Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations

Ramaravind K. Mothilal
Microsoft Research India
t-rakom@microsoft.com

Amit Sharma
Microsoft Research India
amshar@microsoft.com

Chenhao Tan
University of Colorado Boulder
chenhao.tan@colorado.edu

ABSTRACT

Post-hoc explanations of machine learning models are crucial for people to understand and act on algorithmic predictions. An intriguing class of explanations is through *counterfactuals*, hypothetical examples that show people how to obtain a different prediction. We posit that effective counterfactual explanations should satisfy two properties: *feasibility* of the counterfactual actions given user context and constraints, and *diversity* among the counterfactuals presented. To this end, we propose a framework for generating and evaluating a diverse set of counterfactual explanations based on determinantal point processes. To evaluate the actionability of counterfactuals, we provide metrics that enable comparison of counterfactual-based methods to other local explanation methods. We further address necessary tradeoffs and point to causal implications in optimizing for counterfactuals. Our experiments on four real-world datasets show that our framework can generate a set of counterfactuals that are diverse and well approximate local decision boundaries, outperforming prior approaches to generating diverse counterfactuals. We provide an implementation of the framework at https://github.com/microsoft/DiCE.

CCS CONCEPTS

• **Applied computing** → **Law, social and behavioral sciences**.

ACM Reference Format:
Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations. In *Conference on Fairness, Accountability, and Transparency (FAT* '20), January 27–30, 2020, Barcelona, Spain.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3351095.3372850

1 INTRODUCTION

Consider a person who applied for a loan and was rejected by the loan distribution algorithm of a financial company. Typically, the company may provide an explanation on why the loan was rejected, for example, due to "poor credit history". However, such an explanation does not help the person decide *what they should do next* to improve their chances of being approved in the future. Critically, the most important feature may not be enough to flip the decision of the algorithm, and in practice, may not even be changeable such as gender and race. Thus, it is equally important to show decision outcomes from the algorithm with *actionable* alternative profiles, to help people understand what they could have done to change their loan decision. Similar to the loan example, this argument is valid for a range of scenarios involving decision-making on an individual's outcome, such as deciding admission to a university [40], screening job applicants [33], disbursing government aid [3, 5], and identifying people at high risk of a future disease [11]. In all these cases, knowing reasons for a bad outcome is not enough; it is important to know what to do to obtain a better outcome in the future (assuming that the algorithm remains relatively static).

Counterfactual explanations [39] provide this information, by showing feature-perturbed versions of the same person who would have received the loan, e.g., "you would have received the loan if your income was higher by $10,000". In other words, they provide "what-if" explanations for model output. Unlike explanation methods that depend on approximating the classifier's decision boundary [32], counterfactual (CF) explanations have the advantage that they are always truthful w.r.t. the underlying model by giving direct outputs of the algorithm. Moreover, counterfactual examples may also be human-interpretable [39] by allowing users to explore "what-if" scenarios, similar to how children learn through counterfactual examples [6, 7, 41].

However, it is difficult to generate CF examples that are *actionable* for a person's situation. Continuing our loan decision example, a CF explanation may suggest to "change your house rent", but it does not say much about alternative counterfactuals, or consider the relative ease between different changes a person may need to make. Like any example-based decision support system [18], we need a *set* of counterfactual examples to help a person interpret a complex machine learning model. Ideally, these examples should balance between a wide range of suggested changes (*diversity*), and the relative ease of adopting those changes (*proximity* to the original input), and also follow the causal laws of human society, e.g., one can hardly lower their educational degree or change their race.

Indeed, Russell [34] recognizes the importance of diversity and proposes an approach for linear machine learning classifiers based on integer programming. In this work, we propose a method that generates sets of diverse counterfactual examples for any differentiable machine learning classifier. Extending Wachter et al. [39], we construct an optimization problem that considers the diversity of the generated CF examples, in addition to proximity to the original input. Solving the optimization problem requires considering the tradeoff between diversity and proximity, and the tradeoff between continuous and categorical features which may differ in their relative scale and ease of change. We provide a general solution to this optimization problem that can generate any number of CF examples for a given input. To facilitate actionability, our solution is flexible enough to support user-provided inputs based on domain knowledge, such as custom weights for individual features or constraints on perturbation of features.

Further, we provide quantitative evaluation metrics for evaluating any set of counterfactual examples. Due to their inherent subjectivity, CF examples are hard to evaluate. While we cannot replace behavioral experiments, we propose metrics that can help in fine-tuning parameters of the proposed solution to achieve desired properties of validity, diversity, and proximity. We also propose a

second evaluation metric that approximates a behavioral experiment on whether people can *understand* a ML model's decision given a set of CF examples, assuming that people would rationally extrapolate from the CF examples and "guess" the local decision boundary of an ML model.

We evaluate our method on explaining ML models trained on four datasets: COMPAS for bail decision [4], Adult-Income for income prediction [20], German-Credit for assessing credit risk [2], and a dataset from Lending Club for loan decisions [1]. Compared to prior CF generation methods, our proposed solution generates CF examples with substantially higher diversity for these datasets. Moreover, a simple 1-nearest neighbor model trained on the generated CF examples obtains comparable accuracy on locally approximating the original ML model to methods like LIME [32], which are directly optimized for estimating the local decision boundary. Notably, our method obtains higher F1 score on predicting instances in the counterfactual outcome class than LIME in most configurations, especially for Adult-Income and COMPAS datasets wherein both precision and recall are higher. Qualitative inspection of the generated CF examples illustrates their potential utility for making informed decisions. Additionally, CF explanations can expose biases in the original ML model, as we see when some of the generated explanations suggest changes in sensitive attributes like race or gender. The last example illustrates the broad applicability of CF explanations: they are not just useful to an end-user, but can be equally useful to model builders for debugging biases, and for fairness evaluators to discover such biases and other model properties.

Still, CF explanations, as generated, suffer from lack of any causal knowledge about the input features that they modify. Features do not exist in a vacuum; they come from a data-generating process which constrains their modification. Thus, perturbing each input feature independently can lead to infeasible examples, such as suggesting someone to obtain a higher degree but reduce their age. To ensure feasibility, we propose a filtering approach on the generated CF examples based on causal constraints.

To summarize, our work makes the following contributions:

- We propose diversity as an important component for actionable counterfactuals and build a general optimization framework that exposes the importance of necessary tradeoffs, causal implications, and optimization issues in generating counterfactuals.
- We propose a quantitative evaluation framework for counterfactuals that allows fine-tuning of the proposed method for a particular scenario and enables comparison of CF-based methods to other local explanation methods such as LIME.
- Finally, we demonstrate the effectiveness of our framework through empirical experiments on multiple datasets and provide an open-source implementation at https://github.com/microsoft/DiCE.

2 BACKGROUND & RELATED WORK

Explanations are critical for machine learning, especially as machine learning-based systems are being used to inform decisions in societally critical domains such as finance, healthcare, education, and criminal justice. Since many machine learning algorithms are black boxes to end users and do not provide guarantees on input-output relationship, explanations serve a useful role to inspect these models. Besides helping to debug ML models, explanations are hypothesized to improve the interpretability and trustworthiness of algorithmic decisions and enhance human decision making [13, 23, 25, 38]. Below we focus on approaches that provide post-hoc explanations of machine learning models and discuss why diversity should be an important component for counterfactual explanations. There is also an important line of work that focuses on developing intelligible models by assuming that simple models such as linear models or decision trees are interpretable [9, 24, 26, 27].

2.1 Explanation through Feature Importance

An important approach to post-hoc explanations is to determine feature importance for a particular prediction through *local* approximation. Ribeiro et al. [32] propose a feature-based approach, LIME, that fits a sparse linear model to approximate non-linear models locally. Guidotti et al. [16] extend this approach by fitting a decision-tree classifier to approximate the non-linear model and then tracing the decision-tree paths to generate explanations. Similarly, Lundberg and Lee [28] present a unified framework that assigns each feature an importance value for a particular prediction. Such explanations, however, "lie" about the machine learning models. There is an inherent tradeoff between truthfulness about the model and human interpretability when explaining a complex model, and so explanation methods that use proxy models inevitably approximate the true model to varying degrees. Similarly, *global* explanations can be generated by approximating the true surface with a simpler surrogate model and using the simpler model to derive explanations [10, 32]. A major problem with these approaches is that since the explanations are sourced from simpler surrogates, there is no guarantee that they are faithful to the original model.

2.2 Explanation through Visualization

Similar to identifying feature importance, visualizing the decision of a model is a common technique for explaining model predictions. Such visualizations are commonly used in the computer vision community, ranging from highlighting certain parts of an image to activations in convolutional neural networks [29, 42, 43]. However, these visualizations can be difficult to interpret in scenarios that are not inherently visual such as recidivism prediction and loan approvals, which are the cases that our work focuses on.

2.3 Explanation through Examples

The most relevant class of explanations to our approach is through examples. An example-based explanation framework is MMD-critic proposed by Kim et al. [18], which selects both prototypes and criticisms from the original data points. More recently, counterfactual explanations are proposed as a way to provide alternative perturbations that would have changed the prediction of a model. In other words, given an input feature $\boldsymbol{x}$ and the corresponding output by a ML model f, a *counterfactual explanation* is a perturbation of the input to generate a different output y by the same algorithm. Specifically, Wachter et al. [39] propose the following formulation:

$$\boldsymbol{c} = \arg\min_{\boldsymbol{c}} \text{yloss}(f(\boldsymbol{c}), y) + |\boldsymbol{x} - \boldsymbol{c}|, \quad (1)$$

where the first part (yloss) pushes the counterfactual $\boldsymbol{c}$ towards a different prediction than the original instance, and the second part keeps the counterfactual close to the original instance.

ML model (f)	The trained model obtained from the training data.
Original input ($\boldsymbol{x}$)	The feature vector associated with an instance of interest that receives an unfavorable decision from the ML model.
Original outcome	The prediction of the original input from the trained model, usually corresponding to the undesired class.
Original outcome class	The undesired class.
Counterfactual example ($\boldsymbol{c}_i$)	An instance (and its feature vector) close to the original input that would have received a favorable decision from the ML model.
CF class	The desired class.

Table 1: Terminology used throughout the paper.

Extending their work, we provide a method to construct a set of counterfactuals with diversity. In other domains of information search such as search engines and recommendation systems, multiple studies [15, 22, 35, 45] show the benefits of presenting a diverse set of information items to a user. Our hypothesis is that diversity can be similarly beneficial when people are shown counterfactual explanations. For linear models, a recent paper by Russell [34] develops an efficient algorithm to find diverse counterfactuals using integer programming. In this work, we examine an alternative formulation that works for any differentiable model, investigate multiple practical issues on different datasets, and propose a general quantitative evaluation framework for diverse counterfactuals.

3 COUNTERFACTUAL GENERATION ENGINE

The input of our problem is a trained machine learning model, f, and an instance, $\boldsymbol{x}$. We would like to generate a set of k counterfactual examples, $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k\}$, such that they all lead to a different decision than $\boldsymbol{x}$. The instance ($\boldsymbol{x}$) and all CF examples ($\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k\}$) are d-dimensional. Throughout the paper, we assume that the machine learning model is differentiable and static (does not change over time), and that the output is binary. Table 1 summarizes the main terminologies used in the paper.

Our goal is to generate an actionable counterfactual set, that is, the user should be able to find CF examples that they can act upon. To do so, we need individual CF examples to be feasible with respect to the original input, but also need diversity among the generated counterfactuals to provide different ways of changing the outcome class. Thus, we adapt diversity metrics to generate diverse counterfactuals that can offer users multiple options (Section 3.1). At the same time, we incorporate feasibility using the proximity constraint from Wachter et al. [39] and introduce other user-defined constraints. Finally, we point out that counterfactual generation is a post-hoc procedure distinct from the standard machine learning setup, and discuss related practical issues (Section 3.3).

3.1 Diversity and Feasibility Constraints

Although diverse CF examples increase the chances that at least one example will be actionable for the user, examples may end up changing a large set of features, or maximize diversity by considering big changes from the original input. This situation could be worsened when features are high-dimensional. We thus need a combination of diversity and feasibility, as we formulate below.

Diversity via Determinantal Point Processes. We capture diversity by building on determinantal point processes (DPP), which has been adopted for solving subset selection problems with diversity constraints [21]. We use the following metric based on the determinant of the kernel matrix given the counterfactuals:

$$dpp\_diversity = det(\mathbf{K}), \tag{2}$$

where $\mathbf{K}_{i,j} = \frac{1}{1+dist(\boldsymbol{c}_i, \boldsymbol{c}_j)}$ and $dist(\boldsymbol{c}_i, \boldsymbol{c}_j)$ denotes a distance metric between the two counterfactual examples. In practice, to avoid ill-defined determinants, we add small random perturbations to the diagonal elements for computing the determinant.

Proximity. Intuitively, CF examples that are closest to the original input can be the most useful to a user. We quantify *proximity* as the (negative) vector distance between the original input and CF example's features. This can be specified by a distance metric such as ℓ_1-distance (optionally weighted by a user-provided custom weight for each feature). Proximity of a set of counterfactual examples is the mean proximity over the set.

$$Proximity := -\frac{1}{k}\sum_{i=1}^{k} dist(\boldsymbol{c}_i, \boldsymbol{x}). \tag{3}$$

Sparsity. Closely connected to proximity is the feasibility property of sparsity: how many features does a user need to change to transition to the counterfactual class. Intuitively, a counterfactual example will be more feasible if it makes changes to fewer number of features. Since this constraint is non-convex, we do not include it in the loss function but rather handle it through modifying the generated counterfactuals, as explained in Section 3.3.

User constraints. A counterfactual example may be close in feature space, but may not be feasible due to real world constraints. Thus, it makes sense to allow the user to provide constraints on feature manipulation. They can be specified in two ways. First, as box constraints on feasible ranges for each feature, within which CF examples need to be searched. An example of such a constraint is: "income cannot increase beyond 200,000". Alternatively, a user may specify the variables that can be changed.

In general, feasibility is a broad issue that encompasses many facets. We further examine a novel feasibility constraint derived from causal relationships in Section 6.

3.2 Optimization

Based on the above definitions of diversity and proximity, we consider a combined loss function over all generated counterfactuals.

$$C(\boldsymbol{x}) = \underset{\boldsymbol{c}_1,\ldots,\boldsymbol{c}_k}{\arg\min} \frac{1}{k}\sum_{i=1}^{k} \text{yloss}(f(\boldsymbol{c}_i), y) + \frac{\lambda_1}{k}\sum_{i=1}^{k} dist(\boldsymbol{c}_i, \boldsymbol{x}) - \lambda_2\, \text{dpp\_diversity}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k) \tag{4}$$

where $\boldsymbol{c}_i$ is a counterfactual example (CF), k is the total number of CFs to be generated, $f(.)$ is the ML model (a black box to end users), $yloss(.)$ is a metric that minimizes the distance between $f(.)$'s prediction for $\boldsymbol{c}_i$s and the desired outcome y (usually 1 in our experiments), d is the total number of input features, $\boldsymbol{x}$ is the original input, and dpp_diversity(.) is the diversity metric. λ_1 and λ_2 are hyperparameters that balance the three parts of the loss function.

Implementation. We optimize the above loss function using gradient descent. Ideally, we can achieve $f(\boldsymbol{c}_i) = y$ for every counterfactual, but this may not always be possible because the objective is non-convex. We run a maximum of 5,000 steps, or until the loss function converges and the generated counterfactual is valid (belongs to the desired class). We initialize all $\boldsymbol{c}_i$ randomly.

3.3 Practical considerations

Important practical considerations need to be made for such counterfactual algorithms to work in practice, since they involve multiple tradeoffs in choosing the final set. Here we describe four such considerations. While these considerations might seem trivial from a technical perspective, we believe that they are important for supporting user interaction with counterfactuals.

Choice of yloss. An intuitive choice of *yloss* may be ℓ_1-loss ($|y - f(\boldsymbol{c})|$) or ℓ_2-loss. However, these loss functions penalize the distance of $f(\boldsymbol{c})$ from the desired y, whereas a valid counterfactual only requires that $f(\boldsymbol{c})$ be greater or lesser than f's threshold (typically 0.5), not necessarily the closest to desired y (1 or 0). In fact, optimizing for $f(\boldsymbol{c})$ to be close to either 0 or 1 encourages large changes to $\boldsymbol{x}$ towards the counterfactual class, which in turn make the generated counterfactual less feasible for a user. Therefore, we use a hinge-loss function that ensures zero penalty as long as $f(\boldsymbol{c})$ is above a fixed threshold above 0.5 when the desired class is 1 (and below a fixed threshold when the desired class is 0). Further, it imposes a penalty proportional to difference between $f(c)$ and 0.5 when the classifier is correct (but within the threshold), and a heavier penalty when $f(c)$ does not indicate the desired counterfactual class. Specifically, the hinge-loss is:

$$hinge\_yloss = max(0, 1 - z * logit(f(\boldsymbol{c}))),$$

where z is -1 when $y = 0$ and 1 when $y = 1$, and $logit(f(\boldsymbol{c}))$ is the unscaled output from the ML model (e.g., final logits that enter a softmax layer for making predictions in a neural network).

Choice of distance function. For continuous features, we define *dist* as the mean of feature-wise ℓ_1 distances between the CF example and the original input. Since features can span different ranges, we divide each feature-wise distance by the median absolute deviation (MAD) of the feature's values in the training set, following Wachter et al. [39]. Deviation from the median provides a robust measure of the variability of a feature's values, and thus dividing by the MAD allows us to capture the relative prevalence of observing the feature at a particular value.

$$\text{dist\_cont}(\boldsymbol{c}, \boldsymbol{x}) = \frac{1}{d_{cont}} \sum_{p=1}^{d_{cont}} \frac{|\boldsymbol{c}^p - \boldsymbol{x}^p|}{MAD_p}, \tag{5}$$

where d_{cont} is the number of continuous variables and MAD_p is the median absolute deviation for the p-th continuous variable.

For categorical features, however, it is unclear how to define a notion of distance. While there exist metrics based on the relative frequency of different categorical levels for a feature in available data [30], they may not correspond to the *difficulty* of changing a particular feature. For instance, irrespective of the relative ratio of different education levels (e.g., high school or bachelors), it is quite hard to obtain a new educational degree, compared to changes in other categorical features. We thus use a simpler metric that assigns a distance of 1 if the CF example's value for any categorical feature differs from the original input, otherwise it assigns zero.

$$\text{dist\_cat}(\boldsymbol{c}, \boldsymbol{x}) = \frac{1}{d_{cat}} \sum_{p=1}^{d_{cat}} I(\boldsymbol{c}^p \neq \boldsymbol{x}^p), \tag{6}$$

where d_{cat} is the number of categorical variables.

Relative scale of features. In general, continuous features can have a wide range of possible values, while typical encoding for categorical features constrains them to a one-hot binary representation. Since the scale of a feature highly influences how much it matters in our objective function, we believe that the ideal solution is to provide interactive interfaces to allow users to input their preferences across features. As a sensible default, however, we transform all features to $[0, 1]$. Continuous features are simply scaled between 0 and 1. For categorical features, we convert each feature using one-hot encoding and consider it as a continuous variable between 0 and 1. Also, to enforce the one-hot encoding in the learned counterfactuals, we add a regularization term with high penalty for each categorical feature to force its values for different levels to sum to 1. At the end of the optimization, we pick the level with maximum value for each categorical feature.

Enhancing Sparsity. While our loss function minimizes the distance between the input and the generated counterfactuals, an ideal counterfactual needs to be sparse in the number of features it changes. To encourage sparsity in a generated counterfactual, we conduct a post-hoc operation where we restore the value of continuous features back to their values in $\boldsymbol{x}$ greedily until the predicted class $f(\boldsymbol{c})$ changes. For this operation, we consider all continuous features $\boldsymbol{c}^j$ whose difference from $\boldsymbol{x}^j$ is less than a chosen threshold. Although an intuitive threshold is the median absolute distance (MAD), the MAD can be fairly large for features with large variance. Therefore, for each feature, we choose the minimum of MAD and the bottom 10% percentile of the absolute difference between non-identical values from the median.

Hyperparameter choice. Since counterfactual generation is a post-hoc step after training the ML model, it is not necessarily required that we use the same hyperparameter for every original input [39]. However, since hyperparameters can influence the generated counterfactuals, it seems problematic if users are given counterfactuals generated by different hyperparameters.[1] In this work, therefore, we choose $\lambda_1 = 0.5$ and $\lambda_2 = 1$ based on a grid-search with different values and evaluating the diversity and proximity of generated CF examples.

4 EVALUATING COUNTERFACTUALS

Despite recent interest in counterfactual explanations [34, 39], the evaluations are typically only done in a qualitative fashion. In this section, we present metrics for evaluating the quality of a set of counterfactual examples. As stated in Section 3, it is desirable that a method produces diverse and proximal examples and that it can generate valid counterfactual examples for all possible inputs. Ultimately, however, the examples should help a user in understanding the local decision boundary of the ML classifier. Thus, in addition

[1] In general, whether the explanation algorithm should be uniform is a fundamental issue for providing post-hoc explanations of algorithmic decisions and it likely depends on the nature of such explanations.

to diversity and proximity, we propose a metric that approximates the notion of a user's understanding. We do so by constructing a secondary model based on the counterfactual examples that acts as a proxy of a user's understanding, and compare how well it can mimic the ML classifier's decision boundary.

Nevertheless, it is important to emphasize that CF examples are eventually evaluated by end users. The goal of this work is to provide metrics that pave the way towards meaningful human subject experiments, and we will offer further discussion in Section 7.

4.1 Validity, Proximity, and Diversity

First, we define quantitative metrics for validity, diversity, and proximity for a counterfactual set that can be used to evaluate any method for generating counterfactuals. We assume that a set C of k counterfactual examples are generated for an original input.

Validity. Validity is simply the fraction of examples returned by a method that are actually *counterfactuals*. That is, they correspond to a different outcome than the original input. Here we consider only unique examples because a method may generate multiple examples that are identical to each other.

$$\texttt{\%Valid-CFs} = \frac{|\{\text{unique instances in } C \text{ s.t. } f(c) > 0.5\}|}{k}$$

Proximity. We define distance-based proximity separately for continuous and categorical features. Using the definition of *dist* metrics from Equations 5 and 6, we define proximity as:

$$\texttt{Continuous-Proximity} : -\frac{1}{k}\sum_{i=1}^{k} \text{dist\_cont}(\boldsymbol{c}_i, \boldsymbol{x}), \tag{7}$$

$$\texttt{Categorical-Proximity} : 1 - \frac{1}{k}\sum_{i=1}^{k} \text{dist\_cat}(\boldsymbol{c}_i, \boldsymbol{x}), \tag{8}$$

That is, we define proximity as the mean of feature-wise distances between the CF example and the original input. Proximity for a set of examples is simply the average proximity over all the examples. Note that the above metric for continuous proximity is slightly different than the one used during CF generation. During CF generation, we transform continuous features to [0, 1] for reasons discussed in Section 3.3, but we use the features in their original scale during evaluation for better interpretability of the distances.

Sparsity. While proximity quantifies the average change between a CF example and the original input, we also measure another related property, sparsity, that captures the *number* of features that are different. We define sparsity as the number of changes between the original input and a generated counterfactual.

$$\texttt{Sparsity} : 1 - \frac{1}{kd}\sum_{i=1}^{k}\sum_{l=1}^{d} 1_{[\boldsymbol{c}_i^l \neq \boldsymbol{x}_i^l]} \tag{9}$$

where d is the number of input features. For clarity, we can also define sparsity separately for continuous and categorical features (where `Categorical-Proximity` is identical to `Categorical-Sparsity`). Note that greater values of sparsity and proximity are desired.

Diversity. Diversity of CF examples can be evaluated in an analogous way to proximity. Instead of feature-wise distance from the original input, we measure feature-wise distances between each pair of CF examples, thus providing a different metric for evaluation than the loss formulation from Equation 2. Diversity for a set of counterfactual examples is the mean of the distances between each pair of examples. Similar to proximity, we compute separate diversity metrics for categorical and continuous features.

$$\texttt{Diversity} : \Delta = \frac{1}{C_k^2}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} \text{dist}(\boldsymbol{c}_i, \boldsymbol{c}_j),$$

where dist is either dist_cont or dist_cat.

In addition, we define an analagous sparsity-based diversity metric that measures the fraction of features that are different between any two pair of counterfactual examples.

$$\texttt{Count-Diversity} : \frac{1}{C_k^2 d}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\sum_{l=1}^{d} 1_{[\boldsymbol{c}_i^l \neq \boldsymbol{c}_j^l]}$$

It is important to note that the evaluation metrics used here are intentionally different from Equation 4, so there is no guarantee that our generated counterfactuals would do well on all these metrics, especially on the sparsity metric which is not optimized explicitly in CF generation. In addition, given the trade-off between diversity and proximity, no method will be able to maximize both. Therefore, evaluation of a counterfactual set will depend on the relative merits of diversity versus proximity for a particular application domain.

4.2 Approximating the local decision boundary

The above properties are desirable, but ideally, we would like to evaluate whether the examples help a user in *understanding* the local decision boundary of the ML model. As a tool for explanation, counterfactual examples help a user intuitively explore specific points on the other side of the ML model's decision boundary, which then help the user to "guess" the workings of the model. To construct a metric for the accuracy of such guesses, we approximate a user's guess with *another* machine learning model that is trained on the generated counterfactual examples and the original input. Given this secondary model, we can evaluate the effectiveness of counterfactual examples by comparing how well the secondary model can mimic the original ML model. Thus, considering the secondary model as a best-case scenario of how a user may rationally extrapolate counterfactual examples, we obtain a proxy for how well a user may guess the local decision boundary.

Specifically, given a set of counterfactual examples and the input example, we train a 1-nearest neighbor (*1-NN*) classifier that predicts the output class of any new input. Thus, an instance closer to any of the CF examples will be classified as belonging to the desired counterfactual outcome class, and instances closer to the original input will be classified as the original outcome class. We chose 1-NN for its simplicity and connections to people's decision-making in the presence of examples. We then evaluate the accuracy of this classifier against the original ML model on a dataset of simulated test data. To generate the test data, we consider samples of increasing distance from the original input. Consistent with training, we scale distance for continuous features by dividing it by the median absolute deviation (MAD) for each feature. Then, we construct a hypersphere centered at the original input that has dimensions equal to the number of continuous features. Within this hypersphere, we sample feature values uniformly at random.

For categorical features, in the absence of a clear distance metric, we uniformly sample across the range of possible levels.

In our experiments, we consider spheres with radiuses as multiples of the MAD ($r = \{0.5, 1, 2\}MAD$). For each original input, we sample 1000 points at random per sphere to evaluate how well the secondary 1-NN model approximates the local decision boundary. *Note that this 1-NN classifier is trained from a handful of CF examples, and we intentionally choose this simple classifier to approximate what a person could have done given these CF examples.*

4.3 Datasets

To evaluate our method, we consider the following four datasets.

Adult-Income. This dataset contains demographic, educational, and other information based on 1994 Census database and is available on the UCI machine learning repository [20]. We preprocess the data based on a previous analysis [44] and obtain 8 features, namely, hours per week, education level, occupation, work class, race, age, marital status, and sex. The ML model's task is to classify whether an individual's income is over $\$50,000$.

LendingClub. This dataset contains five years (2007-2011) data on loans given by LendingClub, an online peer-to-peer lending company. We preprocess the data based on previous analyses [12, 17, 37] and obtain 8 features, namely, employment years, annual income, number of open credit accounts, credit history, loan grade as decided by LendingClub, home ownership, purpose, and the state of residence in the United States. The ML model's task is to decide loan decisions based on a prediction of whether an individual will pay back their loan.

German-Credit. This dataset contains information about individuals who took a loan from a particular bank [2]. We use all the 20 features in the data, including several demographic attributes and credit history, without any preprocessing. The ML model's task is to determine whether the person has a good or bad credit risk based on their attributes.

COMPAS. This dataset was collected by ProPublica [4] as a part of their analysis on recidivism decisions in the United States. We preprocess the data based on previous work [14] and obtain 5 features, namely, bail applicants' age, gender, race, prior count of offenses, and degree of criminal charge. The ML model's task is to decide bail based on predicting which of the bail applicants will recidivate in the next two years.

These datasets contain different numbers of continuous and categorical features as shown in Table 2. `COMPAS` dataset has a single continuous feature, while `Adult-Income`, `LendingClub` and `German-Credit` have 2, 4, and 5 continuous features respectively. For all three datasets, we transform categorical features by using one-hot-encoding, as described in Section 3. Continuous features are scaled between 0 and 1. To obtain an ML model to explain, we divide each dataset into 80%-20% train and test sets, and use cross-validation on the train set to optimize hyperparameters. To facilitate comparisons with Russell [34], we use TensorFlow library to train both a linear (logistic regression) classifier and a non-linear neural network model with a single hidden layer. Table 2 shows modelling details and test set accuracy on each dataset.

Dataset	Linear	Non-linear	Num cat	Num cont
Adult-Income	0.82	0.82	6	2
LendingClub	0.67	0.66	4	4
German-Credit	0.73	0.77	15	5
COMPAS	0.67	0.67	4	1

Table 2: Model accuracy and feature information.

4.4 Baselines

We employ the following baselines for generating CF examples.

- `SingleCF`: We follow Wachter et al. [39] and generate a single CF example, optimizing for y-loss difference and proximity.
- `MixedIntegerCF`: We use the mixed integer programming method proposed by Russell [34] for generating diverse counterfactual examples. This method works only for a linear model.
- `RandomInitCF`: Here we extend `SingleCF` to generate k CF examples by initializing the optimizer independently with k random starting points from $[0, 1]^d$. Since the optimization loss function is non-convex, one might obtain different CF examples.
- `NoDiversityCF`: This method utilizes our proposed loss function that optimizes the set of k examples simultaneously (Equation 4), but ignores the diversity term by setting $\lambda_2 = 0$.

To these baselines, we compare our proposed method, `DiverseCF`, that generates a set of counterfactual examples and optimizes for both diversity and proximity. As with `RandomInitCF`, we initialize the optimizer with random starting points. In addition, we consider a variant `DiverseCF-Sparse` that performs post-hoc sparsity enhancement on continuous features as described in Section 3.3. Similarly, for `RandomInitCF` and `NoDiversityCF`, we include results both with and without the sparsity correction. For all methods, we use the ADAM optimizer [19] implementation in TensorFlow (learning rate=0.05) to minimize the loss and obtain CF examples.

In addition, we compare `DiverseCF` to one of the major feature-based local explanation methods, `LIME` [32], on how well it can approximate the decision boundary. We construct a 1-NN classifier for each set of CF examples as described in Section 4.2. For LIME, we use the prediction of the linear model for each input instance as a local approximation of the ML model's decision surface. Note that our 1-NN classifiers are based on only $k \leq 10$ counterfactuals, while LIME's linear classifiers are based on 5,000 samples.

5 EXPERIMENT RESULTS

In this section, we show that our approach generates a set of more diverse counterfactuals than the baselines according to the proposed evaluation metrics. We further present examples for a qualitative overview and show that the generated counterfactuals can approximate local decision boundaries as well as LIME, an explanation method specifically designed for local approximation.

5.1 Quantitative Evaluation

We first evaluate `DiverseCF` based on quantitative metrics of valid CF generation, diversity, and proximity. As described in Section 3, we report results with hyperparameters, $\lambda_1 = 0.5$ and $\lambda_2 = 1$ from Equation 4. Figure 1 shows the comparison with `SingleCF`, `RandomInitCF`, and `NoDiversityCF` for explaining the non-linear

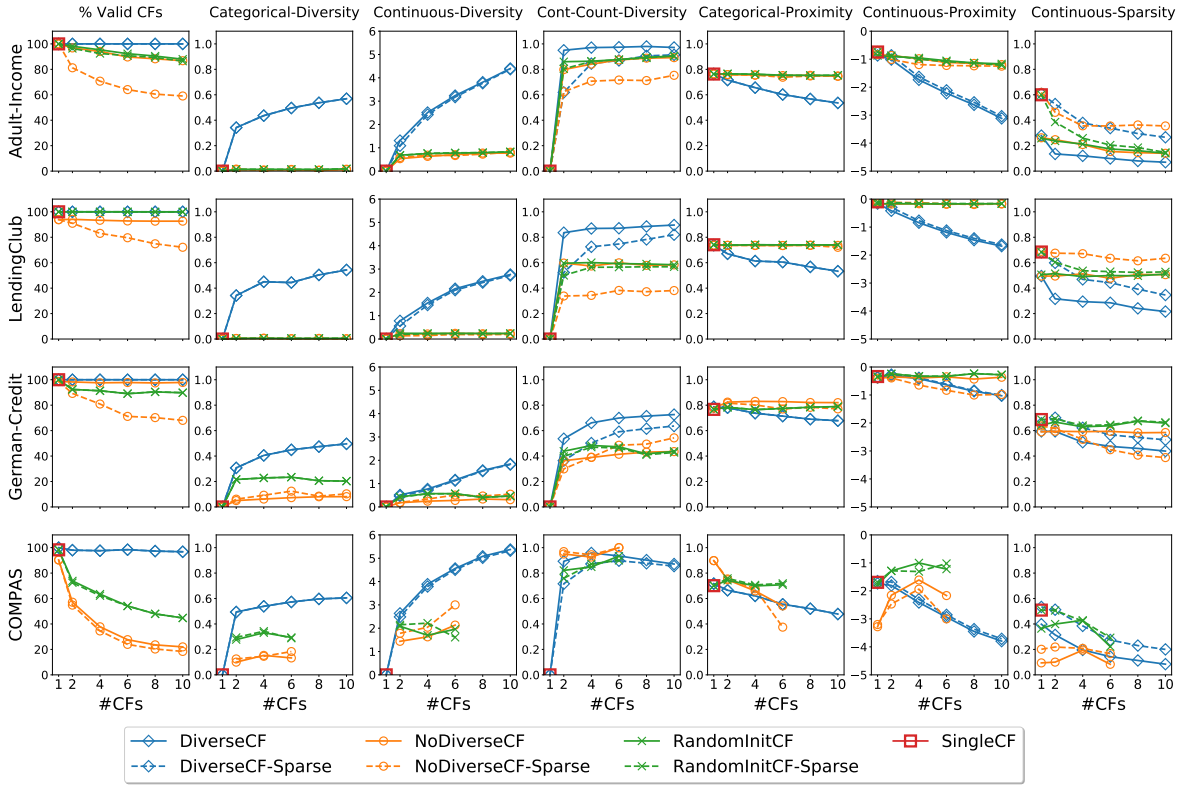


Figure 1: Comparisons of `DiverseCF` with baseline methods on %Valid CFs, diversity, proximity and sparsity (`SingleCF` only shows up for $k = 1$). All y-axes are defined so that higher values are better, while the x-axis represents the number of requested counterfactuals. `DiverseCF` finds a greater number of valid unique counterfactuals for each k and tends to generate more diverse counterfactuals than the baseline methods. For `COMPAS` dataset, none of the baselines could generate $k > 6$ CF examples for any original input, therefore we only show results for `DiverseCF` in COMPAS when $k > 6$.

ML models, while Figure 2 compares with `MixedIntegerCF` for explaining the linear ML models. All results are based on 500 random instances from the test set.

5.1.1 Explaining a non-linear ML model (Figure 1). Given that non-linear ML models are common in real world applications, we focus our discussion on explaining non-linear models.

Validity. Across all four datasets, we find that `DiverseCF` generates nearly 100% valid CF examples for all values of the requested number of examples k. Baseline methods without an explicit diversity objective can generate valid CF examples for $k = 1$, but their percentage of unique valid CFs decreases as k increases. Among the datasets, we find that it is easier to generate valid CF examples for `LendingClub` (`RandomInitCF` also achieves ~100% validity) while `COMPAS` is the hardest, likely driven by the fact that it has only one continuous feature—prior count of offenses. As an example, at $k = 10$ for `COMPAS`, a majority of the CFs generated by the next best method, `RandomInitCF` are either duplicate or invalid.

Diversity. Among the valid CFs, `DiverseCF` also generates more diverse examples than the baseline methods for both continuous and categorical features. For all datasets, Continuous-Diversity for `DiverseCF` is the highest and increases as k increases, reaching up to eleven times the baselines for the `LendingClub` dataset at $k = 10$. Among categorical features, average number of different features between CF examples is higher for all datasets than baseline methods, especially for `Adult-Income` and `LendingClub` datasets where `Cat-Diversity` remains close to zero for baseline methods. Remarkably, `DiverseCF` has the highest number of continuous features changed too (`Cont-Count-Diversity`), even though it was not explicitly optimized for this metric. The only exception is on `COMPAS` data where `NoDiversityCF` has a slightly higher `Cont-Count-Diversity` for $k <= 6$, but is unable to generate any valid CFs for higher k's.

Proximity. To generate diverse CF examples, `DiverseCF` searches a larger space than proximity-only methods such as `RandomInitCF` or `NoDiversityCF`. As a result, `DiverseCF` returns examples with lower proximity than other methods, indicating an inherent tradeoff between diversity and proximity. However, for categorical features, the difference in proximity compared to baselines is small, up to ~30% of the baselines' proximity. Higher proximity over continuous features can be obtained by adding the post-hoc sparsity enhancement (`DiverseCF-Sparse`), which results in higher sparsity than `DiverseCF` for all datasets (but correspondingly lower count-based diversity). Thus, this method can be used to fine-tune `DiverseCF` towards more proximity if desired.

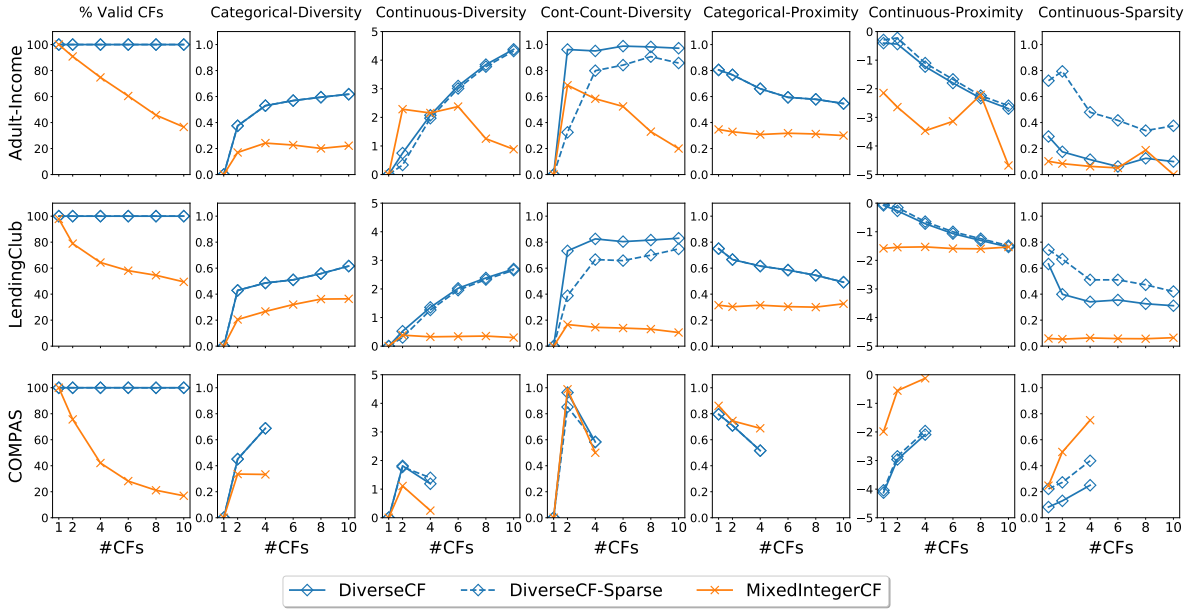


Figure 2: Comparisons of `DiverseCF` with `MixedIntegerCF` on %Valid CFs, diversity, proximity and sparsity on linear ML models. For a fair comparison, we compute average metrics only over the original inputs where `MixedIntegerCF` returned the required number of CF examples. Thus, we omit results when $k > 4$ for COMPAS since `MixedIntegerCF` could not find more than four CFs for any original input. Results for `German-Credit` are in the Supplementary Materials.

5.1.2 Explaining linear ML models (Figure 2). To compare our methods with `MixedIntegerCF` [34], we explain a linear ML model for each dataset. Similar to the results on non-linear models, `DiverseCF` outperforms `MixedIntegerCF` by finding 100% valid counterfactuals, and the gap with `MixedIntegerCF` increases as k increases. We also find that `DiverseCF` has consistently higher diversity among counterfactuals than `MixedIntegerCF` for all datasets. Importantly, better diversity in the CFs from `DiverseCF` does not come at the price of proximity. For `Adult-Income` and `LendingClub` datasets, `DiverseCF` has better proximity and sparsity than `MixedIntegerCF`.

5.2 Qualitative evaluation

To understand more about the resultant explanations, we look at sample CF examples generated by `DiverseCF` with sparsity in Table 3. In the three datasets,[2] the examples capture some intuitive variables and vary them: Education in `Adult-Income`, Income in `LendingClub` dataset, and PriorsCount in `COMPAS`. In addition, the user also sees other features that can be varied for the desired outcome. For example, in the `COMPAS` input instance, a person would have been granted bail if they had been a Caucasian or charged with Misdemeanor instead of Felony. These features do not really lead to actionable insights because the subject cannot easily change them, but nevertheless provide the user an accurate picture of scenarios where they would have been out on bail (and also raise questions about potential racial bias in the ML model itself). In practice, we expect that a domain expert or the user may provide *unmodifiable* features which `DiverseCF` can treat as constants in the counterfactual generation process.

Similarly, in the `Adult-Income` dataset, the set of counterfactuals show that studying for an advanced degree can lead to a higher income, but also shows less obvious counterfactuals such as getting married for a higher income (in addition to finishing professional school and increasing hours worked per week). These counterfactuals are likely generated due to underlying correlations in the dataset (married people having higher income). To counter such correlational outcomes and preserve known causal relationships, we present a post-hoc filtering method in Section 6.

These qualitative examples also confirm our observation regarding sparsity and the choice of the *yloss* function. Continuous variables in counterfactual examples (e.g., income in `LendingClub`) never change to their maximum extreme values thanks to the hinge loss, which was an issue using other *yloss* metrics such as ℓ_1 loss. Furthermore, it does not require changing a large number of features to achieve the desired outcome. However, based on the domain and use case, a user may prioritize changing certain variables or desire more sparse or more diverse CF examples. As we described in Section 3.3, these variations can be achieved by appropriately tuning weights on features and the learning rate for optimization.

Overall, these initial set of CF examples help understand the important variations as learned by the algorithm. We expect the user to engage their actionability constraints with this initial set to iteratively generate focused CF examples, that can help find useful variations. In addition, these examples can also expose biases or odd edge-cases in the ML model, which can be useful for model builders in debugging, or for fairness evaluators in discovering bias.

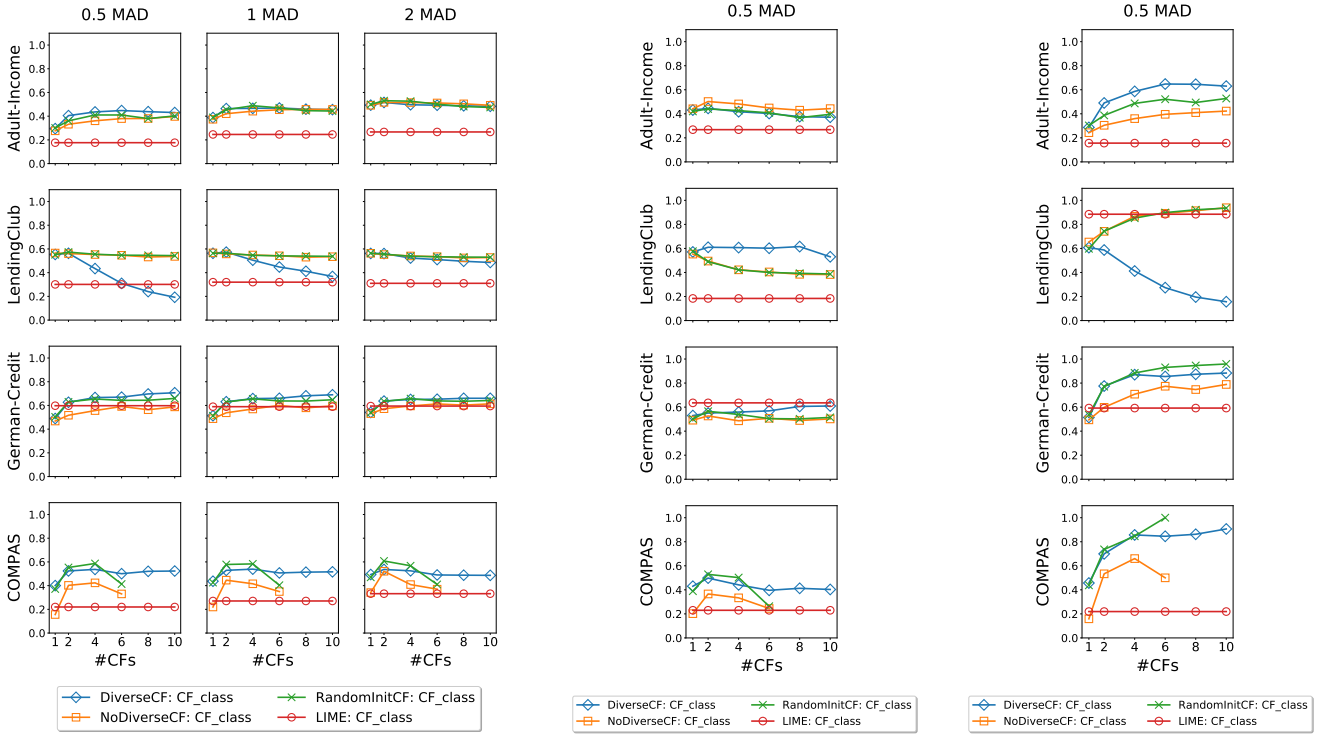
[2]We skip `German-Credit` for space reasons.

Adult	HrsWk	Education	Occupation	WorkClass	Race	AgeYrs	MaritalStat	Sex
Original input (outcome: <=50K)	45.0	HS-grad	Service	Private	White	22.0	Single	Female
Counterfactuals (outcome: >50K)	—	Masters	—	—	—	65.0	Married	Male
	—	Doctorate	—	Self-Employed	—	34.0	—	—
	33.0	—	White-Collar	—	—	47.0	Married	—
	57.0	Prof-school	—	—	—	—	Married	—

LendingClub	EmpYrs	Inc$	#Ac	CrYrs	LoanGrade	HomeOwner	Purpose	State
Original input (outcome: Default)	7.0	69996.0	4.0	26.0	D	Mortgage	Debt	NY
Counterfactuals (outcome: Paid)	—	61477.0	—	—	B	—	Purchase	—
	10.0	83280.0	1.0	23.0	A	—	—	TX
	10.0	69798.0	—	40.0	A	—	—	—
	10.0	130572.0	—	—	A	Rent	—	—

COMPAS	PriorsCount	CrimeDegree	Race	Age	Sex
Original input (outcome: Will Recidivate)	10.0	Felony	African-American	>45	Female
Counterfactuals (outcome: Won't Recidivate)	—	—	Caucasian	—	—
	0.0	—	—	—	Male
	0.0	—	Hispanic	—	—
	9.0	Misdemeanor	—	—	—

Table 3: Examples of generated counterfactuals in `Adult-Income`, `LendingClub` and `COMPAS` datasets.



(a) F1 score of the counterfactual class. **(b) Precision.** **(c) Recall.**

Figure 3: Performance of 1-NN classifiers learned from counterfactuals at different distances from the original input. `DiverseCF` outperforms LIME and baseline CF methods in F1 score on correctly predicting the counterfactual class, except in `LendingClub` dataset. For `Adult-Income` and `COMPAS` datasets, both precision and recall is higher for `DiverseCF` compared to `LIME`.

5.3 Approximating local decision boundary

As a proxy for understanding how well users can guess the local decision boundary of the ML model (see Section 4.2), we compare classifiers based on the proposed `DiverseCF` method, baseline methods, and `LIME`. We use precision, recall, and F1 for the counterfactual outcome class (Figure 3) as our main evaluation metric because of the class imbalance in data points near the original input. To evaluate the sensitivity of these metrics to varying distance from the original input, we show these metrics for points sampled within varying distance thresholds.

Even with a handful (2-11) of training examples (generated counterfactuals and the original input), we find that 1-NN classifiers trained on the output of `DiverseCF` obtain higher F1 score than the LIME classifier in most configurations. For instance, on the

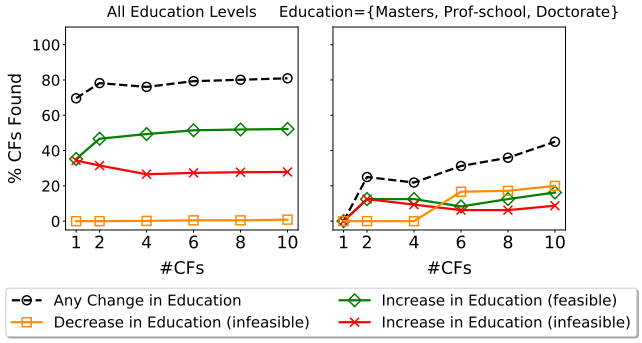


Figure 4: Post-hoc filtering of CF examples based on causal constraints. The left figure shows that there are nearly 80% CFs that include any change in education, out of which more than one-third are infeasible. If we filter to only people with higher degrees, almost half of the changes in educational degrees are infeasible.

`Adult-Income` dataset, at $k = 4$ and 0.5MAD threshold, `DiverseCF` obtains $F1 = 0.44$ while `LIME` obtains $F1 = 0.19$. This result stays consistent as we increase k or the distance from the original input. One exception is `LendingClub` at 0.5 MAD and $k = 10$ where the F1 score for `DiverseCF` drops below `LIME`. Figures 3b and 3c indicate that this drop is due to low recall for `DiverseCF` at this configuration. Still, precision remains substantially higher for `DiverseCF` (0.61) compared to 0.19 for `LIME`. This observation is likely because `LIME` predicts a majority of the instances as the CF class for this dataset, whereas `DiverseCF` has fewer false positives. On `Adult-Income` and `COMPAS` datasets, `DiverseCF` achieves both higher precision and recall than `LIME`.

As for the difference between different methods of generating counterfactuals, `DiverseCF` tend to perform similarly with `NoDiversityCF` and `RandomInitCF` in terms of F1, except in `LendingClub`. An advantage of `DiverseCF` is that it can handle high values of k for which `NoDiversityCF` and `RandomInitCF` cannot find k unique and valid counterfactuals. Another intriguing observation is that the performance improves very quickly as the number of counterfactuals (k) increases, which suggests that two counterfactuals may be sufficient for a 1-NN classifier to get a reasonable idea of the data distribution around $\boldsymbol{x}$ in these four datasets. This observation may also be why `DiverseCF` provides similar F1 score compared to baselines, and merits further study on more complex datasets.

Overall, these results show that examples from `DiverseCF` can approximate the local decision boundary at least as well as local explanation methods like LIME. Still, the gold-standard test will be to conduct a behavioral study where people evaluate whether CF examples provide better explanation than past approaches, which we leave for future work.

6 CAUSAL FEASIBILITY OF CF EXAMPLES

So far, we have generated CF examples by varying each feature independently. However, this can lead to infeasible examples, since many features are causally associated with each other. For example, in the loan application, it can be almost impossible for a person to obtain a higher educational degree without spending time (aging). Consequently, while being valid, diverse and proximal, such a CF example is not feasible and thus not actionable by the person. In this context, we argue that incorporating causal models of data generation is important to prevent infeasible counterfactuals.

Here we present a simple way of incorporating causal knowledge in our proposed method. Users can provide their domain knowledge in the form of pairs of features and the direction of the causal edge between them [31]. Using this, we construct constraints that any counterfactual should follow. For instance, any counterfactual that changes the cause without changing its outcome is infeasible. Given these constraints, we apply a filtering step after CF examples are generated, to increase the feasibility of the output CF set.

As an example, we consider two infeasible changes based on the causal relationship between educational level and age, `Education` $\uparrow \Rightarrow$ `Age` $\uparrow$, and on the practical constraint that educational level of a person cannot be decreased, `Education`$\not\downarrow$. As Figure 4 shows, over one-third of the obtained counterfactuals that include a change in education level are infeasible and need to be filtered: most of them suggest an individual to obtain a higher degree but do not increase their age. In fact, this fraction increases as we look at CF examples for highly educated people (Masters, Doctorate and Professional): as high as 50% of all CFs suggest to switch to a lower education degree. Though post-hoc filtering can ensure feasibility of the resultant CF examples, it is more efficient to incorporate causal constraints during CF generation. We leave this for future work.

7 CONCLUDING DISCUSSION

Building upon prior work on counterfactual explanations [34, 39], we proposed a framework for generating and evaluating a diverse and feasible set of counterfactual explanations. We demonstrated the benefits of our method compared to past approaches on being able to generate a high number of unique, valid, and diverse counterfactuals for a given input for any machine learning model.

Here we note directions for future work. First, our method assumes knowledge of the gradient of the ML model. It is useful to construct methods that can work for fully *black-box* ML models. Second, we would like to incorporate causal knowledge *during* the generation of CF examples, rather than as a post-hoc filtering step. Third, as we saw in §5.2, it is important to understand people's preferences with respect to what additional constraints to add to our framework. Providing an intuitive interface to select scales of features and add constraints, and conducting behavioral experiments to support interactive explorations can greatly enhance the value of CF explanation. It will also be interesting to study the tradeoff between diversity and the cognitive cost of making a choice ("choice overload" [8, 36]), as the number of CF explanations is increased. Finally, while we focused on the utility for an end-user who is the subject of a ML-based decision, we argue that CF explanations can be useful for different stakeholders in the decision making process [38], including model designers, decision-makers such as a judge or a doctor, and decision evaluators such as auditors.

Acknowledgments. We thank Brian Lubars for his insightful comments and Chris Russel for providing assistance in running the diverse CF generation on linear models. This work was supported in part by NSF grant IIS-1927322.

REFERENCES

[1] [n.d.]. Lending Club Statistics. https://www.lendingclub.com/info/download-data.action.

[2] Accessed 2019. German credit dataset . https://archive.ics.uci.edu/ml/support/statlog+(german+credit+data).

[3] Monica Andini, Emanuele Ciani, Guido de Blasio, Alessio D'Ignazio, and Viola Salvestrini. 2017. Targeting policy-compliers with machine learning: an application to a tax rebate programme in Italy. (2017).

[4] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2016. "Machine bias: There's software used across the country to predict future criminals. And it's biased against blacks". https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[5] Susan Athey. 2017. Beyond prediction: Using big data for policy problems. *Science* 355, 6324 (2017), 483–485.

[6] Sarah R Beck, Kevin J Riggs, and Sarah L Gorniak. 2009. Relating developments in childreńs counterfactual thinking and executive functions. *Thinking & reasoning*.

[7] Daphna Buchsbaum, Sophie Bridgers, Deena Skolnick Weisberg, and Alison Gopnik. 2012. The power of possibility: Causal learning, counterfactual reasoning, and pretend play. *Philosophical Trans. of the Royal Soc. B: Biological Sciences* (2012).

[8] M Kate Bundorf and Helena Szrek. 2010. Choice set size and decision making: the case of Medicare Part D prescription drug plans. *Medical Decision Making* 30, 5 (2010), 582–593.

[9] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of KDD*.

[10] Mark Craven and Jude W Shavlik. 1996. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*.

[11] Wuyang Dai, Theodora S Brisimi, William G Adams, Theofanie Mela, Venkatesh Saligrama, and Ioannis Ch Paschalidis. 2015. Prediction of hospitalization due to heart diseases by supervised learning methods. *International journal of medical informatics* 84, 3 (2015), 189–197.

[12] Kevin Davenport. 2015. Lending Club Data Analysis Revisited with Python. http://kldavenport.com/lending-club-data-analysis-revisted-with-python/.

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

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

[15] Michael D Ekstrand, F Maxwell Harper, Martijn C Willemsen, and Joseph A Konstan. 2014. User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 161–168.

[16] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. 2018. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820* (2018).

[17] JFdarre. 2015. Project 1: Lending Club's data. https://rpubs.com/jfdarre/119147.

[18] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. 2016. Examples are not enough, learn to criticize! criticism for interpretability. In *Proceedings of NIPS*.

[19] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

[20] Ronny Kohavi and Barry Becker. 1996. UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/adult

[21] Alex Kulesza, Ben Taskar, et al. 2012. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning* 5, 2–3 (2012), 123–286.

[22] Matevž Kunaver and Tomaž Požrl. 2017. Diversity in recommender systems–A survey. *Knowledge-Based Systems* 123 (2017), 154–162.

[23] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. 2017. Counterfactual fairness. In *Advances in Neural Information Processing Systems*. 4066–4076.

[24] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. 2016. Interpretable decision sets: A joint framework for description and prediction. In *Proc. KDD*.

[25] Zachary C Lipton. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490* (2016).

[26] Yin Lou, Rich Caruana, and Johannes Gehrke. 2012. Intelligible models for classification and regression. In *Proceedings of KDD*.

[27] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. 2013. Accurate intelligible models with pairwise interactions. In *Proceedings of KDD*.

[28] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of NIPS*.

[29] Aravindh Mahendran and Andrea Vedaldi. 2015. Understanding deep image representations by inverting them. In *Proceedings of CVPR*.

[30] PAIR. 2018. What-If Tool. https://pair-code.github.io/what-if-tool/.

[31] Judea Pearl. 2009. *Causality*. Cambridge university press.

[32] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of KDD*.

[33] Jonah E Rockoff, Brian A Jacob, Thomas J Kane, and Douglas O Staiger. 2011. Can you recognize an effective teacher when you recruit one? *Education finance and Policy* 6, 1 (2011), 43–74.

[34] Chris Russell. 2019. Efficient Search for Diverse Coherent Explanations. In *Proceedings of FAT**.

[35] Mark Sanderson, Jiayu Tang, Thomas Arni, and Paul Clough. 2009. What else is there? search diversity examined. In *European Conference on Information Retrieval*. Springer, 562–569.

[36] Benjamin Scheibehenne, Rainer Greifeneder, and Peter M Todd. 2010. Can there ever be too many options? A meta-analytic review of choice overload. *Journal of consumer research* 37, 3 (2010), 409–425.

[37] S Tan, R Caruana, G Hooker, and Y Lou. 2017. Distill-and-compare: Auditing black-box models using transparent model distillation. (2017).

[38] Richard Tomsett, Dave Braines, Dan Harborne, Alun Preece, and Supriyo Chakraborty. 2018. Interpretable to Whom? A Role-based Model for Analyzing Interpretable Machine Learning Systems. *arXiv preprint arXiv:1806.07552* (2018).

[39] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual explanations without opening the black box: Automated decisions and the GDPR.

[40] Austin Waters and Risto Miikkulainen. 2014. Grade: Machine learning support for graduate admissions. *AI Magazine* 35, 1 (2014), 64.

[41] Deena S Weisberg and Alison Gopnik. 2013. Pretense, counterfactuals, and Bayesian causal models: Why what is not real really matters. *Cognitive Science* (2013).

[42] Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Proceedings of ECCV*.

[43] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. 2018. Interpretable basis decomposition for visual explanation. In *Proceedings of ECCV*.

[44] Haojun Zhu. 2016. Predicting Earning Potential using the Adult Dataset. https://rpubs.com/H_Zhu/235617.

[45] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. ACM, 22–32.

A SUPPLEMENTARY MATERIALS

Here we discuss the data properties and the implementation details of ML models relevant for reproducing our results. Our open-source implementation is available at https://github.com/microsoft/DiCE which can be used to generate counterfactual examples for any other dataset or ML model. Further, Figure 5 below compares `DiverseCF` with `MixedIntegerCF` for explaining the linear ML model over the `German-Credit` dataset.

A.1 Building ML Models

First, we build a ML model for each dataset that gives accuracy comparable to previously established benchmarks, using the Adam Optimizer [19] in TensorFlow. We described the datasets that we used in our analysis in Section 4.3. Table 4 provides detailed properties of the processed data and the ML models that we used. We tuned the hyperparameters of the ML model based on previous analyses and found that a single hidden layer neural network gives best generalization ability for all datasets. While 20 hidden neurons worked well for `COMPAS`, `Adult-Income` and `German-Credit` datasets, increasing more than 5 neurons worsened the generalization for `LendingClub` dataset. Furthermore, to handle the class imbalance problem while training with these datasets, we oversampled the training instances belonging to the minority class.

A.2 Explaining linear ML models: `German-Credit`

Similar to results we obtained for other datasets, we observe that for `German-Credit` data, `DiverseCF` consistently generates more diverse counterfactuals compared to `MixedIntegerCF`.

	COMPAS	Adult Income	German Credit	Lending Club
# Continuous Features	1	2	5	4
# Categorical Features	4	6	15	4
Range across all Continuous Features (Min, Avg, Max)	(0, 3.5, 38)	(1, 39.5, 99)	(1, 668, 15945)	(1, 16292, 200000)
# Levels across all Categorical Features (Min, Avg, Max)	(2, 3.25, 6)	(2, 4.5, 8)	(2, 4, 10)	(4, 5.5, 7)
Undesired Class	Will Recidivate	<=50K	Bad	Default
Desired Counterfactual Class	Won't Recidivate	>50K	Good	Paid
Training Data Size	1443	6513	800	8133
Fraction of Instances with Desired CF Outcome	0.55	0.30	0.25	0.8
Nonlinear Model	ANN(1, 20)	ANN(1, 20)	ANN(1, 20)	ANN(1, 5)
Test set accuracy	67%	82%	77%	66%

Table 4: Dataset description.

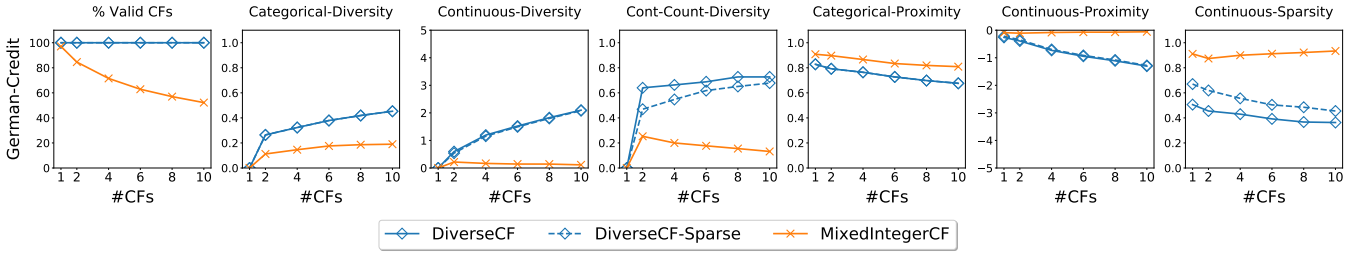


Figure 5: Comparisons of `DiverseCF` with `MixedIntegerCF` on %Valid CFs, diversity, proximity and sparsity on linear ML models for `German-Credit`. For a fair comparison, we compute average metrics only over the original inputs where `MixedIntegerCF` returned the required number of CF examples.