# Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations

Ramaravind Kommiya 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 average distance and 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 three real-world datasets show that our framework can generate a set of counterfactuals that are diverse and well approximate local decision boundaries.

## 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 do should 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 [32], screening job applicants [27], disbursing government aid [2, 4], and identifying people at high risk of a future disease [9]. 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 [31] 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 [26], 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 [31] by allowing users to explore "what-if" scenarios, similar to how children learn through counterfactual examples [5, 6, 33].

Barring simple linear models [28], however, it is difficult to generate CF examples that work for any machine learning model, and that are *actionable* for a person's situation. Continuing our loan decision example, a CF explanation might 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 [14], 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.

To this end, we propose a method that generates sets of diverse counterfactual examples for any machine learning classifier. Extending prior work [31], we construct an optimization problem that considers the diversity of the generated CF examples, in addition to proximity to the original input. Solving this optimization problem requires considering the tradeoff between diversity and accuracy, 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 the 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, except by running human behavioral experiments. While we cannot replace experiments with human subjects, we propose a set of 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 seeks to approximate the results of an actual behavioral experiment that would measure whether people can *understand* an ML model's decisions given the set of CF examples, assuming that people would rationally extrapolate the CF examples and "guess" the local decision boundary of an ML model.

We evaluate our method on explaining neural network models trained on three datasets: COMPAS for bail decision [3], Adult for income prediction [16], 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 all three datasets. Moreover, a simple 1-NN model trained on the generated CF examples obtains comparable accuracy on locally approximating the original ML model to methods like LIME [26], which are directly optimized for estimating the local decision boundary. Notably, our method obtains higher precision on predicting instances in the counterfactual outcome class than LIME in many cases, especially for the Adult dataset wherein both precision and recall are higher. Qualitative inspection of the generated CF examples illustrates the potential usefulness for making informed decisions. Additionally, CF explanations can also 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, and leave including causality while generating CF examples as future work.

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 conduct empirical experiments on multiple datasets to show how our proposed method generates diverse counterfactuals that perform comparably to LIME on estimating the local decision boundary.

## 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 [11, 18, 20, 30]. Below we focus on the main approaches that provide post-hoc explanation of machine learning models. Note that there has also been 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 [7, 19, 21, 22].

### 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. [26] propose a feature-based approach, LIME, that fits a sparse linear model to approximate non-linear models locally. Similarly, Lundberg and Lee [23] 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 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 [8, 26]. 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 [24, 34, 35]. However, this kind of visualization 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. [14], 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 and the corresponding output by an algorithm, a *counterfactual explanation* is a perturbation of the input to generate a different output by the same algorithm. Specifically, Wachter et al. [31] propose the following formulation:

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

where the first part (yloss) pushes the counterfactual towards a different prediction than the original instance, and the second part keeps the counterfactual close to the original instance. Extending this work, we provide a method to construct sets of counterfactuals with diversity. Furthermore, we address a number of practical issues for generating counterfactual explanations.

Concurrently, a recent paper by Russell [28] develops an efficient algorithm to find diverse counterfactuals using integer programming for linear models. In comparison, our work examines an alternative formulation based on diversity metrics that works for

| | |
|---|---|
| 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.**

any differentiable model, investigates multiple practical issues on different datasets, and proposes a novel evaluation framework for quantitative evaluation of 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 $\boldsymbol{x}$. We assume that $\boldsymbol{x}$ and all CF examples 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 terminologies used in the paper.

Our goal is to generate actionable counterfactuals, that is, the user should be able to make the changes indicated by the CF examples. We adapt diversity metrics to this context to generate diverse counterfactuals that can offer users multiple options (Section 3.1). At the same time, we incorporate the proximity constraint from Wachter et al. [31] and introduce user-provided custom constraints (Section 3.2). Finally, we describe how counterfactual generation is a post-hoc procedure distinct from standard machine learning setup and discuss related practical issues (Section 3.3).

### 3.1 Diversity

A set of counterfactual examples should present multiple diverse alternatives to a user, so that the user makes an informed choice among the different decisions available to them. We propose the following two methods to operationalize diversity.

**Average pairwise distance.** Our first intuition is that diverse examples should be far away from each other. We capture this through pairwise distance between the counterfactuals:

$$dist\_diversity = \frac{1}{C_k^2} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} dist(\boldsymbol{c}_i, \boldsymbol{c}_j). \tag{2}$$

**Determinantal point processes.** Alternatively, we can build on determinantal point processes (DPP), a probabilistic model that has been adopted for solving subset selection problems with diversity constraints [17]. We use the following metric based on the determinant of the kernel matrix given the counterfactuals:

$$dpp\_diversity = det(\mathbf{K}), \tag{3}$$

where $\mathbf{K}_{i,j} = \frac{1}{1+dist(\boldsymbol{c}^i, \boldsymbol{c}^j)}$. In practice, to avoid ill-defined determinants, we add small random perturbations to the diagonal elements of the kernel matrix for computing the determinant.

### 3.2 Additional Feasibility Constraints

Diverse CF examples increase the chances that at least one example will be actionable for the user. However, when features are high-dimensional, this may not always be the case. Examples may end up changing a large set of features, or maximize diversity by considering big changes from the original input. We introduce the proximity constraint from Wachter et al. [31] to avoid such CF examples as well as user constraints to satisfy custom requirements.

**Proximity.** Intuitively, CF examples that are closest to the original input can be the most useful to a user. We quantify "proximity" as the vector distance between the original input and CF example's features. This can be specified by minimizing a distance metric such as $\ell_1$-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.

$$Proximity := -\frac{1}{k} \sum_{i=1}^{k} dist(\boldsymbol{c}_i, \boldsymbol{x}). \tag{4}$$

**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 novel feasibility constraints derived from causal graphs in Section 6.

### 3.3 Optimization & Practical Considerations

Based on the above definitions of diversity and proximity, we consider a combined loss function over all generated counterfactuals, setting $dist$ as $\ell_1$-distance: $dist(\boldsymbol{x_i}, \boldsymbol{x_j}) = |\boldsymbol{x_i} - \boldsymbol{x_j}|$.

$$C = \underset{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k}{\arg\min} \frac{1}{k} \sum_{i=1}^{k} \text{yloss}(f(\boldsymbol{c}_i), y) + \frac{\lambda_1}{kd} \sum_{i=1}^{k} |\boldsymbol{x} - \boldsymbol{c}_i| - \lambda_2 \, \text{diversity}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k) \tag{5}$$

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 diversity(.) is a diversity metric. $\lambda_1$ and $\lambda_2$ are two hyperparameters that balance the three parts of the loss function.

**Optimization.** We optimize the above loss function using gradient descent. Ideally, we can achieve $f(\boldsymbol{c}_i) = 1$ 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 also all $f(\boldsymbol{c}_i)$ reach 0.75. We also initialize all $\boldsymbol{c}_i$ as points close to the original input, $\boldsymbol{x}$.

**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 mainly describe two such considerations.

*Relative scale of features.* There are two major issues about features that affect our objective function: 1) categorical features vs. continuous features; 2) the 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 features highly influence how much this feature matters in our objective function, we believe that the ideal approach is to provide interactive interfaces to allow users to input their preferences across features. As a sensible default, however, we transform all features to $[0, 1]$. We convert each categorical variable using one-hot encoding and consider it as a continuous variable between 0 and 1. For continuous variables, we scale all values between 0 and 1. As we will see in Section 5, this decision may influence the found counterfactuals significantly, but we will use this uniform decision for quantitative evaluation and defer further work to future user studies.

Also, to enforce the one-hot encoding in the learned counterfactuals, we add a regularization term for each categorical variable with high penalty to force the values for different levels of each categorical variable to sum to 1. At the end of the optimization, we pick the level with maximum value for each categorical variable.

*Hyperparameter choice.* Since counterfactual generation is a post-hoc step after training the model, it is not necessarily required that we use the same hyperparameter for every original input [31]. However, since hyperparameters can influence the found counterfactuals, it seems problematic if users are given counterfactuals generated by different hyperparameters.[1] In this work, we investigate the robustness of our counterfactual generation algorithms with respect to different values of $\lambda_1$, $\lambda_2$, two choices of diversity (*dist* or *dpp*), and two choices of yloss ($\ell_2$-loss or log-loss) through a grid search.

# 4 EVALUATING COUNTERFACTUALS

Despite recent interest in counterfactual explanations [28, 31], 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 is able to 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 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. In addition, we describe the datasets and baselines used in this work.

Nevertheless, it is important to emphasize that counterfactuals are eventually generated for end users. The effectiveness of CFs should be determined through human subject experiments. The goal of this work is to pave the way towards meaningful human subject experiments, and we will offer further discussions in Section 7.

## 4.1 Validity, Diversity, and Proximity

First, we define quantitative metrics for validity, diversity, and proximity for a counterfactual set that are independent of any particular optimization method. 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.

$$\%\text{Valid CFs} = \frac{|\{\text{unique instances in } C \text{ s.t. } f(c) > 0.75\}|}{k}$$

**Proximity.** We define proximity measures for categorical and continuous features separately. For continuous features, we define proximity as the mean of feature-wise L1 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. 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 [31]. Proximity for a set of examples is simply the average proximity over all the examples.

$$ContinuousProximity: \frac{1}{k}\sum_{i=1}^{k} \text{dist\_cont}(\boldsymbol{c}_i, \boldsymbol{x}),$$

where $\text{dist\_cont}(\boldsymbol{c}, \boldsymbol{x}) = \frac{1}{d_{cont}} \sum_{p=1}^{d_{cont}} \frac{|\boldsymbol{c}_p - \boldsymbol{x}_p|}{MAD_p}$ ($d_{cont}$ is the number of continuous variables and $MAD_p$ is the median absolute deviation for the $p$-th continuous variable).

For categorical features, 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 [25], 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 simple 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. Proximity among categorical features is then the (negative) average of these feature-wise distances.

$$CategoricalProximity: \frac{1}{k}\sum_{i=1}^{k} \text{dist\_cat}(\boldsymbol{c}_i, \boldsymbol{x}),$$

where $\text{dist\_cat}(\boldsymbol{c}, \boldsymbol{x}) = \frac{1}{d_{cat}} \sum_{p=1}^{d_{cat}} I(\boldsymbol{c}_p \neq \boldsymbol{x}_p)$ ($d_{cat}$ is the number of categorical variables).

**Diversity.** Diversity of counterfactual examples can be measured in an analogous way to proximity. Instead of feature-wise distance from the original input, we measure feature-wise distances between

[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.

each pair of counterfactual examples. Diversity for a set of counterfactual examples is the mean of the distances between each pair of examples. As for proximity, we compute separate diversity metrics for categorical and continuous features, based on separate categorical and continuous distance measures.

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

where $d$ is either dist_cont or dist_cat.

It is important to note that the distance metrics used here are intentionally general and different from Equation 5, so there is no guarantee that our generated counterfactuals would lead to strong performance with these metrics. Given the tradeoff between diversity and proximity, no method will be able to maximize both. Therefore, when evaluating a counterfactual generation method, we recommend searching the hyperparameter space of $\lambda_1$ and $\lambda_2$ in Equation 5 to achieve a good balance between the two metrics.

## 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 outputted 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 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. As with the *Proximity* metric, 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 sphere, we sample 1000 points at random per each original input to evaluate how well the secondary 1-NN model approximates the local decision boundary.

## 4.3 Datasets

**COMPAS.** This dataset was collected by ProPublica [3] as a part of their analysis on recidivism decisions in the United States. We preprocess the data based on previous analysis [12] and obtain 5 features, namely, bail applicants' age, gender, race, prior count of offenses, and degree of criminal charge. The machine learning model's task is to decide bail based on predicting which of the bail applicants will recidivate in the next two years.

**Adult.** This dataset contains demographic, educational and other information based on 1994 Census database and is available on the UCI machine learning repository [16]. We preprocess the data based on previous analysis [36] and obtain 8 features, namely, hours per week, education level, occupation, work class, race, age, marital status, and sex. The machine learning model's task is to classify whether an individual's income is over $\$50,000$.

**LendingClub.** This dataset contains five years (2007-2011) data of loans given by LendingClub, an online peer-to-peer lending company. We preprocess the data based on previous analyses [10, 13, 29] 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 machine learning model's task is to decide loan decisions based on a prediction of whether an individual will pay back their loan.

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 TensorFlow library to train a neural network model. For `COMPAS`, we obtain an accuracy of 0.67, comparable to the best accuracy reported on this dataset [3, 12]. Similarly, for `Adult` and `LendingClub` datasets, we obtain accuracies of 0.83 and 0.66 respectively.

## 4.4 Baselines

On these datasets, we employ the following baselines for generating counterfactual examples:

- `SingleCF`: We follow Wachter et al. [31] and generate a single CF example, optimizing for y-loss difference and proximity.
- `RandomInitCF`: Here we extend `SingleCF` to generate $k$ CF examples by initializing the optimizer independently with $k$ random starting points. Since the optimization loss function is non-convex, one should obtain different CF examples.
- `NoDiversityCF`: This method utilizes our proposed loss function that optimizes the set of $k$ examples simultaneously (Equation 5, 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. We initialize `DiverseCF` with random points close to the original input. For all methods, we use the ADAM optimizer [15] implementation in TensorFlow to minimize the loss and obtain CF examples.

In addition, we compare `DiverseCF` to one of the major feature-based local explanation methods, `LIME` [26], on how well it can

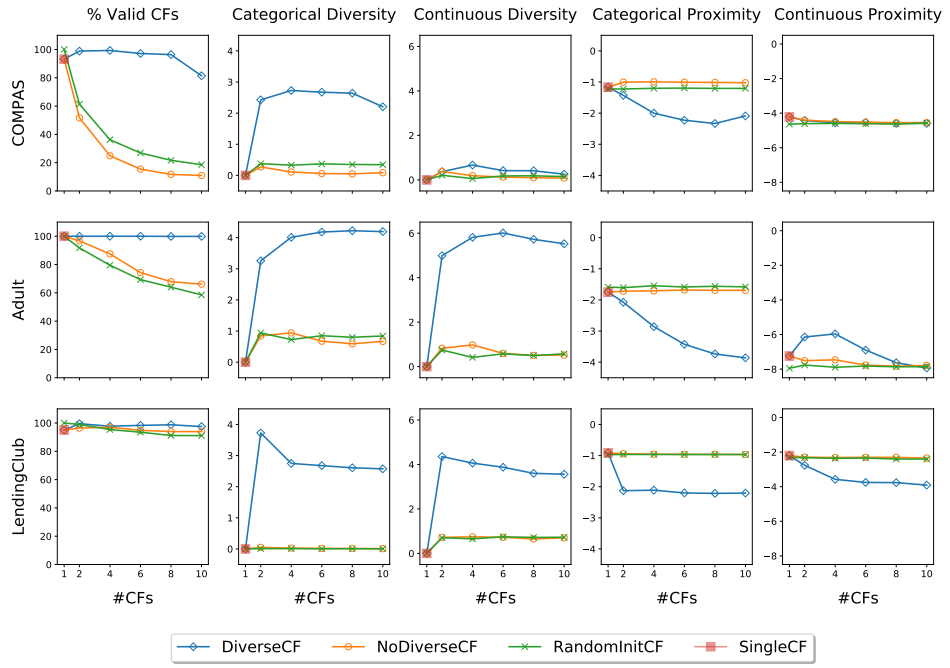



**Figure 1: Performance comparisons based on %Valid CFs, diversity, and proximity. `DiverseCF` clearly generates much more diverse counterfactuals than the baselines (sometimes with 0 diversity), and also finds a greater percentage of valid CFs. Although the proximity is worse as expected, the proximity in continuous variables is comparable with baselines.**

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 outputted linear model for each input instance as a local approximation of the ML model's decision surface.

## 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 found counterfactuals can approximate local decision boundaries as well as LIME, which is specifically designed for local approximation. Finally, we find that our approach is robust to choices of hyperparameters, making it suitable to run with a set of fixed hyperparameters for all potential users.

### 5.1 Quantitative Evaluation

We first evaluate `DiverseCF` based on valid CF generation, diversity, and proximity. We report results with hyperparameters chosen by a grid search as described in Section 5.4. Figure 1 shows the comparison with `SingleCF`, `RandomInitCF`, and `NoDiversityCF`.

**Validity.** Across all three datasets, we find that `DiverseCF` generates the highest fraction of unique valid CF examples, especially as the number of requested examples $k$ increases. The only exception is $k = 1$ where `RandomInitCF` generates a slightly higher fraction of valid CF examples. This is possibly because they use random initialization for optimization, as compared to `DiverseCF` which initializes examples close to the original input. As $k$ increases, baseline methods without an explicit diversity objective fail to generate unique examples, even with random initialization.

**Diversity.** Among the valid CFs, `DiverseCF` also generates more diverse examples than the baseline methods, for both continuous and categorical features. On average, `DiverseCF` results in CF sets where examples differ by 2-4 categorical features, while CF sets from baseline methods differ by at most 1. Among continuous features, average pairwise distance between CF examples is 4 and 6 for `Adult` and `LendingClub` datasets, in comparison to <1 for other methods. Only for the `COMPAS` dataset, `DiverseCF` outputs a low diversity with continuous features, because the dataset has only one continuous feature—prior count of offenses.

Average diversity for `DiverseCF` shows little variation as $k$ increases, indicating that the method can consistently generate diverse counterfactuals for a wide range of requested examples.

**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 continuous features, the difference in proximity compared to baselines is small.

### 5.2 Qualitative evaluation

To understand more about the resultant explanations, we look at some sample CF examples generated by `DiverseCF` in Table 2. In all three datasets, the examples capture some intuitive variables and vary them: PriorsCount in `COMPAS` Hours/Week and Education in `Adult`, and Income in `LendingClub` dataset. In addition, the user also sees other features that can be varied for the desired outcome. In the `COMPAS` input instance, we see that a person would have been granted bail if they had been a younger Asian and charged with Misdemeanor instead of Felony. Due to limited number of features

| **COMPAS** | PriorsCount | CrimeDegree | Race | Age | Sex |
|---|---|---|---|---|---|
| Original input (outcome: Will Recidivate) | 15.0 | Felony | African-American | >45 | Male |
| Counterfactuals (outcome: Won't Recidivate) | 0.0 | Felony | African-American | >45 | Female |
| | 0.0 | Felony | Native American | 25 - 45 | Male |
| | 0.0 | Misdemeanor | Native American | >45 | Male |
| | 33.0 | Misdemeanor | Asian | 25 - 45 | Male |

| **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) | 26.0 | Prof-school | Sales | Self-Employed | Other | 29.0 | Separated | Female |
| | 55.0 | Prof-school | Professional | Private | White | 42.0 | Married | Female |
| | 99.0 | Masters | Service | Private | White | 22.0 | Separated | Male |
| | 99.0 | Doctorate | White-Collar | Private | White | 56.0 | Single | Female |

| **LendingClub** | EmpYrs | Inc$ | #Ac | CrYrs | LoanGrade | HomeOwner | Purpose | State |
|---|---|---|---|---|---|---|---|---|
| Original input (outcome: Default) | 6.0 | 62400.0 | 7.0 | 13.0 | D | Mortgage | Debt | FL |
| Counterfactuals (outcome: Paid) | 6.0 | 76230.0 | 1.0 | 25.0 | A | Own | Purchase | FL |
| | 1.0 | 200000.0 | 1.0 | 1.0 | B | Mortgage | Debt | FL |
| | 4.0 | 200000.0 | 1.0 | 1.0 | D | Rent | Debt | TX |
| | 10.0 | 200000.0 | 14.0 | 60.0 | A | Mortgage | Debt | Other |

**Table 2: Examples of generated counterfactuals in each dataset.**

in the dataset, these CF examples do not provide suggestions on what to do, but nevertheless provide the user an accurate picture of scenarios where they would have been out on bail.

Note that three of the examples in LendingClub increase income to the maximum value possible (200000), probably because income is the most important feature. A person would have received the loan with such a high income even if all the other continuous variables were set to the lowest (just 1 account, 1 year of credit, and 1 year of employment). To explore other possibilities, the user may then choose to fix a different maximum income and explore possibilities within this range. Alternatively, the user may also modify the weights for each feature for computing distance as described in Section 3.2. When the weights for other continuous features are increased to twice their value — 2 instead of 1 — we obtain the CF examples that do not include changing income to 200k. In fact, we obtain one where the income is reduced to 50,622.

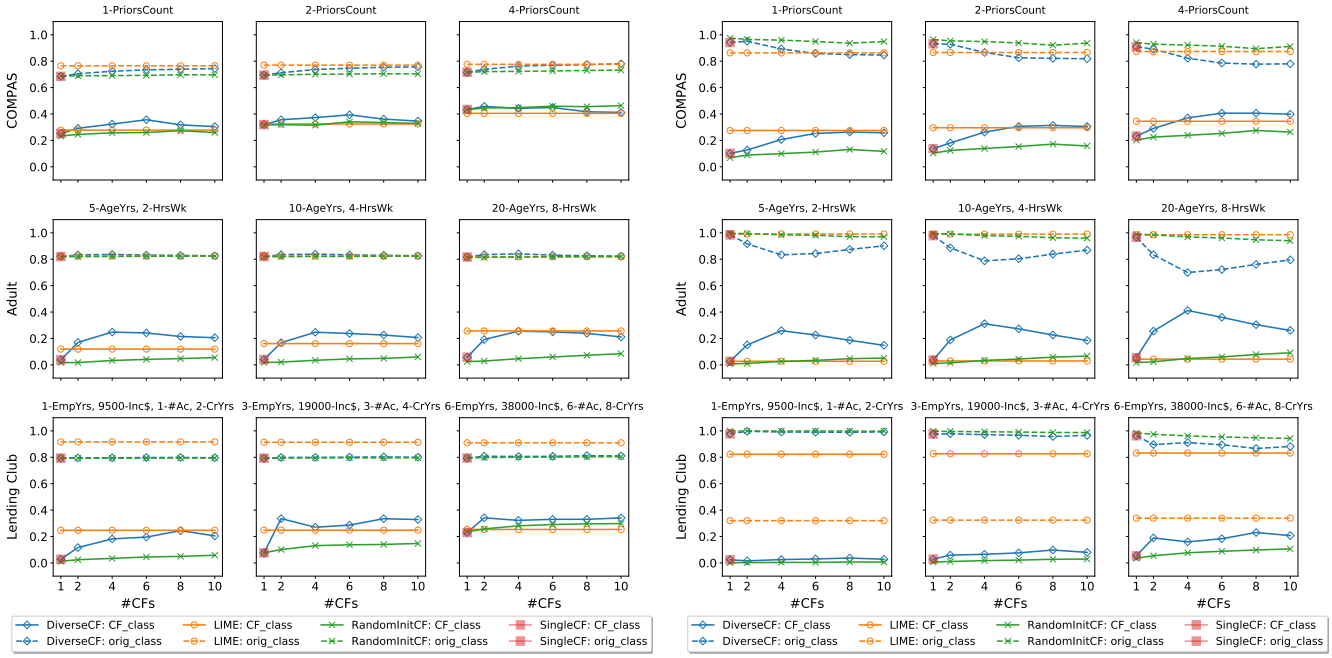
Overall, these initial set of CF examples help in understanding the important variations as learned by the algorithm. We expect the user to engage their actionability constraints with this initial set to iteratively generate focused CF examples, that can help find useful variations. In addition, these examples can also expose biases or odd edge-cases in the ML model itself, that can be useful for the model builder in debugging, or for fairness evaluators in discovering potential bias. For instance, in the Adult dataset, we obtained a CF example which involves changing only someone's gender and race to reach the desired class; such an example prompts further inquiry to ensure that the algorithm is not sustaining an unwanted bias.

## 5.3 Approximating local decision boundary

As a proxy for understanding how well users can guess the local decision boundary of the ML model, we compare classification performance of models based on the proposed DiverseCF method, baseline methods, and LIME. We first look at accuracy. Even with a handful (2-11) training examples, we find that 1-NN classifiers obtain comparable accuracy to the LIME classifier. For COMPAS dataset with $k = 4$, the accuracies of DiverseCF (75%), RandomInitCF (73%) and LIME (77%) are close to each other; we obtain similar results with other $k$ from 2-10. Notably, in the case of LendingClub dataset, DiverseCF (79%) and RandomInitCF (79%) obtain a substantially higher accuracy than LIME (44%). Here LIME performs worse because it mis-estimates the relative prevalence of CF versus original outcome classes near the original input because LIME's sampling strategy around the original input does not cover all possible changes in categorical variables.

Given the class imbalance in data points near the original input, precision and recall for the counterfactual outcome class provide a better understanding of the different methods' performance (Figure 2). For all datasets, DiverseCF obtains the highest precision for the CF outcome class. That is, when guessing the instances to be in the counterfactual class (and thus worth acting upon), using DiverseCF can lead to the lowest false positive rate, even when compared to a local approximator method like LIME. For COMPAS and Adult dataset, DiverseCF also achieves a higher recall than LIME or RandomInitCF. However, its recall is lower for the LendingClub dataset, where LIME obtains nearly 80% recall compared to <20% for DiverseCF. As we saw above for accuracy, this is because LIME predicts a majority of the instances to belong to the CF class, whereas DiverseCF focuses on obtaining a higher precision. In general, we also observe that precision of the counterfactual class increases as the number of CFs grows, which again motivates generating diverse counterfactuals.

Overall, these results show that examples from DiverseCF are able to approximate the local decision boundary better than RandomInitCF, and in many cases, even better than LIME. Still, the gold-standard test will be to conduct a behavioral study where people evaluate whether CF examples are more explanatory than past approaches, which we leave for future work.




**(a) Precision** **(b) Recall**

**Figure 2: Performance comparisons of 1-NN learned from counterfactuals. `DiverseCF` outperforms baseline CF methods and sometimes even LIME.**

## 5.4 Sensitivity to hyperparameters

Finally, we describe how sensitive our method is to changes in hyperparameters. We consider $\lambda 1$ and $\lambda 2$ to vary between $\{0, 2^i (i = -5, \ldots, 1)\}$, consider L2 and log-loss as the two candidates for y-loss, and *dist_diversity* and *dpp_diversity* for diversity part of the loss function from Equation 5.

To select an optimal configuration, we generate CF examples for each possible configuration and evaluate them on the three metrics from Section 4.1: Validity, Diversity and Proximity. For instance, Figure 3 shows the grid search results for the Validity metric, as we vary $\lambda 1$, $\lambda 2$, y-loss and diversity parts of the loss function. Based on this, we find that log-loss with *dpp_diversity* gives the highest number of high-fraction valid CF sets; it has the highest 95th and 50th percentile of percentage of valid CFs (note that the important test is how easy it is to find robust hyperparameters that produce a good results rather than obtaining the max-min performance). Next, we look at similar plots for Diversity and Proximity (omitted for space) to choose suitable values for $\lambda 1$ and $\lambda 2$.

Irrespective of the chosen optimal configuration, Figure 3 also shows that the percentage of valid CFs found does not change substantially with different configurations. That is, the sensitivity of Validity of CFs to a specific configuration is low, indicating that it is possible to generate good quality CF sets with other configurations as well. Such results are reassuring for using the same hyperparameter in practice.

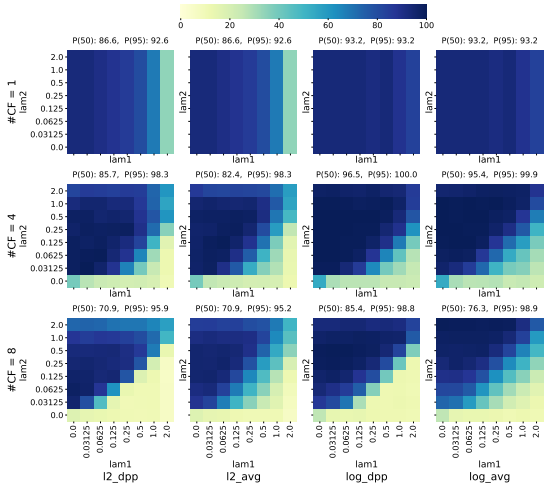



**Figure 3: Hyperparameter sensitivity based on *% valid CFs.***

## 6 FEASIBILITY OF COUNTERFACTUALS

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 valid, diverse and proximal, such a CF example is not feasible and thus not actionable by the user. In this context, we argue that incorporating causal models of data generation is important to avoid presenting infeasible counterfactuals to the user.

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 causal edge between them. Using this, we construct constraints that any countefactual should follow. For instance, any counterfactual that

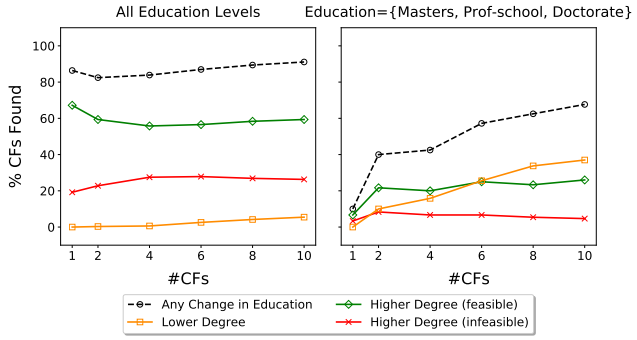



**Figure 4: Filtering based on a causal graph. The left figure shows that there are over 80% CFs that include any change in education, out of which one-fourth are infeasible. If we zoom in to people with higher degrees, almost half of the changes in educational degrees are infeasible.**

changes the cause without 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.

We consider two infeasible changes derived from the causal relation education $\uparrow \Rightarrow$ age $\uparrow$: education$\not\downarrow$ and education $\uparrow \not\Rightarrow$age $\downarrow$. The results of the filtering are shown in Figure 4. More than one-fourth of the obtained counterfactuals which include a change in education level are infeasible. This percentage increases as we look at CF examples for the highly educated people (Masters, Doctorate and Professional) where explanations to switch to a lower education degree are as high as 50% of all CFs that modified educational level.

Post-hoc filtering can ensure the feasibility of the resultant CF set, but it can be more efficient to incorporate causal constraints during the generation of counterfactuals. We leave this for future work.

## 7 CONCLUDING DISCUSSION

Building upon prior work on counterfactual explanations [28, 31], we proposed a framework for generating and evaluating a diverse and feasible set of counterfactual explanations. Here we note directions for future work. First, our method assumes knowledge of the gradient of the ML model. It will be useful to construct methods that can work for fully *black-box* ML models. Second, we would like to incorporate causal knowledge in the generation of CF examples, rather than as a post-hoc filtering step. Third, as we saw in Section 5.2, it is important to understand human 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. Finally, while we focused on the utility for the end-user who is the subject of a ML-based decision, we argue CF explanations can be useful for different stakeholders in the decision making process [30], such as the model designers, decision-makers such as a judge or a doctor, and decision evaluators such as third-party auditors.

## REFERENCES


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

[2] 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).

[3] 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.

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

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

[6] 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).

[7] 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*.

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

[9] 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.

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

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

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

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

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

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

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

[17] 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.

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

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

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

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

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

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

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

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

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

[27] 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.

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

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

[30] 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).

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

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

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

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

[35] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. 2018. Interpretable basis decomposition for visual explanation. In *Proceedings of ECCV*.
[36] Haojun Zhu. 2016. Predicting Earning Potential using the Adult Dataset. https://rpubs.com/H_Zhu/235617.

# A SUPPLEMENTARY MATERIALS

Here we discuss implementation details relevant for reproducibility. We also plan to release our code on GitHub so that people can reproduce our results and obtain counterfactual examples for any new dataset.

## A.1 Building ML Models

First, we build a ML model for each dataset that gives accuracy comparable to previously established benchmarks, using Adam Optimizer [15] in TensorFlow. We discussed the datasets that we used in our analysis in section 4.3. Further, table 3 describes 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-neural network gives best generalization ability for all datasets. Similarly, we also found that while 20 hidden neurons worked well for `COMPAS` and `Adult` dataset, increasing more than 5 neurons worsened the generalization for `LendingClub` dataset. Furthermore, to handle the class imbalance problem while training with the `LendingClub` dataset, we oversampled the training instances belonging to the minority class (with label 'Default') accordingly. Interestingly, though `Adult` dataset also has imbalanced data, the ML model was able to capture both the classes with the hyperparameters that we set.

| | COMPAS | Adult | Lending Club |
|---|---|---|---|
| # Continuous Features | 1 | 2 | 4 |
| # Categorical Features | 4 | 6 | 4 |
| Range across all Continuous Features (Min, Avg, Max) | (0, 3.5, 38) | (1, 39.5, 99) | (1, 16292, 200000) |
| # Levels across all Categorical Features (Min, Avg, Max) | (2, 3.25, 6) | (2, 4.5, 8) | (4, 5.5, 7) |
| Undesired Class | Will Recidivate | <=50K | Default |
| Desired Counterfactual Class | Won't Recidivate | >50K | Paid |
| Training Data Size | 1443 | 6513 | 8133 |
| Fraction of Instances with Desired CF Outcome | 0.55 | 0.25 | 0.8 |
| ML Model Type | ANN(1, 20) | ANN(1, 20) | ANN(1, 5) |
| ML Model Accuracy | 67% | 83% | 66% |

**Table 3: Dataset description.**

## A.2 Hyperparameter tuning

Here we describe the hyperparameters for counterfactul generation process. We used the the Adam Optimizer [15] in TensorFlow for generating counterfactuals too. We experimented with following values for different hyperparameters:

- $\lambda_1$ and $\lambda_2$: $\{0, 2^i (i = -5, \ldots, 1)\}$
- yloss : $\ell_2$-loss or log-loss
- diversity : *dist_diversity* and *dpp_diversity*

We varied $\lambda_1$ and $\lambda_2$ between $\{0, 2^i (i = -5, -4, \ldots, 1)\}$, because we noticed that choosing a value more than 2 does not give valid counterfactuals:the optimization starts to neglect the first loss component pertaining to y-loss. We found that the following parameters worked best for different datasets:

- **COMPAS** : $\lambda_1$: 0.5, $\lambda_2$: 1.0, yloss: log-loss, diversity: *dpp_diversity*
- **Adult** : $\lambda_1$: 0.5, $\lambda_2$: 1.0, yloss: log-loss, diversity: *dpp_diversity*
- **LendingClub** : $\lambda_1$: 0.5, $\lambda_2$: 1.0, yloss: $\ell_2$-loss, diversity: *dist_diversity*

## A.3 LIME implementation

In quantitative evaluation, we noticed that the default LIME implementation discretizes the continuous features and treats them as categorical features. However, we noticed that for all the three datasets, setting the `discretize_continuous` option as False gives better performance for LIME in our experiments. Hence, we ran our quantitative evaluation, as explained in 5.1, with this updated setting for LIME.