GAM Coach: Towards Interactive and User-centered Algorithmic Recourse

Zijie J. Wang
Georgia Tech
Atlanta, USA

Jennifer Wortman Vaughan
Microsoft Research
New York, USA

Rich Caruana
Microsoft Research
Redmond, USA

Duen Horng Chau
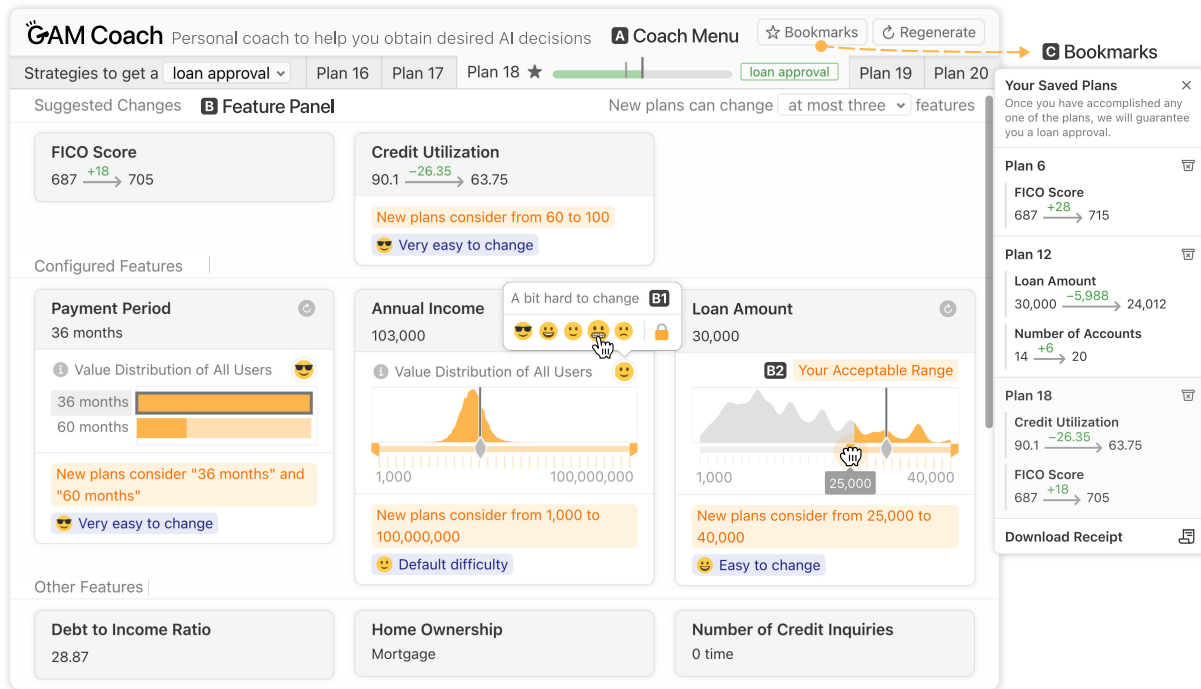Georgia Tech
Atlanta, USA

Figure 1: With a novel interactive interface and an adaptation of integer linear programming, GAM Coach empowers people who are impacted by machine learning-based decision-making systems to iteratively generate algorithmic recourse plans that reflect their preferences. Take loan application as an example. (A) The *Coach Menu* helps a rejected loan applicant browse diverse recourse plans that would lead to loan approval. After the user selects a plan, (B) the *Feature Panel* visualizes all feature information with progressive disclosure, enabling users to explore how hypothetical inputs affect the model's decision and specify recourse preferences—such as (B1) the *difficulty* of changing a feature and (B2) its *acceptable range* of values—guiding GAM Coach to generate actionable plans. (C) The *Bookmarks* window allows users to compare bookmarked plans and save a verifiable receipt.

ABSTRACT

Machine learning (ML) recourse techniques are increasingly used in high-stakes domains, providing end users with actions to alter ML predictions, but they assume ML developers understand what input variables can be changed. However, a recourse plan's actionability is subjective and unlikely to match developers' expectations completely. We present GAM Coach, a novel open-source system that adapts integer linear programming to generate customizable counterfactual explanations for Generalized Additive Models (GAMs), and leverages interactive visualizations to enable end users to iteratively generate recourse plans meeting their needs. A quantitative user study with 41 participants shows our tool is usable and useful, and users prefer personalized recourse plans over generic plans. Through a log analysis, we explore how users discover satisfactory recourse plans, and provide empirical evidence that transparency can lead to more opportunities for everyday users to discover counterintuitive patterns in ML models. GAM Coach is available at: **https://poloclub.github.io/gam-coach/**.

CCS CONCEPTS

• **Computing methodologies → Machine learning**; • **Human-centered computing → Interactive systems and tools**.

KEYWORDS

Algorithmic Recourse, Counterfactual Explanation, Interpretability

1 INTRODUCTION

As machine learning (ML) is increasingly used in high-stakes decision-making, such as lending [77], hiring [49], and college admissions [93], there has been a call for greater transparency and increased opportunities for algorithmic recourse [88]. Algorithmic recourse aims to help those impacted by ML systems learn about the decision rules used [74], and provide suggestions for *actions* to change decision outcome in the future [83]. This often involves generating counterfactual (CF) examples, which suggest minimal changes in a few features that would have led to the desired decision outcome [88], such as "if you had decreased your requested loan amount by $9k and changed your home ownership from renting to mortgage, your loan application would have been approved." (Fig. 2A)

For such approaches to be useful, it is necessary for the suggested actions to be *actionable*—realistic actions that users can appreciate and follow in their real-life circumstances. In the example above, changing home ownership status would arguably not be an actionable suggestion for most loan applicants. To provide actionable recourse, recent work proposes techniques such as generating concise CF examples [48], creating a diverse set of CF examples [57, 70], and grouping features into different actionability categories [36]. These approaches often rely on the underlying assumption that ML developers can measure and predict which CF examples are actionable for all users. However, the actionability of recourse is ultimately subjective and varies from one user to another [3, 87], or even for a single user at different times [50, 99]. Therefore, there is a pressing need to capture and integrate user preferences into algorithmic recourse [3, 42]. GAM Coach aims to take a user-centered approach (Fig. 2B–C) to fill this critical research gap. In this work, we **contribute**:

- **GAM Coach, the first interactive algorithmic recourse tool that empowers end users** to specify their recourse *preferences*, such as difficulty and acceptable range for changing a feature, and iteratively *fine-tune* actionable recourse plans (Fig. 1). With an exploratory interface design [76], our tool helps users understand the ML model behaviors by experimenting with hypothetical input values and inspecting their effects on the model outcomes. Our tool advances over existing interactive ML tools [19, 95], overcoming unique design challenges identified from a literature review of recent algorithmic recourse work (§ 3, § 5).

- **Novel adaptation of integer linear programming to generate CF examples.** To operationalize interactive recourse, we ground our research in generalized additive models (GAMs) [6, 59], a popular class of models that performs competitively to other state-of-the-art models yet has a transparent and simple structure [7, 60, 89, 94]. GAMs enable end users to probe model behaviors with hypothetical inputs in real time directly in web browsers. Adapting integer linear programming, we propose an
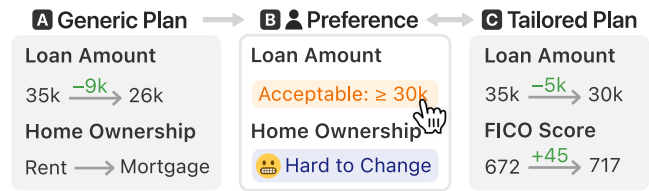

Figure 2: GAM Coach enables end users to iteratively fine-tune recourse plans. (A) If a user finds the initial generic plan less actionable, (B) they can specify their recourse preferences through simple interactions. (C) Our tool will then generate tailored plans that reflect the user's preferences.

efficient and flexible method to generate optimal CF examples for GAM-based classifiers and regressors with continuous and categorical features and pairwise feature interactions [52] (§ 4).

- **Design lessons distilled from a user study with log analysis.** We conducted an online user study with 41 Amazon Mechanical Turk workers to evaluate GAM Coach and investigate how everyday users would use an interactive algorithmic recourse tool. Through analyzing participants' interaction logs and subjective ratings in a hypothetical lending scenario, our study highlights that GAM Coach is usable and useful, and users prefer personalized recourse plans over generic plans. We discuss the *characteristics* of users' satisfactory recourse plans, *approaches* users take to discover them, and *design lessons* for future interactive recourse tools. We also provide empirical evidence that with transparency, everyday users can discover and are often puzzled by counterintuitive patterns in ML models (§ 6).

- **An open-source, web-based implementation** that broadens people's access to developing and using interactive algorithmic recourse tools. We implement our CF generation method in both Python and JavaScript, enabling future researchers to use it on diverse platforms. We develop GAM Coach with modern web technologies such as WebAssembly, so that anyone can access our tool using their web browsers without the need for installation or a dedicated backend server. We open-source[1] our CF generation library and GAM Coach system with comprehensive documentation[2] (§ 5.5). For a demo video of GAM Coach, visit **https://youtu.be/ubacP34H9XE**.

To design and evaluate a prospective interface [76] for interactive algorithmic recourse, we situate GAM Coach in loan application scenarios. However, we caution that adapting GAM Coach for real lending settings would require further research with financial and legal experts as well as people who would be impacted by the system. Our goal is for this work to serve as a foundation for the design of future user-centered recourse and interpretable ML tools.

2 RELATED WORK

2.1 Algorithmic Recourse

Algorithmic recourse aims to design techniques that provide those impacted by ML systems with actionable feedback about how to

---

[1] GAM Coach code: **https://github.com/poloclub/gam-coach**
[2] GAM Coach documentation: **https://poloclub.github.io/gam-coach/docs**

alter the outcome of ML models. Popularized by Wachter et al. [88], researchers typically generate this actionable feedback by creating CF examples. Here, a CF example represents a recourse plan that contains minimal changes to the original input but leads to a different model prediction [35, 83]. For example, a bank that uses ML models to inform loan application decisions can provide a rejected loan applicant with a recourse plan that suggests the applicant increase their annual income by $5k so that they can obtain a loan approval. CF examples not only inform end users about the key features contributing to the decision, but also provide suggestions that end users can act on to obtain the desired outcome [83]. Researchers have developed various methods to generate CF examples, such as casting it as an optimization problem [e.g., 11, 33, 56, 70, 83, 88], searching through similar samples [e.g., 12, 21, 40, 73, 85], and developing generative models [e.g., 13, 30, 41, 78].

It is challenging to generate helpful CF examples in practice. Besides making minimal changes, a helpful CF example should also be *actionable* for the end user [39, 83]. To generate actionable recourse plans, recent research includes proposals to find concise CF examples [48], consider causality [36, 37, 54], present diverse plans [57, 70], and assign features with different actionability scores [36]. However, the actionability of recourse is ultimately subjective and varies among end users [42, 50, 87, 99]. To restore users' autonomy with CF examples, some researchers explore the potential of interactive tools. For example, Prospector [46], What-If Tool [95], Polyjuice [96], and AdViCE [20] leverage interactive visualizations to help ML developers debug models with CF examples. Context Sight [98] allows ML developers to analyze model errors by customizing the acceptable feature range and desired number of changes in CF examples. CEB [58] interactively presents CF examples to help non-experts understand neural networks. In comparison, GAM Coach aims to empower *end users* to discover *actionable* strategies to alter undesirable ML decisions.

DECE [9] is a visual analytics tool designed to help ML developers and end users interpret neural network predictions with CF examples. It allows users to customize CF examples by specifying acceptable feature ranges. In comparison, while the interface for GAM Coach is model agnostic, the recourse generation technique it employs is tailored to GAMs, a different model family, and our tool especially focuses on end users without an ML background. We evaluate GAM Coach through an observational log study with 41 crowdworkers, while DECE is evaluated through three expert interviews. These evaluations provide complementary viewpoints and insights into how interactive recourse tools may be used in practice. Possibly closest in spirit to our work is ViCE [19], an interactive visualization tool that generates CF examples on end users' selected continuous features. In contrast, GAM Coach—which supports both continuous and categorical features, as well as their pairwise interactions—allows end users to specify a much wider range of recourse preferences including feature difficulty, acceptable range, and the number of features to change. Our tool then generates *optimal* and *diverse* CF examples meeting specified preferences.

2.2 Interactive Tools for Interpretable ML

Besides CF explanations, researchers have developed interactive tools to help different ML stakeholders interpret ML models [e.g.,

28, 31, 65, 92]. In particular, the simple structure and high performance of GAMs have attracted many researchers to use this model to explore how interactivity plays a role in interpretable ML. For example, Gamut [27] provides both global and local explanations by visualizing the shape functions in GAMs. Similarly, TeleGam [29] helps users understand GAM predictions by combining both graphical and textual explanations. GAM Changer [90] supports users to edit GAM model parameters through interactive visualization. However, the target users of these tools are ML experts, such as ML researchers and model developers, or domain experts who need to vet and correct models before deployment. In comparison, GAM Coach targets people who are impacted by ML models and who are less knowledgeable about ML and domain-specific concepts [81].

There is an increasing body of research in developing interactive systems to help *non-experts* interact with ML models. The main goal of these tools is to educate non-experts about the underlying mechanisms of ML models. For example, Teachable Machine [5] helps users learn about basic ML concepts through interactive demos. Tensorflow Playground [80], GAN Lab [32], and CNN Explainer [91] use interactive visualizations to help novices learn about the underlying mechanisms of neural networks, generative adversarial networks, and convolutional neural networks, respectively. In contrast, instead of educating non-experts on the technical inner workings of ML models, we focus on helping non-experts who are impacted by ML models understand why a model makes a particular decision and what actions they can take to alter that decision.

3 DESIGN GOALS

Our goal is to design and develop an interactive, visual experimentation tool that respects end users' autonomy in algorithmic recourse, helping them discover and fine-tune recourse plans that reflect their preferences and needs. We identify five main design goals of GAM Coach through synthesizing the trends and limitations of traditional algorithmic recourse systems [e.g., 2, 3, 35, 39, 55, 76, 88].

G1. Visual summary of diverse algorithmic recourse plans. To help end users find actionable recourse plans, researchers suggest presenting diverse CF options that users can pick from [3, 57]. Thus, GAM Coach should efficiently generate diverse recourse plans (§ 4.2) and present a visual summary of each plan as well as display multiple plans at the same time (§ 5.1). This could help users compare different strategies and inform interactions to generate better recourse plans.

G2. Easy ways to specify recourse preferences. What makes a recourse plan actionable varies from one user to another—it is crucial for a recourse tool to enable users to specify a wide range of recourse preferences [3, 42, 55]. Therefore, we would like to allow users to easily configure (1) the *difficulty* of changing a feature, (2) the *acceptable range* within which a feature can change, and (2) the *maximum number of features* that a recourse plan can change (§ 5.2), and GAM Coach should generate plans reflecting users' specified preferences (§ 4.3). This interactive recourse design would empower users to iteratively customize recourse plans until they find satisfactory plans.

G3. Exploratory interface to experiment with hypothetical inputs. The goal of algorithmic recourse is not only to help users identify actions to alter unfavorable model decisions, but

also to help them understand how a model makes decisions [35, 88]. When explaining a model's decision-making, research shows that interfaces allowing users to probe an ML model with different inputs help users understand model behaviors and lead to greater satisfaction with the model [10, 62, 76, 95]. Therefore, we would like GAM Coach to enable users to experiment with different hypothetical inputs and inspect how these changes affect the model's decision (§ 5.2).

G4. **Clear communication and engagement.** The target users of GAM Coach are everyday people who are usually less knowledgeable about ML and domain-specific concepts [81]. Our goal is to design and develop an interactive system that is easy to understand and engaging to use, requiring the tool to communicate and explain recourse plans and domain-specific information to end users (§ 5.2, § 5.3).

G5. **Open-source and model-agnostic implementation.** We aim to develop an interactive recourse tool that is easily accessible to users, with no installation required. By using web browsers as the platform, users can directly access GAM Coach through their laptops or tablets. Additionally, we aim to make our interface model-agnostic so that future researchers can use it with different ML models and recourse techniques. Finally, we would like to open-source our implementation and provide documentation to support future design, research, and development of interactive algorithmic recourse (§ 5.5).

4 TECHNIQUES FOR CUSTOMIZABLE RECOURSE GENERATION

Given our design goals (G1–G5), it is crucial for GAM Coach to generate customizable recourse plans interactively with a short response time. Therefore, we base our design on GAMs, a family of ML models that perform competitively to state-of-the-art models yet have a transparent and simple structure—enabling end users to probe model behaviors in real-time with hypothetical inputs. In addition, with a novel adaptation of integer linear programming (§ 4.2), GAMs allow us to efficiently generate recourse plans that respect users' preferences and thus achieve our design goals (§ 4.3).

4.1 Model Choice

To operationalize our design of interactive algorithmic recourse, we ground our research in GAMs [24]. More specifically, we make use of a type of GAMs called *Explainable Boosting Machines*, (EBMs) [6, 60], which perform competitively to the state-of-the-art black-box models yet have a transparent and simple structure [7, 60, 89, 94]. Compared to simple models like linear models or decision trees, EBMs achieve superior accuracy by learning complex relations between features through gradient-boosting trees [52], and thus deploying our design is realistic. Compared to complex models like neural networks, EBMs have a similar performance on tabular data but a simpler structure; therefore, users can probe model behaviors in real-time with hypothetical inputs (G3).

Given an input $x \in \mathbb{R}^k$ with k features, the output $y \in \mathbb{R}$ of an EBM model can be written as:

$$y = l\,(S_x)$$
$$S_x = \beta_0 + f_1\,(x_1) + f_2\,(x_2) + \cdots + f_k\,(x_k) + \cdots + f_{ij}(x_i, x_j) \quad (1)$$

Here, each shape function f_j for single features $j \in \{1, 2, \ldots, k\}$ or $f_{ij}(x_i, x_j)$ for pairwise interactions between features [52] is learned using gradient-boosted trees [51]. S_x is the sum of all shape function outputs as well as the intercept constant β_0. The model converts S_x to the output y through a link function l that is determined by the ML task. For example, a sigmoid function is used for binary classifications, and an identity function for regressions.

What distinguishes EBMs from other GAMs is that the shape function f_j or f_{ij} is an ensemble of trees, mapping a main effect feature value x_j or a pairwise interaction (x_i, x_j) to a scalar score. Before training, EBM applies *equal-frequency binning* on each continuous feature, where bins have different widths but the same number of training samples. This discrete bucketing process is commonly used to speed up gradient-boosting tree methods with little cost in accuracy, such as in popular tree-based models LightGBM [38] and XGBoost [8]. For categorical features, EBMs treat each discrete level as a bin. Once an EBM model is trained, the learned parameters for each ensemble of trees which defines the feature split points and scores in each region defined by these split points are transformed to a *lookup histogram* (for univariate features) and a *lookup table* (for pairwise interactions). When predicting on a data point, the model first looks up corresponding scores for all feature values and interaction terms and then applies Equation 1 to compute the output.

4.2 CF Generation: Integer Linear Programming

A recourse plan is a CF example c that makes minimal changes to the original input x but leads to a different prediction. Without loss of generality, we use binary classification as an example, with sigmoid function $\sigma(a) = \frac{1}{1+e^{-a}}$ as a link function. If $\sigma\,(S_x) \geq 0.5$ or $S_x \geq 0$, the model predicts the input x as positive; otherwise it predicts x as negative. To generate c, we can change x so that the new score S_c has a different sign from S_x. Note that S_x is a linear combination of shape function scores and so is $S_c - S_x$. Thus, we can express this counterfactual constraint as a linear constraint (derivation in § A.2). To enforce c to only make minimal changes to x, we can minimize the distance between c and x, which can also be expressed as a linear function (§ A.3). Since all constraints are linear, and there are a finite number of bins for each feature, we express the GAM Coach recourse generation as an *integer linear program*:

$$\min \ \text{distance} \qquad\qquad\qquad\qquad\qquad\qquad (2a)$$

$$\text{s.t. distance} = \sum_{i=1}^{k} \sum_{b \in B_i} d_{ib} v_{ib} \qquad\qquad\qquad (2b)$$

$$-S_x \leq \sum_{i=1}^{k} \sum_{b \in B_i} g_{ib} v_{ib} + \sum_{(i,j) \in N} \sum_{b_1 \in B_i} \sum_{b_2 \in B_j} h_{ijb_1b_2} z_{ijb_1b_2} \quad (2c)$$

$$z_{ijb_1b_2} = v_{ib_1} v_{jb_2} \quad \text{for } (i,j) \in N, \ \ b_1 \in B_i, \ \ b_2 \in B_j \quad (2d)$$

$$\sum_{b \in B_i} v_{ib} \leq 1 \qquad \text{for } i = 1, \ldots, k \qquad\qquad (2e)$$

$$v_{ib} \in \{0, 1\} \qquad \text{for } i = 1, \ldots, k, \ \ b \in B_i \qquad (2f)$$

$$z_{ijb_1b_2} \in \{0, 1\} \qquad \text{for } (i,j) \in N, \ \ b_1 \in B_i, \ \ b_2 \in B_j \quad (2g)$$

We use an indicator variable v_{ib} (2f) to denote if a main effect bin is active: if $v_{ib} = 1$, we change the feature value of x_i to the closest value in its bin b. All bin options of x_i are included in a set B_i. For each feature x_i, there can be at most one active bin (2e); if there is no active bin, then we do not change the value of x_i. We use an

indicator variable $z_{ijb_1b_2}$ (2g) to denote if a pairwise interaction effect is active—it is active if and only if bin b_1 of x_i and bin b_2 of x_j are both active (2d). The set N includes all available interaction effect terms. Constraint 2b determines the total distance cost for a potential CF example; it uses a set of pre-computed distance costs d_{ib} of changing one feature x_i to the closest value in bin b. Constraint 2c ensures that any solution would flip the model prediction, by gaining enough total score from main effect scores (g_{ib}) and interaction effect scores ($h_{ijb_1b_2}$). Constants g_{ib} and $h_{ijb_1b_2}$ are pre-computed and adjusted for cases where a single active main effect bin results in changes in interaction terms (see § A.4 for details).

Novelty. Advancing existing works that use integer linear programs for CF generation (on linear models [83] or using a linear approximation of neural networks [56]), our algorithm is the first that works on non-linear models without approximation. Our algorithm is also the first and only CF method specifically designed for EBM models. Without it, users would have to rely on model-agnostic techniques such as genetic algorithm [73] and KD-tree [85] to generate CF examples. These model-agnostic methods do not allow for customization. Also, by quantitatively comparing our method with these two model-agnostic CF techniques on three datasets, we find CFs generated by our method are significantly *closer* to the original input, *more sparse*, and encounter *less failures* (see § A.9 and Table S1 for details).

Generalizability. Our algorithm can easily be adapted for EBM regressors and multiclass classifiers. For regression, we modify the left side and the inequality of constraint 2c to bound the prediction value in the desired range (see § A.6 for details). For multiclass classification, we can modify constraint 2c to ensure that the desired class has the largest score (see § A.7 for details). In addition to EBMs, one can also adapt our algorithm to generate CF examples for linear models [83]. For other non-linear models (e.g., neural networks), one can first use a linear approximation [56] and then apply our algorithm, verifying suggested recourse plans with respect to the original model. If the suggested recourse plan would not change the output of the original model, an alternative can be generated by solving the program again with the previous solution blocked.

Scalability. Modern linear solvers can efficiently solve our integer linear programs. The complexity of solving an integer linear program increases along two factors: the number of variables and the number of constraints. In Equation 2, all variables are binary—making the program easier to solve than a program with non-binary integer variables. For any dataset, there are always exactly 3 constraints from 2b, 2c, and 2e. The number of constraints from 2d increases along the number of interaction terms $|N|$ and the number of bins per feature $|B_i|$ on these interaction terms. In practice, $|N|$ and $|B_i|$ are often bounded to ensure EBM are interpretable. For example, by default the popular EBM library InterpretML [60] bounds $|N| \leq 10$ and $|B_i| \leq 32$. Therefore, in the worst-case scenario with 10 continuous-continuous interaction terms, there will be at most $10 \times 32 \times 32 = 10,240$ constraints from 2d. For instance, on the Communities and Crime dataset [67] with 119 continuous features, 1 categorical feature, and 10 pairwise interaction terms, there are about 7.2k constraints and 3.6k variables in our program. It only takes about 0.5–3.0 seconds to generate a recourse plan using Firefox Browser on a MacBook (see § A.10 for details).

4.3 Recourse Customization

With integer linear programming, we can generate recourse plans that reflect a wide range of user preferences (G2). For example, to prioritize a feature that is *easier for a user to change*, we can lower the distance cost d_{ib} for that feature (§ A.5). To enforce recourse plans to only change a feature in a user specified *acceptable range*, we can remove out-of-range binary variables v_{ib}. If a user requires the recourse plans to only change *at most p features*, we can add an additional linear constraint $\sum_{i=1}^{k} \sum_{b \in B_i} v_{ib} \leq p$. Finally, with modern linear solvers, we can efficiently generate diverse recourse plans (G1) by solving the program multiple times while blocking previous solutions (see § A.6–§ A.8 for details).

5 USER INTERFACE

Given the design goals (G1–G5) described in § 3, we present GAM Coach, an interactive tool that empowers end users to specify preferences and iteratively fine-tune recourse plans (Fig. 4). The interface tightly integrates three components: the *Coach Menu* that provides overall controls and organizes multiple recourse plans as tabs (§ 5.1), the *Feature Panel* containing *Feature Cards* that allow users to specify recourse preferences with simple interactions (§ 5.2), and the *Bookmark Window* summarizing saved recourse plans (§ 5.3). To explain these views in this section, we use a loan application scenario with the LendingClub dataset [1], where a bank refers a rejected loan applicant to GAM Coach pre-loaded with the applicant's input data. Our tool can be easily applied to GAMs trained on different datasets while providing a consistent user experience. On GAM Coach's **public demo page**, we present five additional examples with five datasets that are commonly used in algorithmic recourse literature: Communities and Crime [67] (also used in the second usage scenario in § 5.4), Taiwan Credit [97], German Credit [14], Adult [45], and COMPAS [47].

5.1 Coach Menu

The *Coach Menu* (Fig. 1A) is the primary control panel of GAM Coach. Users can use the dropdown menu and input fields to specify desired decisions for classification and regression. For each recourse plan generation iteration, the tool generates five diverse plans (G1) to help users achieve their goal, with each plan representing a CF example. Users can access each plan by clicking the corresponding tab on the plan tab bar. When a plan is selected, the *Feature Panel* updates to show details about the plan, and the plan's corresponding tab extends to show the model's decision score (Fig. 3). Users can click the ☆Bookmarks button to open the *Bookmarks* window and click the ↻Regenerate button to generate five new recourse plans that reflect the currently specified recourse preferences.

Figure 3: A bar chart visualizes the model's decision score of a recourse plan: the bar is marked with the user's original score (shorter vertical line on the left) and the threshold needed to obtain the desired decision (longer vertical line on the right).
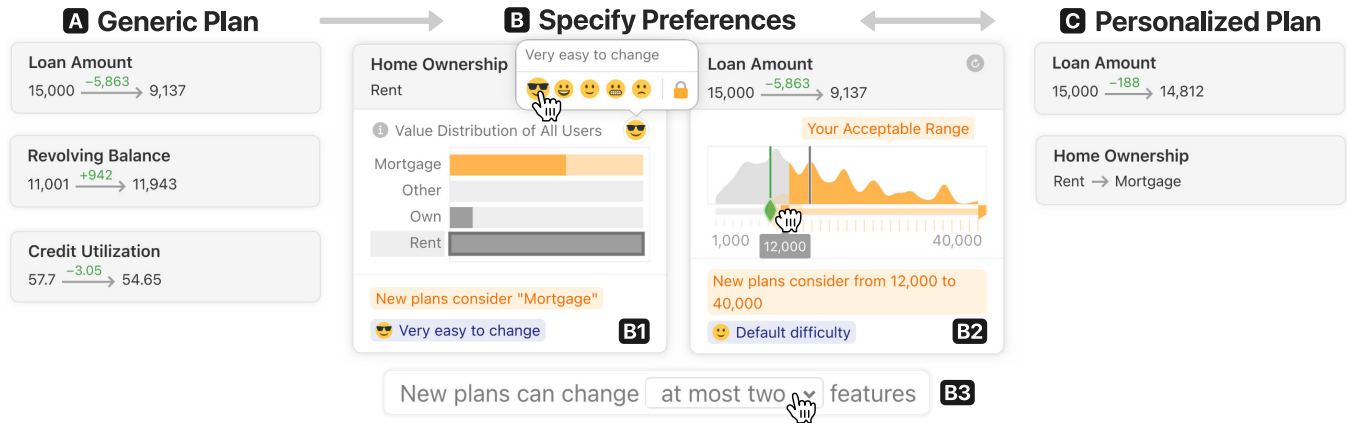
Figure 4: GAM Coach enables end users to inspect and customize recourse plans through simple interactions. (A) *Initial generic plans* **are generated with the same configurations for all users. (B) Users can** *specify recourse preferences* **if they are not satisfied with the initial plans; by configuring (B1) the** *difficulty* **to change a feature; (B2) the** *acceptable range* **that a feature can change between, and (B3) the** *max number of features* **that a recourse plan can alter. (C) GAM Coach then generates** *personalized plans* **that respect users' preferences. Users can iteratively refine their preferences until a satisfactory plan is found.**

5.2 Feature Panel

Each recourse plan has a unique *Feature Panel* (Fig. 1B) that visualizes plan details and allows users to provide preferences guiding the generation of new plans (G2). A *Feature Panel* consists of *Feature Cards* where each card represents a data feature used in the model. To help users easily navigate through different features, the panel groups *Feature Cards* into three sections: (1) features that are changed in the plan, (2) features that are configured by the user, (3) and all other features. To prevent overwhelming users with too much information (G4), all cards are collapsed by default—only displaying the feature name and feature values. Users can hover over the feature name to see a tooltip explaining the definition of the feature (G4). With a *progressive disclosure* design [61, 75], details of a feature, such as the distribution of feature values, are only shown on demand after users click that *Feature Card*. Progressive disclosure also makes GAM Coach interface scalable, as users can easily scroll and browse over hundreds of collapsed *Feature Cards*. Since EBMs process continuous and categorical features differently, we employ different card designs based on the feature type.

Continuous Feature Card. For continuous features, such as FICO score, the *Feature Card* (Fig. 5) uses a filled curved chart to visualize the distribution of feature values in the training set. Users can
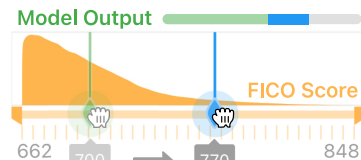

Figure 5: Users can test hypothetical input values in real time.

drag the diamond-shaped thumb ◆ on a slider below the chart to experiment with hypothetical values. During dragging, the decision score bar updates its width to reflect a new prediction score in real time. Therefore, users can better understand the underlying decision-making process by probing the model with different inputs (G3). Also, users can drag the orange thumbs 🔶🔶 to set the lower and upper bounds of acceptable feature changes. For example,

one user might only accept recourse plans that include loan amount at $12k or higher (Fig. 4-B2).

Categorical Feature Card. For categorical features, such as home ownership, users can inspect the value distribution with a horizontal bar chart (Fig. 4-B1), where a longer bar represents more frequent options in the training data. To specify acceptable ranges, users can click the bars to select or deselect acceptable options for new recourse plans. Acceptable options are highlighted as orange, whereas unacceptable options are colored as gray. Users can also click text labels next to the bars to experiment with hypothetical options and observe how they affect the model decision.

Specify Difficulty to Change a Feature. Besides selecting a feature's acceptable range, users can also specify how hard it would be for them to change a feature. For example, it might be easier for some users to lower credit utilization than to change home ownership. To configure feature difficulties, users can click the smiley button on any *Feature Card* and then select a suitable difficulty option on the pop-up window (Fig. 4-B1). Internally,

Figure 6: Distance multipliers of difficulties.

GAM Coach multiplies the distance costs of all bins in that feature with a constant multiplier (Fig. 6). If the user selects the "impossible to change" difficulty, the tool will remove all variables associated with this feature in the internal integer program (§ 4.3). Therefore, when generating new recourse strategies, GAM Coach would prioritize features that are easier to change and would not consider features that are impossible to change.

5.3 Bookmarks and Receipt

During the recourse iterations, users can save any suitable plans by clicking the star button ☆ on the plan tab (Fig. 3). Then, users
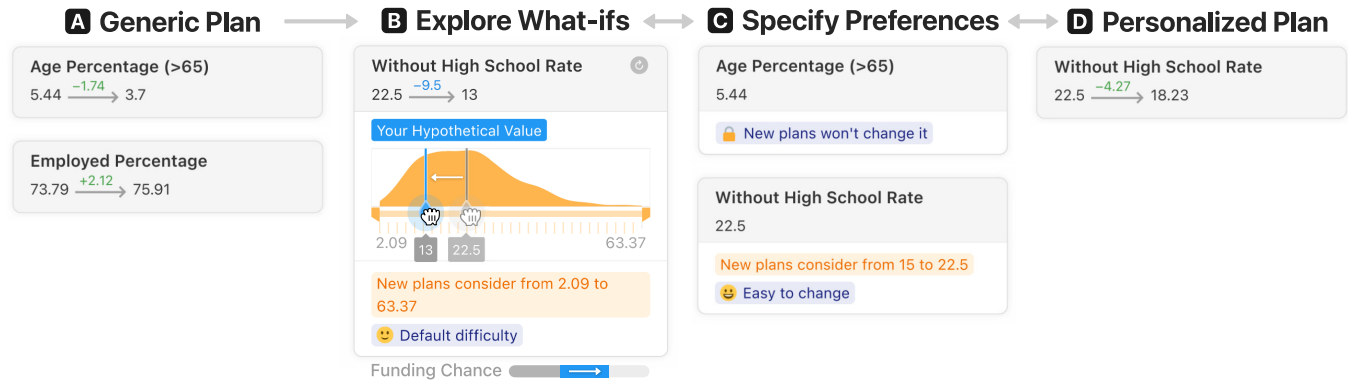
Figure 7: GAM Coach allows end users to experiment with hypothetical input values and customize recourse plans. (A) Our tool first shows *generic plans* generated with default configurations. (B) Users can explore how different input values affect the model's prediction in real time through simple interactions on the *Feature Card*: for example, lowering the percentage of adults without a high school diploma increases the chance of getting a government grant. (C) Users can then specify recourse preferences—such as feature *difficulties* and *acceptable ranges*—based on their circumstances and understanding of the model's prediction patterns. (D) GAM Coach then generates more actionable recourse plans based on the user-specified preferences.

can compare and update their saved plans in the *Bookmarks window* (Fig. 1C). Once users are satisfied with bookmarked plans, they can save a *recourse receipt* as proof of the generated recourse plans. Wachter et al. [88] first introduced the recourse receipt concept as a contract guaranteeing that a bank will approve a loan application if the applicant achieves all changes listed in the recourse plan. GAM Coach is the first tool to realize this concept by creating a plaintext file that records the timestamp, a hash of EBM model weights, the user's original input, and details of bookmarked plans (G4). In addition, we propose a novel security scheme that uses Pretty Good Privacy (PGP) to sign the receipt with the bank's private key [17]. With public-key cryptography, users can hold the bank accountable by being able to prove the receipt's authenticity to third-party authorities with the bank's public key. Also, banks can use their private key to verify a receipt's integrity during recourse redemption to avoid counterfeit receipts.

5.4 Usage Scenarios

We present two hypothetical usage scenarios to illustrate how GAM Coach can potentially help everyday users identify actionable strategies to alter undesirable ML-generated decisions.

Individual Loan Application. Eve is a rejected loan applicant, and she wants to identify ways to get a loan in the future. In this hypothetical usage scenario, to inform loan decisions, the bank has trained an EBM model on past data (we use LendingClub [1] to illustrate this scenario in Fig. 4). Their dataset has 9 continuous features and 11 categorical features (Fig. S2), and the outcome variable is binary—indicating whether a person can pay back the loan in time. The bank gives Eve a **link to GAM Coach** when informing her of the loan rejection decision. After Eve opens GAM Coach in a web browser, the tool pre-loads Eve's input data and generates five recourse plans based on the default configurations. Each plan lists a set of minimal changes in feature values that would lead to loan approval. One plan suggests Eve lower the requested loan amount from $15k to $9k along with two other changes (Fig. 4A). Eve

does not like this suggestion because she is unwilling to compromise a loss of $6k in the requested loan. Therefore, she clicks the loan amount *Feature Card* and drags the left thumb ▌ to set the *acceptable range* of loan amount to $12k and above (Fig. 4-B2). After browsing all recourse plans in the *Coach Menu*, Eve finds that none of the plans suggest changes to home ownership. Eve and her partner are actually moving to their newly-purchased condo next month. Therefore, Eve sets the *acceptable range* of home ownership to "mortgage" and changes its *difficulty* to "very easy" 😎 (Fig. 4-B1). Eve also prefers plans that change fewer features, so she clicks the dropdown menu on the *Feature Panel* to ask the tool to only generate plans that change at most two features (Fig. 4-B3). After Eve clicks the ↻Regenerate button, GAM Coach quickly generates five personalized plans that respect Eve's preferences. Among these plans, Eve especially likes the one suggesting she lower the loan amount by about $200 and change home ownership to mortgage (Fig. 4C). Finally, Eve bookmarks this plan and downloads a recourse receipt that guarantees her a loan if all suggested terms are met. Eve plans to apply for the loan again at the same bank next month.

Government Grant Application. Hal is a county manager in the United States. He has applied for a federal grant for his county. Unfortunately, his application is rejected. He wants to learn about the decision-making process and what actions he can take to succeed in future applications. In this hypothetical usage scenario, to inform funding decisions, the federal government has trained an EBM model on past data (we use the Communities and Crime dataset [67] to illustrate this scenario in Fig. 7). This dataset has 119 continuous features and 1 categorical feature describing the demographic and economic information of different counties in the United States, and is used to predict the risk of violent crime. As part of a performance incentive funding program [86], the federal government provides more funding opportunities to counties with lower predicted crime risk [79]. Before training the EBM model, the federal government has removed protected features (e.g., black population) and features with many (more than half) missing values, resulting in a total of 94 continuous features and 1 categorical feature.

The federal government provides rejected counties with a **link to GAM Coach** when informing them of the funding decisions. Hal opens GAM Coach in his browser; this tool has pre-loaded the demographic and economic features of his county and quickly suggested five recourse plans that would lead to funding. These generic plans are generated with the default configuration. One plan (Fig. 7A) suggests Hal decrease `age percentage (>65)` and increase `employed percentage` in his county. Hal likes the recommendation of increasing `employed percentage` because a higher employment rate is also beneficial for the economy of his county. However, Hal is puzzled by the suggestion of lowering `age percentage (>65)`. He is not sure why the population age is used to decide funding decisions. Besides, lowering the percentage of the elderly population is not actionable. Therefore, Hal "locks" this feature by setting its *difficulty* to "impossible" 🔒 (Fig. 7C).

To gain a better understanding of how the funding decision is made, Hal expands several *Feature Cards* and experiments with hypothetical feature values by dragging the blue thumbs 🔵; GAM Coach visualizes the model's prediction scores with these hypothetical inputs in real time (Fig. 7B). Hal quickly finds that lowering `without high school rate` can increase his chance of getting a grant. This is good news as Hal's county has just started a high school dropout prevention program aiming to lower the percentage of adults without a high school diploma to below 15% in eight years. Hal then sets this feature's *difficulty* to "easy to change" 😀 and drags the orange thumbs 🔶 to set its *acceptable range* to between 15% and 22.5% (Fig. 7C). After Hal clicks the `⟳ Regenerate` button, GAM Coach generates five new personalized plans in only 3 seconds despite there being almost 100 features. Among these five plans, Hal likes the one that recommends decreasing `without high school rate` by 4.27% (Fig. 7D). Finally, Hal saves a recourse receipt, and he will apply for this grant again once the percentage of adults without a high school diploma in his county drops by 4.27%.

5.5 Open-source & Generalizable Tool

GAM Coach is a web-based algorithmic recourse tool that users can access with any web browser on their laptops or tablets, no installation required (G5). We use *GLPK.js* [84] to solve integer programs with WebAssembly, *OpenPGP.js* [23] to sign recourse receipts with PGP, and *D3.js* [4] for visualizations. Therefore, the entire system runs locally in users' browsers without dedicated backend servers. We also provide an additional Python package[3] for developers to generate customizable recourse plans for EBM models without a graphical user interface. With this Python package, developers and researchers can also easily extract model weights from any EBM model to build their own GAM Coach. Finally, despite its name, GAM Coach's interface is model-agnostic—it supports any ML models where (1) one can control the difficulty and acceptable range of changing a feature during CF generation, and (2) model inference is available. With our open-source and generalizable implementation, detailed documentation, and examples on six datasets across a wide range of tasks and domains—LendingClub [1], Taiwan Credit [97], German Credit [14], Adult [45], COMPAS [47], and Communities and Crime [97]—future researchers can easily adapt our interface design to their models and datasets.

---

[3]Python package: **https://poloclub.github.io/gam-coach/docs/gamcoach**

6 USER STUDY

To evaluate GAM Coach and investigate how everyday users would use an interactive algorithmic recourse tool, we conducted an online user study with 41 United States-based crowdworkers. For possible datasets to use in this user study, we compared five public datasets that are commonly used in the recourse literature: LendingClub [e.g., 57, 82], Taiwan Credit [e.g., 73, 82, 83], German Credit [e.g., 57, 79, 82], Adult [e.g., 34, 56, 73], and COMPAS [e.g., 34, 57, 66]. We decided to use LendingClub in our study for the following three reasons. First, we chose a lending scenario as it is one scenario that many people, including crowdworkers, may encounter in real-life. Second, there is no expert knowledge needed to understand the setting, making our tasks appropriate for crowdworkers. Finally, our institute requires research participants to be United States-based: among the four datasets that can be used in a lending setting (LendingClub, Taiwan Credit, German Credit, and Adult), LendingClub is the only United States-based dataset collected from a real lending website. In this user study, we aimed to answer the following three research questions:

RQ1. What makes a satisfactory recourse plan for end users? (§ 6.3.1)

RQ2. How do end users discover their satisfactory recourse plans? (§ 6.3.2)

RQ3. How does interactivity play a role in providing algorithmic recourse? (§ 6.3.3)

6.1 Participants

We recruited 50 anonymous and voluntary United States-based participants from Amazon Mechanical Turk (MTurk), an online crowdsourcing platform. We did not collect any personal information. Collected interaction logs and subjective ratings are stored in a secure location where only the authors have access. The authors' Institutional Review Board (IRB) has approved the study. The average of three self-reported task completion times on a worker-centered forum[4] is 32½-minutes. We paid 41 participants $6.50 per study and 9 participants who had not passed our quality control $5.50.[5] Recruited participants self-report an average score of 2.7 for ML familiarity in a 5-point Likert-scale, where 1 represents "I have never heard of ML" and 5 represents "I have developed ML models."

6.2 Study Design

To start, each participant signed a consent form and filled out a background questionnaire (e.g., familarity with ML).

GAM Coach Tutorial and Short Quiz. We directed participants to a Google Survey form and **a website** containing GAM Coach, task instructions, and tutorial videos. Our tool, loaded with an EBM binary classifier that predicts loan approval on the LendingClub dataset [1], also contains input values of 500 random test samples on which the model predicts loan rejection. Participants

---

[4]TurkerView: **https://turkerview.com/**
[5]Originally the task was posted with a base payment of $3.50 and $1 bonus for quality. However, when analyzing participants' responses, we realized that the task required more time than we originally expected, so we provided an additional $2 bonus to all participants after the study to ensure appropriate compensation for their time. This brought the payment to $6.50 for those who passed the quality control quiz and $5.50 for those who did not.

Figure 8: We asked user study participants to explain why they had chosen their satisfactory plans, and why they had not chosen two other random plans (not shown in the figure).

were asked to watch a 3-minute tutorial video and complete eight multiple-choice quiz questions. These questions are simple—asking what is shown in the tool after certain interactions. All participants were asked to perform these interactions on the same data sample, so we had "ground truth" answers for the quiz questions. We used the quiz as a "gold standard" question to detect fraudulent responses [43, 63]. Although participants were prompted that they would need to answer all questions correctly to receive the base compensation, we paid all participants regardless of their answers. However, in our analysis, we only included responses from participants who had correctly answered at least four questions.

Free Exploration with an Imaginary Usage Scenario. After completing the tutorial and quiz, participants were asked to pretend to be a rejected loan applicant and freely use GAM Coach *until finding at least one satisfactory recourse plan.* These satisfactory recourse plans could be chosen from the first five generic plans that GAM Coach generates with a default configuration *or* follow-up plans that are generated based on participants' configured preferences. To help participants imagine the scenario, we asked them to change the input sample (one of 500 random samples) until they find one that they feel comfortable pretending to be. Participants could also manually adjust the input values (Fig. S2 in the appendix). After identifying and bookmarking their satisfactory plans, participants were asked to rate the importance of configured preferences or briefly explain why no configuration is needed. Then, participants were asked to explain why they had chosen their saved plans (Fig. 8) and why they had not chosen two other plans, which were randomly picked from the initial recourse plans. To incentivize participants to write good-quality explanations [26, 64], we told participants that they could get a $1 bonus reward if their explanations are well-justified. Regardless of their responses, all participants who had correctly answered at least four quiz questions were rewarded with this bonus.

Interaction Logging and Survey. While participants were using GAM Coach, the tool logged all interactions, such as preference configuration, hypothetical value experiment, and recourse plan generation. Each log event includes a timestamp and associated values. After finishing the exploration task, participants were asked to click a button that uploads their interaction logs and recourse plan reviews as a JSON file to a secured Dropbox directory. The filenames

included a random number. Participants were given this number as a verification code to report in the survey response and MTurk submission—we used this number to link a participant's MTurk ID with their log data and survey response. Finally, participants were asked to complete the survey consisting of subjective ratings and open-ended comments regarding the tool. As the EBM model used in the study is non-monotonic, the tool sometimes can suggest counterintuitive changes [3], such as to lower `annual income` for loan approval. We asked participants to report counterintuitive recourse plans in the survey if they had seen any.

6.3 Results

Out of 50 recruited participants, 41 (P1–P41) correctly answered at least four "quality-control" questions. In the following sections, we summarize our findings through analyzing these 41 participants' interaction logs, recourse plan reviews, and survey responses. We denote the Wald Chi-Square statistical test score as χ^2.

6.3.1 RQ1: Characteristics of Satisfactory Recourse Plans. During the exploration task, participants were asked to identify at least one recourse plan that they would be satisfied with if they were a rejected loan applicant using GAM Coach. On average, each participant chose 1.54 📊 satisfactory plans. Participants preferred *concise plans* that changed only a few features, with an average of 2.11 📊 features per plan. Chosen plans changed a diverse set of features, including 13 out of 20 features. The most popular features changed by chosen plans were `loan amount` (26.3%), `FICO score` (18.8%), and `credit utilization` (11.3%). Features that were not changed by any chosen plans were mostly hard to change in real life, such as `number of bankruptcies` and `employment length`.

Reasons for Choosing Satisfactory Plans. Three main reasons that participants reported choosing plans were that the plans were (1) controllable, (2) requiring small changes or less compromise, or (3) beneficial for life in general. Most participants chose recourse plans that felt realistic and controllable. For example, P30 wrote "*I think it's very possible to reduce my credit utilization in a short amount of time.*" In particular, participants preferred plans that only changed a few features and required a small amount of change. Participants described these plans as "*simple and fast*" (P5), "*straightforward*" (P7), and "*easy to do*" (P16). Some participants chose plans because they could tolerate the compromises. For example, P8 wrote "*I'm fine with the lower loan amount.*" Similarly, P11 reported "*[The decreased] loan amount is close to what I need.*" Interestingly, some participants favored plans that could benefit their lives in addition to helping them get loan approval. For example, P14 wrote "*[...] lower utilization is good for me anyway from what I know, so this seems like the best plan.*" Similarly, P28 wrote "*[this plan] in my opinion would guarantee greater monetary flexibility.*"

Reasons for Not Choosing a Plan. Participants' explanations for not choosing a plan mostly complemented the reasons for choosing a plan. Some participants also skipped plans because they were puzzled by counterintuitive suggestions, did not understand the suggestions, or just wanted to see more alternatives. First, participants disliked unrealistic suggestions: P2 explained "*It tells me to increase my income. My income is fixed. I cannot just increase them at a whim.*" Similarly, P6 wrote "*With inflation it might be harder to use less*

credit." Participants also disliked plans requiring too many changes or a large amount of change. For example, P30 wrote "*The amount of loan suggested to be reduced is too large. Assuming I'm applying for 9,800 for real, I wouldn't want to reduce the amount by more than 30%.*" Interestingly, some participants skipped a plan because it suggested counterintuitive changes. For example, P14 wrote "*It seemed like a bug because why would asking for an extra 13 dollars [in loan amount] result in a loan approval?*" Participants also skipped plans when they did not understand the suggestion: P9 wrote "*I'm not exactly sure what credit utilization is. I looked at the tooltip, but still wasn't sure.*" Finally, some participants skipped the initial plans because they just wanted to explore more alternatives: P22 explained "*I wanted to check out a few more things before I made my decision.*"

Design Lessons. By analyzing the characteristics of satisfactory recourse plans, our user study is the first study that provides empirical evidence to support several hypotheses from the recourse literature. We find that participants preferred plans that suggested changes on actionable features [35, 42], are concise and make small changes [48, 88], and could benefit participants beyond the recourse goal [3]. Additionally, participants were likely to save multiple satisfactory plans from one recourse session, highlighting the importance of providing diverse recourse plans [57]. Our study also shows that with transparency, end users can identify and dislike counterintuitive recourse plans (see more discussion in § 6.3.3). Therefore, future researchers and developers should help users identify concise and diverse plans that change actionable features and are beneficial overall. Also, researchers and developers should carefully audit and improve their models to prevent a CF generation algorithm from generating counterintuitive plans. Our findings also highlight that communicating recourse plans and providing a good user experience are as important as generating good recourse plans.

6.3.2 RQ2: Path to Discover Satisfactory Recourse Plans. In the exploration task, participants could freely choose their satisfactory recourse plans from the initial batch, where plans were generated with default configurations, or from follow-up batches, where plans reflected participants' specified preferences. We find that participants were more likely to choose satisfactory plans that respect participants' preference configurations (33 participants out of 41) than the default plans (8 participants). In addition, each recourse session had a median of 3 ▄▃▁▁▁ plan iterations. In other words, on average, a participant discovered satisfactory plans after seeing about 15 plans, where the last 10 plans were generated based on their preferences. The average time to identify satisfactory plans was 8 minutes and 38 seconds.

Preference configuration is helpful. In GAM Coach, users can specify the *difficulty* and *acceptable range* to change a feature and the *max number of features* a plan can change. We find all three preferences helped participants discover satisfactory plans. Among 63 total satisfactory plans chosen by 41 participants, 49 plans (77.78%) reflected at least one difficulty configuration and 44 plans (69.84%) reflected at least one range configuration. Also, 12 participants configured the max number of features—seven participants changed it to 1 and five changed it to 2 (default is 4).

Diverse Preference Configurations. By further analyzing participants' preferences associated with their chosen plans, we find
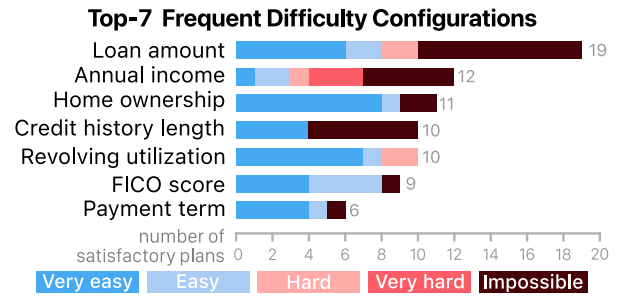

Figure 9: Difficulty configuration counts across frequent features highlighting variability of participants' preferences.

(1) participants specified preferences on a wide range of features; (2) some features were more popular than others; (3) different participants set different preferences on a given feature. Of the 20 features, at least one participant changed the difficulty of 16 features (80%) and acceptable range of 13 features (65%). Among these configured features, participants were more likely to specify preferences on some than others [$\chi^2 = 54.37$, $p < 0.001$ for the difficulty, $\chi^2 = 27.68$, $p = 0.006$ for the acceptable range]. For example, 19 satisfactory plans reflected difficulty for `loan amount`, whereas only 1 plan reflected the difficulty for `number of past dues`. Also, there was high variability in configured preferences on popular configured feature (Fig. 9). For instance, 6 plans considered `loan amount` as "very easy to change," while 9 plans deemed it as "impossible to change." Our findings confirm hypotheses that recourse preferences can be incorporated to identify satisfactory plans [3, 94], and these preferences are idiosyncratic [42, 87].

Design Lessons. When designing recourse systems, it is useful to allow end users to specify a wide range of recourse preferences, such as difficulties to change a feature, acceptable feature ranges, and max number of features to change. Additionally, there can be predictable patterns in users' recourse preferences—researchers can leverage these patterns to further improve user experiences. For example, developers can use the log data of an interactive recourse tool to train a new ML model to predict users' preference configurations. Then, for a new user, developers can predict their recourse preference and use it as the tool's default configuration.

6.3.3 RQ3: Interactive Algorithmic Recourse. How did participants use and perceive various *interactions* throughout the exploration task? Interestingly, 28% of participants who configured difficulty preferences had also immediately altered the difficulty levels on the same features; most of them have changed "easy" to "very easy" and "hard" to "very hard." For acceptable ranges, the percentage is higher at 88%. It suggests participants may need iterations to learn how preference configuration works in GAM Coach and then fine-tune configurations to generate better plans—highlighting the key role of iteration in interactive recourse. Survey response show that participants found both preference configuration and iteration helpful in finding good recourse plans (Fig. 10B). For example, P30 commented "*[I like] how easy it was to make changes to the priority of each thing. Showing that some things can be easy changes, or impossible to change, and making plans built around those.*" Similarly, P19 wrote "*[I like] regenerating unlimited plans until I find a fit one.*"
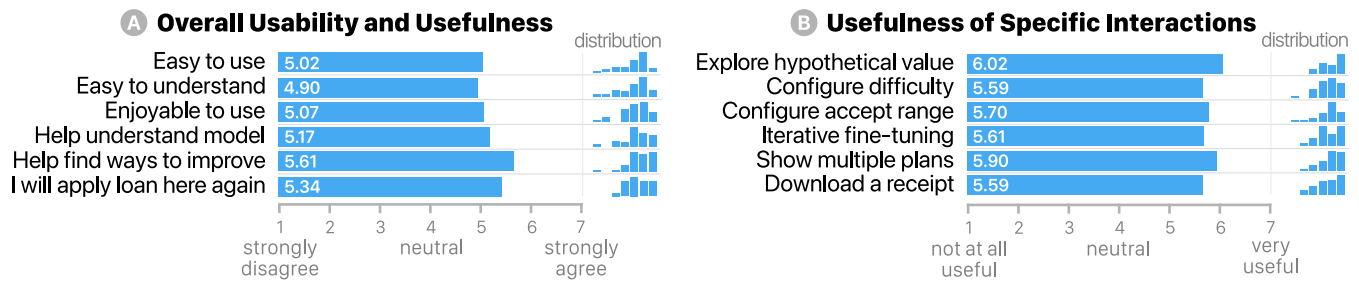
Figure 10: Average ratings and rating distributions from 41 participants on the usability and usefulness of GAM Coach. (A) Participants thought GAM Coach was relatively easy and enjoyable to use, and the tool helped them identify actions to obtain a preferred ML decision. (B) All interaction techniques, especially experimenting with hypothetical values, were rated favorably.

"What-if" Questions. Besides configuring preferences, participants also engaged in other modes of interaction with GAM Coach. For example, 32 out of 41 participants experimented with hypothetical feature values (§ 5.2), even though it did not affect recourse generations and was not required in the task. These participants explored median of 3 unique features ▆▁▁, and a median of 5.5 hypothetical feature values ▆▁▁▁▁▁. These 32 participants asked what-if questions on a total of 99 features, and only 39 (39.4%) of these features were from the presented recourse plan. It suggests that participants were more interested in learning about the predictive effects of features that have not been changed by GAM Coach. After exploring what-ifs on these 99 features, participants configured at least one preference (difficulty or acceptable range) on about half of them (49 features, 49.5%). In comparison, these participants only configured preferences on 13.72% features (87 out of 634) on which they had not explored what-ifs or had explored what-ifs *after* configuring preferences. It shows that participants were more likely to customize features on which they had explored hypothetical values [$\chi^2 = 85.459$, $p < 0.00001$]. Finally, 20 out of these 32 participants (62.5%) chose a satisfactory plan with a changed feature on which they had explored what-ifs. It may suggest participants preferred recourse plans that changed features on which they had explored what-ifs, but this result is not statistically significant [$\chi^2 = 2.0$, $p = 0.1573$].

By analyzing survey responses, we also find that asking what-if questions was one of the participants' favorite features (Fig. 10B). For example, P12 wrote "*[I like] how it adjusts the plans in real time and gives you an answer if the loan will be approved.*" Throughout the task, participants also frequently used the tooltip annotations to inspect the decision score bar (median 8 times per participant) and check the meaning of different features (median 25 times)—highlighting the importance of clearly explaining visual representations and terminologies in interactive recourse tools.

Counterintuitive recourse plans. We asked participants to report strange recourse plans that GAM Coach could rarely suggest, such as to lower annual income for loan approval. To our surprise, 7 out of 41 participants had encountered and reported these counterintuitive plans! For example, P6 was confused that some plans suggested conflicting changes on the same feature: "*One plan told me to increase the loan amount by $13 while another plan told me to decrease by $1,613.*" Another interesting case was P39: "*I don't understand how purpose changes approval decision. Something like*

'mortgage' I understand, but changing something and all of a sudden you can do a wedding but not home improvement? Like what?" First, P39 found it counterintuitive that GAM Coach includes the categorical feature loan purpose as a changeable feature because they thought the model decision should be independent of the loan purpose. Then, through experimenting with hypothetical values, P39 was baffled by the observation that two different purposes (wedding and home improvement) resulted in two distinct model decisions. Some other participants also attributed these strange patterns as reasons why they skipped some plans (§ 6.3.1). This finding provides empirical evidence that with transparency, everyday users can discover potentially problematic behaviors in ML models.

Design Lessons. Overall, interactivity helps users identify satisfactory recourse plans, and users appreciate being able to control recourse generation. In addition, users like being able to ask what-if questions; experimenting with hypothetical feature values also helps them find satisfactory recourse plans. However, it takes time and trial and error for users to understand how preference configurations affect recourse generation. Therefore, future interactive recourse tools can improve user experience by focusing on improving learnability and reversibility. Also, our study shows that interactivity and transparency could occasionally confuse users with counterintuitive recourse plans. Therefore, future researchers and developers should carefully audit and improve their ML models before deploying interactive recourse tools.

6.3.4 Usability. Our survey included a series of 7-point Likert-scale questions regarding the usability of GAM Coach (Fig. 10A). The results suggest that the tool is relatively easy to use (average 5.02), easy to understand (average 4.90), and enjoyable to use (average 5.07). However, some participants commented that the tool was not easy to learn at first and may be too complex for users with less knowledge about loans. For example, P5 wrote "*Without the tutorials, it would have taken me much longer to learn how to navigate the program, because it is not very intuitive at first.*" Similarly, P8 wrote "*I am decent with finances, but I'd imagine that other people would have more difficulty [using the tool].*" Our participants were MTurk workers, who are similar to the demographics of American internet users as a whole, but slightly younger and more educated [25, 63]. Therefore, GAM Coach might be overwhelming for real-life loan applicants who are less familiar with web technology or finance. Participants also provided specific feedback for improvement, such

as designing a better way to *store* and *compare* all generated plans. Currently, users would lose unsaved plans when generating new plans, and users could only compare different recourse plans in the *Bookmarks window* (§ 5.3). We plan to continue improving the design of GAM CᴏᴀᴄH based on participants' feedback.

7 LIMITATIONS

We acknowledge limitations regarding our tool's generalizability, usage scenarios, and user study design.

Generalizability of GAM CᴏᴀᴄH. To design and develop the first interactive algorithmic recourse tool that enables end users to fine-tune recourse plans with preferences, we ground our research in GAMs, a class of accurate and transparent ML models with simple structures. This approach enables us to generate customizable CF examples efficiently. However, not all CF generation algorithms allow users to specify the feature-level distance functions, acceptable ranges, and max number of features that a CF example can change. Therefore, while the GAM CᴏᴀᴄH interface is model-agnostic, it does not directly support all existing ML models and CF generation methods. Also, our novel CF generation algorithm is tailored to EBMs. However, one can easily adapt our linear constraints to generate customizable CF examples for linear models [83]. For more complex non-linear models (e.g., random forest, neural networks), one can apply our method to a linear approximation [56] of these models (§ 4.2). We also acknowledge that similar to most existing CF generation algorithms [3, 39], our algorithm assumes all features to be independent. However, in practice, many features can be associated. For example, changing credit utilization is likely to also affect a user's FICO score. Future work can generalize our algorithm to dependent features by modeling their casual relationships [36].

Hypothetical Usage Scenarios. We situate GAM CᴏᴀᴄH in lending and government funding settings (§ 5.4), two most cited scenarios in existing CF literature [3, 35]. It is important to note that none of the authors have expertise in law, finance, or political science. Therefore, to adapt GAM CᴏᴀᴄH for use in real lending and government funding settings, it would require more research and engaging with experts in the legal and financial domains as well as people who would be impacted by the systems. In addition, we use LendingClub [1] and Communities and Crime [67], two largest suitable datasets we have access to (§ 6), to simulate two usage scenarios and design our user study. These two datasets can have different features and sizes from the data that are used in practice. Therefore, before adapting GAM CᴏᴀᴄH, researchers and developers should thoroughly test our tool on their own datasets.

Simulated Study Design. To study how end users would use interactive recourse tools, we recruited MTurk workers and asked them to pretend to be rejected loan applicants, and we logged and analyzed their interactions with GAM CᴏᴀᴄH. We designed the task to encourage and help participants simulate the scenario (e.g., rewarding bonus, supporting participants to input data or choose data from multiple random samples). However, participants' usage patterns and reactions may not fully represent real-life loan applicants. We chose to simulate a lending scenario because (1) crowdworkers may have encountered lending, (2) it does not require expert knowledge, and (3) we have access to a large and real US-based lending dataset. We acknowledge that participants' usage patterns

may not full represent users in other domains. Therefore, it would require further research with actual end users (e.g., loan applicants, county executives, and bail applicants) to study how GAM CᴏᴀᴄH can aid them in real-world settings. In our study, we only collected participants' familiarity with ML. As MTurk workers tend to be younger and more educated than average internet users [25, 63], future researchers can collect more self-reported demographic information (e.g., age, education, sex) to study if different user groups would use an interactive recourse tool differently.

Observational Study Design. Our observational log study can provide a portrait of users' natural behaviors when interacting with interactive algorithmic recourse tools and scale to a large number of participants [15]. However, it lacks a control group. As algorithmic recourse research and applications are still nascent, the community has not yet established a recommended workflow or system that we can use as a baseline in our study (§ 2.1). Our main goal is to study how *recourse customizability* can help users discover useful recourse plans. Therefore, to mitigate the lack of a control group, we offer participants the option to *abstain from customizing recourse plans* to probe into the usefulness of recourse customizability. In our analysis, we compare both (1) the numbers of participants who specify recourse preferences and who do not, (2) and the numbers of satisfactory plans generated with a default configuration and satisfactory plans generated with a participant-configured preference (§ 6.3.2). Finally, with our open-source implementation (§ 5.5), future researchers can use GAM CᴏᴀᴄH as a baseline system to evaluate their interactive recourse tools.

8 DISCUSSION AND FUTURE WORK

Reflecting on our end-to-end realization of interactive algorithmic recourse—from UI design to algorithm development and a user study—we distill lessons and provide a set of future directions for algorithmic recourse and ML interpretability.

Too much transparency. GAM CᴏᴀᴄH uses a glass-box model, provides end users with complete control of recourse plan generation, and supports users to ask "what-if" questions with any feature values. One might argue that GAM CᴏᴀᴄH is too transparent and too much transparency makes the tool unfavorable, because (1) end users can use this tool for gaming the ML model [22, 44] and (2) this tool fails to protect the decision maker's model intellectual property [88]. We acknowledge these concerns. As recourse research and applications are still nascent, it is challenging to know how we can balance the benefits of transparency and human agency and the risk of revealing too much information about the ML model. Our user study shows that with transparency end users can discover and are often puzzled by counterintuitive patterns in ML models. We believe if GAM CᴏᴀᴄH is adopted, it has the potential to incentivize decision makers to create better models in order to avoid confusion as well as model exploitations. As one of the furthest realizations of ML transparency, GAM CᴏᴀᴄH can be a research instrument that facilitates future researchers to study the tension between *decision makers* and *decision subjects*, and identify the right amount of transparency that most benefits both parties. Then, to adopt GAM CᴏᴀᴄH in practice, ML developers can remove certain functionalities or impose recourse constraints accordingly. For example, if a bank is offering GAM CᴏᴀᴄH and is worried about

people gaming the system by changing certain features that do not actually improve their creditworthiness (e.g., opening more credit cards), they could insert their own optimization constraints that prevent these features from being modified.

Transparent ML models for algorithmic recourse. Black-box ML models are popular across different domains. To interpret these models, researchers have developed post-hoc techniques to identify feature importance [e.g. 53, 68] and generate CF examples [e.g. 48, 57]. However, Rudin [69] argues that researchers and practitioners should use transparent ML models instead of black-box models in high-stake domains due to transparent models' high accuracy and explanation fidelity. The design of GAM Coach is based on GAMs, a state-of-the-art transparent model [6, 89]. We would like to broaden the perspective of using transparent models reflecting on our study. We find that GAM Coach provides opportunities for everyday users to discover counterintuitive patterns in the ML model. It implies that ML developers and researchers can also use GAM Coach as a penetration testing tool to detect potentially problematic behaviors in their models. Note that both black-box and transparent learning methods would have learned these counterintuitive behaviors [6], but with a transparent model, developers can further *vet* and *fix* these behaviors. As an example, an ML developer training a GAM can use GAM Coach to iteratively generate recourse plans for potential users (e.g., training data where the model gives unfavorable predictions). If they identify strange suggestions, they can use existing interactive tools [60, 90] to visualize corresponding shape functions to pinpoint the root cause of these counterintuitive patterns, and then edit shape function parameters to avoid them from happening during recourse deployment. Future research can leverage transparent models to distill guidelines to audit and fix models before recourse deployment.

Put users at the center. During the design and implementation of GAM Coach, we have encountered many challenges in transforming technically sound recourse plans into a seamless user experience. As the end users of recourse tools are everyday people who are less familiar with ML and domain-specific concepts, one of our design goals is to help them understand necessary concepts and have a frictionless experience (G4). GAM Coach aims to achieve this goal by following a progressive disclosure and details-on-demand design strategy [61, 75] and presenting textual annotations to explain visual representations in the tool. However, our user study suggests that few users might still find it challenging to use GAM Coach at first (§ 6.3.4). During our development process, we identify many edge cases that a recourse application would encounter in practice, such as features requiring integer values (e.g., FICO score), features using log transformations (e.g., annual income), or features less familiar to everyday users (e.g., credit utilization). Our open-source implementation handles these edge cases, and we provide ML developers with simple APIs to add descriptions for domain-specific feature names in their own instances of GAM Coach. However, these practical edge cases are rarely discussed or handled in the recourse research community, since (1) the field of algorithmic recourse is relatively nascent, (2) and the main evaluation criteria of recourse research are distance-based statistics instead of *user experience* [39]. Therefore, in addition to developing faster techniques to generate more actionable recourse plans, we hope future

researchers engage with end users and incorporate user experience into their research agenda. Besides interactive visualization, researchers can also explore alternative mediums to communicate and personalize ML recourse plans and model explanations, such as through a textual [16] or multi-modal approach [29].

9 CONCLUSION

As ML models are increasingly used to inform high-stakes decision-making throughout our everyday life, it is crucial to provide decision subjects ways to alter unfavorable model decisions. In this work, we present GAM Coach, an interactive algorithmic recourse tool that empowers end users to specify their preferences and iteratively fine-tune recourse plans. Our tool runs in web browsers and is open-source, broadening people's access to responsible ML technologies. We discuss lessons learned from our realization of interactive algorithmic recourse and an online user study. We hope our work will inspire future research and development of user-centered and interactive tools that help end users restore their human agency and eventually trust and enjoy ML technologies.

REFERENCES

[1] 2018. Lending Club: Online Personal Loans at Great Rates. https://www.lendingclub.com/

[2] Ashraf Abdul, Jo Vermeulen, Danding Wang, Brian Y. Lim, and Mohan Kankanhalli. 2018. Trends and Trajectories for Explainable, Accountable and Intelligible Systems: An HCI Research Agenda. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. https://doi.org/10.1145/3173574.3174156

[3] Solon Barocas, Andrew D. Selbst, and Manish Raghavan. 2020. The Hidden Assumptions behind Counterfactual Explanations and Principal Reasons. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. https://doi.org/10.1145/3351095.3372830

[4] M. Bostock, V. Ogievetsky, and J. Heer. 2011. D³ Data-Driven Documents. *IEEE TVCG* 17 (2011).

[5] Michelle Carney, Barron Webster, Irene Alvarado, Kyle Phillips, Noura Howell, Jordan Griffith, Jonas Jongejan, Amit Pitaru, and Alexander Chen. 2020. Teachable Machine: Approachable Web-Based Tool for Exploring Machine Learning Classification. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3334480.3382839

[6] 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. *KDD* (2015). https://doi.org/10.1145/2783258.2788613

[7] Chun-Hao Chang, Sarah Tan, Ben Lengerich, Anna Goldenberg, and Rich Caruana. 2021. How Interpretable and Trustworthy Are GAMs? *KDD* (2021). https://doi.org/10.1145/3447548.3467453

[8] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939785

[9] Furui Cheng, Yao Ming, and Huamin Qu. 2021. DECE: Decision Explorer with Counterfactual Explanations for Machine Learning Models. *IEEE TVCG* 27 (2021). https://doi.org/10.1109/TVCG.2020.3030342

[10] Hao-Fei Cheng, Ruotong Wang, Zheng Zhang, Fiona O'Connell, Terrance Gray, F. Maxwell Harper, and Haiyi Zhu. 2019. Explaining Decision-Making

Algorithms through UI: Strategies to Help Non-Expert Stakeholders. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3290605.3300789

[11] Zhicheng Cui, Wenlin Chen, Yujie He, and Yixin Chen. 2015. Optimal Action Extraction for Random Forests and Boosted Trees. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2783258.2783281

[12] Eoin Delaney, Derek Greene, and Mark T. Keane. 2021. Instance-Based Counterfactual Explanations for Time Series Classification. *arXiv:2009.13211 [cs, stat]* (2021). http://arxiv.org/abs/2009.13211

[13] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. 2018. Explanations Based on the Missing: Towards Contrastive Explanations with Pertinent Negatives. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*.

[14] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[15] Susan Dumais, Robin Jeffries, Daniel M. Russell, Diane Tang, and Jaime Teevan. 2014. Understanding User Behavior Through Log Data and Analysis. In *Ways of Knowing in HCI*. https://doi.org/10.1007/978-1-4939-0378-8_14

[16] Upol Ehsan, Brent Harrison, Larry Chan, and Mark O. Riedl. 2018. Rationalization: A Neural Machine Translation Approach to Generating Natural Language Explanations. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. https://doi.org/10.1145/3278721.3278736

[17] Simson Garfinkel. 1995. *PGP: Pretty Good Privacy*.

[18] Fred Glover. 1975. Improved Linear Integer Programming Formulations of Nonlinear Integer Problems. *Management Science* 22 (1975). https://doi.org/10.1287/mnsc.22.4.455

[19] Oscar Gomez, Steffen Holter, Jun Yuan, and Enrico Bertini. 2020. ViCE: Visual Counterfactual Explanations for Machine Learning Models. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*. https://doi.org/10.1145/3377325.3377536

[20] Oscar Gomez, Steffen Holter, Jun Yuan, and Enrico Bertini. 2021. AdViCE: Aggregated Visual Counterfactual Explanations for Machine Learning Model Validation. In *2021 IEEE Visualization Conference (VIS)*. https://doi.org/10.1109/VIS49827.2021.9623271

[21] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Counterfactual Visual Explanations. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*. https://proceedings.mlr.press/v97/goyal19a.html

[22] Moritz Hardt, Nimrod Megiddo, Christos Papadimitriou, and Mary Wootters. 2016. Strategic Classification. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*. https://doi.org/10.1145/2840728.2840730

[23] Tankred Hase. 2014. OpenPGP.Js: OpenPGP JavaScript Implementation. https://openpgpjs.org/

[24] Trevor Hastie and Robert Tibshirani. 1999. *Generalized Additive Models*.

[25] Paul Hitlin. 2016. Research in the Crowdsourcing Age: A Case Study. (2016).

[26] Chien-Ju Ho, Aleksandrs Slivkins, Siddharth Suri, and Jennifer Wortman Vaughan. 2015. Incentivizing High Quality Crowdwork. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15)*. https://doi.org/10.1145/2736277.2741102

[27] Fred Hohman, Andrew Head, Rich Caruana, Robert DeLine, and Steven M. Drucker. 2019. Gamut: A Design Probe to Understand How Data Scientists Understand Machine Learning Models. *CHI* (2019). https://doi.org/10.1145/3290605.3300809

[28] Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng Chau. 2019. SUMMIT: Scaling Deep Learning Interpretability by Visualizing Activation and Attribution Summarizations. *IEEE TVCG* (2019). https://doi.org/10.1109/TVCG.2019.2934659

[29] Fred Hohman, Arjun Srinivasan, and Steven M. Drucker. 2019. TeleGam: Combining Visualization and Verbalization for Interpretable Machine Learning. In *2019 IEEE Visualization Conference (VIS)*. https://doi.org/10.1109/VISUAL.2019.8933695

[30] Shalmali Joshi, Oluwasanmi Koyejo, Warut Vijitbenjaronk, Been Kim, and Joydeep Ghosh. 2019. Towards Realistic Individual Recourse and Actionable Explanations in Black-Box Decision Making Systems. *arXiv:1907.09615 [cs, stat]* (2019). http://arxiv.org/abs/1907.09615

[31] Minsuk Kahng, Pierre Y. Andrews, Aditya Kalro, and Duen Horng Chau. 2018. ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models. *IEEE TVCG* 24 (2018). https://doi.org/10.1109/TVCG.2017.2744718

[32] Minsuk Kahng, Nikhil Thorat, Duen Horng Chau, Fernanda B. Viegas, and Martin Wattenberg. 2019. GAN Lab: Understanding Complex Deep Generative Models Using Interactive Visual Experimentation. *IEEE Transactions on Visualization and Computer Graphics* 25 (2019).

[33] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Hiroki Arimura. 2020. DACE: Distribution-Aware Counterfactual Explanation by Mixed-Integer Linear Optimization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2020/395

[34] Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. 2020. Model-Agnostic Counterfactual Explanations for Consequential Decisions. *arXiv:1905.11190 [cs, stat]* (2020). http://arxiv.org/abs/1905.11190

[35] Amir-Hossein Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. 2021. A Survey of Algorithmic Recourse: Definitions, Formulations, Solutions, and Prospects. *arXiv:2010.04050 [cs, stat]* (2021). http://arxiv.org/abs/2010.04050

[36] Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. 2021. Algorithmic Recourse: From Counterfactual Explanations to Interventions. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. https://doi.org/10.1145/3442188.3445899

[37] Amir-Hossein Karimi, Julius von Kügelgen, Bernhard Schölkopf, and Isabel Valera. 2020. Algorithmic Recourse under Imperfect Causal Knowledge: A Probabilistic Approach. In *Advances in Neural Information Processing Systems*, Vol. 33. https://proceedings.neurips.cc/paper/2020/file/02a3c7fb3f489288ae6942498498db20-Paper.pdf

[38] Guolin Ke, Qi Meng, Thomas Finely, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30 (NIP 2017)*. https://www.microsoft.com/en-us/research/publication/lightgbm-a-highly-efficient-gradient-boosting-decision-tree/

[39] Mark T. Keane, Eoin M. Kenny, Eoin Delaney, and Barry Smyth. 2021. If Only We Had Better Counterfactual Explanations: Five Key Deficits to Rectify in the Evaluation of Counterfactual XAI Techniques. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2021/609

[40] Mark T Keane and Barry Smyth. 2020. Good Counterfactuals and Where to Find Them: A Case-Based Technique for Generating Counterfactuals for Explainable AI (XAI). In *International Conference on Case-Based Reasoning*.

[41] Eoin M. Kenny and Mark T Keane. 2021. On Generating Plausible Counterfactual and Semi-Factual Explanations for Deep Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 35 (2021). https://ojs.aaai.org/index.php/AAAI/article/view/17377

[42] Lara Kirfel and Alice Liefgreen. 2021. What If (and How...)? - Actionability Shapes People's Perceptions of Counterfactual Explanations in Automated Decision-Making. In *ICML Workshop on Algorithmic Recourse*.

[43] Aniket Kittur, Ed H. Chi, and Bongwon Suh. 2008. Crowdsourcing User Studies with Mechanical Turk. In *Proceeding of the Twenty-Sixth Annual CHI Conference on Human Factors in Computing Systems - CHI '08*. https://doi.org/10.1145/1357054.1357127

[44] Jon Kleinberg and Manish Raghavan. 2020. How Do Classifiers Induce Agents to Invest Effort Strategically? *ACM Transactions on Economics and Computation* 8 (2020). https://doi.org/10.1145/3417742

[45] Ron Kohavi et al. 1996. Scaling up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid.. In *KDD*, Vol. 96.

[46] Josua Krause, Adam Perer, and Kenney Ng. 2016. Interacting with Predictions: Visual Inspection of Black-box Machine Learning Models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/2858036.2858529

[47] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. 2016. How We Analyzed the COMPAS Recidivism Algorithm. *ProPublica* 9 (2016).

[48] Thai Le, Suhang Wang, and Dongwon Lee. 2020. GRACE: Generating Concise and Informative Contrastive Sample to Explain Neural Network Model's Prediction. *arXiv:1911.02042 [cs, stat]* (2020). http://arxiv.org/abs/1911.02042

[49] Cynthia C. S. Liem, Markus Langer, Andrew Demetriou, Annemarie M. F. Hiemstra, Achmadnoer Sukma Wicaksana, Marise Ph. Born, and Cornelius J. König. 2018. Psychology Meets Machine Learning: Interdisciplinary Perspectives on Algorithmic Job Candidate Screening. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*. https://doi.org/10.1007/978-3-319-98131-4_9

[50] Tania Lombrozo. 2016. Explanatory Preferences Shape Learning and Inference. *Trends in Cognitive Sciences* 20 (2016). https://doi.org/10.1016/j.tics.2016.08.001

[51] Yin Lou, Rich Caruana, and Johannes Gehrke. 2012. Intelligible Models for Classification and Regression. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '12*. https://doi.org/10.1145/2339530.2339556

[52] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. 2013. Accurate Intelligible Models with Pairwise Interactions. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2487575.2487579

[53] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. https://doi.org/10.48550/arXiv.1705.07874

[54] Divyat Mahajan, Chenhao Tan, and Amit Sharma. 2020. Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers. *arXiv:1912.03277 [cs, stat]* (2020). http://arxiv.org/abs/1912.03277

[55] Brent Mittelstadt, Chris Russell, and Sandra Wachter. 2019. Explaining Explanations in AI. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. https://doi.org/10.1145/3287560.3287574

[56] Kiarash Mohammadi, Amir-Hossein Karimi, Gilles Barthe, and Isabel Valera. 2021. Scaling Guarantees for Nearest Counterfactual Explanations. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. https://doi.org/10.1145/3461702.3462514

[57] Ramaravind K. Mothilal, Amit Sharma, and Chenhao Tan. 2020. Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. https://doi.org/10.1145/3351095.3372850

[58] Chelsea M. Myers, Evan Freed, Luis Fernando Laris Pardo, Anushay Furqan, Sebastian Risi, and Jichen Zhu. 2020. Revealing Neural Network Bias to Non-Experts Through Interactive Counterfactual Examples. *arXiv:2001.02271 [cs]* (2020). http://arxiv.org/abs/2001.02271

[59] J. A. Nelder and R. W. M. Wedderburn. 1972. Generalized Linear Models. *Journal of the Royal Statistical Society. Series A (General)* 135 (1972). https://doi.org/10.2307/2344614

[60] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. 2019. InterpretML: A Unified Framework for Machine Learning Interpretability. *arXiv* (2019). http://arxiv.org/abs/1909.09223

[61] Donald A. Norman and Stephen W. Draper. 1986. *User Centered System Design: New Perspectives on Human-Computer Interaction*.

[62] Seyednaser Nourashrafeddin, Ehsan Sherkat, Rosane Minghim, and Evangelos E. Milios. 2018. A Visual Approach for Interactive Keyterm-Based Clustering. *ACM Transactions on Interactive Intelligent Systems* 8 (2018). https://doi.org/10.1145/3181669

[63] Judith S. Olson and Wendy Kellogg. 2014. *Ways of Knowing in HCI*.

[64] Gabriele Paolacci and Jesse Chandler. 2014. Inside the Turk: Understanding Mechanical Turk as a Participant Pool. *Current Directions in Psychological Science* 23 (2014). https://doi.org/10.1177/0963721414531598

[65] Nicola Pezzotti, Thomas Hollt, Jan Van Gemert, Boudewijn P.F. Lelieveldt, Elmar Eisemann, and Anna Vilanova. 2018. DeepEyes: Progressive Visual Analytics for Designing Deep Neural Networks. *IEEE Transactions on Visualization and Computer Graphics* 24 (2018). https://doi.org/10.1109/TVCG.2017.2744358

[66] Kaivalya Rawal and Himabindu Lakkaraju. 2020. Beyond Individualized Recourse: Interpretable and Interactive Summaries of Actionable Recourses. *arXiv:2009.07165 [cs, stat]* (2020). http://arxiv.org/abs/2009.07165

[67] Michael Redmond and Alok Baveja. 2002. A Data-Driven Software Tool for Enabling Cooperative Information Sharing among Police Departments. *European Journal of Operational Research* 141 (2002). https://doi.org/10.1016/S0377-2217(01)00264-8

[68] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939778

[69] Cynthia Rudin. 2019. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence* 1 (2019). https://doi.org/10.1038/s42256-019-0048-x

[70] Chris Russell. 2019. Efficient Search for Diverse Coherent Explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. https://doi.org/10.1145/3287560.3287569

[71] Harvey M. Salkin and Cornelis A. De Kluyver. 1975. The Knapsack Problem: A Survey. *Naval Research Logistics Quarterly* 22 (1975). https://doi.org/10.1002/nav.3800220110

[72] Matthew J. Saltzman. 2002. Coin-Or: An Open-Source Library for Optimization. In *Programming Languages and Systems in Computational Economics and Finance*. Vol. 18. https://doi.org/10.1007/978-1-4615-1049-9_1

[73] Maximilian Schleich, Zixuan Geng, Yihong Zhang, and Dan Suciu. 2021. GeCo: Quality Counterfactual Explanations in Real Time. *arXiv:2101.01292 [cs]* (2021). http://arxiv.org/abs/2101.01292

[74] Andrew D. Selbst and Solon Barocas. 2018. The Intuitive Appeal of Explainable Machines. *SSRN Electronic Journal* (2018). https://doi.org/10.2139/ssrn.3126971

[75] B. Shneiderman. 1996. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*. https://doi.org/10.1109/VL.1996.545307

[76] Ben Shneiderman. 2020. Bridging the Gap Between Ethics and Practice: Guidelines for Reliable, Safe, and Trustworthy Human-centered AI Systems. *ACM Transactions on Interactive Intelligent Systems* 10 (2020). https://doi.org/10.1145/3419764

[77] Naeem Siddiqi. 2013. *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring*. http://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=4035275

[78] Sumedha Singla, Brian Pollack, Junxiang Chen, and Kayhan Batmanghelich. 2020. Explanation by Progressive Exaggeration. In *ICLR*.

[79] Dylan Slack, Anna Hilgard, Himabindu Lakkaraju, and Sameer Singh. 2021. Counterfactual Explanations Can Be Manipulated. In *Advances in Neural Information Processing Systems*, Vol. 34. https://proceedings.neurips.cc/paper/2021/

[80] Daniel Smilkov, Shan Carter, D. Sculley, Fernanda B. Viégas, and Martin Wattenberg. 2017. Direct-Manipulation Visualization of Deep Networks. *arXiv:1708.03788* (2017).

[81] Harini Suresh, Steven R. Gomez, Kevin K. Nam, and Arvind Satyanarayan. 2021. Beyond Expertise and Roles: A Framework to Characterize the Stakeholders of Interpretable Machine Learning and Their Needs. *arXiv:2101.09824 [cs]* (2021). https://doi.org/10.1145/3411764.3445088

[82] Stratis Tsirtsis and Manuel Gomez Rodriguez. 2020. Decisions, Counterfactual Explanations and Strategic Behavior. In *Advances in Neural Information Processing Systems*, Vol. 33. https://proceedings.neurips.cc/paper/2020/file/c2ba1bc54b239208cb37b901c0d3b363-Paper.pdf

[83] Berk Ustun, Alexander Spangher, and Yang Liu. 2019. Actionable Recourse in Linear Classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. https://doi.org/10.1145/3287560.3287566

[84] Jan Vaillant. 2021. Glpk.Js. https://github.com/jvail/glpk.js/

[85] Arnaud Van Looveren and Janis Klaise. 2020. Interpretable Counterfactual Explanations Guided by Prototypes. *arXiv:1907.02584 [cs, stat]* (2020). http://arxiv.org/abs/1907.02584

[86] Vera Institute of Justice. 2012. Performance Incentive Funding: Aligning Fiscal and Operational Responsibility to Produce More Safety at Less Cost. Vera Institute of Justice Report.

[87] Sahil Verma, John Dickerson, and Keegan Hines. 2020. Counterfactual Explanations for Machine Learning: A Review. *arXiv:2010.10596 [cs, stat]* (2020). http://arxiv.org/abs/2010.10596

[88] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. *SSRN Electronic Journal* (2017). https://doi.org/10.2139/ssrn.3063289

[89] Caroline Wang, Bin Han, Bhrij Patel, Feroze Mohideen, and Cynthia Rudin. 2020. In Pursuit of Interpretable, Fair and Accurate Machine Learning for Criminal Recidivism Prediction. *arXiv:2005.04176* (2020). http://arxiv.org/abs/2005.04176

[90] Zijie J. Wang, Alex Kale, Harsha Nori, Peter Stella, Mark E. Nunnally, Duen Horng Chau, Mihaela Vorvoreanu, Jennifer Wortman Vaughan, and Rich Caruana. 2022. Interpretability, Then What? Editing Machine Learning Models to Reflect Human Knowledge and Values. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22)*. https://doi.org/10.1145/3534678.3539074

[91] Zijie J. Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng Chau. 2020. CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* (2020).

[92] Zijie J. Wang, Chudi Zhong, Rui Xin, Takuya Takagi, Zhi Chen, Duen Horng Chau, Cynthia Rudin, and Margo Seltzer. 2022. TimberTrek: Exploring and Curating Sparse Decision Trees with Interactive Visualization. In *2022 IEEE Visualization and Visual Analytics (VIS)*. https://doi.org/10.1109/VIS54862.2022.00021

[93] Austin Waters and Risto Miikkulainen. 2014. GRADE: Machine Learning Support for Graduate Admissions. *AI Magazine* 35 (2014). https://doi.org/10.1609/aimag.v35i1.2504

[94] Daniel S. Weld and Gagan Bansal. 2019. The Challenge of Crafting Intelligible Intelligence. *Commun. ACM* 62 (2019). https://doi.org/10.1145/3282486

[95] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viegas, and Jimbo Wilson. 2019. The What-If Tool: Interactive Probing of Machine Learning Models. *IEEE TVCG* 26 (2019). https://doi.org/10.1109/TVCG.2019.2934619

[96] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2021. Polyjuice: Generating Counterfactuals for Explaining, Evaluating, and Improving Models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. https://doi.org/10.18653/v1/2021.acl-long.523

[97] I-Cheng Yeh and Che-hui Lien. 2009. The Comparisons of Data Mining Techniques for the Predictive Accuracy of Probability of Default of Credit Card Clients. *Expert Systems with Applications* 36 (2009). https://doi.org/10.1016/j.eswa.2007.12.020

[98] Jun Yuan and Enrico Bertini. 2022. Context Sight: Model Understanding and Debugging via Interpretable Context. In *Proceedings of the Workshop on Human-in-the-Loop Data Analytics (HILDA '22)*. Article 1. https://doi.org/10.1145/3546930.3547502

[99] Zahra Zahedi, Alberto Olmo, Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. 2019. Towards Understanding User Preferences for Explanation Types in Model Reconciliation. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. https://doi.org/10.1109/HRI.2019.8673097

[100] Xuezhou Zhang, Sarah Tan, Paul Koch, Yin Lou, Urszula Chajewska, and Rich Caruana. 2019. Axiomatic Interpretability for Multiclass Additive Models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. https://doi.org/10.1145/3292500.3330898

A RECOURSE GENERATION DETAILS

A.1 EBM CF Generation Problem Definition

Given a trained EBM model M and an instance $x \in \mathbb{R}^k$, our goal is to generate a set of CF examples $\{c^{(1)}, c^{(2)}, \ldots, c^{(l)}\}$, where M gives a different decision than the original input x. In other words, we would like to find c such that $M(c) \neq M(x)$. Without loss of generality, we use binary classification as an example in this section. For binary classifications, EBM use sigmoid function $\sigma(a) = \frac{1}{1+e^{-a}}$ as a link function. This link function rescales the sum of shape function values $S_x = \beta_0 + f_1(x_1) + f_2(x_2) + \cdots + f_k(x_k) + \cdots + f_{i,j}(x_i, x_j)$ to a probability $\sigma(S_x)$, ranging from 0 to 1. If $\sigma(S_x) \geq 0.5$ or $S_x \geq 0$, M predicts the input x as positive; otherwise M predicts x as negative. To generate a CF example c that leads to a different decision than the original input x, we need to make some changes to x so that the new score S_c has a different sign from S_x.

A.2 Counterfactual Constraint

A CF example c is valid if it changes the sign of the original score S_x. If the model predicts the original input x as positive ($s_x \geq 0$), then the score gain $g(x, c) = S_c - S_x$ should be smaller than $-S_x$. Similarly, if the model predicts x as negative ($s_x < 0$), then the score gain $g(x, c)$ should be at least $-S_x$. Since EBM is additive during inference, we can write $g(x, c)$ as:

$$
\begin{aligned}
g(x, c) &= S_c - S_x \\
&= (\beta_0 + f_1(c_1) + \cdots + f_k(c_k) + \cdots + f_{i,j}(c_i, c_j)) - \\
&\quad (\beta_0 + f_1(x_1) + \cdots + f_k(x_k) + \cdots + f_{i,j}(x_i, x_j)) \\
&= (f_1(c_1) - f_1(x_1)) + \cdots + (f_k(c_k) - f_k(x_k)) + \cdots + \\
&\quad (f_{i,j}(c_i, c_j) - f_{i,j}(x_i, x_j)) \\
&= g(x_1, c_2) + \cdots + g(x_k, c_k) + \cdots + g(x_i, x_j, c_i, c_j) \quad (3)
\end{aligned}
$$

We define the local score gain $g(x_i, c_i) = f_i(c_i) - f_i(x_i)$ as the shape function value difference of changing the main feature x_i to c_i. Similarly, we define the local score gain of a pair-wise interaction term as $g(x_i, x_j, c_i, c_j) = f_{ij}(c_i, c_j) - f_{ij}(x_i, x_j)$. Then, we can see that the counterfactual constraint $g(x, c) \geq -S_x$ or $g(x, c) < -S_x$ is just a linear constraint that consists of a linear combination of shape function value differences.

A.3 Proximity Requirement

To provide helpful recourse to end users, we want CF examples to be actionable. One of the most critical measurements of recourse actionability is high proximity between the CF example and the original input, where we want the CF example to only make minimal changes to the original input values [83, 88]. For example, a CF example that suggests increasing annual income by \$5k would be more actionable than another CF example suggesting to increase annual income by \$10k. We can formulate this proximity requirement as to minimize the distance $d(x, c)$ between the original input and the CF example—sum of the distances across all features.

$$
d(x, c) = d(x_1, c_1) + d(x_2, c_2) + \cdots + d(x_k, c_k) \quad (4)
$$

Note that there is no distance cost for pair-wise interaction terms after considering the main effects. We will discuss our choice of distance functions for continuous and categorical features in-depth

in § A.5. If all distance functions are linear, or we can pre-compute each $d(x_k, c_k)$, then the proximity requirement can be formulated as a linear objective function that we want to minimize.

A.4 Integer Linear Optimization

As a gradient-boost tree model, EBM applies equal-frequency binning on continuous features to speed up the training process with a minimal accuracy cost. For categorical features, EBM uses the discrete levels as bins. For pair-wise interaction terms, EBM also bins two feature values to construct a lookup table. Therefore, a CF example can alter the model output if and only if it changes the active bins that some feature values are in. There are finite number of bins, where each bin provides a local score gain $g(x_i, c_i)$ and has a distance cost $d(x_i, c_i)$. Therefore, generating CF examples for EBM can be thought as solving a variation of *Knapsack Problems* [71]. A knapsack problem considers a set of items where each item has a reward and a weight, and the goal is to find the optimal way to pack items to maximize the total reward under a weight budget. Popular methods used to solve knapsack problems include integer programming (IP) and dynamic programming. GAM Coach uses IP because (1) it allows users to easily customize optimization constraints (§ A.8); (2) users can generate multiple optimal and sub-optimal CF example as recourse (§ A.8); (3) modern IP solvers can quickly find a globally optimal solution (§ A.10).

We express the GAM Coach CF generation method as an integer linear programming of the form:

$$
\text{min distance} \tag{5a}
$$

$$
\text{s.t. distance} = \sum_{i=1}^{k} \sum_{b \in B_i} d_{ib} v_{ib} \tag{5b}
$$

$$
-S_x \leq \sum_{i=1}^{k} \sum_{b \in B_i} g_{ib} v_{ib} + \sum_{(i,j) \in N} \sum_{b_1 \in B_i} \sum_{b_2 \in B_j} h_{ijb_1b_2} z_{ijb_1b_2} \tag{5c}
$$

$$
z_{ijb_1b_2} = v_{ib_1} v_{jb_2} \quad \text{for } (i, j) \in N, \ b_1 \in B_i, \ b_2 \in B_j \tag{5d}
$$

$$
\sum_{b \in B_i} v_{ib} \leq 1 \quad \text{for } i = 1, \ldots, k \tag{5e}
$$

$$
v_{ib} \in \{0, 1\} \quad \text{for } i = 1, \ldots, k, \ b \in B_i \tag{5f}
$$

$$
z_{ijb_1b_2} \in \{0, 1\} \quad \text{for } (i, j) \in N, \ b_1 \in B_i, \ b_2 \in B_j \tag{5g}
$$

Here, we use an indicator variable v_{ib} (5f) to denote if a main effect bin is active. If $v_{ib} = 1$, it means that we change the feature value of x_i to the closest value in its bin b. All bin options of x_i are listed in a set B_i. For each feature x_i, there can be at most one active bin (5e); if there is no active bin, then we do not change the feature value of x_i. Similarly, we use an indicator variable $z_{ijb_1b_2}$ (5g) to denote if an interaction effect is active. This interaction effect is active if and only if bin b_1 of feature x_i and bin b_2 of feature x_j are both active (5d). N denotes a set of feature pairs that the given EBM computes interaction effects from. Constraint (5b) determines the total distance cost for a potential CF example; it uses a set of pre-computed distance costs d_{ib} of changing one feature x_i to the closest value in bin b (§ A.3).

Constraint (5c) ensures that any solution would flip the prediction of the given EBM model (§ A.2). Constraint (5c) is used when the model predicts the original input as negative; if the original prediction is positive, we only need to change $\leq$ to $>$ (§ A.2). Here, g_{ib} and $h_{ijb_1b_2}$ denote pre-computed local score gains of activating bin b in x_i and activating the interaction effect $z_{ijb_1b_2}$, respectively. Note that activating one bin can trigger multiple interaction effects, but $h_{ijb_1b_2}$ is only counted when both v_{ib_1} and v_{jb_2} are active (5c and 5g). Therefore, we compute g_{ib} by preemptively adding the shape function differences of all partially affected interaction effects to the shape function difference of the main effect. For example, if $N = \{(i, j), (i, m), (l, m)\}$, we compute g_{ib} and g_{jb} as:

$$g_{ib} = (f_i(x_{ib}) - f_i(x_{i0})) + (f_{ij}(x_{ib}, x_{j0}) - f_{ij}(x_{i0}, x_{j0})) + \\ (f_{im}(x_{ib}, x_{m0}) - f_{im}(x_{i0}, x_{m0})) \tag{6a}$$

$$g_{jb} = \left(f_j\left(x_{jb}\right) - f_j\left(x_{j0}\right)\right) + \left(f_{ij}\left(x_{i0}, x_{jb}\right) - f_{ij}\left(x_{i0}, x_{j0}\right)\right) \tag{6b}$$

Here, x_{ib} denotes the closest value of bin b of feature x_i, and x_{i0} denotes the original value of feature x_i. In 6a, we add two partial interaction score gains because activating bin b of feature x_i affects two interaction terms (i, j) and (i, m). Similarly, 6a only includes one partial interaction score gain because activating bin b of feature x_j only affects one interaction term (i, j).

However, when both v_{ib_1} and v_{jb_2} are active, the interaction score gain should be $f_{ij}\left(x_{ib_1}, x_{jb_2}\right) - f_{ij}\left(x_{i0}, x_{j0}\right)$. Therefore, we need to offset two partial interaction score gains added preemptively when computing g_{ib} and g_{jb} (6a and 6b). To do that, we simply subtract them when computing the interaction score gain $h_{ijb_1b_2}$:

$$h_{ijb_1b_2} = \left(f_{ij}\left(x_{ib_1}, x_{jb_2}\right) - f_{ij}\left(x_{i0}, x_{j0}\right)\right) - \\ \left(f_{ij}\left(x_{ib_1}, x_{j0}\right) - f_{ij}\left(x_{i0}, x_{j0}\right)\right) - \\ \left(f_{ij}\left(x_{i0}, x_{jb_2}\right) - f_{ij}\left(x_{i0}, x_{j0}\right)\right) \tag{7}$$

Once trained, the EBM model transforms all parameters into lookup histograms and lookup tables (§ 4.1), so we can quickly pre-compute all g_{ib} and $h_{ijb_1b_2}$ terms. Furthermore, we can linearize the binary variable multiplication constraint (5d) as three linear constraints: (1) $z_{ijab} \leq v_{ia}$; (2) $z_{ijab} \leq v_{jb}$; (3) $z_{ijab} \geq v_{ia} + v_{jb} - 1$. Then, all constraints (5b–5g) are linear, and (5) is an integer linear program with all binary variables, which can be efficiently solved by modern IP solvers [72]. As this formulation considers all possible effective changes to the original input, the solution to (5) is guaranteed to be the optimal CF example regarding the given distance functions.

A.5 Choice of Distance Function

It is challenging to define a distance function that can accurately measure the difficulty for end users to change a feature [3]. In GAM Coach, we use the ℓ_1 distance to measure the distance between the original input and the CF example across continuous features. As different continuous features often have different scales, we divide each feature-wise distance by the median absolute deviation (MAD) of that feature on the training set, which is a common choice among other CF generation methods [e.g., 33, 57, 88]. MAD

provides a robust way to measure the variance within each feature. Here, n is the size of the training set. Dividing the ℓ_1 distance with MAD implies that it is relatively easier for end users to change a high-variance features than low-variance features.

$$d_{\text{cont}}(x_i, c_i) = \frac{|x_i - c_i|}{\text{Median}_{j=1}^n \left(x_i^{(j)} - \text{Median}_{p=1}^n \left(x_i^{(p)}\right)\right)} \tag{8}$$

It is harder to define the distance for categorical features. Some CF methods use 1 for features having the same level and 0 for different level [57], and others consider the probability that two examples would share the same level [95]. In GAM Coach, we use the complement of the probability of seeing one level based on its frequency in the training set. Here, n is the size of the training set and $\mathbb{I}$ is the indicator function. This distance definition implies that it is easier for end users to change to a more frequent level in a given categorical feature.

$$d_{\text{cat}}(x_i, c_i) = 1 - \frac{\sum_{j=1}^n \mathbb{I}\left(x_i^{(j)} = c_i\right)}{n} \tag{9}$$

After counting distance costs of all bins of main effects, we re-weight distance costs of all categorical bins so that the average of continuous feature distances is the same as the average of categorical feature distances. There is no right way to choose distance functions [3, 57]. Fortunately, all distances are pre-computed before solving the actual IP, and GAM Coach provides flexible APIs to let developers use their own distance functions.

Ultimately, we believe that instead of researchers searching for a one-fit-all distance function, we should enable end users to directly specify their own difficulty to change features (G2). To do that, GAM Coach provides end users with an interface to select feature *difficulties* by clicking buttons (Fig. 4-B1). Internally, GAM Coach assigns each difficulty level with a constant multiplier (Fig. S1). Before solving the IP, the tool multiplies the pre-computed distances of all bins in a feature with this constant multiplier. For example, if a user selects "very easy" for feature i, then the distance between the original value c_i and the closest value in bin b_{ij} of feature i is computed as $0.1 \times d\left(b_{ij}, c_i\right)$. If a user selects the "impossible to change" difficulty, GAM Coach will remove all variables associated with this feature in the IP. Therefore, when generating new recourse plans, GAM Coach would prioritize features that are easier to change and would not consider features that are impossible to change. We choose six levels of feature difficulties because we observe that we can mix and match these six levels on different features to flexibly fine-tune recourse generation in our experiments with six datasets. We choose the four constant multipliers [0.1, 0.5, 2, 10] because they can noticeably affect the IP solutions with "appropriate" strengths. However, researchers and developers can easily change these constant values and also the difficulty granularity (e.g., with only three levels "very easy", "neutral", and "impossible") in their specific use cases.

Difficulty	Distance
😎 Very easy	$\times 0.1$
😃 Easy	$\times 0.5$
🙂 Neutral	$\times 1$
😬 Hard	$\times 2$
😣 Very hard	$\times 10$
🔒 Impossible	$\times \infty$

Figure S1: Distance multipliers of difficulties.

A.6 Generalization to Regression

Barocas et al. [3] finds that algorithmic recourse literature often assumes the ML model outcome to be binary, such as loan approval, school acceptance, and hiring decision. However, in reality, end users also need recourse for AI-generated decisions on continuous values such as a loan's interest rate. GAM Coach supports generating CF examples for regression problems. To do that, we only need to modify the CF constraint to bound the needed score gain to meet the desired range provided by the end user (§ A.2). Then, we can update the left side value $-S_x$ and the inequality in 5c to reflect the score gain boundaries. This constraint would still be linear, and IP solver can solve the whole program. For example, to increase the predicted continuous value (e.g., interest rate) by at least δ, we only need to modify 5c to be:

$$\delta \leq \sum_{i=1}^{k} \sum_{b \in B_i} g_{ib} v_{ib} + \sum_{(i,j) \in N} \sum_{b_1 \in B_i} \sum_{b_2 \in B_j} h_{ijb_1b_2} z_{ijb_1b_2} \quad (10)$$

A.7 Generalization to Multiclass Classification

In addition to regression, our IP can be easily generalized for multiclass classification. Compared to binary EBM, multiclass EBM [100] uses a multiclass cross entropy as its loss function and softmax as its link function. Once trained, an n-class EBM has a similar structure as the binary EBM. However, there are no interaction terms in a multiclass EBM, and each bin of a feature now has n associated additive scores instead of just 1 score as in binary EBM. During inference, the n-class EBM adds up the additive scores from all features and an intercept for each class. For example, we use S_x^1 to denote the score for class 1 of input x, then $S_x^1 = \beta_0^1 + f_1^1(x_1) + f_2^1(x_2) + \cdots + f_k^1(x_k)$. Next, the softmax link function (Equation 11) rescales n scores $S_x^1, S_x^2, \ldots, S_x^n$ to n class probabilities $\sigma_x^1, \sigma_x^2, \ldots, \sigma_x^n$, where $\sum_{j=0}^{n} \sigma_x^j = 1$. Finally, the multiclass EBM chooses the class j with the largest σ_x^j as the final prediction.

$$\sigma_x^p = \frac{\exp\left(S_x^p\right)}{\sum_{j=1}^{n} \exp\left(S_x^j\right)} \quad (11)$$

Note that the softmax function is monotonic and it preserves the rank order of its input values. In other words, to make a multiclass EBM predict class p on a CF example c, we only need to make $S_c^j < S_c^p$ for $j = 1, \ldots, n$ and $j \neq p$, which can be written as $n-1$ linear constraints. Therefore, the GAM Coach CF generation method for multiclass classification (target class is p) can be written as the following integer linear program:

$$\text{min distance} \quad (12a)$$

$$\text{s.t. distance} = \sum_{i=1}^{k} \sum_{b \in B_i} d_{ib} v_{ib} \quad (12b)$$

$$S_x^j + \sum_{i=1}^{k} \sum_{b \in B_i} g_{ib}^j v_{ib} < S_x^p + \sum_{i=1}^{k} \sum_{b \in B_i} g_{ib}^p v_{ib}$$
$$\text{for } j = 1, \ldots, n \text{ and } j \neq p \quad (12c)$$

$$\sum_{b \in B_i} v_{ib} \leq 1 \qquad \text{for } i = 1, \ldots, k \quad (12d)$$

$$v_{ib} \in \{0, 1\} \qquad \text{for } i = 1, \ldots, k, \ \ b \in B_i \quad (12e)$$

In constraint 12c, S_x^j is the total score for class j of the original input x. Similar to g_{ib} in 5c, g_{ib}^j denotes the score gain for class j of changing feature x_i to the closest value in its bin b. All constants S_x^j and g_{ib}^j can be pre-computed.

A.8 Support Various Actionability Constraints

To generate recourses that are actionable for end users, we not only prefer CF examples that are close to the original input (§ A.3), but also *concise* [48], *diverse* [57, 70], and respect to individual end users' *preferences* [3, 39]. With GAM Coach, we can generate CF examples with these desired properties by formulating these requirements as linear constraints in the IP. For example, to generate concise or sparse CF examples—examples that only change a few features from the original input—we can introduce a linear constraint to bound the total number (up to p) of active variables for main effects: $\sum_{i=1}^{k} \sum_{b \in B_i} v_{ib} \leq p$. To generate diverse CF examples, we can solve the same IP multiple times, where each time we add a new constraint to force the solver to avoid previous solutions. For example, we can set $v_{ib_i} v_{jb_j} v_{kb_k} = 0$ for new iterations where $\{v_{ib_i} = 1, v_{jb_j} = 1, v_{kb_k} = 1\}$ is a previous solution. Since all variables are binary, we can linearize these multiplication constraints [18]. With this approach, the generated k diverse solutions are also guaranteed to be the top-k optimal solutions. Similarly, if we have prior knowledge of end users' preferences, such as difficulties and actionable ranges of individual features, we can adjust the distance costs during the pre-computation process. Therefore, the flexibility of IP helps us operationalize the design of GAM Coach (G2).

A.9 CF Generation Method Comparison

Our CF generation method is the first and only CF algorithm specifically developed for EBM models. Before our method, ML researchers and developers would need to use model-agnostic algorithms like genetic algorithm [73] and KD-tree [85] to generate recourse plans for EBM models. Our technique is guaranteed to outperform or tie with these algorithms if we measure the quality of CFs by their distances (e.g., ℓ_1 distance) to the original input. This is because our technique formulates CF generation as a linear optimization program (§ A.4) that minimizes the distance between the modified and original inputs. For completeness, we have included such comparison results in Table S1 to give readers a sense of how far from optimal existing CF generation methods are in terms of distance.

In the comparison experiment, we train three EBM binary classifiers on LendingClub [1], Adult [45], and German Credit [14] datasets. We use our IP approach, genetic algorithm, and KD-tree to generate CFs for test samples that are rejected for a loan (378, 400, and 239 samples from three datasets). We use the DICE library's implementation [57] of the genetic algorithm and KD-tree. We disable our method's default categorical distance (§ A.5) to match the other two algorithms (distance is 1 if the category is changed and 0 otherwise). All three algorithms use MAD adjusted ℓ_1 to measure the distance of continuous variables. The distance between two samples is defined as the mean of all categorical and continuous distances. The results (Table S1) highlight that compared to existing methods, CFs generated by our method are significantly *closer* to the original input, *more sparse*, and encounter *fewer failures*.

Table S1: We compare our method with two existing CF generation methods: genetic algorithm and KD-tree. We train three EBM binary classifiers on LendingClub, German Credit, and Adult datasets, and then apply three CF algorithms to generate CFs for test samples that are rejected for a loan. The results highlight that our method significantly outperforms existing methods. In particular, CFs generated by our method are *closer* to the original input, *more sparse,* and encounter *less failures.*

	Mean Distance	Mean Number of Features Changed	Number of Failures
Lending Club (378 samples)			
Our Method	**0.1836**	**2.2222**	**0**
Genetic Algorithm [73]	3.1950	10.2520	1
KD Tree [85]	3.7388	10.8360	6
German Credit (239 samples)			
Our Method	**1.1392**	**2.0962**	**0**
Genetic Algorithm [73]	6.8573	9.3305	0
KD Tree [85]	7.3565	9.9414	0
Adult (400 samples)			
Our Method	**1.6856**	**2.4075**	**0**
Genetic Algorithm [73]	4.9231	4.6475	0
KD Tree [85]	5.1082	4.9500	0

A.10 Fast CF Generation

In many cases of providing algorithmic recourse, we need to prioritize CF example generation speed over the optimality of generated CF examples [73]. With GAM Coach, modern IP solvers can efficiently solve the program (Equation 5). The complexity of solving an integer linear program increases along two factors: the number of variables and the number of constraints. Here, all variables are binary—making the program easier to solve than a program with non-binary integer variables. For any dataset, there are always exactly 3 constraints from 5b, 5c, and 5e. The number of constraints from 5d increases along the number of interaction terms $|N|$ and the number of bins per feature $|B_i|$ on these interaction terms. In practice, $|N|$ and $|B_i|$ are often bounded to ensure GAMs are interpretable. For example, by default the popular GAM library InterpretML [60] bounds $|N| \leq 10$ and $|B_i| \leq 32$. Therefore, in the worst-case scenario with 10 continuous-continuous interaction terms, there will be at most $10 \times 32 \times 32 = 10,240$ constraints from 5d. For example, on the Communities and Crime dataset [67] with

119 continuous features, 1 categorical feature, and 10 pairwise interaction terms, there are about 7.2k constraints and 3.6k variables in our program. It only takes about 0.5–3.0 seconds to generate a recourse plan using Firefox Browser on a MacBook.

In addition, in applications where the generation speed is critical, developers can significantly improve the run time by filtering less effective bins during the pre-computation process, which decreases the number of variables quadratically. First, developers can filter out main effect bins that give opposite score gains from the objective (i.e., positive score gain when the goal is to lower the prediction score). By default, GAM Coach does not apply this filtering, because in rare cases the score gains of associated interaction terms can offset the opposite score gain from the main effect. By filtering out bins with opposite score gains, GAM Coach can consistently generate CF examples in under 1 second in end users' browsers (§ 5). To further improve the speed, developers can also filter out main effect bins that give similar score gains as existing bins but have a higher distance cost.
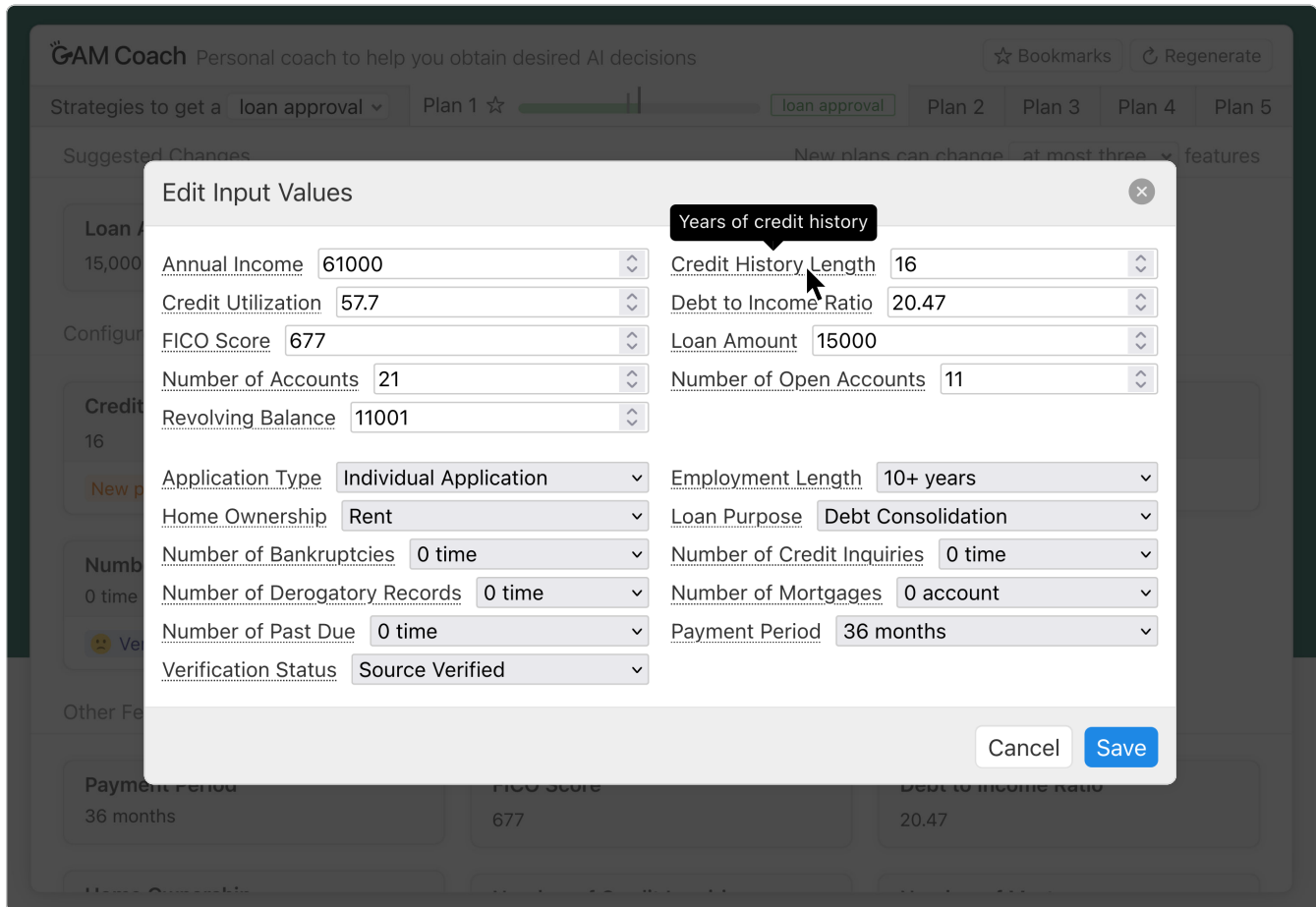
B SUPPLEMENTARY FIGURES

Figure S2: To help user study participants imagine the loan application scenario, the GAM Coach interface allows participants to change the input values of the hypothetical loan applicant. The top row includes input fields for the 9 continuous features, and the bottom row contains dropdowns for the 11 categorical features used in the EBM model. Users can hover over the feature name to see the detailed description for that feature.