate from equation 1, λ_1 to λ_3 are the balance parameters, and d_c quantifies the distance between the input and the prescribed recourse. To arrive at a output probability of 0.5, the target score for $f(\mathbf{x})$ for a sigmoid function is $s = 0$, where the score corresponds to a 0.5 probability for $y = 1$.

The new component R is a Hinge loss encouraging that the prescribed recourse has a low probability of invalidation, and the parameter σ^2 is the uncertainty magnitude and controls the size of the neighbourhood in which the recourse has to be robust. The middle term encourages the score at the prescribed recourse $f(\check{\mathbf{x}})$ to be close to the target score s, while the last term promotes the distance between the input $\mathbf{x}$ and the recourse $\check{\mathbf{x}}$ to be small.

In practice, the choice of r depends on the risk-aversion of the end-user. If the end-user is not confident about achieving a 'precision landing', then a rather low invalidation target should be chosen (i.e., $r < 0.5$).

4.2 Optimizing the recourse invalidation rate aware objective

Algorithm 1 `PROBE`

```
Input: $\mathbf{x}$ s.t. $f(\mathbf{x}) < 0$, $f$, $\sigma^2$, $\lambda > 0$, $\alpha, r > 0$
Init.: $\mathbf{x}' = \mathbf{x}$;
Compute $\tilde{\Delta}(\mathbf{x}')$   ▷ from Theorem 1
while $\tilde{\Delta}(\mathbf{x}') > r$ and $f(\mathbf{x}') < 0$ do
    $\tilde{\Delta}$ = ClosedFormIR($f, \sigma^2, \mathbf{x}'$)   ▷ from Theorem 1
    $\mathbf{x}' = \mathbf{x}' - \alpha \cdot \nabla_{\mathbf{x}'} \mathcal{L}(\mathbf{x}'; \sigma^2, r, \lambda)$   ▷ Opt. equation 3
end while
Return: $\check{\mathbf{x}} = \mathbf{x}'$
```

In order for the objective in equation 3 to guide us reliably towards recourses with low target invalidation rate r, we need to approximate the invalidation rate $\Delta(\mathbf{x}')$ at any $\mathbf{x}' \in \mathbb{R}^d$. However, such an approximation becomes non-trivial since the recourse invalidation rate, which depends on the classifier h, is generally non-differentiable since the classifier $h(\mathbf{x}) = I(f(\mathbf{x}) > \xi)$ as defined in Section 3 involves an indicator function acting on the score f. To circumvent this issue, we derive a closed-form expression for the IR using a local approximation of the predictive model f. The procedure suggested here remains generalizable even for non-linear models since the local behavior of a given non-linear model can often be well approximated by fitting a locally linear model (Ribeiro et al., 2016; Ustun et al., 2019).

Theorem 1 (Closed-Form Recourse Invalidation Rate)**.** *A first-order approximation $\tilde{\Delta}$ to the recourse invalidation rate Δ in equation 2 under Gaussian distributed noise in human responses $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma\mathbf{I})$ is given by:*

$$\tilde{\Delta}(\check{\mathbf{x}}_E; \sigma^2\mathbf{I}) = 1 - \Phi\left(\frac{f(\check{\mathbf{x}}_E)}{\sqrt{\nabla f(\check{\mathbf{x}}_E)^\top \sigma^2 \mathbf{I} \nabla f(\check{\mathbf{x}}_E)}}\right), \tag{4}$$

where Φ is the CDF of the univariate standard normal distribution $\mathcal{N}(0,1)$, $f(\check{\mathbf{x}}_E)$ denotes the logit score at $\check{\mathbf{x}}_E$ which is the recourse output by a recourse method E, and $h(\check{\mathbf{x}}_E) \in \{0,1\}$.

All theoretical proofs along with the proof to the above proposition can be found in Appendix D. In Algorithm 1, we show pseudo-code of our optimization procedure. Using gradient descent we update the recourse repeatedly until the class label flips from 0 to 1 and the IR $\tilde{\Delta}$ is smaller than the targeted invalidation rate r. In essence, the result in Theorem 1 serves as our regularizer since it steers recourses towards low-invalidation regions. For example, when $f(\check{\mathbf{x}}_E) = 0$, then $\tilde{\Delta} = 0.5$ since $\Phi(0) = \frac{1}{2}$. This means that the prescribed recourse and the recourse implemented by the user *agree* 50% *of the time*. On the other hand, when $f(\check{\mathbf{x}}_E) \to +\infty$, then $\tilde{\Delta} \to 0$ since $\Phi \to 1$, which means that the prescribed recourse and the recourse implemented by the user *always agree*. Figure 3 demonstrates how `PROBE` finds recourses relative to a standard low-cost algorithm (Wachter et al., 2018).

We now leverage the recourse invalidation rate derived in Theorem 1 to show how the recourses output by Wachter et al. (2018) can be made more robust. Pawelczyk et al. (2022) provide a closed-form solution for the recourse output by Wachter et al. (2018) w.r.t. the special case of a logistic regression classifier when $d_c = \|\mathbf{x} - \mathbf{x}'\|_2$ and the MSE-loss is used. This solution takes the following form: $\check{\mathbf{x}}_{\text{Wachter}}(s) = \mathbf{x} + \frac{s - f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|_2^2} \nabla f(\mathbf{x})$, where s is the target logit score. More specifically, to arrive at the desired class with probability of 0.5, the target score for a sigmoid function is $s = 0$, where the logit corresponds to a 0.5 probability for $y = 1$. The next statement quantifies the IR of recourses output by Wachter et al. (2018).

Proposition 1 (Exact Recourse IR). *For logistic regression, consider the recourse output by Wachter et al. (2018): $\check{\mathbf{x}}_{Wachter}(s) = \mathbf{x} + \frac{s - f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|_2^2} \nabla f(\mathbf{x})$. Then the recourse invalidation rate is given by:*

$$\Delta(\check{\mathbf{x}}_{Wachter}(s); \sigma^2 \mathbf{I}) = 1 - \Phi\left(\frac{s}{\sigma \|\nabla f(\mathbf{x})\|_2}\right), \tag{5}$$

where s is the target logit score.

A recourse generated by Wachter et al. (2018) such that $f(\check{\mathbf{x}}_{\text{Wachter}}) = s = 0$ will result in $\Delta = 0.5$. To obtain recourse that is more robust to noisy responses from users, i.e., $\Delta \rightarrow 0$, the decision maker can choose a higher logit target score of $s' > s \geq 0$ since this decreases the recourse invalidation rate, i.e., $\Delta(\check{\mathbf{x}}_{\text{Wachter}}(s)) > \Delta(\check{\mathbf{x}}_{\text{Wachter}}(s'))$. The next statement makes precise how s should be chosen to achieve a desired robustness level.

Corollary 1. *Under the conditions of Proposition 1, choosing $s_r = \sigma \|\nabla f(\mathbf{x})\|_2 \Phi^{-1}(1-r)$ guarantees a recourse invalidation rate of r, i.e., $\Delta(\check{\mathbf{x}}_{Wachter}(s_r); \sigma^2 \mathbf{I}) = r$.*

On extensions to general noise distributions, and tree-based classifiers. In Appendix A we present extensions of our framework to obtain (i) reliable recourses for general noise distributions and (ii) tree-based classifiers. These two cases pose non-trivial difficulties as the recourse invalidation rate is generally non-differentiable. As for the more general noise distributions, we develop a Monte-Carlo approach in appendix A.1, which relies on a differentiable approximation of the indicator function required to obtain a Monte-Carlo estimate of the invalidation rate. For tree-based classifiers, we develop a closed-form solution for the recourse invalidation rate (see Theorem 2).

4.3 Additional Theoretical Results

In this section, we leverage the recourse invalidation rate expression derived in the previous section to theoretically show i) that an additional cost has to be incurred to generate robust recourses in the face of noisy human responses, and ii) we derive a general upper bound on the IR which is applicable to any valid recourse provided by any method with the underlying classifier being a differentiable model.

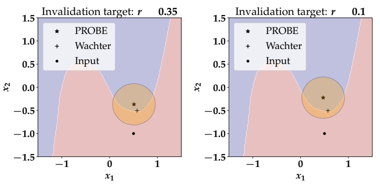


Figure 3: Navigating between high and low invalidation recourses. The circles around `PROBE`'s recourses have radius 2σ, i.e., this is the region where 95% of recourse inaccuracies fall when $\sigma^2 = 0.05$. For instance, on the left we set an invalidation target of $r = 0.35$, i.e., 35% of the recourse responses would fail under spherical inaccuracies $\varepsilon \sim \mathcal{N}(\mathbf{0}, 0.05 \cdot \mathbf{I})$.

Next, we show that there exists a tradeoff between robustness to noisy human responses and cost. To this end, we fix the target invalidation rate r, and ask what costs are needed to achieve a fixed level r:

Proposition 2 (General Cost of Recourse). *For a linear classifier, let $r \in (0, 1)$ and let $\check{\mathbf{x}}_E = \mathbf{x} + \boldsymbol{\delta}_E$ be the output produced by some recourse method E such that $h(\check{\mathbf{x}}_E) = 1$. Then the cost required to achieve a fixed invalidation target r is:*

$$\|\boldsymbol{\delta}_E\|_2 = \frac{\sigma}{\omega}\left(\Phi^{-1}(1-r) - c\right), \tag{6}$$

where $c = \frac{f(\mathbf{x})}{\sigma \cdot \|\nabla f(\mathbf{x})\|_2}$ is a constant, and $\omega > 0$ is the cosine of the angle between $\nabla f(\mathbf{x})$ and $\boldsymbol{\delta}_E$.

From Proposition 2, we see that the target invalidation rate r decreases as the recourse cost increases for a given uncertainty magnitude σ^2. To make this more precise the next statement demonstrates the cost-robustness tradeoff.

Proposition 3 (Cost-Robustness Tradeoff). *Under the same conditions as in Proposition 2, we have $\frac{\partial \|\boldsymbol{\delta}_E\|_2}{\partial (1-r)} = \frac{\sigma}{\omega} \frac{1}{\phi(\Phi^{-1}(1-r))} > 0$, i.e., an infinitesimal increase in robustness (i.e.,$1-r$) increases the cost of recourse by $\frac{\sigma}{\omega} \frac{1}{\phi(\Phi^{-1}(1-r))}$.*

Now, we derive a general upper bound on the recourse invalidation rate. This bound is applicable to any method E that provides recourses resulting in a positive outcome.

Proposition 4 (Upper Bound)**.** *Let $\check{\mathbf{x}}_E$ be the output produced by some recourse method E such that $h(\check{\mathbf{x}}_E) = 1$. Then, an upper bound on $\tilde{\Delta}$ from equation 4 is given by:*

$$\tilde{\Delta}(\check{\mathbf{x}}_E; \sigma^2 \mathbf{I}) \leq 1 - \Phi\left(c + \frac{\omega}{\sigma} \frac{\|\nabla f(\mathbf{x})\|_2}{\|\nabla f(\check{\mathbf{x}}_E)\|_2} \frac{\|\boldsymbol{\delta}_E\|_1}{\sqrt{\|\boldsymbol{\delta}_E\|_0}} \right), \tag{7}$$

where $c = \frac{f(\mathbf{x})}{\sigma \cdot \|\nabla f(\mathbf{x})\|_2}$, $\boldsymbol{\delta}_E = \check{\mathbf{x}}_E - \mathbf{x}$, and $\omega > 0$ is the cosine of the angle between $\nabla f(\mathbf{x})$ and $\boldsymbol{\delta}_E$.

The right term in the inequality entails that the upper bound depends on the ratio of the ℓ_1 and ℓ_0-norms of the recourse action $\boldsymbol{\delta}_E$ provided by recourse method E. The higher the ℓ_1/ℓ_0 ratio of the recourse actions, the tighter the bound. The bound is tight when $\|\boldsymbol{\delta}_E\|_0$ assumes minimum value i.e., $\|\boldsymbol{\delta}_E\|_0 = 1$ since at least one feature needs to be changed to flip the model prediction.

5 Experimental Evaluation

We now present our empirical analysis. First, we validate our theoretical results on the recourse invalidation rates across various recourse methods. Second, we study the effectiveness of PROBE at finding robust recourses in the presence of noisy human responses.

Real-World Data and Noisy Responses. Regarding real-world data, we use the same data sets as provided in the recourse and counterfactual explanation library CARLA (Pawelczyk et al., 2021). The *Adult* data set Dua & Graff (2017) originates from the 1994 Census database, consisting of 14 attributes and 48,842 instances. The class label indicates whether an individual has an income greater than 50,000 USD/year. The *Give Me Some Credit* (GMC) data set Kaggle-Competition (2011) is a credit scoring data set, consisting of 150,000 observations and 11 features. The class label indicates if the corresponding individual will experience financial distress within the next two years (*SeriousDlqin2yrs* is 1) or not. The *COMPAS* data set Angwin et al. (2016) contains data for more than 10,000 criminal defendants in Florida. It is used by the jurisdiction to score defendant's likelihood of re-offending. The class label indicates if the corresponding defendant is high or low risk for recidivism. All the data sets were normalized so that $\mathbf{x} \in [0,1]^d$. Across all experiments, we add noise $\boldsymbol{\varepsilon}$ to the prescribed recourse $\check{\mathbf{x}}_E$, where $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbf{I})$ and $\sigma^2 = 0.01$.

Methods. We compare the recourses generated by PROBE to four different baseline methods which aim to generate low-cost recourses using fundamentally different principles: AR (-LIME) uses an integer-programming-based objective Ustun et al. (2019), Wachter uses a gradient-based objective (Wachter et al., 2018), DICE uses a diversity-based objectve (Mothilal et al., 2020), and GS is based on a random search algorithm (Laugel et al., 2017). Further, we compare with methods that use adversarial minmax objectives to generate robust recourse (Dominguez-Olmedo et al., 2022; Upadhyay et al., 2021). We used the recourse implementations from CARLA (Pawelczyk et al., 2021). Following Upadhyay et al. (2021), all methods search for counterfactuals over the same set of balance parameters $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$ when applicable.

Prediction Models. For all data sets, we trained both ReLU-based NN models with 50 hidden layers (App. B) and a logistic regerssion (LR). All recourses were generated with respect to these classifiers.

Measures. We consider three measures in our evaluation: 1) We measure the *average cost* (AC) required to act upon the prescribed recourses where the average is taken with respect to all instances in the test set for which a given method provides recourse. Since all our algorithms are optimizing for the ℓ_1-norm we use this as our cost measure. 2) We use *recourse accuracy* (RA) defined as the fraction of instances in the test set for which acting upon the prescribed recourse results in the desired prediction. 3) We compute the *average IR* across every instance in the test set. To do that, we sample 10,000 points from $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ for every instance and compute IR in equation 2. Then the *average IR* quantifies recourse robustness where the individual IRs are averaged over all instances from the test set for which a given method provides recourse.

5.1 Validating our Theoretical Bounds

Computing Bounds. We empirically validate the theoretical upper bounds derived in Section 4.3. To do that, we first estimate the bounds for each instance in the test set according to Proposition 4,

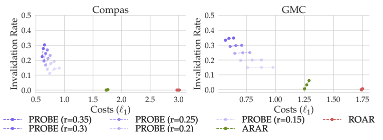
		Adult				Compas				GMC			
	Measures	AR	Wachter	GS	PROBE	AR	Wachter	GS	PROBE	AR	Wachter	GS	PROBE
LR	RA (↑)	0.98	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	AIR (↓)	0.5 ± 0.01	0.46 ± 0.02	0.35 ± 0.11	**0.34 ± 0.02**	0.48 ± 0.04	0.47 ± 0.02	0.3 ± 0.18	**0.28 ± 0.02**	0.47 ± 0.06	0.45 ± 0.03	0.48 ± 0.04	**0.24 ± 0.01**
	AC (↓)	**0.55 ± 0.4**	0.62 ± 0.43	2.12 ± 1.05	1.56 ± 0.92	**0.16 ± 0.17**	0.22 ± 0.17	0.73 ± 0.45	0.63 ± 0.39	**0.29 ± 0.27**	0.49 ± 0.51	**0.28 ± 0.31**	0.60 ± 0.56
NN	RA (↑)	0.38	**1.0**	**1.0**	**0.99**	0.84	**1.0**	**1.0**	**1.0**	0.38	**1.0**	**1.0**	**1.0**
	AIR (↓)	0.49 ± 0.03	0.5 ± 0.02	0.48 ± 0.02	**0.35 ± 0.01**	0.34 ± 0.09	0.46 ± 0.02	0.43 ± 0.07	**0.33 ± 0.02**	0.34 ± 0.07	0.43 ± 0.03	0.45 ± 0.03	**0.25 ± 0.03**
	AC (↓)	1.05 ± 0.22	**0.3 ± 0.19**	2.99 ± 1.51	1.43 ± 0.49	1.15 ± 0.52	**0.2 ± 0.16**	0.81 ± 0.45	0.8 ± 0.34	0.2 ± 0.19	0.26 ± 0.18	**0.12 ± 0.09**	0.47 ± 0.21

(a) Comparing PROBE to baseline recourse methods.

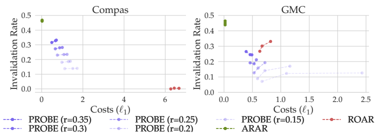
		Adult			Compas			GMC		
	Measures	ROAR	ARAR	PROBE	ROAR	ARAR	PROBE	ROAR	ARAR	PROBE
LR	RA(↑)	**1.0**	**1.0**	**1.0**	**1.0**	0.99	**1.0**	**1.0**	**1.0**	**1.0**
	AIR (↓)	**0.0 ± 0.0**	0.02 ± 0.01	0.34 ± 0.02	**0.0 ± 0.0**	**0.0 ± 0.0**	0.28 ± 0.02	**0.0 ± 0.0**	0.35 ± 0.01	0.24 ± 0.01
	AC (↓)	3.56 ± 0.8	2.68 ± 0.79	**1.56 ± 0.92**	2.99 ± 0.31	1.74 ± 0.3	**0.63 ± 0.39**	1.74 ± 0.45	1.27 ± 0.45	**0.60 ± 0.56**
NN	RA(↑)	0.94	0.03	**0.99**	0.97	0.02	**1.0**	0.06	0.06	**1.0**
	AIR (↓)	**0.0 ± 0.0**	0.51 ± 0.0	0.35 ± 0.01	**0.01 ± 0.06**	0.46 ± 0.0	0.33 ± 0.02	0.3 ± 0.21	0.45 ± 0.01	**0.25 ± 0.03**
	AC (↓)	19.8 ± 3.39	0.04 ± 0.0*	**1.43 ± 0.49**	6.41 ± 1.07	0.02 ± 0.0*	**0.8 ± 0.34**	0.67 ± 0.94	0.02 ± 0.0*	**0.47 ± 0.21**

(b) Comparing PROBE to adversarially robust recourse methods.

Table 1: Comparing PROBE to recourse methods from literature using recourse accuracy (RA), average recourse invalidation rate (AIR) for $\sigma^2 = 0.01$ and average cost (AC) across different recourse methods. For PROBE, we generated recourses by setting $r = 0.35$ and $\sigma^2 = 0.01$. (a): Recourses that use our framework PROBE are more robust compared to those produced by existing baselines. (b): Adversarially robust recourses are more costly than recourses output by PROBE. For ARAR and ROAR we set $\epsilon = 0.01$. *: Results with recourse accuracies less than 10% have not been considered.



(a) Logistic Regression



(b) Neural Network

Figure 4: Comparing PROBE to adversarially robust recourse methods using pareto plots that show the tradeoff between average costs and average invalidation rate (towards bottom left indicates a better performance). For PROBE, the invalidation target is $r \in \{0.35, 0.3, 0.25, 0.20, 0.15\}$, and we generated recourses by setting $\sigma^2, \epsilon \in \{0.005, 0.01, 0.015\}$. The latter are used for ARAR and ROAR.

and compare them with the empirical estimates of the IR. The empirical IR, in turn, we obtain from Monte-Carlo estimates of the IR in equation 2; we used 10,000 samples to get a stable estimate of IR.

Results. In Figure 5, we validate the bounds obtained in Proposition 4 for the GMC data sets. We relegated results for the Compas and Adult data set and other values of σ^2 to Appendix C. Note that the trivial upper bound is 1 since $\Delta \leq 1$, and we see that our bounds usually lie well below this value, which suggests that our bounds are meaningful. We observe that these upper bounds are quite tight, thus providing accurate estimates of the worst case recourse invalidation rates. It is noteworthy that GS tends to provide looser bounds, since its recourses tend to have lower ℓ_1/ℓ_0 ratios; for GS, its random search procedure increases the ℓ_0-norms of the recourse relative to the recourses output by other recourse methods. This contributes to a looser bound saying that the randomly sampled recourses by GS tend to provide looser worst-case IR estimates relative to all the other methods, which do use gradient information (e.g., Wachter , AR and PROBE).

5.2 Evaluating the PROBE Framework

Results. Here, we evaluate the robustness, costs and recourse accuracy of the recourses generated by our framework PROBE relative to the baselines. We consider a recourse robust if the recourse remains valid (i.e., results in positive outcome) even after small changes are made to it (i.e., humans implement it in a noisy manner). Table 1 shows the average IR for different methods across different real world data sets and classifiers when $\sigma^2 = 0.01$. Further, in Table 1a we see that PROBE has the lowest invalidation rate across all real-world data sets and classifiers among the non-robust recourse methods, while PROBE provides the lowest cost recourses among the robust recourse methods (see

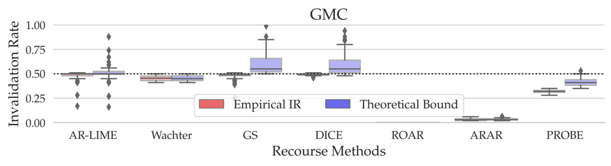


Figure 5: Verifying the theoretical upper bound from Proposition 4 on the logistic regression model. The red boxplots show the empirical recourse invalidation rates for `AR(-LIME)`, `Wachter`, `GS`, `DICE`, `ARAR` ($\epsilon = 0.01$), `ROAR` ($\epsilon = 0.01$) and `PROBE` ($r = 0.35, \sigma^2 = 0.01$). The blue boxplots show the distribution of upper bounds evaluated by plugging in the corresponding quantities (i.e., σ^2, ω, etc.) into the bound. The results show no violations of our theoretical bounds. See appendix C for the full set of results.

Table 1b). We also consider if the robustness achieved by our framework is coming at an additional cost i.e., by sacrificing recourse accuracy (RA) or by increasing the average recourse cost (AC). We compute AC of the recourses output by all the algorithms and find that `PROBE` usually has the highest or second highest recourse costs, while the RA is at 100% across classifiers and data sets.

Finally, we provide a more detailed comparison between `PROBE` and the adversarially robust recourse methods `ARAR` and `ROAR`. To do so, we plot pareto frontiers in Figure 4 which demonstrate the inherent tradeoffs between the average cost of recourse and the average recourse invalidation rate computed over all recousre seeking individuals for different uncertainty magnitudes $\sigma^2, \epsilon \in \{0.005, 0.01, 0.15\}$. For `ARAR` and `ROAR` we expect to see AIRs close to 0 (by construction). However, this is only the case for the linear classifiers. Moreover, `ROAR` provide recourses with up to 3 times higher cost relative to our method `PROBE`. Note also that `ARAR` and `ROAR` have trouble finding recourses for non-linear classifiers, resulting in RA scores of around 5% in the worst case, while not being able to maintain low invalidation scores. This is likely due to the local linear approximation used by these methods. In summary, `PROBE` finds recourses for 100% of the test instances in line with the promise of having an invalidation probability of at most r, while being less costly than `ROAR` and `ARAR`.

Relegated results. The relegated experiments in Appendix C (i) demonstrate that baseline recourse methods are not robust to noisy human responses (Figures 8 - 9), (ii) verify that the targeted invalidation rates match the empirical recourse invalidation rates (Figures 13 - 15) and (iii) demonstrate the trade-off between recourse costs and robustness verifying Corollary 3 (Figures 16 - 17).

6 Conclusion

In this work, we proposed a novel algorithmic framework called Probabilistically ROBust rEcourse (PROBE) which enables end users to effectively manage the recourse cost vs. robustness trade-offs by letting users choose the probability with which a recourse could get invalidated (recourse invalidation rate) if small changes are made to the recourse i.e., the recourse is implemented somewhat noisily. To the best of our knowledge, this work is the first to formulate and address the problem of enabling users to navigate the trade-offs between recourse costs and robustness. Our framework can ensure that a resulting recourse is invalidated at most $r\%$ of the time when it is noisily implemented, where r is provided as input by the end user requesting recourse. To operationalize this, we proposed a novel objective function which simultaneously minimizes the gap between the achieved (resulting) and desired recourse invalidation rates, minimizes recourse costs, and also ensures that the resulting recourse achieves a positive model prediction. We developed novel theoretical results to characterize the recourse invalidation rates corresponding to any given instance w.r.t. different classes of underlying models (e.g., linear models, tree based models etc.), and leveraged these results to efficiently optimize the proposed objective. Experimental evaluation with multiple real world datasets not only demonstrated the efficacy of the proposed framework, but also validated our theoretical findings. Our work also paves the way for several interesting future research directions in the field of algorithmic recourse. For instance, it would be interesting to build on this work to develop approaches which can generate recourses that are simultaneously robust to noisy human responses, noise in the inputs, as well as shifts in the underlying models.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their insightful feedback. This work is supported in part by the NSF awards #IIS-2008461 and #IIS-2040989, and research awards from Google, JP Morgan, Amazon, Bayer, Harvard Data Science Initiative, and D^3 Institute at Harvard. HL would like to thank Sujatha and Mohan Lakkaraju for their continued support and encouragement. The views expressed here are those of the authors and do not reflect the official policy or position of the funding agencies.

REFERENCES

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks, 2016.

Javier Antorán, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. Getting a clue: A method for explaining uncertainty estimates. In *International Conference on Learning Representations (ICLR)*, 2021.

André Artelt, Valerie Vaquet, Riza Velioglu, Fabian Hinder, Johannes Brinkrolf, Malte Schilling, and Barbara Hammer. Evaluating robustness of counterfactual explanations. *arXiv:2103.02354*, 2021.

Daniel Björkegren, Joshua E. Blumenstock, and Samsun Knight. Manipulation-proof machine learning. *arXiv:2004.03865*, 2020.

Emily Black, Zifan Wang, Matt Fredrikson, and Anupam Datta. Consistent counterfactuals for deep models. *arXiv:2110.03109*, 2021.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *arXiv:2110.01889*, 2021.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2006.

Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*, pp. 448–469. Springer, 2020.

Pedro Domingos. Knowledge acquisition from examples via multiple models. In *14th International Conference on Machine Learning*, pp. 98–106, 1997.

Ricardo Dominguez-Olmedo, Amir-Hossein Karimi, and Bernhard Schölkopf. On the adversarial robustness of causal algorithmic recourse. In *International Conference on Machine Learning (ICML)*, 2022.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015.

Kaggle-Competition. "give me some credit data set", 2011.

Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020a.

Amir-Hossein Karimi, Julius von Kügelgen, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020b.

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Inverse classification for comparison-based interpretability in machine learning. *arXiv preprint arXiv:1712.08443*, 2017.

Ana Lucic, Harrie Oosterhuis, Hinda Haned, and Maarten de Rijke. Focus: Flexible optimizable counterfactual explanations for tree ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

Divyat Mahajan, Chenhao Tan, and Amit Sharma. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277*, 2019.

Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv:1912.01703*, 2019.

Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. Learning model-agnostic counterfactual explanations for tabular data. In *Proceedings of The Web Conference 2020 (WWW)*. ACM, 2020a.

Martin Pawelczyk, Klaus Broelemann, and Gjergji. Kasneci. On counterfactual explanations under predictive multiplicity. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 809–818. PMLR, 2020b.

Martin Pawelczyk, Sascha Bielawski, Johan Van den Heuvel, Tobias Richter, and Gjergji Kasneci. Carla: A python library to benchmark algorithmic recourse and counterfactual explanation algorithms. In *Advances in Neural Information Processing Systems (NeurIPS) (Benchmark and Datasets Track)*, volume 34, 2021.

Martin Pawelczyk, Chirag Agarwal, Shalmali Joshi, Sohini Upadhyay, and Himabindu Lakkaraju. Exploring counterfactual explanations through the lens ofadversarial examples: A theoretical and empirical analysis. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

Martin Pawelczyk, Tobias Leemann, Asia Biega, and Gjergji Kasneci. On the Trade-Off between Actionable Explanations and the Right to be Forgotten. In *International Conference on Learning Representations (ICLR)*, 2023.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 2011.

Mary Phuong and Christoph Lampert. Towards understanding knowledge distillation. In *International Conference on Machine Learning (ICML)*. PMLR, 2019.

Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C Duchi, and Percy Liang. Adversarial training can hurt generalization. *arXiv preprint arXiv:1906.06032*, 2019.

Kaivalya Rawal and Himabindu Lakkaraju. Interpretable and interactive summaries ofactionable recourses. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Kaivalya Rawal, Ece Kamar, and Himabindu Lakkaraju. Algorithmic recourse in the wild: Understanding the impact of data and model shifts. *arXiv:2012.11788*, 2021.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (KDD)*, pp. 1135–1144, 2016.

Dylan Slack, Sophie Hilgard, Himabindu Lakkaraju, and Sameer Singh. Counterfactual explanations can be manipulated. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Thomas Spooner, Danial Dervovic, Jason Long, Jon Shepard, Jiahao Chen, and Daniele Magazzeni. Counterfactual explanations for arbitrary regression models. *arXiv:2106.15212*, 2021.

Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. ACM, 2017.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

Sohini Upadhyay, Shalmali Joshi, and Himabindu Lakkaraju. Towards robust and reliable algorithmic recourse. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

Berk Ustun, Alexander Spangher, and Y. Liu. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2019.

Arnaud Van Looveren and Janis Klaise. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584*, 2019.

Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv:2010.10596*, 2020.

Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10:3152676, 2017.

Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: automated decisions and the gdpr. *Harvard Journal of Law & Technology*, 31(2), 2018.

A Extensions to Other Noise Distributions and Tree Based Classifiers

A.1 Extensions to general noise distributions

A.1.1 A Monte-Carlo approach for general noise distributions

Algorithm 2 PROBE-MC

```
Input: $\mathbf{x}$ s.t. $f(\mathbf{x}) < 0$, $f$, $\sigma^2$, $\lambda > 0$, $t, \alpha, r > 0$
Init.: $\mathbf{x}' = \mathbf{x}$;
Compute $\hat{\Delta}_{\mathrm{MC}}(\mathbf{x}')$    ▷ from equation 11
while $\hat{\Delta}_{\mathrm{MC}}(\mathbf{x}') > r$ and $f(\mathbf{x}') < 0$ do
    Compute $\hat{\Delta}_{\mathrm{MC}}(\mathbf{x}')$    ▷ from equation 11
    $\mathbf{x}' = \mathbf{x}' - \alpha \cdot \nabla_{\mathbf{x}'} \mathcal{L}(\mathbf{x}'; \sigma^2, r, \lambda)$
                                        ▷ Opt. equation 3
end while
Return: $\check{\mathbf{x}} = \mathbf{x}'$
```

In section 4 we have introduced our PROBE framework, which enables us to guide the search for counterfactual explanations towards regions with a targeted low invalidation rate. Recall that the optimization procedure in Section 4 relied on a first-order approximation to the recourse invalidation rate under Gaussian distributed noisy human responses. In this section, we develop an algorithm that is agnostic to the specifics of the parameterized noise distribution. To this end, we suggest a Monte Carlo estimator of the recourse IR from Def. 1, i.e.,

$$\tilde{\Delta}_{\mathrm{MC}} = \frac{1}{K} \sum_{k=1}^{K} (1 - h(\mathbf{x}' + \boldsymbol{\varepsilon}_k)). \tag{8}$$

We highlight that the estimator $\tilde{\Delta}_{\mathrm{MC}}$ allows for a flexible specification of various noise distributions, and thus does not depend on specific distributional assumptions of $\boldsymbol{\varepsilon}$. The following result suggests that we can estimate the true IR $\Delta(\mathbf{x}')$ to desired precision using the Monte-Carlo estimator $\tilde{\Delta}_{\mathrm{MC}}(\mathbf{x}')$.

Proposition 5. *The mean-squared-error (MSE) between the true IR $\Delta(\mathbf{x}')$ and the empirical Monte-Carlo estimate $\tilde{\Delta}_{MC}(\mathbf{x}')$ is upper bounded such that:*

$$\mathbb{E}_{\boldsymbol{\varepsilon}}\Big[(\Delta(\mathbf{x}') - \tilde{\Delta}_{MC}(\mathbf{x}'))^2\Big] \leq \frac{1}{4K}. \tag{9}$$

Since it is up to us to choose K, we can make the MSE arbitrarily small and reliably estimate the true invalidation rate $\Delta(\mathbf{x}')$.

A.1.2 A differentiable approximation to $\tilde{\Delta}_{\mathrm{MC}}$

A problem with the estimator $\tilde{\Delta}_{\mathrm{MC}}$ is that it is not amenable to automatic differentiation required for our gradient based algorithm to operate. This is due to the discontinuity at the threshold ξ introduced by the indicator function which, in turn, is applied to the logit score when computing the recourse invalidation rate (i.e., $h(\mathbf{x}) = \mathbb{I}(f(\mathbf{x}) > \xi)$ and see Definition 1). To mitigate this issue, we suggest to use a sigmoid function with appropriate temperature t to approximate the indicator at the threshold ξ:

$$S\big((x-\xi)\cdot t\big) = \frac{1}{1+\exp\big(-(x-\xi)\cdot t\big)}. \tag{10}$$

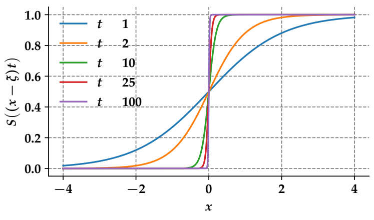


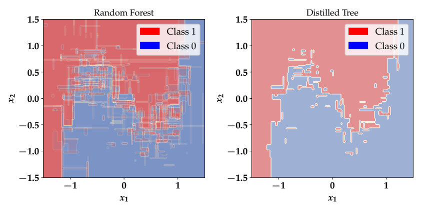
Figure 6: Differentiable approximations of the indicator function $\mathbb{I}(x > 0)$ using the sigmoid function $S(y) = \frac{1}{1+\exp(-y)}$ evaluated at different temperatures $t \in \{1, 2, 10, 25, 100\}$ when $\xi = 0$.

Therefore, as $t \to \infty$ the sigmoid S converges to the indicator function $\mathbb{I}(x > \xi)$. We illustrate this behaviour in Figure 6 for different temperature levels $t \in \{1, 2, 10, 25, 100\}$ when the threshold is $\xi = 0$. Using the differentiable approximation to the indicator function, we are now ready to state a differentiable estimator for the recourse invalidation rate, which we can use to guide our gradient descent procedure to low recourse invalidation regions:

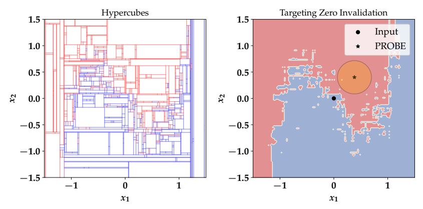
$$\hat{\Delta}_{\mathrm{MC}}(\mathbf{x}'; 0, t) = \frac{1}{K} \sum_{k=1}^{K} \Big(1 - S\big(t \cdot f(\mathbf{x}' + \boldsymbol{\varepsilon}_k)\big)\Big). \tag{11}$$

A.2 Extensions to tree based classifiers

The recourse literature commonly considers consequential decision problems which heavily rely on the usage of tabular data. For this data modality, ensembles of decision trees such as Random Forest (RF) (Breiman, 2001) or Gradient Boosted Boosted Decision Trees (GBDT) (Friedman, 2001) are considered among the state-of-the-art models (Borisov et al., 2021). As a consequence, some recourse methods were developed to find recourses for tree ensembles (Tolomei et al., 2017; Lucic et al., 2022) where the non-differentiability prevents a direct application of the recourse objective in equation 1. To extend our method to tree-based classifiers, we also derive an IR expression for tree ensembles, and develop a method which computes low IR recourses for these models.



(a) Distilling a RF classifier (left) into a single tree (right). In the left panel, the RF classifier averages 30 decision trees, indicating that the final axis-aligned regions (not shown) are complicated functions of all 30 decision trees.



(b) Computing recourse for the RF model (right) based on the hypercubes (left). The circle has radius 2σ, i.e., it shows the region where 95% of recourse inaccuracies fall when $\sigma^2 = 0.025$. The input $\mathbf{x}$ has IR ≈ 0.5. The CE has IR ≈ 0.05.

Figure 7: Computing certified recourses on the 2d Moon data set (Pedregosa et al., 2011) for a RF classifier. **Figure a)**: Distilling a RF classifier (left panel) into a single decision tree (right panel) using knowledge distillation (Domingos, 1997). **Figure b)**: Using the distilled tree, we form the hypercubes (left panel) required to compute IR according to Theorem 2. We then optimize equation 3 to find certified recourses for the RF model (right panel).

Tree Ensemble Classifiers An object of interest is the predicted output of a decision tree:

$$\mathcal{T}(\mathbf{x}) = \sum_{R \in \mathcal{R}_{\mathcal{T}}} c_{\mathcal{T}}(R) \cdot \mathbb{I}(\mathbf{x} \in R), \tag{12}$$

where $c_{\mathcal{T}}(R) \in \{0, 1\}$ is the constant prediction assigned in region $R \in \mathcal{R}_{\mathcal{T}}$ for tree $\mathcal{T}$. Moreover, a decision forest is formed by a set of M_T decision trees, and forms the probabilistic output:

$$f_{\text{Forest}}(\mathbf{x}) = \frac{1}{M_T} \sum_{m=1}^{M_T} \mathcal{T}_m(\mathbf{x}). \tag{13}$$

The predicted class of an input $\mathbf{x}$ is formed via a vote by the trees where each tree assigns a probability estimate to the input. That is, the predicted class is the one with highest mean probability estimate across the trees. After the trees are combined, the multiple models form a single model again (Domingos, 1997). Thus, the corresponding predicted class of equation 13 is given by:

$$\mathcal{F}(\mathbf{x}) = \sum_{R \in \mathcal{R}_{\mathcal{F}}} c_{\mathcal{F}}(R) \cdot \mathbb{I}(\mathbf{x} \in R), \tag{14}$$

where $c_{\mathcal{F}}(R) \in \{0, 1\}$ is the constant prediction assigned in region $R \in \mathcal{R}_{\mathcal{F}}$ for the ensemble of trees $\mathcal{F}$. Furthermore, note that for each ensemble, there is an active subset of ensemble-specific features $\mathcal{S}_{\mathcal{F}} \subseteq \{1, \ldots, d\}$ on which axis-aligned splits took place. Finally, we note that this formulation is quite general as it subsumes a large class of popular tree-based models such as Random Forests (RF) and Gradient Boosted Decision Trees (GBDT).

A.3 The Recourse IR for Tree Ensemble Classifiers

Theorem 2 (IR for Tree-Ensemble Classifiers)**.** *Consider the decision forest classifier in equation 14. The recourse invalidation rate under Gaussian distributed response inconsistencies* $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma}^2\mathbf{I})$ *is* *given by:*

$$\Delta(\check{\mathbf{x}}_E;\mathbf{\Sigma}) = 1 - \sum_{R\in\mathcal{R}_{\mathcal{F}}} c_{\mathcal{F}}(R) \prod_{j\in\mathcal{S}_{\mathcal{F}}} d_{j,R}(\check{x}_{E,j}), \tag{15}$$

where

$$d_{j,R}(\check{x}_{E,j}) = \left[\Phi\left(\frac{\bar{t}_{j,R} - \check{x}_{E,j}}{\sigma_j}\right) - \Phi\left(\frac{\underline{t}_{j,R} - \check{x}_{E,j}}{\sigma_j}\right)\right], \tag{16}$$

and where Φ is the Gaussian CDF, $\bar{t}_{j,R}$ and $\underline{t}_{j,R}$ are the upper and lower points corresponding to feature $j \in \mathcal{S}_{\mathcal{F}}$ that define the hypercube formed by region R.

Proof Sketch. The proof uses the insight that a decision forest based on trees with axis-aligned splits partions the input space into hypercubes where the prediction is either 0 or 1. It then remains to evaluate Gaussian integrals subject to the constrains set by the hypercubes. The full proof is given in Appendix D.3. □

Our proof of Theorem 2 assumed that the split points $\bar{t}_{j,R}$ and $\underline{t}_{j,R}$, corresponding to the tree-ensemble, are readily available. However, the hypercubes formed by the tree-ensemble, for which the prediction is constant, is a function of all individual trees, and of how they are combined. Thus, the clear-cut division into hypecrubes present in each of the trees got lost in the process of model averaging.

Model Distillation to Evaluate IR We suggest a solution to this problem by using a technique called *model distillation* (Domingos, 1997; Bucilua et al., 2006; Hinton et al., 2015; Phuong & Lampert, 2019). In a nutshell: We wish to change the form of the model (to a simpler decision tree) while keeping the same knowledge (from our tree ensemble) (Hinton et al., 2015). Thus, the goal of this technique is to distil the knowledge of a larger model (possibly an ensemble) into a single, small (and interpretable) model. In our case, the ensemble is formed by decision trees, and the target model is a decision tree as well. Second, the method is simple to operationalize: let h be your complex model, and g denotes the simple model. Then we use our data $\{\mathbf{x}_i, y_i\}_{i=1}^n$ to train and validate the model h. The target model, however, is trained on samples from $\{\mathbf{x}_i, h(\mathbf{x}_i)\}_{i=1}^n$ to mimic the behaviour of the complex model. We refer to panels 1 to 3 in Figure 7 to gain some intuition on how this technique works on a non-linear 2-dimensional data set.

B Experimental Details

In this section, we describe the hyperparameter choices and how the classification models were fitted. We have used `CARLA`'s built-in functionality to fit classifiers using PyTorch (Paszke et al., 2019) and treat all variables as continuous. We set $\lambda_1 = 2$, $\lambda_2 = 1$ and search over λ_3 in the usual way (Wachter et al., 2018). All models use a 80 – 20 train-test split for model training and evaluation. We evaluate model quality based on the model accuracy. All models are trained with the same architectures across the data sets:

	Neural Network	Logistic Regression
Units	[Input dim., 50, 2]	[Input dim. , 2]
Type	Fully connected	Fully connected
Intermediate activations	ReLU	N/A
Last layer activations	Softmax	Softmax

Table 2: Classification model details

		Adult	COMPAS	Give Me Credit
Batch-size	NN	512	32	64
	Logistic Regression	512	32	64
Epochs	NN	50	40	30
	Logistic Regression	50	40	30
Learning rate	NN	0.002	0.002	0.001
	Logistic Regression	0.002	0.002	0.001

Table 3: Training details

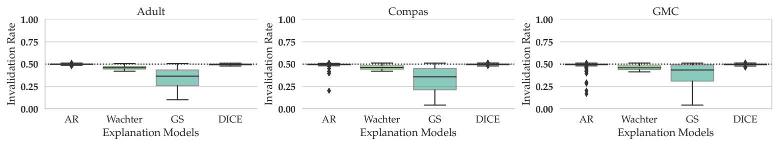
	Adult	COMPAS	Give Me Credit
Logistic Regression	0.83	0.84	0.92
Neural Network	0.85	0.85	0.93

Table 4: Performance of models used for generating recourses

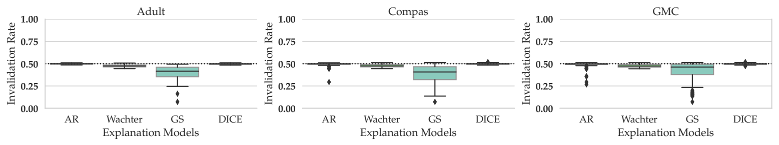
C Additional Experiments

C.1 Algorithmic Recourse in the Face of Noisy Human Responses

In this Section we show a set of additional experiments. Since this work is the first to highlight and address the problem of recourse invalidation in the face of noisy human responses, we demonstrate in Figures 8 and 9 that recourses generated by state-of-the-art approaches are, on average, invalidated up to 50% of the time when small changes are made to them. It is worth highlighting that the maximum invalidation scores can become as high as 61%, which motivates the need for a recourse method that rightly controls the invalidation rate.

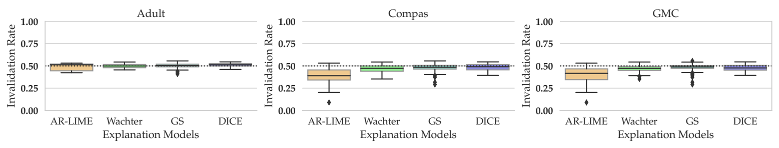


(a) $\sigma^2 = 0.01$

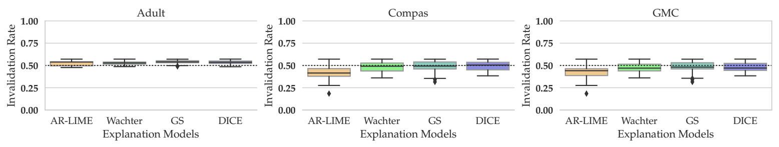


(b) $\sigma^2 = 0.025$

Figure 8: Boxplots of recourse invalidation probabilities across sucessfully generated recourses $\check{\mathbf{x}}$ for **logistic regression** on three data sets. Recourses were generated by four different explanation methods (i.e., `AR`, `Wachter`, and `GS`, `DICE`), which use different techniques (i.e., integer programming, gradient search, random search, diverse recourse) to find minimum cost recourses. We perturbed the recourses by adding small normally distributed response inaccuracies $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbf{I})$ to $\check{\mathbf{x}}$.



(a) σ^2 = 0.01

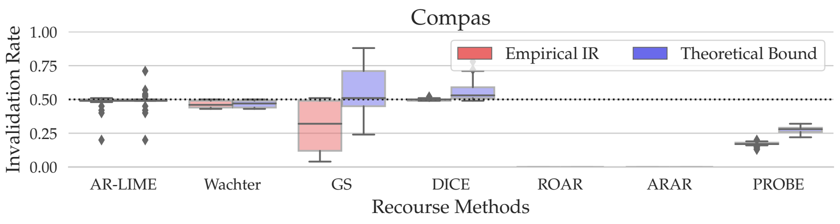


(b) σ^2 = 0.025

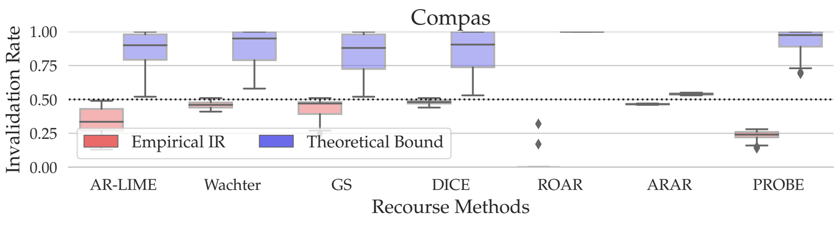
Figure 9: Boxplots of recourse invalidation probabilities across sucessfully generated recourses $\check{\mathbf{x}}$ for **NN classifiers** on three data sets. The recourses were generated by four different explanation methods (i.e., `AR`, `Wachter`, and `GS`, `DICE`), which use different techniques (i.e., integer programming, gradient search, random search, diverse recourse) to find minimum cost recourses. We perturbed the recourses by adding small normally distributed response inaccuracies $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot \mathbf{I})$ to $\check{\mathbf{x}}$.

C.2 Missing Figures from the Main Text

Below, we show the Figure that was missing from the main text due to space constraints. To keep the plots below more readable, we have omitted `DICE` from them as both the bounds implied by `DICE`, the results on cost and the remaining measures are similar to the one by `Wachter`.

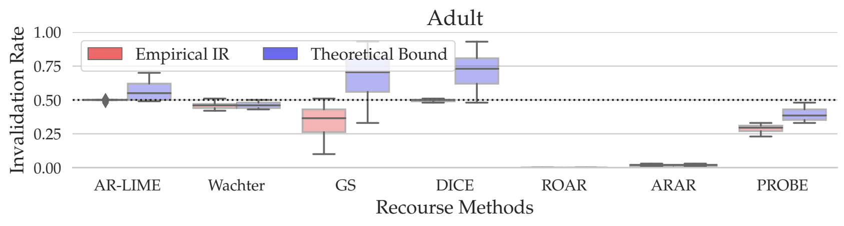


(a) Logistic Regression

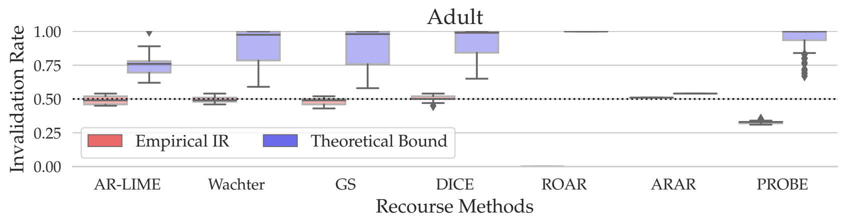


(b) Neural Network

Figure 10: Missing figures from the main text (Compas).

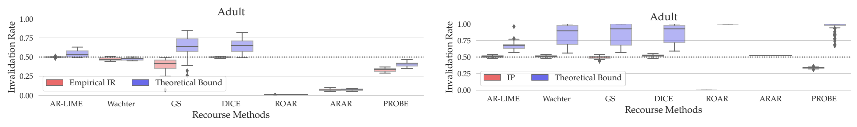


(a) Logistic Regression

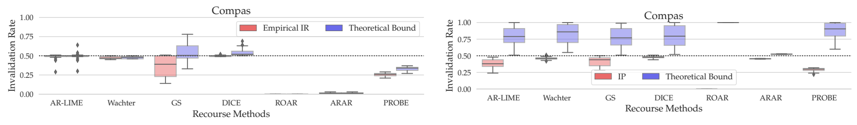


(b) Neural Network

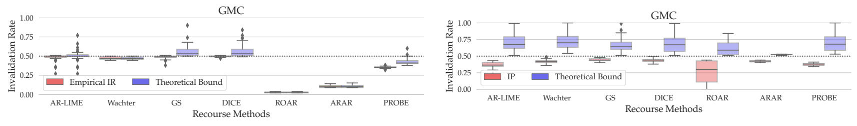
Figure 11: Missing figures from the main text (Adult).



(a) Logistic Regression (Left), ANN (Right), $\sigma^2 = 0.025$



(b) Logistic Regression (Left), ANN (Right), $\sigma^2 = 0.025$



(c) Logistic Regression (Left), ANN (Right), $\sigma^2 = 0.025$

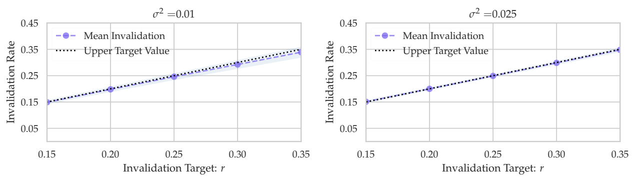
Figure 12: Verifying the theoretical upper bound from Lemma 4 for the logistic regression and artificial neural network classifiers on all data sets when $\sigma^2 = 0.025$. The green boxplots show the empirical recourse IRs for `AR(-LIME)`, `Wachter`, `GS`, and `PROBE`. The blue boxplots show the distribution of upper bounds, which we evaluated by plugging in the corresponding quantities (i.e., σ^2, ω, etc.) into the upper bound from Lemma 4. The results show no violations of our bounds.

		Adult				Compas				GMC			
		AR	Wachter	GS	PROBE	AR	Wachter	GS	PROBE	AR	Wachter	GS	PROBE
LR	RA (↑)	0.98	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	AIR (↓)	0.5 ± 0.01	0.48 ± 0.01	0.4 ± 0.08	0.28 ± 0.02	0.49 ± 0.03	0.48 ± 0.02	0.36 ± 0.14	0.31 ± 0.01	0.48 ± 0.04	0.47 ± 0.02	0.49 ± 0.02	0.3 ± 0.01
	AC (↓)	0.55 ± 0.4	0.62 ± 0.43	2.06 ± 1.03	2.21 ± 3.17	0.16 ± 0.17	0.22 ± 0.17	0.73 ± 0.45	0.68 ± 0.28	0.29 ± 0.27	0.49 ± 0.51	0.28 ± 0.32	1.22 ± 2.29
NN	RA(↑)	0.38	**1.0**	**1.0**	**1.0**	0.84	**1.0**	**1.0**	**1.0**	0.4	**1.0**	**1.0**	**1.0**
	AIR (↓)	0.51 ± 0.02	0.51 ± 0.01	0.5 ± 0.02	0.33 ± 0.01	0.39 ± 0.06	0.46 ± 0.02	0.41 ± 0.07	0.25 ± 0.02	0.37 ± 0.05	0.42 ± 0.03	0.44 ± 0.02	0.34 ± 0.02
	AC (↓)	1.05 ± 0.22	0.3 ± 0.19	3.11 ± 1.62	1.98 ± 2.35	1.15 ± 0.52	0.2 ± 0.16	**1.0** ± 0.17	0.84 ± 0.34	0.2 ± 0.16	0.26 ± 0.18	0.11 ± 0.09	0.41 ± 0.23

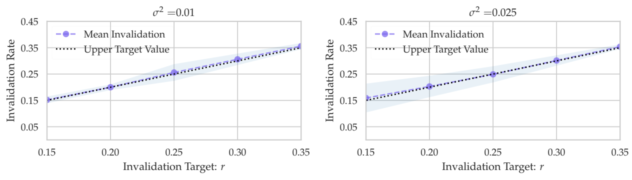
Table 5: Recourse accuracy (RA), average recourse invalidation rate (AIR) for $\sigma^2 = 0.025$ and average cost (AC) across different recourse methods. Recourses that use our framework PROBE are more robust compared to those produced by existing baselines. For PROBE, we generated recourses by setting $r = 0.35$. Thus, the AIR should be at most 0.35, in line with our results.

C.3 Verifying the Validity of the Empirical Invalidation Rate

In Figures 13, 14, and 15 we show that the IRs of the recourses by our framework can be controlled setting r to desired values.



(a) LR



(b) NN

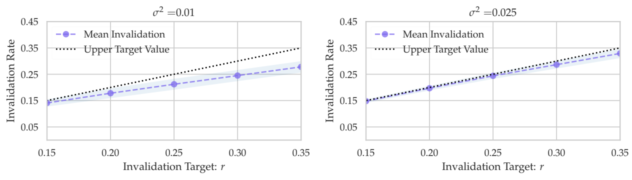
Figure 13: Verifying that the invalidation rate for our framework PROBE (blue line) is at most equal to the invalidation target r on the **Adult** data set for different $\sigma^2 \in \{0.01, 0.025\}$ across both classifiers. We compute the *mean IR* across every instance in the test set. To do that, we sample 10,000 points from $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ for every instance and compute IR in equation 2. Then the *mean IR* quantifies recourse robustness where the individual IRs are averaged over all instances from the test set. The shaded regions indicate the corresponding standard deviations.

C.4 Demonstrating the Cost-Robustness Tradeoff

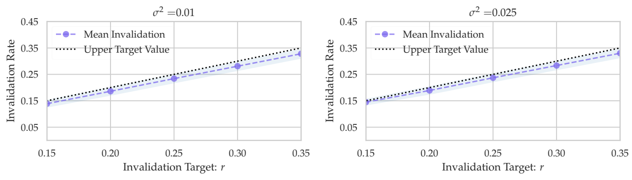
In Figures 16 and 17 we demonstrate that there exists a tradeoff between recourse costs and the robustness of recourse to noisy response.

C.5 Detailed Comparison with ROAR and ARAR

In this section we compare our method with two approaches that aim at generating robust algorithmic recourse in different settings. We further report results by DICE, which does not generate robust recourse. Thus, we PROBE the cost performance (i.e., AC) by DICE to serve as a lower bound, while its robustness performance would serve as an upper bound (i.e., AIR). Regarding the methods that suggest robust recourse we refer to Upadhyay et al. (2021) who proposed a minimax objective to generate recourses that are robust to model updates (ROAR), while Dominguez-Olmedo et al. (2022) use a slight variation of this objective to find recourses that are robust to uncertainty in the

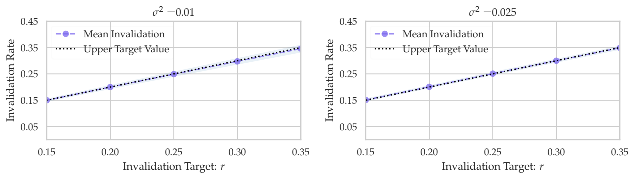


(a) LR

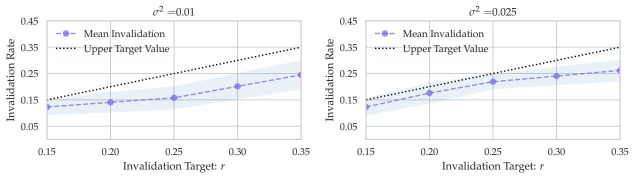


(b) NN

Figure 14: Verifying that the invalidation rate for our framework PROBE (blue line) is at most equal to the invalidation target r on the **Compas** data set for different $\sigma^2 \in \{0.01, 0.025\}$ across both classifiers. We compute the *mean IR* across every instance in the test set. To do that, we sample 10,000 points from $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ for every instance and compute IR in equation 2. Then the *mean IR* quantifies recourse robustness where the individual IRs are averaged over all instances from the test set. The shaded regions indicate the corresponding standard deviations.



(a) LR



(b) NN

Figure 15: Verifying that the invalidation rate for our framework PROBE (blue line) is at most equal to the invalidation target r on the **GMC** data set for different $\sigma^2 \in \{0.01, 0.025\}$ across both classifiers. We compute the *mean IR* across every instance in the test set. To do that, we sample 10,000 points from $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$ for every instance and compute IR in equation 2. Then the *mean IR* quantifies recourse robustness where the individual IRs are averaged over all instances from the test set. The shaded regions indicate the corresponding standard deviations.

inputs (ARAR). Moreover, on a high-level, these objectives differ from our approach since the epsilon neighborhoods that PROBE constructs are probabilistic.

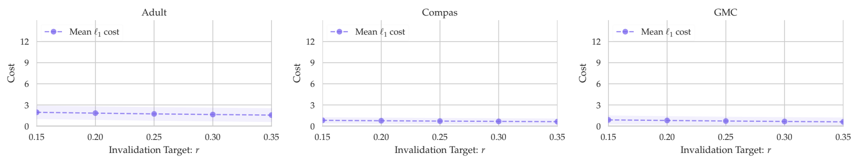


Figure 16: Trading off recourse costs against robustness by choosing the invalidation target r in our PROBE framework. We generated recourses by setting $r \in \{0.20, 0.25, 0.30, 0.35.0.40\}$ and $\sigma^2 = 0.01$ for the **logistic regression classifier**.

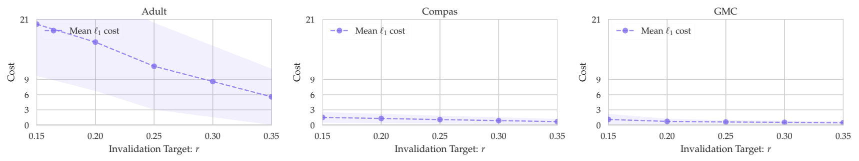


Figure 17: Trading off recourse costs against robustness by choosing the invalidation target r in our PROBE framework. We generated recourses by setting $r \in \{0.20, 0.25, 0.30, 0.35.0.40\}$ and $\sigma^2 = 0.01$ for the **NN classifier**.

Cost versus invalidation rate performances. The table shown below summarizes the performance comparison across the aforementioned methods, and Figures 18 and 19 provide Pareto plots, which demonstrate the tradeoff between the average costs measured in terms of ℓ_1 norm and the average invalidation rate.

Discussion. The AIR for PROBE should be at most 0.35, in line with our results. For ARAR and ROAR, we should expect AIRs close to 0, which is only the case for the linear classifiers. Additionally, ARAR and ROAR provide recourses with up to 10 times higher cost relative to our method PROBE. Note also that ARAR and ROAR have trouble finding recourses for non-linear classifiers, resulting in RA scores of around 5% in the worst case, while not being able to maintain low invalidation scores. This is likely due to the local linear approximation that needs to be used by these methods. For ARAR, only up to 5 percent of all recourse are found (i.e., it only finds recourse with low cost to the decision boundary), and for those identified recourses the average invalidation rate is close to a random coin flip. In summary, PROBE finds recourses for 100% of the test instances in line with the promise of having an invalidation probability of at most 0.35, while being substantially less costly than ROAR.

		Adult			Compas			GMC		
	Measures	ROAR	ARAR	PROBE	ROAR	ARAR	PROBE	ROAR	ARAR	PROBE
LR	RA(↑)	**1.0**	**1.0**	**1.0**	**1.0**	0.99	**1.0**	**1.0**	**1.0**	**1.0**
	AIR (↓)	0.0 ± 0.0	0.02 ± 0.01	0.34 ± 0.02	0.0 ± 0.0	0.0 ± 0.0	0.33 ± 0.02	0.0 ± 0.0	0.35 ± 0.01	0.24 ± 0.01
	AC (↓)	3.56 ± 0.8	2.68 ± 0.79	**1.56 ± 0.92**	2.99 ± 0.31	1.74 ± 0.3	**0.63 ± 0.39**	1.74 ± 0.45	1.27 ± 0.45	**0.60 ± 0.56**
NN	RA(↑)	0.94	0.03	**0.99**	0.97	0.02	**1.0**	0.06	0.06	**1.0**
	AIR (↓)	0.0 ± 0.0	0.51 ± 0.0	0.35 ± 0.01	0.01 ± 0.06	0.46 ± 0.0	0.33 ± 0.02	0.3 ± 0.21	0.45 ± 0.01	**0.25 ± 0.03**
	AC (↓)	19.8 ± 3.39	0.04 ± 0.0	1.43 ± 0.49	6.41 ± 1.07	0.02 ± 0.0	0.8 ± 0.34	0.67 ± 0.94	0.02 ± 0.0	**0.47 ± 0.21**

Table 6: Recourse accuracy (RA), average recourse invalidation rate (AIR) for $\sigma^2 = 0.01$ and average cost (AC) across different recourse methods. Recourses that use our framework PROBE provide a strong recourse-robustness tradeoff. For PROBE, we generated recourses by setting $r = 0.35$, $\sigma^2 = 0.01$. For ROAR and ARAR, we generated recourses by setting $\varepsilon = 0.01$.

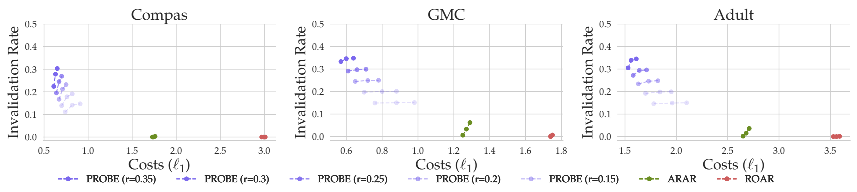


Figure 18: Pareto plots showing the tradeoff between average costs and average invalidation rate when the underlying model is linear. For PROBE, the invalidation target r (dotted line) is set to 0.3, and we generated recourses by setting $\sigma^2 \in \{0.005, 0.01, 0.015\}$, and for ARAR and ROAR we set $\epsilon \in \{0.005, 0.01, 0.015\}$. Following the suggestion by Upadhyay et al. (2021), all recourse methods search for the optimal counterfactuals over the same set of balance parameters $\lambda \in \{0. 0.25. 0.5. 0.75. 1\}$.

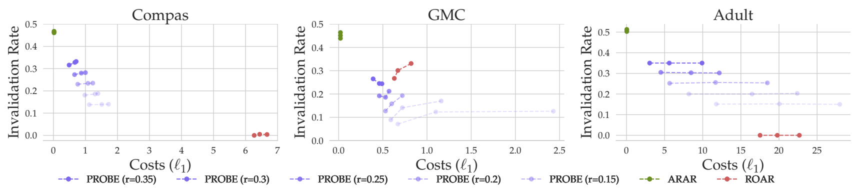


Figure 19: Pareto plots showing the tradeoff between average costs and average invalidation rate when the underlying model is a neural network. For PROBE, the invalidation target r (dooted line) is set to 0.35, and we generated recourses by setting $\sigma^2 \in \{0.005, 0.01, 0.015\}$, and for ARAR and ROAR we set $\epsilon \in \{0.005, 0.01, 0.015\}$. Following the suggestion by Upadhyay et al. (2021), all recourse methods search for the optimal counterfactuals over the same set of balance parameters $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$.

D Proofs

D.1 Proof of Proposition 5

Proposition 5. *The mean-squared-error (MSE) between the true IR $\Delta(\mathbf{x}')$ and the empirical Monte-Carlo estimate $\tilde{\Delta}_{MC}(\mathbf{x}')$ is upper bounded such that:*

$$\mathbb{E}_{\boldsymbol{\varepsilon}}\Big[(\Delta(\mathbf{x}') - \tilde{\Delta}_{MC}(\mathbf{x}'))^2\Big] \leq \frac{1}{4K}. \tag{17}$$

Proof. First, recall that the empirical Monte-Carlo estimator is given by:

$$\tilde{\Delta}_{\text{MC}} = \frac{1}{K}\sum_{k=1}^{K}(1 - h(\mathbf{x}' + \boldsymbol{\varepsilon}_k)). \tag{18}$$

Next, note $\mathbb{E}_{\boldsymbol{\varepsilon}}[1 - h(\mathbf{x}' + \boldsymbol{\varepsilon})] = \Delta(\mathbf{x}')$. Further, the mean-squared error between the nominal invalidation rate $\Delta(\mathbf{x}')$ and the Monte-Carlo estimate $\tilde{\Delta}_{\text{MC}}$ is given by:

$$\mathbb{E}_{\boldsymbol{\varepsilon}}\Big[(\Delta(\mathbf{x}') - \tilde{\Delta}_{\text{MC}})^2\Big] = \mathbb{V}_{\boldsymbol{\varepsilon}}(\tilde{\Delta}_{\text{MC}}) + \mathbb{E}_{\boldsymbol{\varepsilon}}[\tilde{\Delta}_{\text{MC}} - \Delta(\mathbf{x}')]^2, \tag{19}$$

which gives the bias-variance decomposition. We first compute the squared bias term:

$$\mathbb{E}_{\boldsymbol{\varepsilon}}[\tilde{\Delta}_{\text{MC}} - \Delta(\mathbf{x}')]^2 = \left[\frac{1}{K}\cdot K \cdot \mathbb{E}_{\boldsymbol{\varepsilon}}[1 - h(\mathbf{x} + \boldsymbol{\varepsilon})] - \Delta(\mathbf{x}')\right]^2 \tag{20}$$

$$= 0, \tag{21}$$

where we have used that the $\boldsymbol{\varepsilon}$s are identically distributed. We now turn to the variance term for which we find the following expression:

$$\mathbb{V}_{\boldsymbol{\varepsilon}}(\tilde{\Delta}_{\text{MC}}) = \frac{1}{K^2}\cdot K \cdot \mathbb{V}_{\boldsymbol{\varepsilon}}[1 - h(\mathbf{x} + \boldsymbol{\varepsilon})] = \frac{1}{K}\cdot \mathbb{V}_{\boldsymbol{\varepsilon}}[h(\mathbf{x} + \boldsymbol{\varepsilon})]. \tag{22}$$

It remains to identify an upper bound for $\mathbb{V}_{\varepsilon}(h(\mathbf{x}+\varepsilon))$. Since $h(\mathbf{x}+\varepsilon)$ is binary, a simple upper bound is given by:

$$\mathbb{V}_{\varepsilon}(h(\mathbf{x}+\varepsilon)) \leq \frac{1}{4}. \tag{23}$$

Combining the expression for the squared bias and the upper bound on the variance yields the desired result. □

D.2 Proofs of Theorem 1, Propositions 1 - 4 and Corollary 1

Theorem 1. *A first-order approximation $\tilde{\Delta}$ to the recourse invalidation rate Δ in equation 2 under a Gaussian distribution $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ capturing the noise in human responses is given by:*

$$\tilde{\Delta}(\check{\mathbf{x}}_E; \mathbf{\Sigma}) = 1 - \Phi\left(\frac{f(\check{\mathbf{x}}_E)}{\sqrt{\nabla f(\check{\mathbf{x}}_E)^\top \mathbf{\Sigma} \nabla f(\check{\mathbf{x}}_E)}}\right), \tag{24}$$

where Φ is the CDF of the univariate standard normal distribution $\mathcal{N}(0,1)$, $f(\check{\mathbf{x}}_E)$ denotes the logit score at $\check{\mathbf{x}}_E$ which is the recourse output by a recourse method E, and $h(\check{\mathbf{x}}_E) \in \{0,1\}$.

Proof. Let the random variable ε follow a multivariate normal distribution, i.e., $\varepsilon \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$. The following result is a well-known fact: $\mathbf{v}^\top \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{v}^\top \boldsymbol{\mu}, \mathbf{v}\mathbf{\Sigma}\mathbf{v}^T)$ where $\mathbf{v} \in \mathbb{R}^d$. Let $\mathbf{x}$ denote the input sample for which we wish to find a counterfactual $\check{\mathbf{x}}_E = \mathbf{x} + \boldsymbol{\delta}_E$. Recall from Definition 1 that we have to evaluate:

$$\begin{aligned}\Delta &= \mathbb{E}_{\varepsilon}\Big[\underbrace{h(\check{\mathbf{x}}_E)}_{\text{CE class}} - \underbrace{h(\check{\mathbf{x}}_E + \varepsilon)}_{\text{class after response}}\Big] \\ &= 1 - \mathbb{E}_{\varepsilon}\big[h(\check{\mathbf{x}}_E + \varepsilon)\big],\end{aligned} \tag{25}$$

where we have used that the first term is a constant and evaluates to 1 by the definition of a counterfactual explanation. It remains to evaluate the expectation: $\mathbb{E}_{\varepsilon}\big[h(\check{\mathbf{x}}_E + \varepsilon)\big]$. Next, we note that equation 25 can equivalently be expressed in terms of the logit outcomes:

$$\Delta = \mathbb{E}_{\varepsilon}\Big[\underbrace{\mathbb{I}\big[f(\check{\mathbf{x}}_E) > 0\big]}_{\text{CE class}} - \underbrace{\mathbb{I}\big[f(\check{\mathbf{x}}_E + \varepsilon) > 0\big]}_{\text{class after perturbation}}\Big] = \Big(1 - \mathbb{E}_{\varepsilon}\big[\mathbb{I}\big[f(\check{\mathbf{x}}_E + \varepsilon) > 0\big]\big]\Big). \tag{26}$$

Again, we are interested in the second term, which evaluates to:

$$\mathbb{E}_{\varepsilon}\big[\mathbb{I}\big[f(\check{\mathbf{x}}_E + \varepsilon) > 0\big]\big] = 0 \cdot \mathbb{P}\Big(f(\check{\mathbf{x}}_E + \varepsilon) < 0\Big) + 1 \cdot \mathbb{P}\Big(f(\check{\mathbf{x}}_E + \varepsilon) > 0\Big). \tag{27}$$

Next, consider the first-order Taylor approximation: $f(\check{\mathbf{x}}_E + \varepsilon) \approx f(\check{\mathbf{x}}_E) + \nabla f(\check{\mathbf{x}}_E)^\top \varepsilon$. Hence, we know $\nabla f(\check{\mathbf{x}}_E)^\top \varepsilon$ approximately follows $\mathcal{N}(\mathbf{0}, \nabla f(\check{\mathbf{x}}_E)\mathbf{\Sigma}\nabla f(\check{\mathbf{x}}_E)^\top)$. Now, the second term can be computed as follows:

$$\mathbb{P}\Big(f(\check{\mathbf{x}}_E + \varepsilon) > 0\Big) \approx \mathbb{P}\Big(f(\check{\mathbf{x}}_E) > -\nabla f(\check{\mathbf{x}}_E)^\top \varepsilon\Big) = \mathbb{P}\Big(-f(\check{\mathbf{x}}_E) < \nabla f(\check{\mathbf{x}}_E)^\top \varepsilon\Big) \tag{28}$$

$$= 1 - \mathbb{P}\Big(-f(\check{\mathbf{x}}_E) > \nabla f(\check{\mathbf{x}}_E)^\top \varepsilon\Big) \tag{29}$$

$$= 1 - \mathbb{P}\Bigg(\underbrace{\frac{\nabla f(\check{\mathbf{x}}_E)^\top \varepsilon}{\sqrt{\nabla f(\check{\mathbf{x}}_E)^\top \mathbf{\Sigma} \nabla f(\check{\mathbf{x}}_E)}}}_{\text{Mean 0 Gaussian RV}} < \underbrace{-\frac{f(\check{\mathbf{x}}_E)}{\sqrt{\nabla f(\check{\mathbf{x}}_E)^\top \mathbf{\Sigma} \nabla f(\check{\mathbf{x}}_E)}}}_{\text{Constant}}\Bigg)$$

$$= 1 - \Phi\left(-\frac{f(\check{\mathbf{x}}_E)}{\sqrt{\nabla f(\check{\mathbf{x}}_E)^\top \mathbf{\Sigma} \nabla f(\check{\mathbf{x}}_E)}}\right)$$

$$= \Phi\left(\frac{f(\check{\mathbf{x}}_E)}{\sqrt{\nabla f(\check{\mathbf{x}}_E)^\top \mathbf{\Sigma} \nabla f(\check{\mathbf{x}}_E)}}\right), \tag{30}$$

where the last line follows due to symmetry of the standard normal distribution (i.e., $\Phi(-u) = 1 - \Phi(u)$). Putting the pieces together, we have:

$$\mathbb{E}_{\boldsymbol{\varepsilon}}\big[\mathbb{I}\big[f(\check{\mathbf{x}}_E + \boldsymbol{\varepsilon}) > 0\big]\big] = 0 \cdot \mathbb{P}\bigg(f(\check{\mathbf{x}}_E + \boldsymbol{\varepsilon}) < 0\bigg) + 1 \cdot \mathbb{P}\bigg(f(\check{\mathbf{x}}_E + \boldsymbol{\varepsilon}) \geq 0\bigg) \tag{31}$$

$$= \Phi\Bigg(\frac{f(\check{\mathbf{x}}_E)}{\sqrt{\nabla f(\check{\mathbf{x}}_E)^\top \boldsymbol{\Sigma} \nabla f(\check{\mathbf{x}}_E)}}\Bigg). \tag{32}$$

Thus, we have:

$$\Delta \approx \tilde{\Delta} = 1 - \Phi\Bigg(\frac{f(\check{\mathbf{x}}_E)}{\sqrt{\nabla f(\check{\mathbf{x}}_E)^\top \boldsymbol{\Sigma} \nabla f(\check{\mathbf{x}}_E)}}\Bigg), \tag{33}$$

which completes our proof. Note that this is equivalent to $\mathbb{P}\bigg(f(\check{\mathbf{x}}_E + \boldsymbol{\varepsilon}) < 0\bigg)$, and thus we are "counting" how often perturbations to $\check{\mathbf{x}}_E$ sampled from $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ result in flips back to the undesired class. □

Proposition 2. *For a linear classifier, let $r \in (0,1)$ and let $\check{\mathbf{x}}_E = \mathbf{x} + \boldsymbol{\delta}_E$ be the output produced by some recourse method E such that $h(\check{\mathbf{x}}_E) = 1$. Then the cost required to achieve a fixed invalidation target r is given by:*

$$\|\boldsymbol{\delta}_E\|_2 = \frac{\sigma}{\omega}\big(\Phi^{-1}(1-r) - c\big), \tag{34}$$

where $c = \frac{f(\mathbf{x})}{\sigma \cdot \|\nabla f(\mathbf{x})\|_2}$ is a constant, and $\omega > 0$ is the cosine of the angle between the vectors $\nabla f(\mathbf{x})$ and $\boldsymbol{\delta}_E$.

Proof. Under a logistic classifier, the result immediately follows by setting the expression from Theorem 1 equal to r, using the identity $\nabla f(\mathbf{x})^\top \boldsymbol{\delta}_E = \omega \cdot \|\nabla f(\mathbf{x})\|_2 \cdot \|\boldsymbol{\delta}_E\|_2$ where ω is the cosine of the angle between the vectors $\nabla f(\mathbf{x})$ and $\boldsymbol{\delta}_E$, and rearranging for $\|\boldsymbol{\delta}_E\|_2$. □

Proposition 3. *Under the same conditions as in Proposition 2, we have $\frac{\partial \|\boldsymbol{\delta}_E\|_2}{\partial(1-r)} = \frac{\sigma}{\omega}\frac{1}{\phi(\Phi^{-1}(1-r))} > 0$, i.e., an infinitesimal increase in robustness (i.e.,$1-r$) increases the cost of recourse by $\frac{\sigma}{\omega}\frac{1}{\phi(\Phi^{-1}(1-r))}$.*

Proof. We will compute the derivative of $\|\boldsymbol{\delta}_E\|_2 = \frac{\sigma}{\omega}\big(\Phi^{-1}(1-r) - c\big)$ with respect to $1-r$ and show that it is positive for all $r \in (0,1)$:

$$\frac{\partial \|\boldsymbol{\delta}_E\|_2}{\partial(1-r)} = \frac{\sigma}{\omega}\frac{1}{\phi(\Phi^{-1}(1-r))} > 0, \tag{35}$$

where ϕ is the probability density function (PDF) of the standard Gaussian distribution. Since the PDF must be positive, we have that $\phi(\Phi^{-1}(1-r)) > 0$, and we know that $\sigma, \omega > 0$. Thus, the results follows. □

Proposition 4. *Let $\check{\mathbf{x}}_E$ be the output produced by some recourse method E such that $h(\check{\mathbf{x}}_E) = 1$. Then, an upper bound on $\tilde{\Delta}$ from equation 4 is given by:*

$$\tilde{\Delta}(\check{\mathbf{x}}_E; \sigma^2 \mathbf{I}) \leq 1 - \Phi\Bigg(c + \frac{\omega}{\sigma}\frac{\|\nabla f(\mathbf{x})\|_2}{\|\nabla f(\check{\mathbf{x}}_E)\|_2}\frac{\|\boldsymbol{\delta}_E\|_1}{\sqrt{\|\boldsymbol{\delta}_E\|_0}}\Bigg), \tag{36}$$

where $c = \frac{f(\mathbf{x})}{\sigma \cdot \|\nabla f(\mathbf{x})\|_2}$ is a constant, $\boldsymbol{\delta}_E = \check{\mathbf{x}}_E - \mathbf{x}$, and $\omega > 0$ is the cosine of the angle between the vectors $\nabla f(\mathbf{x})$ and $\boldsymbol{\delta}_E$.

Proof. We start by noting the following basic inequality:

$$\|\mathbf{z}\|_1 \leq \sqrt{\|\mathbf{z}\|_0} \cdot \|\mathbf{z}\|_2.$$

Going forward, we will refer to these inequalities as basic inequalities. Moreover, note that Φ is a monotonic function. Thus, we have $\Phi(a) \leq \Phi(a')$ for $a \leq a'$. Note that $f(\check{\mathbf{x}}_E) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \boldsymbol{\delta}_E$. Thus we obtain the following approximation:

$$\tilde{\Delta} = 1 - \Phi\left(\frac{f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \boldsymbol{\delta}_E}{\sqrt{\nabla f(\check{\mathbf{x}}_E) \boldsymbol{\Sigma}^\top \nabla f(\check{\mathbf{x}}_E)}}\right). \tag{37}$$

Next, we will find upper bounds for the term on the right: Before we will do that, we will express the above expression more conveniently to highlight the impact of the counterfactual action $\boldsymbol{\delta}_E$ more explicitly. To do that, note that $\nabla f(\mathbf{x})^\top \boldsymbol{\delta}_E = \omega \cdot \|\nabla f(\mathbf{x})\|_2 \cdot \|\boldsymbol{\delta}_E\|_2$ where ω is the cosine of the angle between the vectors $\nabla f(\mathbf{x})$ and $\boldsymbol{\delta}_E$. Using $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$, we obtain:

$$\Phi\left(\frac{f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \boldsymbol{\delta}_E}{\sigma \|\nabla f(\check{\mathbf{x}}_E)\|_2}\right) = \Phi\left(c + \frac{\|\nabla f(\mathbf{x})\|_2}{\|\nabla f(\check{\mathbf{x}}_E)\|_2} \cdot \frac{\omega}{\sigma} \cdot \|\boldsymbol{\delta}_E\|_2\right), \tag{38}$$

where we defined a constant $c = \frac{f(\mathbf{x})}{\sigma \|\nabla f(\check{\mathbf{x}}_E)\|_2}$ using quantities that we will keep fixed in our analysis, namely $\mathbf{x}, \nabla f(\mathbf{x})$ and σ. Also note that $\mathbf{x}$ is the factual input, and thus its logit score satisfies: $f(\mathbf{x}) < 0$. Since $\boldsymbol{\delta}_E$ is a valid perturbation, we must have that $\omega > 0$ for the perturbation to change the class prediction.

Note that the following *lower bound* holds by the basic inequality stated above:

$$\Phi\left(c + \frac{\|\nabla f(\mathbf{x})\|_2}{\|\nabla f(\check{\mathbf{x}}_E)\|_2} \cdot \frac{\omega}{\sigma} \cdot \|\boldsymbol{\delta}_E\|_2\right) \geq \Phi\left(c + \frac{\|\nabla f(\mathbf{x})\|_2}{\|\nabla f(\check{\mathbf{x}}_E)\|_2} \cdot \frac{\omega}{\sigma} \cdot \frac{\|\boldsymbol{\delta}_E\|_1}{\sqrt{\|\boldsymbol{\delta}_E\|_0}}\right). \tag{39}$$

As a consequence we obtain the following *upper bound on the IR*:

$$\tilde{\Delta} \leq 1 - \Phi\left(c + \frac{\|\nabla f(\mathbf{x})\|_2}{\|\nabla f(\check{\mathbf{x}}_E)\|_2} \cdot \frac{\omega}{\sigma} \cdot \frac{\|\boldsymbol{\delta}_E\|_1}{\sqrt{\|\boldsymbol{\delta}_E\|_0}}\right), \tag{40}$$

as claimed. □

Proposition 1. *For the logistic regression classifier, consider the recourse output by Wachter et al. (2018): $\check{\mathbf{x}}_{Wachter}(s) = \mathbf{x} + \frac{s - f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|_2^2} \nabla f(\mathbf{x})$. Then the recourse invalidation rate has the following closed-form:*

$$\Delta(\check{\mathbf{x}}_{Wachter}(s); \sigma^2 \mathbf{I}) = 1 - \Phi\left(\frac{s}{\sigma \|\nabla f(\mathbf{x})\|_2}\right), \tag{41}$$

where s is the target logit score.

Proof. Since we are in the linear case, we have: $\nabla f(\check{\mathbf{x}}_E) = \nabla f(\mathbf{x})$. Also, note that $f(\check{\mathbf{x}}_E) = f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \boldsymbol{\delta}_E$. Using $\boldsymbol{\Sigma} = \sigma^2 \mathbf{I}$, we obtain the following exact expression:

$$\Delta = 1 - \Phi\left(\frac{f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \boldsymbol{\delta}_E}{\sigma \|\nabla f(\mathbf{x})\|_2}\right). \tag{42}$$

From Pawelczyk et al. (2022), we have:

$$\boldsymbol{\delta}_{\text{Wachter}} = \frac{s - f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|_2^2} \nabla f(\mathbf{x}). \tag{43}$$

Plugging equation 43 into equation 42 we obtain:

$$\begin{aligned}
\Delta &= 1 - \Phi\left(\frac{f(\mathbf{x})}{\sigma \|\nabla f(\mathbf{x})\|_2} + \frac{\nabla f(\mathbf{x})^\top \boldsymbol{\delta}_E}{\sigma \|\nabla f(\mathbf{x})\|_2}\right) && (44)\\
&= 1 - \Phi\left(\frac{f(\mathbf{x})}{\sigma \|\nabla f(\mathbf{x})\|_2} + \frac{1}{\sigma \|\nabla f(\mathbf{x})\|_2} \cdot \nabla f(\mathbf{x})^\top \nabla f(\mathbf{x}) \frac{s - f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|_2^2}\right)\\
&= 1 - \Phi\left(\frac{f(\mathbf{x})}{\sigma \|\nabla f(\mathbf{x})\|_2} + \frac{s - f(\mathbf{x})}{\sigma \|\nabla f(\mathbf{x})\|_2}\right)\\
&= 1 - \Phi\left(\frac{s}{\sigma \|\nabla f(\mathbf{x})\|_2}\right), && (45)
\end{aligned}$$

which concludes the proof. □

Corollary 1. *Under the conditions of Proposition 1, choosing* $s_r = \sigma \|\nabla f(\mathbf{x})\|_2 \Phi^{-1}(1-r)$ *guarantees a recourse invalidation rate of* r*, i.e.,* $\Delta(\check{\mathbf{x}}_{\textit{Wachter}}(s_r); \sigma^2 \mathbf{I}) = r$.

Proof. The result directly follows from plugging in $s_r = \sigma \|\nabla f(\mathbf{x})\|_2 \Phi^{-1}(1-r)$ into the optimal recourse from $\boldsymbol{\delta}_{\text{Wachter}}$ from equation 43 and subsequently evaluating the recourse invalidation rate from equation 5. □

D.3 PROOF OF THEOREM 2

Proof. From Definition 1 we know:

$$\Delta_{\text{Forest}} = \mathbb{E}_{\boldsymbol{\varepsilon}}\Big[\underbrace{\mathcal{F}(\check{\mathbf{x}}_E)}_{\text{CE class}} - \underbrace{\mathcal{F}(\check{\mathbf{x}}_E + \boldsymbol{\varepsilon})}_{\text{class after response}}\Big] \tag{46}$$

$$= 1 - \mathbb{E}_{\boldsymbol{\varepsilon}}\big[\mathcal{F}(\check{\mathbf{x}}_E + \boldsymbol{\varepsilon})\big]. \tag{47}$$

It remains to evaluate: $\mathbb{E}_{\boldsymbol{\varepsilon}}\big[\mathcal{F}(\check{\mathbf{x}}_E + \boldsymbol{\varepsilon})\big]$. Using equation 14, we have:

$$\mathbb{E}_{\boldsymbol{\varepsilon}}\big[\mathcal{F}(\check{\mathbf{x}}_E + \boldsymbol{\varepsilon})\big] = \mathbb{E}_{\boldsymbol{\varepsilon}}\Bigg[\sum_{R \in \mathcal{R}_{\mathcal{F}}} c_{\mathcal{F}}(R) \cdot \mathbb{I}(\check{\mathbf{x}}_E + \boldsymbol{\varepsilon} \in R)\Bigg] \tag{48}$$

$$= \sum_{R \in \mathcal{R}_{\mathcal{F}}} c_{\mathcal{F}}(R) \cdot \mathbb{E}_{\boldsymbol{\varepsilon}}\big[\mathbb{I}(\check{\mathbf{x}}_E + \boldsymbol{\varepsilon} \in R)\big] \qquad \text{(Linearity of Expectation)}$$

$$= \sum_{R \in \mathcal{R}_{\mathcal{F}}} c_{\mathcal{F}}(R) \cdot \int_R p(\mathbf{y}) d\mathbf{y} \qquad (p(\mathbf{y}) = \mathcal{N}(\check{\mathbf{x}}_E, \boldsymbol{\sigma}^2 \mathbf{I}))$$

$$= \sum_{R \in \mathcal{R}_{\mathcal{F}}} c_{\mathcal{F}}(R) \cdot \prod_{j \in \mathcal{S}_{\mathcal{F}}} \int_{R_j} \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{1}{2}\frac{(y_j - \check{x}_{E,j})^2}{\sigma_j^2}\right) dy_j$$

(Since $\boldsymbol{\varepsilon}$ is an independent Gaussian)

$$= \sum_{R \in \mathcal{R}_{\mathcal{F}}} c_{\mathcal{F}}(R) \cdot \prod_{j \in \mathcal{S}_{\mathcal{F}}} \left[\Phi\left(\frac{\bar{t}_{j,R} - \check{x}_{E,j}}{\sigma_j}\right) - \Phi\left(\frac{\underline{t}_{j,R} - \check{x}_{E,j}}{\sigma_j}\right)\right].$$

(Since $\boldsymbol{\varepsilon}$ is Gaussian)

Using our Definition of robustness, we have

$$\Delta_{\text{Forest}} = 1 - \sum_{R \in \mathcal{R}_{\mathcal{F}}} c_{\mathcal{F}}(R) \prod_{j \in \mathcal{S}_{\mathcal{F}}} \left[\Phi\left(\frac{\bar{t}_{j,R} - \check{x}_{E,j}}{\sigma_j}\right) - \Phi\left(\frac{\underline{t}_{j,R} - \check{x}_{E,j}}{\sigma_j}\right)\right], \tag{49}$$

which completes the proof. □

A.6 CARLA: A Python Library to Benchmark Algorithmic Recourse Algorithms

Publication: Published in the 35th Conference on Neural Information Processing Systems (**NeurIPS**) 2021.

Contribution: I developed the idea for the benchmarking library. Tobias Richter and I developed the initial outline for the software architecture. Sascha Bielawaski and I developed the initial codebase and refined the software architecture. While Sascha Bielawaski focused on integrating algorithms with existing code bases, I re-implemented all algorithms that only contained pseudocode from scratch (e.g., `REVISE`, `GS`, `FACE`). Sascha Bielawaski then transferred the code base to the `CARLA`-repository. Johannes van den Heuvel joined the project in the late stages and ran the benchmarking pipeline. Tobias Richter and I wrote the introduction. The remainder of the paper was written by me. Gjergji Kasneci made valuable suggestions by challenging and improving ideas, and by making suggestions to refine the text and the narrative.

CARLA: A Python Library to Benchmark Algorithmic Recourse and Counterfactual Explanation Algorithms

Martin Pawelczyk[*]
University of Tübingen
martin.pawelczyk@uni-tuebingen.de

Sascha Bielawski
University of Tübingen
sascha.bielawski@uni-tuebingen.de

Johannes van den Heuvel
University of Tübingen
johannes.van-den-heuvel@uni-tuebingen.de

Tobias Richter [†]
CarePay International
t.richter@carepay.com

Gjergji Kasneci [†]
University of Tübingen
gjergji.kasneci@uni-tuebingen.de

Abstract

Counterfactual explanations provide means for prescriptive model explanations by suggesting actionable feature changes (e.g., increase income) that allow individuals to achieve favourable outcomes in the future (e.g., insurance approval). Choosing an appropriate method is a crucial aspect for meaningful counterfactual explanations. As documented in recent reviews, there exists a quickly growing literature with available methods. Yet, in the absence of widely available open–source implementations, the decision in favour of certain models is primarily based on what is readily available. Going forward – to guarantee meaningful comparisons across explanation methods – we present `CARLA` (**C**ounterfactual **A**nd **R**ecourse **L**ibr**A**ry), a python library for benchmarking counterfactual explanation methods across both different data sets and different machine learning models. In summary, our work provides the following contributions: (i) an extensive benchmark of 11 popular counterfactual explanation methods, (ii) a benchmarking framework for research on future counterfactual explanation methods, and (iii) a standardized set of integrated evaluation measures and data sets for transparent and extensive comparisons of these methods. We have open sourced `CARLA` and our experimental results on Github, making them available as competitive baselines. We welcome contributions from other research groups and practitioners.

1 Introduction

Machine learning (ML) methods have found their way into numerous everyday applications and have become an indispensable asset in various sensitive domains, like disease diagnostics [13], criminal justice [4], or credit risk scoring [29]. While ML models bear the great potential to provide effective support in human decision making processes, their predictions may have considerable impact on personal lives, where the final decisions might be disadvantageous for an end user. For example, the rejection of a loan or the denial of parole might have negative effects on the future development of the corresponding person's life.

[*]Corresponding author
[†]Equal senior author contribution

35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks.

When ML systems involve humans in the loop, it is crucial to build a strong foundation for long-term acceptance of these methods. To this end, it is critical (1) to *explain* the predictions of a model and (2) to *offer constructive means for the improvement* of those predictions to the advantage of the end–user. Counterfactual explanations – popularized by the seminal work of [62] – provide means for prescriptive model explanations by suggesting actionable feature changes (e.g., increase income) that allow individuals to achieve favourable outcomes in the future (e.g., insurance approval).

When counterfactual explainability is employed in systems that involve humans in the loop, the community refers to it as *recourse*. Algorithmic recourse subsumes precise recipes on how to obtain desirable outcomes after being subjected to an automated decision, emphasizing feasibility constraints that have to be taken into account. Those explanations are found by making the smallest possible change to an input vector to influence the prediction of a pretrained classifier in a positive way; for example, from 'loan denial' to 'loan approval', subject to the constraint that an individual's `sex` may not change. As documented in recent reviews, there exists a quickly growing literature with available methods (see Figure 1 and [54, 24, 60]), reflecting the insight that the understanding of complex machine learning models is an elementary ingredient for a wide and safe technology adoption.

In practice, the counterfactual explanation (CE) that an individual receives crucially depends on the method that computes the recourse suggestions. Hence, there is a substantial need for a standardized benchmarking platform, which ensures that methods can be compared in a transparent and meaningful way. Researchers need to be able to easily evaluate their proposed methods against the overwhelming diversity of already available methods and practitioners need to make sure that they are using the right recourse mechanism for the problem at hand. Therefore, a standardized framework for comparison and quality assurance is an essential and indispensable prerequisite.

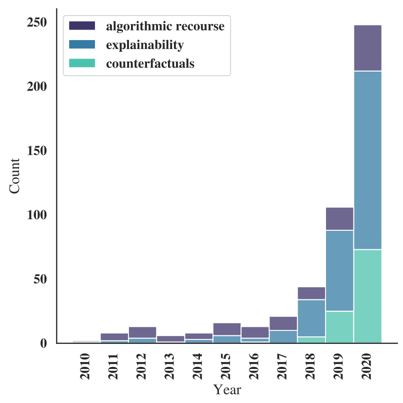


Figure 1: ArXiv submissions over time on explainability, counterfactual explanations and algorithmic recourse.

In this work, we present `CARLA` (**C**ounterfactual **A**nd **R**ecourse **L**ibr**A**ry), a python library with the following merits: First, `CARLA` provides **competitive baselines** for researchers to benchmark *new* counterfactual explanation and recourse methods for the standardized and transparent comparison of CE methods on different integrated data sets. Second, `CARLA` is a **common framework** with more than 10 counterfactual explanation methods in combination with the possibility to easily integrate new methods into a commonly accessible and easily distributable Python library. Moreover, the built-in integrated evaluation measures allow users to plug-in their custom black-box predictive models into the available counterfactual explanation methods and conduct extensive evaluations in comparison with other recourse mechanisms across different data sets. The same is true for researchers, who can use `CARLA` to extensively benchmark available counterfactual methods on popular data sets across various ML models. Third, `CARLA` **supports popular optimization frameworks** such as Tensorflow [1] and PyTorch [43], and provides a generic abstraction layer to support custom implementations. Users can can define problem–specific data set characteristics like immutable features and explicitly specify hyperparameters for the chosen counterfactual explanation method.

The remainder of this work is structured as follows: Section 2 presents related work, Section 3 formally introduces the recourse problem, Section 4 presents the benchmarking process. In Section 5, we describe our main findings, before concluding in Section 6. Appendices A - E describe `CARLA`'s software architecture and usage instructions, as well as additional experimental results, used ML classifiers, data sets and hyperparameters settings.

2 Related Work

Explainable machine learning is concerned with the problem of providing explanations for complex ML models. Towards this goal, various streams of research follow different explainability paradigms which can be categorized into the following groups [17, 14].

2.1 Feature Highlighting Explanations

Local input attribution techniques seek to explain the behaviour of ML models instance by instance. Those methods aim to understand how all inputs available to the model are being used to arrive at a certain prediction. Some popular approaches for model explanations aim at explainability by design [34, 2, 5, 63]. For white-box models – the internal model parameters are known – gradient-based approaches, e.g. [27, 6] (for deep neural networks), and rule-based or probabilistic approaches for tree ensembles, e.g. [19, 9] have been proposed. In cases where the parameters of the complex models cannot be accessed, model-agnostic approaches can prove useful. This group of approaches seeks to explain a model's behavior locally by applying surrogate models [50, 35, 51, 36], which are interpretable by design and are used to explain individual predictions of black-box ML models.

2.2 Counterfactual Explanations

The main purpose of counterfactual explanations is to suggest constructive interventions to the input of a complex model so that the output changes to the advantage of an end user. By emphasizing both the feature importance and the recommendation aspect, counterfactual explanation methods can be further divided into three different groups: independence-based, dependence-based, and causality-based approaches.

In the class of **independence-based methods**, where the input features of the predictive model are assumed to be independent, some approaches use combinatorial solvers or evolutionary algorithms to generate recourse in the presence of feasibility constraints [57, 52, 49, 23, 28, 8]. Notable exceptions from this line of work are proposed by [56, 32, 31, 18, 15], who use decision trees, random search, support vector machines (SVM) and information networks that are aligned with the recourse objective. Another line of research deploys gradient-based optimization to find low-cost counterfactual explanations in the presence of feasibility and diversity constraints [10, 38, 39, 53, 59, 46]. The main problem with these approaches is that they abstract from input correlations. That implies that the intervention costs (i.e., the costs of changing the input to achieve the proposed counterfactual state) are too optimistically estimated. In other words, the estimated costs do not reflect the true costs that an individual would need to incur in practical scenarios, where feature dependencies are usually present: e.g., *income* is dependent on *tenure*, and if *income* changes, *tenure* also changes (see Figure 2 for a schematic comparison).

In the class of **causality-based** approaches, all methods make use of Pearl's causal modelling framework [47]. As such, they usually require knowledge of the system of causal structural equations [20, 16, 25, 42] or the causal graph [26]. The authors of [25] show that these models can generate minimum-cost recourse, if the access to the true causal data generating process was available. However, in practical scenarios, the guarantee for such minimum-cost recommendations is vacuous, since, in complex settings, the causal model is likely to be miss-specified [26]. Since these methods usually require the *true* causal graph – which is the limiting factor in practice – we have not considered them at this point, but we plan to do that in the future.

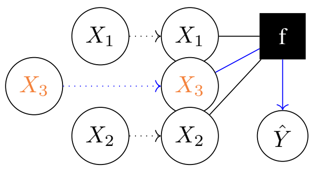
Dependence-based methods bridge the gap between the strong independence assumption and the strong causal assumption. This class of models builds recourse suggestions on generative models [44, 11, 20, 37, 45]. The main idea is to change the geometry of the intervention space to a lower dimensional latent space, which encodes different factors of variation while capturing input dependencies. To this end, these methods primarily use variational autoencoders (VAE) [30, 41]. In particular, Mahajan et al. [37] demonstrate how to encode various feasibility constraints into VAE-based models. Most recently, [3] proposed `CLUE`, a generative recourse model that takes a classifier's uncertainty into account. Work that deviates from this line of research was done by [48, 22]. The authors of [48] provide `FACE`, which uses a shortest path algorithm on graphs to find counterfactual explanations. In contrast, Kanamori et al. [22] use integer programming techniques to account for input dependencies.

3 Preliminaries

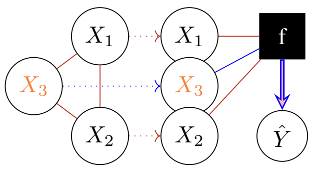
In this Section, we review the algorithmic recourse problem and draw a distinction between two observational (i.e., non–causal) methods.

3.1 Counterfactual Explanations for Independent Inputs

Let $\mathcal{D}$ be the data set consisting of N input data points, $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$. We denote by $f : \mathbb{R}^d \rightarrow [0, 1]$ the fixed classifier for which recourse is to be determined. We denote the



(a) Indep. recourse intervention



(b) Dep. recourse intervention

Figure 2: Different views on recourse generation. In (a) a change to X_3 only impacts f through X_3 , while in (b) the same change induces indirect effects on f, if X_3 is correlated with other inputs.

set of outcomes by $y(x) \in \{0, 1\}$, where $y = 1$ indicates the desirable outcome. Moreover, $\hat{y} = \mathbb{I}[f(x) > \theta]$ is the predicted class, where $\mathbb{I}[\cdot]$ denotes the indicator function and θ is a threshold (e.g., 0.5). Our goal is to find a set of actionable changes in order to improve the outcomes of instances x, which are assigned an undesirable prediction under f. Moreover, one typically defines a distance measure in inputs space $c : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}_+$. We discuss typical choices for c in Section 4.

Assuming inputs are pairwise statistically independent, the recourse problem is defined as follows:

$$\delta_x^* = \underset{\delta_x \in \mathcal{A}_d}{\arg\min}\, c(x, \check{x}) \text{ s.t. } \check{x} = x + \delta_x, f(x + \delta_x) > \theta, \tag{I}$$

where $\mathcal{A}_d$ is the set of admissible changes made to the factual input x. For example, $\mathcal{A}_d$ could specify that no changes to sensitive attributes such as `age` or `sex` may be made. For example, using the independent input assumption, existing approaches [57] use mixed-integer linear programming to find counterfactual explanations. In the next paragraph, we present a problem formulation that relaxes the strong independence assumption by introducing generative models.

3.2 Recourse for Correlated Inputs

We assume the factual input $x \in \mathcal{X} = \mathbb{R}^d$ is generated by a generative model g such that:

$$x = g(z),$$

where $z \in \mathcal{Z} = \mathbb{R}^k$ are latent codes. We denote the counterfactual explanation in an input space by $\check{x} = x + \delta_x$. Thus, we have $\check{x} = x + \delta_x = g(z + \delta_z)$. Assuming inputs are dependent, we can rewrite the recourse problem in (I) to faithfully capture those dependencies using the generative model g:

$$\delta_z^* = \underset{\delta_z \in \mathcal{A}_k}{\arg\min}\, c(x, \check{x}) \text{ s.t. } \check{x} = g(z + \delta_z), f(\check{x}) > \theta, \tag{D}$$

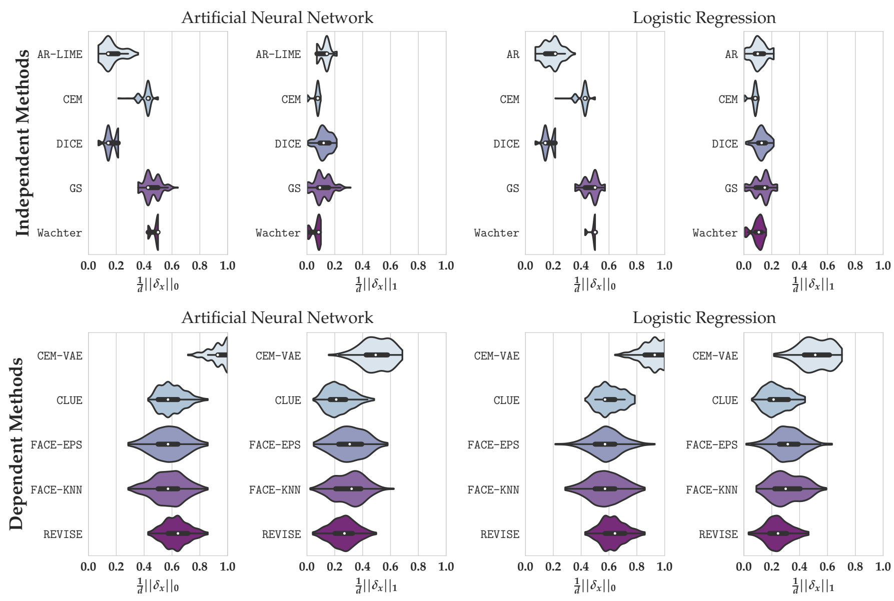
where $\mathcal{A}_k$ is the set of admissible changes in the k-dimensional latent space. For example, $\mathcal{A}_k$ would ensure that the counterfactual latent space lies within range of z. The problem in (**D**) is an abstraction from how the problem is usually solved in practice: most existing approaches first train a type of autoencoder model (e.g., a VAE), and then use the model's trained decoder as a deterministic function g to find counterfactual explanations [20, 44, 37, 11, 3]. Our benchmarked explanation models roughly fit in one of these two categories.

4 Benchmarking Process

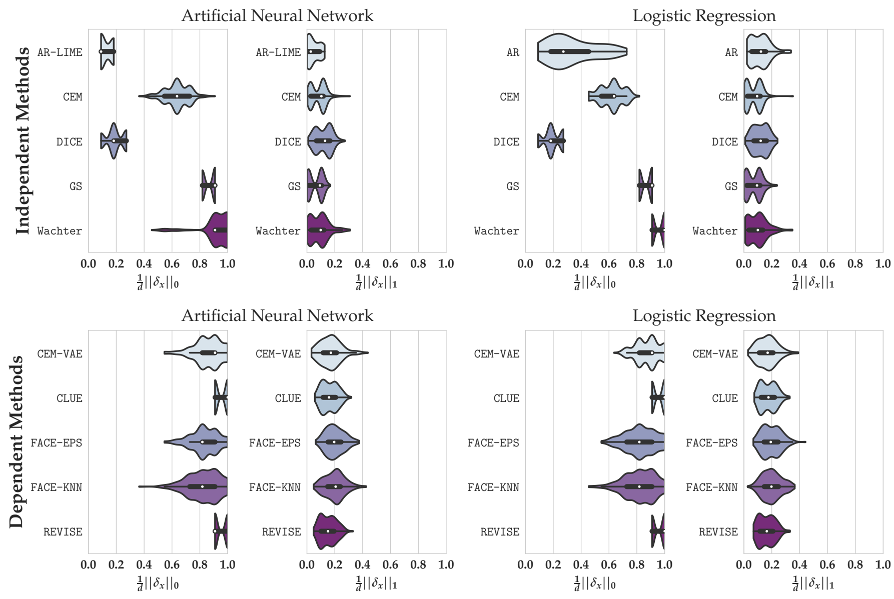
In this Section, we provide a brief explanation model overview and introduce a variety of explanation measures used to evaluate the quality of the generated counterfactual explanations. In Table 1 we present a concise explanation model overview.

4.1 Counterfactual Explanation Methods

`AR` (**I**) Ustun et al. [58] provide a method to generate minimal cost actions δ_x^* for linear classification models such as logistic regression models. `AR` requires the linear model's coefficients, and uses these coefficients for its search for counterfactual explanations. To provide reasonable actions it is possible to restrict δ_x^* to *user–specified constraints* (e.g., *has_phd* can only change from *False* to *True*) or to set a subset of inputs as *immutable* (e.g., *age*). The problem to find these changes is a discrete optimization problem. Given a set of actions, `AR` finds the action which minimizes a defined cost function, using integer programming solvers like `CPLEX` or `CBC`.



(a) Adult Data



(b) Give Me Some Credit Data

Figure 3: Evaluating the distribution of costs of counterfactual explanations on 2 different data sets (the results on COMPAS are relegated to Appendix B). For all instances with a negative prediction ($\{x \in \mathcal{D} : f(x) < \theta\}$), we plot the distribution of ℓ_0 and ℓ_1 costs of algorithmic recourse as defined in (1) for a logistic regression and an artificial neural network classifier. The white dots indicate the medians (lower is better), and the black boxes indicate the interquartile ranges. We distinguish between independence based and dependence based methods. The results are discussed in Section 5

Approach	Method	Model Type	Algorithm	Immutable	Categorical	Other
Independent (**I**)	`AR`	Linear	Integer Prog.	Yes	Binary	Direction of change
	`AR-LIME`	Agnostic	Integer Prog.	Yes	Binary	Direction of change
	`CEM`	Gradient based	Gradient based	No	No	None
	`DICE`	Gradient Based	Gradient based	Yes	Binary	Generative model
	`GS`	Agnostic	Random search	Yes	Binary	None
	`Wachter`	Gradient based	Gradient based	No	Binary	None
Dependent (**D**)	`CEM-VAE`	Gradient based	Gradient based	No	No	Gen. Model regularizer
	`CLUE`	Gradient based	Gradient based	No	No	Generative model
	`FACE-EPS`	Agnostic	Graph search	Binary	Binary	CE is from data set
	`FACE-KNN`	Agnositc	Graph search	Binary	Binary	CE is from data set
	`REVISE`	Gradient based	Gradient based	Binary	Binary	Generative model

Table 1: Explanation method summary: we categorize different approaches based on their underlying assumptions and list what kind of ML model they work with (Model Type), the Method's underlying algorithm (Algorithm), whether the method can handle immutable features (Immutable), whether it can handle categorical features (Categorical) and any other outstanding characteristics (Other).

`AR-LIME` (**I**) Most classification tasks do not have linearly separable classes and complex non–linear models usually provide more accurate predictions. Non–linear models are not per se interpretable and usually do not provide coefficients similar to linear models. We use a **reduction** to apply `AR` to non–linear models by computing a local linear approximation for the point of interest x, using `LIME` [50]. For an arbitrary black–box model f, `LIME` estimates post–hoc local explanations in form of a set of linear coefficients per instance. Using the coefficients we apply `AR`.

`CEM` (**I**) Dhurandhar et al. [10] use an elastic–net regularization inspired objective to find low-cost counterfactual instances. Different weights can be assigned to ℓ_1 and ℓ_2 norms, respectively. There exists no immutable feature handling. However, we provide support for their VAE type regularizer, which should help ensure that counterfactual instances look more realistic.

`CLUE` (**D**) Antorán et al. [3] propose `CLUE`, a generative recourse model that takes a classifier's uncertainty into account. This model suggests feasible counterfactual explanations that are likely to occur under the data distribution. The authors use a variational autoencoder (VAE) to estimate the generative model. Using the VAE's decoder, `CLUE` uses an objective that guides the search of CEs towards instances that have low uncertainty measured in terms of the classifier's entropy.

`DICE` (**I**) Mothilal et al. [40] suggest `DICE`, which is an explanation model that seeks to generate minimum costs counterfactual explanations according to (**I**) subject to a diversity constraint which aims to promote a diverse set of counterfactual explanations. Diversity is achieved by using the whole range of suggested changes, while still keeping proximity to a given input. Regarding the optimization problem, `DICE` uses gradient descent to find a solution that trades-off proximity and diversity. Domain knowledge – in form of feature ranges or immutability constraints – can be added.

`FACE` (**D**) The authors of [48] provide `FACE`, which uses a shortest path algorithm (for graphs) to find counterfactual explanations from high–density regions. Those explanations are actual data points from either the training or test set. Immutability constraints are enforced by removing incorrect neighbors from the graph. We implemented two variants of this model: the first variant uses an epsilon–graph (`FACE-EPS`), whereas the second variant uses a knn–graph (`FACE-KNN`).

`Growing Spheres (GS)` (**I**) `Growing Spheres` – suggested in [32] – is a random search algorithm, which generates samples around the factual input point until a point with a corresponding counterfactual class label was found. The random samples are generated around x using growing hyperspheres. For binary input dimensions, the method makes use of Bernoulli sampling. Immutable features are readily specified by excluding them from the search procedure.

`REVISE` (**D**) Joshi et al. [20] propose a generative recourse model. This model suggests feasible counterfactual explanations that are likely to occur under the data distribution. The authors use a variational autoencoder (VAE) to estimate the generative model. Using the VAE's decoder, `REVISE` uses the latent space to search for CEs. No handling of immutable features exists.

`Wachter et al. (Wachter)` (**I**) The optimization approach suggested by Wachter et al. [61] generates counterfactual explanations by minimizing an objective function using gradient descent to find counterfactuals $\tilde{x}$ which are as close as possible to x. Closeness is measured in ℓ_1-norm.

		Artificial Neural Network					Logistic Regression				
Data Set	Method	*yNN*	redund.	violation	success	$\bar{t}(s)$	*yNN*	redund.	violation	success	$\bar{t}(s)$
Adult	AR(-LIME)	0.62	**0.00**	0.14	0.28	1.59	**0.72**	0.67	0.13	0.52	10.49
	CEM	0.26	3.96	0.66	**1.00**	1.10	0.20	3.98	0.66	**1.00**	0.92
	DICE	**0.71**	0.53	0.17	**1.00**	0.13	0.58	**0.51**	0.23	**1.00**	0.13
	GS	0.30	3.77	**0.09**	**1.00**	**0.01**	0.30	3.94	**0.10**	**1.00**	**0.01**
	Wachter	0.23	4.45	0.83	0.50	15.72	0.16	1.67	0.94	**1.00**	0.03
GMC	AR(-LIME)	0.89	**0.00**	0.29	0.07	0.55	**1.00**	2.33	**0.14**	0.39	3.42
	CEM	**0.95**	5.46	0.65	**1.00**	0.97	0.74	5.07	0.67	**1.00**	0.87
	DICE	0.90	0.58	0.27	**1.00**	0.28	0.88	**0.61**	0.27	**1.00**	0.29
	GS	0.40	6.64	**0.17**	**1.00**	**0.01**	0.49	5.29	0.17	**1.00**	**0.01**
	Wachter	0.58	6.56	0.71	**1.00**	0.02	0.59	6.12	0.83	**1.00**	**0.01**

(a) Independence based methods

		Artificial Neural Network					Logistic Regression				
Data Set	Method	*yNN*	redund.	violation	success	$\bar{t}(s)$	*yNN*	redund.	violation	success	$\bar{t}(s)$
Adult	CEM-VAE	0.12	9.68	1.82	**1.00**	**0.93**	0.43	10.05	1.80	**1.00**	**0.81**
	CLUE	**0.82**	8.05	**1.28**	**1.00**	2.70	0.33	7.30	1.33	**1.00**	2.56
	FACE-EPS	0.65	5.19	1.45	0.99	4.36	**0.64**	5.11	1.44	0.94	4.35
	FACE-KNN	0.60	**5.11**	1.41	**1.00**	4.31	0.57	**4.97**	1.38	1.00	4.31
	REVISE	0.20	8.65	1.33	1.00	8.33	0.62	7.92	**1.23**	**1.00**	7.52
GMC	CEM-VAE	**1.00**	8.40	**0.66**	**1.00**	**0.87**	**1.00**	8.54	**0.36**	**1.00**	**0.88**
	CLUE	**1.00**	9.39	0.90	0.93	1.91	**1.00**	9.56	0.96	**1.00**	1.76
	FACE-EPS	0.99	**8.06**	0.99	**1.00**	19.44	0.98	7.98	0.96	**1.00**	19.50
	FACE-KNN	0.98	9.00	0.98	**1.00**	15.87	0.98	**7.88**	0.95	**1.00**	16.09
	REVISE	**1.00**	9.50	0.97	**1.00**	4.56	**1.00**	9.59	0.96	1.00	3.76

(b) Dependence based methods

Table 2: Summary of a subset of results for independence and dependence based methods. For all instances with a negative prediction ($\{x \in \mathcal{D} : f(x) < \theta\}$), we compute counterfactual explanations for which we then measure yNN (higher is better), redundancy (lower is better), violation (lower is better), success rate (higher is better) and time (lower is better). We distinguish between a logistic regression and an artificial neural network classifier. Detailed descriptions of these measures can be found in Section 4. The results are discussed in Section 5.

4.2 Evaluation Measures for Counterfactual Explanation Methods

As algorithmic recourse is a multi–modal problem we introduce a variety of measures to evaluate the methods' performances. We use six baseline evaluation measures. Besides distance measures it is important to consider measures that emphasize the *quality* of recourse.

Costs When answering the question of generating the nearest counterfactual explanation, it is essential to define the distance of the factual x to the nearest counterfactual $\check{x}$. The literature has formed a consensus to use either the normalized ℓ_0 or ℓ_1 norm or any convex combination thereof (see for example [49, 39, 45, 23, 57, 62]). The ℓ_0 norm puts a restriction on the number of feature changes between factual and counterfactual instance, while the ℓ_1 norm restricts the average change:

$$c_0(\check{x}, x) = \frac{1}{d}\|x - \check{x}\|_0 = \frac{1}{d}\|\delta_x\|_0, \qquad c_1(\check{x}, x) = \frac{1}{d}\|x - \check{x}\|_1 = \frac{1}{d}\|\delta_x\|_1. \tag{1}$$

Constraint violation This measure counts the number of times the CE method violates user-defined constraints. Depending on the data set, we fixed a list of features which should not be changed by the used method (e.g., *sex*, *age* or *race*).

yNN We use a measure that evaluates how much data support CEs have from positively classified instances. Ideally, CEs should be close to positively classified individuals which is a desideratum formulated by Laugel et al. [33]. We define the set of individuals who received an undesirable prediction under f as $H^- := \{x \in \mathcal{D} : f(x) < \theta\}$. The counterfactual instances (instances for which the label was successfully changed) corresponding to the set H^- are denoted by $\check{H}^-$. We use a measure that captures how differently neighborhood points around a counterfactual instance $\check{x}$ are classified:

$$\text{yNN} = 1 - \frac{1}{nk} \sum_{i \in \check{H}^-}^{n} \sum_{j \in \text{kNN}(\check{x}_i)} |f_b(\check{x}_i) - f_b(x_j)|, \tag{2}$$

where kNN denotes the k-nearest neighbours of x, and $f_b(x) = \mathbb{I}[f(x) > 0.5]$ is the binarized classifier. Values of yNN close to 1 imply that the neighbourhoods around the counterfactual explanations consists of points with the same predicted label, indicating that the neighborhoods around these points have already been reached by positively classified instances. We use a value of $k := 5$, which ensures sufficient data support from the positive class.

Redundancy We evaluate how many of the proposed feature changes were not necessary. This is a particularly important criterion for independence–based methods. We measure this by *successively* flipping one value of $\check{x}$ after another back to x, and then we inspect whether the label flipped from 1 back to 0: e.g., we check whether flipping the value for the second dimension would change the counterfactual outcome 1 back to the predicted factual outcome of 0: $\mathbb{I}[f_b([\check{x}_1, x_2, \check{x}_3, \dots, \check{x}_d]) = 0]$. If the predicted outcome does not change, we increase the redundancy counter, concluding that a sparser counterfactual explanation could have been found. We iterate this process over all dimensions of the input vector.[3] A low number indicates few redundancies across counterfactual instances.

Success Rate Some generated counterfactual explanations do not alter the predicted label of the instance as anticipated. To keep track how often the generated CE does hold its promise, the success rate shows the fraction of respective models' correctly determined counterfactuals.

Average Time By measuring the average time a CE method needs to generate its result, we evaluate the effectiveness and feasibility for real–time prediction settings.

5 Experimental Evaluation

Using `CARLA` we conduct extensive empirical evaluations to benchmark the presented counterfactual explanations methods using three real-world data sets. Our main findings are displayed in Figure 3, and Table 2. We split the benchmarking evaluation by CE method category. In the following Sections, we provide an overview over the used data sets (see Table 3) and the classification models. Detailed information on hyperparameter search for the CE methods is provided in Appendix E.

Data sets The **Adult** data set [12] originates from the 1994 Census database, consisting of 14 attributes and 48,842 instances. The classification consists of deciding whether an individual has an income greater than 50,000 USD/year. Since several CE methods cannot handle non-binary categorical data, we binarized these features by partitioning them into the most frequent value, and its counterpart (e.g., *US* and *Non-US*, *Husband* and *Non-Husband*). The features *age*, *sex* and *race* are set as immutable. The **Give Me Some Credit** (GMC) data set [7] from a 2011 Kaggle Competition is a credit scoring data set, consisting of 150,000 observations and 11 features. The classification task consists of deciding whether an instance will experience financial distress within the next two years (*SeriousDlqin2yrs* is 1) or not. We dropped missing data, and set *age* as immutable.

Data Set	Task	Positive Class	Size (N | d)	Features	Immutable Features
Adult	Predict Income	High Income (24%)	(45,222 | 20)	Work, Education, Income	Sex, Age, Race
COMPAS	Predict Recidivism	No Recid. (65%)	(10,000 | 8)	Crim. History, Jail & Prison Time	Sex, Race
GMC	Predict Financial Distress	No deficiency (93%)	(150,000 | 11)	Pay. History, Balance, Loans	Age

Table 3: Summarizing the used data sets, where N and d are the number of samples and input dimension after processing of the data. Results on the COMPAS data set are relegated to Appendix B.

Black-box models We briefly describe how the black–box classifiers f were trained. `CARLA` supports different ML libraries (e.g., Pytorch, Tensorflow) to estimate these classifiers as the implementations of the various explanation methods work particular ML libraries only. The first model is a multi-layer perceptron, consisting of three hidden layers with 18, 9 and 3 neurons, respectively. To allow a more extensive comparison (`AR` only works on linear models) between CE methods, we chose logistic regression models as the second classification model for which we evaluate the CE methods. Detailed information on the classifiers' training for each data set is provided in Appendix C.

[3]We do not consider all possible subsets of changes.

Benchmarking For **independence based** methods, we find that no one single CE method outperformed all its competitors. This is not too surprising since algorithmic recourse is a multi–modal problem. Instead, we found that some methods dominated certain measures across all data sets. AR, AR-LIME, DICE performed strongest with respect to ℓ_0 (see the top left panels in Figures 3a and 3b). AR-LIME does so despite our use of the LIME reduction. Therefore, it makes sense that AR, AR-LIME and DICE offer the lowest *redundancy* scores (Table 2a). CEM performed strongest with respect to the overall cost measure ℓ_1 across data sets. GS is the clear winner when it comes to the measurement of time (Table 2a). Since the algorithm behind GS is based on a rather rudimentary sampling strategy, we expect that savvier sampling strategies should boost its cost performance significantly.

For **dependence based** methods, the results are mixed as well. While CLUE and REVISE are the winner with respect to the cost of recourse (ℓ_1), the margins between these generative recourse models and the graph-based ones (FACE) are small (Figure 3). The FACE-EPS method performs strongest with respect to the ynn measure (usually well above 0.60) (Table 2b) indicating that the generated CEs have sufficient data support from positively classified individuals relatively to the remaining dependence–based methods. As expected, the ynn measures are on average higher for the dependence based methods. This suggest that dependence based CEs are less often outliers. Notably, CLUE and REVISE perform best with respect to ℓ_1 (with REVISE being the clear winner on 3 out of 4 cases), while they perform worst on ℓ_0 – likely due to the decoder's imprecise reconstruction. In this respect, it is not surprising that these methods have average redundancy values that are up to twice as high as those by FACE. Finally, the generative model approaches (CEM-VAE, CLUE, REVISE) performed best with respect to time since the autoencoder training time amortizes with more samples.

6 Conclusion and Broader Impact of CARLA

The current implementations of recourse methods, mentioned in Section 4.1 are based on the original implementation of the respective research groups. Researchers mostly implement their experiments and models for specific ML frameworks and data sets. For example, some explanation methods are restricted to Tensorflow and are not applicable to Pytorch models. In the future, we will extend CARLA to decouple each recourse method from the frameworks and data contraints.

When trying to combine different CE methods into a common benchmarking framework we encountered the following issues: First, a great number of repositories only contain remarks about installation and script calls to recreate the results from the corresponding research papers. Second, missing information about interfaces for data sets or black–box models further complicated the process of integrating different CE methods into the benchmarking workflow. In order to add more CE methods and data sets to CARLA, we are currently in contact with several authors in this exciting and rapidly growing field. With a growing open-source community, CARLA can evolve to be the main library for generating counterfactual explanations and benchmarks for recourse methods. Therefore we are continuously expanding the catalog of explanation methods and data sets, and welcome researchers to add their own recourse methods to the library. To facilitate this process, we provide a step-by-step user-guide to integrate new CE methods into CARLA, which we present in Appendix A.

The rapidly growing number of available CE methods calls for standardized and efficient ways to assure the quality of a new technique in comparison with other approaches on different data sets. Quality assurance is a key aspect of actionable recourse, since complex models and CE mechanisms can have a considerable impact on personal lives. In this work, we presented CARLA, a versatile benchmarking platform for the standardized and transparent comparison of CE methods on different integrated data sets. In the explainability field, CARLA bears the potential to help researchers and practitioners alike to efficiently derive more realistic and use–case–driven recourse strategies and assure their quality through extensive comparative evaluations. We hope that this work contributes to further advances in explainability research.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX}*.

[2] David Alvarez-Melis and Tommi S Jaakkola. 2018. Towards robust interpretability with self-explaining neural networks. In *Conference on Neural Information Processing Systems (NeurIPS)*.

[3] Javier Antorán, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. 2021. Getting a CLUE: A Method for Explaining Uncertainty Estimates. In *International Conference on Learning Representations (ICLR)*.

[4] Richard Berk. 2012. *Criminal justice forecasts of risk: A machine learning approach*. Springer Science & Business Media.

[5] Klaus Broelemann and Gjergji Kasneci. 2019. A gradient-based split criterion for highly accurate and transparent model trees. In *Proceedings of the International Joint Conference on Artificial Intelligence IJCAI*.

[6] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. 2018. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE.

[7] Kaggle Competition. [n.d.]. "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".

[8] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. 2020. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*. Springer, 448–469.

[9] Houtao Deng. 2019. Interpreting tree ensembles with intrees. *International Journal of Data Science and Analytics* (2019).

[10] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. 2018. Explanations based on the Missing: Towards Contrastive Explanations with Pertinent Negatives. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[11] Michael Downs, Jonathan L. Chu, Yaniv Yacoby, Finale Doshi-Velez, and Weiwei Pan. 2020. CRUDS: Counterfactual Recourse Using Disentangled Subspaces. *ICML Workshop on Human Interpretability in Machine Learning (WHI 2020)* (2020).

[12] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository.

[13] Meherwar Fatima, Maruf Pasha, et al. 2017. Survey of machine learning algorithms for disease diagnostic. *Journal of Intelligent Learning Systems and Applications* (2017).

[14] Krishna Gade, Sahin Cem Geyik, Krishnaram Kenthapadi, Varun Mithal, and Ankur Taly. 2019. Explainable AI in industry. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*.

[15] Azin Ghazimatin, Oana Balalau, Rishiraj Saha Roy, and Gerhard Weikum. 2020. PRINCE: provider-side interpretability with counterfactual explanations in recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM)*.

[16] Yash Goyal, Amir Feder, Uri Shalit, and Been Kim. 2019. Explaining classifiers with causal concept effect (cace). *arXiv preprint arXiv:1907.07165* (2019).

[17] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. (2018).

[18] Vivek Gupta, Pegah Nokhiz, Chitradeep Dutta Roy, and Suresh Venkatasubramanian. 2019. Equalizing recourse across groups. *arXiv preprint arXiv:1909.03166* (2019).

[19] Satoshi Hara and Kohei Hayashi. 2018. Making tree ensembles interpretable: A bayesian model selection approach. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR.

[20] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. 2019. Towards Realistic Individual Recourse and Actionable Explanations in Black-Box Decision Making Systems. *arXiv preprint arXiv:1907.09615* (2019).

[21] Surya Mattu Julia Angwin, Jeff Larson and Lauren Kirchner. 2016. Machine bias: There's software used across the country to predict future criminals. And it's biased against blacks".

[22] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Hiroki Arimura. 2020. DACE: Distribution-Aware Counterfactual Explanation by Mixed-Integer Linear Optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*.

[23] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. 2020. Model-Agnostic Counterfactual Explanations for Consequential Decisions. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[24] Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. 2020. A survey of algorithmic recourse: definitions, formulations, solutions, and prospects. *arXiv preprint arXiv:2010.04050* (2020).

[25] Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. 2021. Algorithmic Recourse: from Counterfactual Explanations to Interventions. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAccT)*.

[26] Amir-Hossein Karimi, Julius von Kügelgen, Bernhard Schölkopf, and Isabel Valera. 2020. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. In *Conference on Neural Information Processing Systems (NeurIPS)*.

[27] Gjergji Kasneci and Thomas Gottron. 2016. Licon: A linear weighting scheme for the contribution ofinput variables in deep artificial neural networks. In *CIKM*.

[28] Eoin M. Kenny and Mark T. Keane. 2020. On Generating Plausible Counterfactual and Semi-Factual Explanations for Deep Learning. arXiv:2009.06399 [cs.LG]

[29] Amir E. Khandani, Adlar J. Kim, and Andrew W. Lo. 2010. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance* (2010).

[30] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. In *International Conference on Learning Representations*.

[31] Michael T Lash, Qihang Lin, Nick Street, Jennifer G Robinson, and Jeffrey Ohlmann. 2017. Generalized inverse classification. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM.

[32] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. 2017. Inverse Classification for Comparison-based Interpretability in Machine Learning. *arXiv preprint arXiv:1712.08443* (2017).

[33] Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. 2019. The Dangers of Post-hoc Interpretability: Unjustified Counterfactual Explanations. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI*.

[34] Yin Lou, Rich Caruana, and Johannes Gehrke. 2012. Intelligible models for classification and regression. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*.

[35] Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Conference on Neural Information Processing Systems (NeurIPS)*.

[36] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. From local explanations to global understanding with explainable AI for trees. *Nature machine intelligence* (2020).

[37] Divyat Mahajan, Chenhao Tan, and Amit Sharma. 2019. Preserving causal constraints in counterfactual explanations for machine learning classifiers. *arXiv preprint arXiv:1912.03277* (2019).

[38] Brent Mittelstadt, Chris Russell, and Sandra Wachter. 2019. Explaining explanations in AI. In *Proceedings of the conference on fairness, accountability, and transparency (FAT*).*

[39] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*).*

[40] R. K. Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining machine learning classifiers through diverse counterfactual explanations.

[41] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. 2020. Handling incomplete heterogeneous data using vaes. *Pattern Recognition* (2020).

[42] Matthew O'Shaughnessy, Gregory Canal, Marissa Connor, Mark Davenport, and Christopher Rozell. 2020. Generative causal explanations of black-box classifiers. In *Conference on Neural Information Processing Systems (NeurIPS).*

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703* (2019).

[44] Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. 2020. Learning Model-Agnostic Counterfactual Explanations for Tabular Data. In *Proceedings of The Web Conference 2020 (WWW).* ACM.

[45] Martin Pawelczyk, Klaus Broelemann, and Gjergji Kasneci. 2020. On Counterfactual Explanations under Predictive Multiplicity. In *Conference on Uncertainty in Artificial Intelligence (UAI).* PMLR, 809–818.

[46] Martin Pawelczyk, Shalmali Joshi, Chirag Agarwal, Sohini Upadhyay, and Himabindu Lakkaraju. 2021. On the Connections between Counterfactual Explanations and Adversarial Examples. arXiv:2106.09992 [cs.LG]

[47] Judea Pearl. 2009. *Causality.* Cambridge university press.

[48] Rafael Poyiadzi, Kacper Sokol, Raul Santos-Rodriguez, Tijl De Bie, and Peter Flach. 2020. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES).*

[49] Kaivalya Rawal and Himabindu Lakkaraju. 2020. Beyond Individualized Recourse: Interpretable and Interactive Summaries of Actionable Recourses. In *Conference on Neural Information Processing Systems (NeurIPS).*

[50] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD).* ACM.

[51] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI).*

[52] Christopher Russell. 2019. Efficient Search for Diverse Coherent Explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*).*

[53] Lisa Schut, Oscar Key, Rory Mc Grath, Luca Costabello, Bogdan Sacaleanu, Yarin Gal, et al. 2021. Generating Interpretable Counterfactual Explanations By Implicit Minimisation of Epistemic and Aleatoric Uncertainties. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1756–1764.

[54] Ilia Stepin, Jose M Alonso, Alejandro Catala, and Martín Pereira-Fariña. 2021. A Survey of Contrastive and Counterfactual Explanation Generation Methods for Explainable Artificial Intelligence. *IEEE Access* (2021).

[55] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* (2012).

[56] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. 2017. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. ACM.

[57] Berk Ustun, Alexander Spangher, and Yang Liu. 2019. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*.

[58] Berk Ustun, Alexander Spangher, and Y. Liu. 2019. Actionable Recourse in Linear Classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*.

[59] Arnaud Van Looveren and Janis Klaise. 2019. Interpretable counterfactual explanations guided by prototypes. *arXiv preprint arXiv:1907.02584* (2019).

[60] Sahil Verma, John Dickerson, and Keegan Hines. 2020. Counterfactual Explanations for Machine Learning: A Review. *arXiv preprint arXiv:2010.10596* (2020).

[61] Sandra Wachter, B. Mittelstadt, and Chris Russell. 2017. Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *Cybersecurity* (2017).

[62] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2018. Counterfactual explanations without opening the black box: automated decisions and the GDPR. *Harvard Journal of Law & Technology* 31, 2 (2018).

[63] Danding Wang, Qian Yang, Ashraf Abdul, and Brian Y Lim. 2019. Designing theory-driven user-centric explainable AI. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–15.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] . As we state in the abstract, our goal is to provide a Python framework for benchmarking counterfactual explanation methods. Users can easily evaluate our results by accessing our Github repository, where we host our Python framework and our benchmarking results.
   (b) Did you describe the limitations of your work? [Yes] . In Section 6, we discuss the current limitations of our approach. The counterfactual explanation methods are based on the original implementation of the respective research groups. Researchers mostly implement their experiments and models for specific ML frameworks and data sets. For example, some explanation methods are restricted to Tensorflow and are not applicable to Pytorch models.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] . We discuss the broader impact of our benchmarking library in Section 6; we mainly see positive impacts on the literature of algorithmic recourse.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] . We have read the ethics review guidelines and attest that our paper conforms to the guidelines.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] . We did not provide theoretical results.
   (b) Did you include complete proofs of all theoretical results? [N/A] . We did not provide theoretical results.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] . Details of implementations, data sets and instructions can be found here: Appendices A, C, E, and our Github repository.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] . Please see Appendices E and C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] . Error bars have been reported for our cost comparisons in terms of the 25th and 75ht percentiles of the cost distribution, see for example Figure 3.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] . All models are evaluated on an i7-8550U CPU with 16 Gb RAM, running on Windows 10.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] . The data sets, which are publicly available are appropriately cited in Section 5. We cite and link to any additional code used, for example [3].
   (b) Did you mention the license of the assets? [Yes] . All assets are publicly available and attributed.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] . Our implementation and code is accessible through our Github repository.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] . We use publicly available data sets without any personal identifying information.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] . We use publicly available data sets without any personal identifying information.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] . We did not use crowdsourcing or conduct research with human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] . We did not use crowdsourcing or conduct research with human subjects.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] . We did not use crowdsourcing or conduct research with human subjects.

A CARLA's Software Interface

In the following, we introduce our open-source benchmarking software CARLA. we describe the architecture in more detail and provide examples of different use-cases and their implementation.

A.1 CARLA's High Level Software Architecture

The purpose of this Python library is to provide a simple and standardized framework to allow users to apply different state-of-the-art recourse methods to arbitrary data sets and black-box-models. It is possible to compare different approaches and save the evaluation results, as described in Section 4.2. For research groups, CARLA provides an implementation interface to integrate new recourse methods in an easy-to-use way, which allows to compare their method to already existing methods.

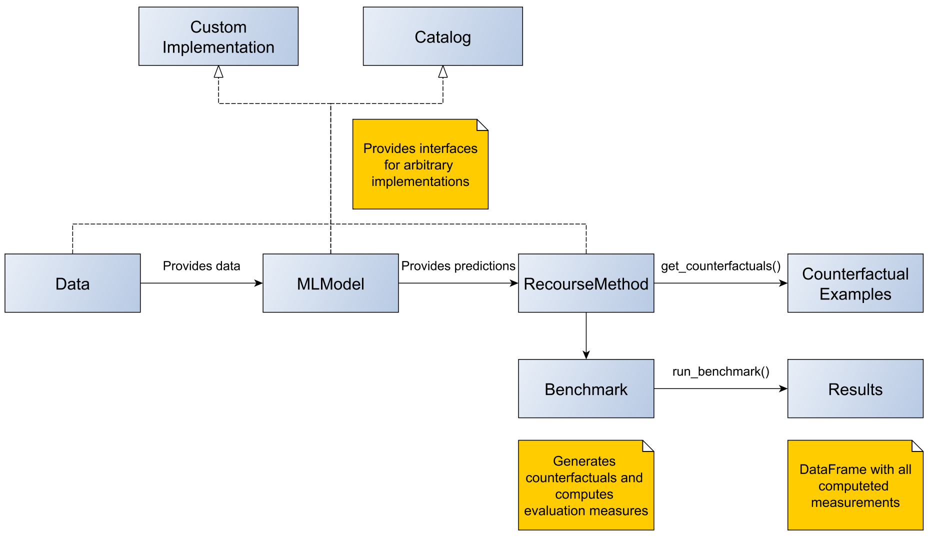


Figure 4: Architecture of the CARLA python library. The silver boxes show the individual objects that will be created to generate counterfactual explanations and evaluate recourse methods. Useful explanations to specific processes are illustrated as yellow notes. The dashed arrows are showing the different implementation possibilities, either use pre-defined catalog objects or provide custom implementation. All dependencies between these objects are visualised by solid arrows with an additional description.

A simplified visualization of the CARLA software architecture is depicted in Figure 4. For every component (*Data*, *MLModel*, and *RecourseMethod*) the library provides the possibility to use existing methods from our catalog, or extend the users custom methods and implementations. The components represent an interface to the key parts in the process of generating counterfactual explanations. *Data* provides a common way to access the data across the software and maintains information about the features. *MLModel* wraps each black-box model and stores details on the encoding, scaling and feature order specific to the model. The primary purpose of *RecourseMethod* is to provide a common interface to easily generate counterfactual examples.

Besides the possibility to use pretrained black-box-models and preprocessed data, CARLA provides an easy way to load and define own data sets and model structures independent of their framework (e.g., Pytorch, Tensorflow, sklearn). The following sections will give an overview and provide example implementations of different use cases.

A.2 CARLA for Research Groups

One of the most exciting features of CARLA is, that research groups can make use of the *RecourseMethod*-wrapper to implement their own method to generate counterfactual examples. This opens up a way of standardized and consistent comparisons between different recourse methods.

Strong and weak points of new algorithms can be stated, benchmarked and analysed in forthcoming publications with the help of CARLA.

In Figure 5, we show how an implementation of a custom recourse method can be structured. After defining the recourse method in the shown way, it can be used with the library to generate counterfactuals for a given data set and benchmark its results against other methods. Research groups have the choice to do this using our provided catalog of data sets, recourse methods and black-box models (Figure 6) or use their own models and data sets (see Figures 7 and 8).

```python

from carla import RecourseMethod

# Custom recourse implementations need to
# inherit from the RecourseMethod interface
class MyRecourseMethod(RecourseMethod):
    def __init__(self, mlmodel):
        super().__init__(mlmodel)

    # Generate and return encoded and
    # scaled counterfactual examples
    def get_counterfactuals(self, factuals: pd.DataFrame):
    [...]
    return counterfactual_examples

```

Figure 5: Pseudo-implementation of the CARLA recourse method wrapper

A.3 CARLA as a Recourse Library

A common usage of the package is to generate counterfactual examples. This can be done by loading black-box-models and data sets from our provided catalogs, or by user-defined models and datasets via integration with the defined interfaces. Figure 6 shows an implementation example of a simple use-case, applying a recourse method to a pre-defined data set and model from our catalog. After importing both catalogs, the only necessary step is to describe the data set name (e.g., adult, give me some credit, or compas) and the model type (e.g., ann, or linear) the user wants to load. Every recourse method contains the same properties to generate counterfactual examples.

```python

from carla import DataCatalog, MLModelCatalog
from carla.recourse_methods import GrowingSpheres

# 1. Load data set from the DataCatalog
data_name = "adult"
dataset = DataCatalog(data_name)

# 2. Load pre-trained black-box model from the MLModelCatalog
model = MLModelCatalog(dataset, "ann")

# 3. Load recourse model with model specific hyperparameters
gs = GrowingSpheres(model)

# 4. Generate counterfactual examples
factuals = dataset.raw.sample(10)
counterfactuals = gs.get_counterfactuals(factuals)

```

Figure 6: Example implementation of CARLA, using the data and model catalog.

To give users the possiblity to explore their own black-box-model on a custom data set, we implemented in CARLA easy-to-use interfaces, that are able to wrap every possible model or data set. These interfaces specify particular properties users have to implement, to be able to work with the library.

Figure 7 shows an example implementation of the data wrapper, and Figure 8 depicts the same for an arbitrary black-box-model. After defining data set and black-box model classes, users simply need to call the canonical methods and generate counterfactual examples, similar to the process in Figure 6.

```python

from carla import Data
from carla.recourse_methods import GrowingSpheres

# Custom data set implementations need to inherit from the Data
interface
class MyOwnDataSet(Data):
    def __init__(self):
        # The data set can e.g. be loaded in the constructor
        self._dataset = load_dataset_from_disk()

    # List of all categorical features
    def categoricals(self):
        return [...]

    # List of all continous features
    def continous(self):
        return [...]

    # List of all immutable features which
    # should not be changed by the recourse method
    def immutables(self):
        return [...]

    # Feature name of the target column
    def target(self):
        return "label"

    # Non-encoded and  non-normalized, raw data set
    def raw(self):
        return self._dataset

```

Figure 7: Pseudo-implementation of the CARLA data wrapper

A.4 Benchmarking Recourse Methods

Besides the generation of counterfactual examples, the focus of CARLA lies on benchmarking recourse methods. Users are able to compute evaluation measures to make qualitative statements about usability and applicability.

All measurements, which are described in Section 4.2, are implemented in the *Benchmarking* class of CARLA and can be used for every wrapped recourse method. Figure 9 shows an example implementation of a benchmarking process based on the variables of Figure 6.

B Additional Experimental Results

In this Section, we depict the missing experiments from the COMPAS data set in Figure 10 and Table 4. These results underline the trends that we have already highlighted in Section 5.

C ML Classifiers

In this section, we describe how the black–box models f were fitted. CARLA supports different ML libraries to estimate these models (e.g., Pytorch, Tensorflow) as the implementations of the various explanation methods work with a particular ML library. We note that the various explanation methods rely on different binary feature encodings. DICE, for example, requires that binary inputs are supplied as one–hot vectors, while FACE needs binary features encoded in a single column. If this was the

```
from carla import MLModel

# Custom black-box models need to inherit from
# the MLModel interface
class MyOwnModel(MLModel):
    def __init__(self, data):
        super().__init__(data)
        # The constructor can be used to load or build an
        # arbitrary black-box-model
        self._mymodel = load_model()

        # Define a fitted sklearn scaler to normalize input data
        self.scaler = MySklearnScaler().fit()

        # Define a fitted sklearn encoder for binary input data
        self.encoder = MySklearnEncoder.fit()

    # List of the feature order the ml model was trained on
    def feature_input_order(self):
        return [...]

    # The ML framework the model was trained on
    def backend(self):
        return "pytorch"

    # The black-box model object
    def raw_model(self):
        return self._mymodel

    # The predict function outputs
    # the continous prediction of the model
    def predict(self, x):
        return self._mymodel.predict(x)

    # The predict_proba method outputs
    # the prediction as class probabilities
    def predict_proba(self, x):
        return self._mymodel.predict_proba(x)

```

Figure 8: Pseudo-implementation of the CARLA black-box-model wrapper

case, we fitted two ML models, using the same hyperparameters, and generated CEs with respect to the same set of samples.

To ensure similar behavior between the different ML libraries and encoding variations, each black-box model type has the same structure (e.g., number of hidden layer, number of neurons), and training parameters (e.g., learning rate, epochs, etc.).

The first model is a multi-layer perceptron, consisting of three hidden layers with 18, 9 and 3 neurons, respectively. We use ReLu activation functions and binary cross entropy to calculate class probabilities. Optimization of the loss function is done by RMSProp [55] using a learning rate of 0.002 for every data set. By performing 25 epochs on COMPAS and 10 epochs on Adult and GMC we reached acceptable performance. Further increasing epochs gave rise to very marginal performance increases. For Adult we use a batch–size of 1024, for COMPAS 25 and for GMC 2048.

To allow a more extensive comparison between CE methods, we choose linear models as the second black–box model category for which we evaluate the CE methods. Again, we optimized these models with RMSProp using a binary cross entropy loss. For Adult, we used 100 epochs and a batch–size of 2048, for COMPAS we choose 25 epochs and batch–size of 128, and for GMC we chose 10 epochs

```python

from carla import Benchmark

# 1. Initilize the benchmarking class by passing
# black-box-model, recourse method, and factuals into it
benchmark = Benchmark(model, gs, factuals)

# 2. Either only compute the distance measures
distances = benchmark.compute_distances()

# 3. Or run all implemented measurements and create a
# DataFrame which consists of all results
results = benchmark.run_benchmark()

```

Figure 9: Pseudo-implementation of the CARLA recourse method wrapper

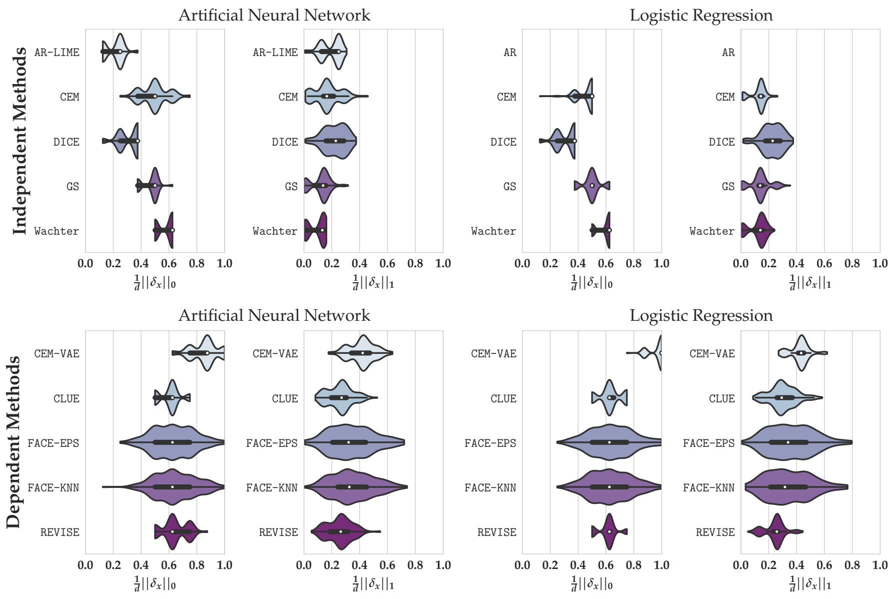


Figure 10: COMPAS Data

Figure 11: Evaluating the distribution of costs of counterfactual explanations on the COMPAS dataset. For all instances with a negative prediction ($\{x \in \mathcal{D} : f(x) < \theta\}$), we plot the distribution of ℓ_0 and ℓ_1 costs of algorithmic recourse as defined in (1) for a logistic regression and an artificial neural network classifier. The white dots indicate the medians (lower is better), and the black boxes indicate the interquartile ranges. We distinguish between independence based and dependence based methods.

with a batch–size of 2048. The learning rate on every data set is set 0.002. Table 5 provides an overview of the model's classification accuracies.

	Adult	COMPAS	Give Me Credit
Logistic Regression	0.83	0.84	0.92
Neural Network	0.84	0.85	0.93

Table 5: Performance of classification models used for generating algortihmic recourse.

		Artificial Neural Network					Logistic Regression				
Data Set	Method	*yNN*	redund.	violation	success	$\bar{t}(s)$	*yNN*	redund.	violation	success	$\bar{t}(s)$
COMPAS	`AR(-LIME)`	**0.91**	**0.00**	**0.02**	0.53	0.06	–	–	–	0.00	**0.01**
	`CEM`	**0.98**	2.29	0.43	**1.00**	0.89	0.93	1.88	0.99	**1.00**	0.86
	`DICE`	0.89	0.88	1.03	**1.00**	0.09	**0.95**	0.94	0.90	**1.00**	0.09
	`GS`	0.44	0.97	0.03	**1.00**	**0.01**	0.60	**0.64**	**0.02**	**1.00**	**0.01**
	`Wachter`	0.56	1.77	0.74	0.66	10.90	0.50	1.21	0.79	**1.00**	0.02

(a) Independence based methods

		Artificial Neural Network					Logistic Regression				
Data Set	Method	*yNN*	redund.	violation	success	$\bar{t}(s)$	*yNN*	redund.	violation	success	$\bar{t}(s)$
COMPAS	`CEM-VAE`	**1.00**	5.59	1.98	**1.00**	0.89	1.00	6.91	2.14	**1.00**	0.87
	`CLUE`	0.99	4.06	**1.08**	**1.00**	2.03	**1.00**	4.62	1.25	**1.00**	1.88
	`FACE-EPS`	0.94	3.71	1.55	0.99	0.45	0.97	3.94	1.62	0.99	0.45
	`FACE-KNN`	0.94	3.83	1.63	**1.00**	**0.44**	0.97	3.86	1.57	**1.00**	**0.44**
	`REVISE`	**1.00**	**3.29**	1.29	**1.00**	6.06	0.92	**3.15**	**1.03**	**1.00**	5.35

(b) Dependence based methods

Table 4: Summary of COMPAS results for independence and dependence based methods. For all instances with a negative prediction ($\{x \in \mathcal{D} : f(x) < \theta\}$), we compute counterfactual explanations for which we then measure yNN (higher is better), redundancy (lower is better), violation (lower is better), success rate (higher is better) and time (lower is better). We distinguish between a logistic regression and an artificial neural network classifier. Detailed descriptions of these measures can be found in Section 4. The results are discussed in Appendix B.

D COMPAS Data Set Description

The **COMPAS** data set [21] contains data for more than 10,000 criminal defendants in Florida. It is used by the jurisdiction to score defendant's likelihood of reoffending. We kept a small part of the raw data as features like *name*, *id*, *casenumbers* or *date-time* were dropped. The classification task consists of classifying an instance into high risk of recidivism (*score_text* is high). By converting the feature *race* into *white* and *non-white*, we keep the categorical input binary. Similar to Adult, the immutable features for COMPAS are *age, sex* and *race*.

E Hyperparameter Search for the Counterfactual Explanation and Recourse Methods

We generated counterfactual explanations for instances from H^-, the set of factuals with negative class predictions.

`AR` and `AR-LIME` It frequently occurred that the action with the lowest cost did not flip the prediction of the black-box classifier. To overcome this problem, we let `AR` compute a flipset of 150 actions per instance, and subsequently search this set for low–cost CEs. For `AR-LIME`, we used `LIME` [50] and required `sampling around the instance` to make sure that the coefficients at x were truly local.

`CEM` After performing grid search, we set the ℓ_1 weight to 0.9 and the ℓ_2 weight to 0.1, yielding the strongest performance. For `CEM-VAE` we set the ℓ_2 weight to 0.1, and the VAE–weight to 0.9.

`CLUE` We use the default hyperparameters from [3], which are set as a function of the data set dimension d. Performing hyperparameter search did not yield results that were improving distances while keeping the same success rate.

`DICE` Since `DICE` is able to compute a set of counterfactuals for a given instance, we only chose to generate one CE per input instance. We use a *grid search* for the *proximity* and *diversity* weights.

`FACE` To determine the strongest hyperparameters for the graph size we conducted a *grid search*. We found that values of $k_{FACE} = 50$ gave rise to the best balance of success rate and costs. For the epsilon graph, a radius of 0.25 yields the strongest results to balance between high yNN and low cost.

`GS` We chose 0.02 as the step size with which the sphere is grown. Lower values yield similar results at the costs of higher computational time, while higher values gave worse results.

`REVISE` The *grid search* to find an acceptable learning rate and similarity weight λ yielded $\eta = 0.1$ and $\lambda = 0.5$ for about 1500 iterations.

`Wachter` For the target loss, we choose the Binary Cross Entropy with a learning rate of 0.01 and an initial λ of 0.01. For the distance loss, we use the ℓ_1- norm to measure the similarity between the factual input and the counterfactual point $\check{x}$.