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 (Counterfactual And Recourse Library), 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

*Corresponding author
†Equal senior author contribution

rejection of a loan or the denial of parole might have negative effects on the future development of the corresponding person’s life.

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\(^3\) – popularized by the seminal work of [61] – 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 [53, 24, 59]), 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.

In this work, we present CARLA (Counterfactual And Recourse LibrAry), 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 explanation 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 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.

\(^3\)The terms counterfactual explanations [61], contrastive explanations [24], and algorithmic recourse [56] have been used interchangeably in prior work. We use these terms interchangeably to refer to the notion introduced by Wachter et al. [61].
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, 62]. 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 [49, 35, 50, 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 [56, 51, 48, 23, 28, 8]. Notable exceptions from this line of work are proposed by [55, 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, 52, 58]. 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 [46]. 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 do not consider them further.

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 [47, 22]. The authors of [47] 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.
We assume the factual input \( z \) would ensure that the counterfactual latent space lies within range of \( A \). Assuming inputs are pairwise statistically independent, the recourse problem is defined as follows:

\[
\delta^*_x = \arg \min_{\delta_x \in A_d} c(x, \hat{x}) \text{ s.t. } \hat{x} = x + \delta_x, f(\hat{x}) > \theta,
\]

where \( A_d \) is the set of admissible changes made to the factual input \( x \). For example, \( A_d \) 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 [56] 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 \( \hat{x} = x + \delta_x \). Thus, we have \( \hat{x} = x + \delta_x = g(z + \delta_z) \). Assuming inputs are dependent, we can rewrite the recourse problem in (I) to faithfully capture those dependencies using the generative model \( g \):

\[
\delta^*_z = \arg \min_{\delta_z \in A_k} c(x, \hat{x}) \text{ s.t. } \hat{x} = g(z + \delta_z), f(\hat{x}) > \theta,
\]

where \( A_k \) is the set of admissible changes in the \( k \)-dimensional latent space. For example, \( A_k \) 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.
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 ($\{ x \in D : f(x) < \theta \}$), we plot the distribution of $\ell_0$ and $\ell_1$ 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.
<table>
<thead>
<tr>
<th>Approach</th>
<th>Method</th>
<th>Model Type</th>
<th>Algorithm</th>
<th>Immutable</th>
<th>Categorical</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
<td>Linear</td>
<td>Integer Prog.</td>
<td>Yes</td>
<td>Binary only</td>
<td>Direction of change</td>
<td></td>
</tr>
<tr>
<td>AR-LIME</td>
<td>Agnostic</td>
<td>Integer Prog.</td>
<td>Yes</td>
<td>Binary only</td>
<td>Direction of change</td>
<td></td>
</tr>
<tr>
<td>CEM</td>
<td>Gradient based</td>
<td>Gradient based</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>DICE</td>
<td>Gradient Based</td>
<td>Gradient Based</td>
<td>Yes</td>
<td>Binary Only</td>
<td>Generative model</td>
<td></td>
</tr>
<tr>
<td>GS</td>
<td>Agnostic</td>
<td>Random search</td>
<td>Yes</td>
<td>Binary Only</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Wachter</td>
<td>Gradient based</td>
<td>Gradient based</td>
<td>No</td>
<td>Binary Only</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>CEM-VAE</td>
<td>Gradient based</td>
<td>Gradient based</td>
<td>No</td>
<td>No</td>
<td>Gen. Model regularizer</td>
<td></td>
</tr>
<tr>
<td>CLUE</td>
<td>Gradient based</td>
<td>Gradient based</td>
<td>No</td>
<td>No</td>
<td>Generative model</td>
<td></td>
</tr>
<tr>
<td>FACE-EPS</td>
<td>Agnostic</td>
<td>Graph search</td>
<td>Binary Only</td>
<td>Binary Only</td>
<td>CE is from data set</td>
<td></td>
</tr>
<tr>
<td>FACE-KNN</td>
<td>Agnostic</td>
<td>Graph search</td>
<td>Binary Only</td>
<td>Binary Only</td>
<td>CE is from data set</td>
<td></td>
</tr>
<tr>
<td>REVISE</td>
<td>Gradient based</td>
<td>Gradient based</td>
<td>Binary Only</td>
<td>Binary Only</td>
<td>Generative model</td>
<td></td>
</tr>
<tr>
<td>GROWING SPHERES</td>
<td>Independent (I)</td>
<td>Categorical</td>
<td>Any other outstanding characteristics (Other)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GROWING SPHERES</td>
<td>Dependent (D)</td>
<td>No</td>
<td>Direction of change</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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).

4.1 Counterfactual Explanation Methods

AR (I) Ustun et al. [57] 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.

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 [49]. 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 ℓ1 and ℓ2 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 [47] 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
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 essential to define the distance of the factual instance, while the \[ \ell_1 \] norm restricts the average change:

\[
c_0(\hat{x}, x) = \frac{1}{d} \| x - \hat{x} \|_0, \quad c_1(\hat{x}, x) = \frac{1}{d} \| x - \hat{x} \|_1 = \frac{1}{d} \| \delta x \|_1.
\] (1)

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 \( \hat{x} \). The literature has formed a consensus to use either the normalized \( \ell_0 \) or \( \ell_1 \) norm or any convex combination thereof (see for example [48, 39, 45, 23, 56, 61]). The \( \ell_0 \) norm puts a restriction on the number of feature changes between factual and counterfactual instance, while the \( \ell_1 \) norm restricts the average change:

\[
c_0(\hat{x}, x) = \frac{1}{d} \| x - \hat{x} \|_0, \quad c_1(\hat{x}, x) = \frac{1}{d} \| x - \hat{x} \|_1 = \frac{1}{d} \| \delta x \|_1.
\] (1)

### Table 2: Summary of a subset of results for independence and dependence based methods.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Method</th>
<th>( yNN )</th>
<th>( \text{redund.} )</th>
<th>( \text{violation} )</th>
<th>( \text{success} )</th>
<th>( \ell(s) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adult</td>
<td>AR(-LIME)</td>
<td>0.62</td>
<td>0.00</td>
<td>0.14</td>
<td>0.28</td>
<td>1.59</td>
</tr>
<tr>
<td></td>
<td>CEM</td>
<td>0.26</td>
<td>3.96</td>
<td>0.66</td>
<td>1.00</td>
<td>1.10</td>
</tr>
<tr>
<td></td>
<td>DICE</td>
<td>0.71</td>
<td>0.53</td>
<td>0.17</td>
<td>1.00</td>
<td>0.13</td>
</tr>
<tr>
<td></td>
<td>GS</td>
<td>0.30</td>
<td>3.77</td>
<td>0.09</td>
<td>1.00</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>Wachter</td>
<td>0.23</td>
<td>4.45</td>
<td>0.83</td>
<td>0.50</td>
<td>15.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \ell_0 )</td>
<td>1.00</td>
<td>2.33</td>
<td>0.14</td>
<td>0.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \ell_1 )</td>
<td>0.72</td>
<td>0.67</td>
<td>0.13</td>
<td>0.52</td>
</tr>
<tr>
<td>GMC</td>
<td>AR(-LIME)</td>
<td>0.89</td>
<td>0.00</td>
<td>0.29</td>
<td>0.07</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>CEM</td>
<td>0.95</td>
<td>5.46</td>
<td>0.65</td>
<td>1.00</td>
<td>0.97</td>
</tr>
<tr>
<td></td>
<td>DICE</td>
<td>0.90</td>
<td>0.58</td>
<td>0.27</td>
<td>1.00</td>
<td>0.28</td>
</tr>
<tr>
<td></td>
<td>GS</td>
<td>0.40</td>
<td>6.64</td>
<td>0.17</td>
<td>1.00</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>Wachter</td>
<td>0.58</td>
<td>6.56</td>
<td>0.71</td>
<td>1.00</td>
<td>0.02</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \ell_0 )</td>
<td>1.00</td>
<td>2.33</td>
<td>0.14</td>
<td>0.39</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \ell_1 )</td>
<td>0.72</td>
<td>0.67</td>
<td>0.13</td>
<td>0.52</td>
</tr>
</tbody>
</table>

(a) Independence based methods

(b) Dependence based methods

The optimization approach suggested by Wachter et al. [60] forms a consensus to use either the normalized \( \ell_0 \) or \( \ell_1 \) norm or any convex combination thereof (see for example [48, 39, 45, 23, 56, 61]). The \( \ell_0 \) norm puts a restriction on the number of feature changes between factual and counterfactual instance, while the \( \ell_1 \) norm restricts the average change:

\[
c_0(\hat{x}, x) = \frac{1}{d} \| x - \hat{x} \|_0, \quad c_1(\hat{x}, x) = \frac{1}{d} \| x - \hat{x} \|_1 = \frac{1}{d} \| \delta x \|_1.
\] (1)
The measure 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 $\hat{H}^-$. We use a measure that captures how differently neighborhood points around a counterfactual instance $\hat{x}$ are classified:

$$y_{\text{NN}} = 1 - \frac{1}{nk} \sum_{i \in H^-} \sum_{j \in \text{nn}(\hat{x}_i)} |f_b(\hat{x}_i) - f_b(x_j)|,$$

(2)

where $\text{nn}$ denotes the $k$-nearest neighbours of $x$, and $f_b(x) = \mathbb{I}[f(x) > 0.5]$ is the binarized classifier. Values of $y_{\text{NN}}$ 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 $\hat{x}$ 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: $\mathbb{I}[f_b([\hat{x}_1, x_2, \hat{x}_3, \ldots, x_d]) = 0]$. 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.\footnote{We do not consider all possible subsets of changes.} 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.

**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.
When trying to combine different CE methods into a common benchmarking framework we encounter with respect to time since the autoencoder training time amortizes with more samples. We expect that savvier sampling strategies should boost its cost performance significantly.

The rapidly growing number of available CE methods calls for standardized and efficient ways to assure their quality through extensive comparative evaluations. We hope that this work contributes to further advances in explainability research.
References


[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”.


22 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.


[38] Brent Mittelstadt, Chris Russell, and Sandra Wachter. 2019. Explaining explanations in AI. In Proceedings of the conference on fairness, accountability, and transparency (FAT*).


Checklist

481 The checklist follows the references. Please read the checklist guidelines carefully for information on
482 how to answer these questions. For each question, change the default [TODO] to [Yes], [No], or
483 [N/A]. You are strongly encouraged to include a justification to your answer, either by referencing
484 the appropriate section of your paper or providing a brief inline description. For example:

485 • Did you include the license to the code and datasets? [Yes] See Section ??.
486 • Did you include the license to the code and datasets? [No] The code and the data are
487 proprietary.
488 • Did you include the license to the code and datasets? [N/A]

489 Please do not modify the questions and only use the provided macros for your answers. Note
490 that the Checklist section does not count towards the page limit. In your paper, please delete this
491 instructions block and only keep the Checklist section heading above along with the questions/answers
492 below.

1. For all authors...
493 (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s
494 contributions and scope? [Yes] As we state in the abstract, our goal is to provide
495 a Python framework for benchmarking counterfactual explanation methods. Users
496 can easily evaluate our results by accessing our Github repository, where we host our
497 Python framework and our benchmarking results.
498 (b) Did you describe the limitations of your work? [Yes]. In Section 6, we discuss the
499 current limitations of our approach. The counterfactual explanation methods are based
500 on the original implementation of the respective research groups. Researchers mostly
501 implement their experiments and models for specific ML frameworks and data sets. For
502 example, some explanation methods are restricted to Tensorflow and are not applicable
503 to Pytorch models.
504 (c) Did you discuss any potential negative societal impacts of your work? [N/A]. We
505 discuss the broader impact of our benchmarking library in Section 6; we mainly see
506 positive impacts on the literature of algorithmic recourse.
507 (d) Have you read the ethics review guidelines and ensured that your paper conforms to
508 them? [Yes]. We have read the ethics review guidelines and attest that our paper
509 conforms to the guidelines.

2. If you are including theoretical results...
510 (a) Did you state the full set of assumptions of all theoretical results? [N/A]. We did not
511 provide theoretical results.
512 (b) Did you include complete proofs of all theoretical results? [N/A]. We did not provide
513 theoretical results.

3. If you ran experiments...
514 (a) Did you include the code, data, and instructions needed to reproduce the main experi-
515 mental results (either in the supplemental material or as a URL)? [Yes]. Details of
516 implementations, data sets and instructions can be found here: Appendices A, C, E,
517 and our Github repository.
518 (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they
519 were chosen)? [Yes]. Please see Appendices E and C.
520 (c) Did you report error bars (e.g., with respect to the random seed after running experi-
521 ments multiple times)? [Yes]. Error bars have been reported for our cost comparisons
522 in terms of the 25th and 75th percentiles of the cost distribution, see for example Figure
523 3.
524 (d) Did you include the total amount of compute and the type of resources used (e.g., type
525 of GPUs, internal cluster, or cloud provider)? [Yes]. All models are evaluated on an
526 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.