Scientific Research Publishing

Journal of Computer and Communications, 2024, 12, 169-198
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

What-If XAI Framework (WiXAI): From Counterfactuals towards Causal Understanding

Neelabh Kshetry, Mehmed Kantardzic

Department of Computer Science, Data Mining Laboratory, University of Louisville, Louisville, USA
Email: neelabh.kshetry@louisville.edu, mehmed.kantardzic@louisville.edu

How to cite this paper: Kshetry, N. and Kantardzic, M. (2024) What-If XAI Framework (WiXAI): From Counterfactuals towards Causal Understanding. *Journal of Computer and Communications*, **12**, 169-198.
https://doi.org/10.4236/jcc.2024.126011

Received: May 20, 2024
Accepted: June 25, 2024
Published: June 28, 2024

Abstract

People learn causal relations since childhood using counterfactual reasoning. Counterfactual reasoning uses counterfactual examples which take the form of "what if *this* has happened differently". Counterfactual examples are also the basis of counterfactual explanation in explainable artificial intelligence (XAI). However, a framework that relies solely on optimization algorithms to find and present counterfactual samples cannot help users gain a deeper understanding of the system. Without a way to verify their understanding, the users can even be misled by such explanations. Such limitations can be overcome through an interactive and iterative framework that allows the users to explore their desired "what-if" scenarios. The purpose of our research is to develop such a framework. In this paper, we present our "what-if" XAI framework (**WiXAI**), which visualizes the artificial intelligence (AI) classification model from the perspective of the user's sample and guides their "what-if" exploration. We also formulated how to use the **WiXAI** framework to generate counterfactuals and understand the feature-feature and feature-output relations in-depth for a local sample. These relations help move the users toward causal understanding.

Keywords

XAI, AI, WiXAI, Causal Understanding, Counterfactuals, Counterfactual Explanation

1. Introduction

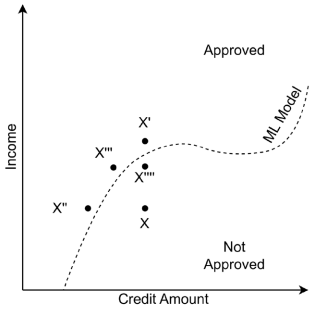
As Artificial Intelligence (AI) has become more ubiquitous in our daily lives, being used in our smartphones [1], banking systems [2], cybersecurity [3], insurance [4], law enforcement [5], etc., it becomes very important to be able to understand these systems to narrow down any flaws, biases, and evaluate the

performances. Most of what we refer to as AI is simply Machine Learning (ML), which uses large amounts of data to create a model that represents the actual system. Traditional approaches to ML like linear models, decision trees, etc. are easier to explain [6] but their performances pale in comparison to methods like Deep Neural Networks (DNN) [7]. These complicated methods are, however, much harder to explain due to their much higher complexity. For that reason, research in explainable AI (XAI), has become more intense in recent years. The explanations generated by an XAI framework can be either data explainability, model explainability, or post-hoc explainability [8]. Data explainability refers to explaining the data, while model explainability refers to explaining the model as a whole; but post-hoc explainability simply refers to a method of explaining the ML model's decision [9]. As the data usually dealt with are very large and the ML models quite complex, going the route of data or model explainability is difficult and thus a large portion of research focuses on post-hoc explanations.

One of these post-hoc explanation methods is the example-based counterfactual explanation (CFE) method. In simple words, counterfactuals are contrary examples to past events that help the thinker understand them [9]. Simply put, CFE explains to the users (including domain-experts and decision-subjects) in the form of "if *x* event had occurred differently, the outcome would've been different". Here, domain-experts refer to the experts in non-ML fields like doctors, nurses, accountants, economists, etc. who serve a client or a patient, whereas the decision-subjects refer to the people who are impacted by the decisions from the ML systems like medical patients, banking clients, suspects, etc. [10].

In **Figure 1(a)**, we can see a 2-dimensional example of a credit approval ML system, which uses data from the users (annual income and credit amount requested) and predicts credit "Approved" or "Not Approved". Here, the user has one set of feature values, represented by sample $\mathbf{X}$ in the graph 1, and the ML model predicts "Not Approved". In **Figure 1(b)**, we see the same scenario being represented in a tabular form. As mentioned before, a counterfactual explanation approach to this system is using counterfactuals. In **Figure 1(c)**, we see that a counterfactual example has been provided: "If the value of annual income had been \$57k instead of \$52k, the system would've given the prediction 'Approved'." This counterfactual point is also shown in **Figure 1(a)** as $\mathbf{X}'$. Similarly, we also see counterfactuals $\mathbf{X}''$ (changing the value of Credit Amount from \$7.5k to \$6.5k) and $\mathbf{X}'''$ (changing the value of both Income and Credit Amount as seen in **Figure 1(e)**). All 3 counterfactual examples in **Figure 1** give some idea about the sample and its relation to the classification model. All of them change the original classification, making them "valid" counterfactuals. Similarly, $\mathbf{X}''''$ doesn't change the original classification which makes it an "invalid" counterfactual. In this sense, the counterfactual explanation is a "retrospective reasoning" method [11], one that helps users understand the relation between the sample and the ML model. Once this relation between the sample and the ML model is understood, it becomes easier to find recourse for unfavorable outcomes. With enough variety of counterfactuals, it is also possible to gather feature-feature and fea-

ture-outcome relations. CFEs are also able to provide causal conclusions when phrased as conditional propositions [9].



(a) Simple Credit Approval Model

Income	$52,000
Credit Amount	$7,500
Prediction	Not Approved

(b) User's Sample $\mathbf{X}$

Income	$57,000
Credit Amount	$7,500
Prediction	Approved

(c) Counterfactual sample $\mathbf{X}'$

Income	$52,000
Credit Amount	$6,500
Prediction	Approved

(d) Counterfactual sample $\mathbf{X}''$

Income	$54,000
Credit Amount	$7,000
Prediction	Approved

(e) Counterfactual sample $\mathbf{X}'''$

Figure 1. Demonstration of counterfactual explanation.

The use of CFE in XAI has also been widely researched recently, with methods such as MAPLE [12], CEM [13], DiCE [14], FACE [15], CFX [16], etc. These methods solve the optimization problem of finding appropriate singular or a handful of counterfactuals. However, they fail to fully capture the user's preference. The goal of these methods is also limited to finding actionable recourse for the user rather than providing a framework that promotes learning and causal understanding. Allowing the users to interact with the explanations to explore their desired "what-if" scenarios, and generating a very large number of diverse counterfactuals are the foundational steps in allowing the users to move towards causal understanding [17].

In this work, we present our **WiXAI** framework, which visualizes the ML model in 2-dimensions and guides the users towards various counterfactuals, allowing them to try any number of desired "what-if" scenarios. The information gained from the visualization of the ML model (explained more in section), prevents the users from having to blindly try out these "what-if" scenarios. We also argue that our framework meets the requirements for a system that promotes causal understanding formalized by Chou *et al.* [17]. Thus, the **WiXAI** framework can be a step towards promoting a causal understanding of the ML system and a learning process for the users to have a better understanding of the model's decision and subsequently more trust and confidence in the system.

Our main contributions are:

1) Formalizing a method of visualizing N-dimensional classification model in 2 dimensions via Model Projections

2) Defining an interactive and iterative framework to allow for user-generated counterfactuals

3) Systematizing rules to help users move towards causal understanding with "what-if" scenarios

Helping users understand the decision of the model is very important in increasing their trust and confidence in an ML system. To allow this, we formalize

a method of visualizing the N-dimensional model into 2 dimensions. We are utilizing the projection of the model onto each feature space and stacking them together. The model projections allow the users to visualize the model according to their sample, making the visual explanations local. We also define an interactive and iterative framework that allows the users to dynamically explore their desired "what-if" scenarios and observe the changes in the view of the model. Observing the changes in the model's projections allows the users to understand the feature-feature and feature-output relations. We systematize the rules that help the users move from their "what-if" explorations toward causal understanding. The interactive framework also allows the users to test their understanding by making use of more "what-if" explorations.

2. Related Works

Since the first paper on Counterfactual Explanation in AI [18], there has been a lot of research done on various methods and applications of CFE [19]. This takes form in CFE methods for any form of data [13]-[15] [20] [21], image specific data [22]-[26], tabular data [10] [16] [27] [28], etc. Because a significant portion of data stored and used today is still in tabular form, data specifically in tabular form will be the focus of our framework this time.

Most of the research on CFE has been focused on finding counterfactuals by solving an optimization problem. These methods sometimes focus on finding the closest singular counterfactual [29]-[31], and sometimes multiple counterfactuals [14] [32] [33], some are model specific [14] [31] [33], and some model agnostic [29] [30] [32]. However, almost all of them have the common element of generating calculated counterfactuals using one of the four methods: solving optimization, heuristic search, instance-based search, and decision tree [34]. When generating automated or guided counterfactuals, these are useful tools, however, they can take away some or all user choices and thus sometimes provide inapplicable counterfactuals. Users not being able to verify their understanding, can erode trust and confidence in the explanation or the ML system.

A more user-friendly approach for CFE is to use an interactive tool that keeps the end-users in the loop (like in **Figure 2**). Many such tools have also been proposed and developed with various goals in mind [10] [28] [35]-[37]. ***WIT*** [35], that allowed the ML-experts to focus on sub-group analysis with "what-if" exploration was deprecated in Tensor Flow $v > 2.11$. It was replaced with ***LIT*** [36], which specifically focuses on text data. ***SystemD*** [37] was developed for and specifically focuses on business scenario. Most of them also suffer from the problem of complexity in their UI and packed features; also, most of them are either focused on specific domain experts (like doctors, nurses, economists, bankers, etc.) or designed for ML experts. The end goal of these tools is rarely to make them useful for the decision subjects. They mostly focus on providing explanations that meet the properties of CFE. These properties of CFE [38] are given in **Table 1**.

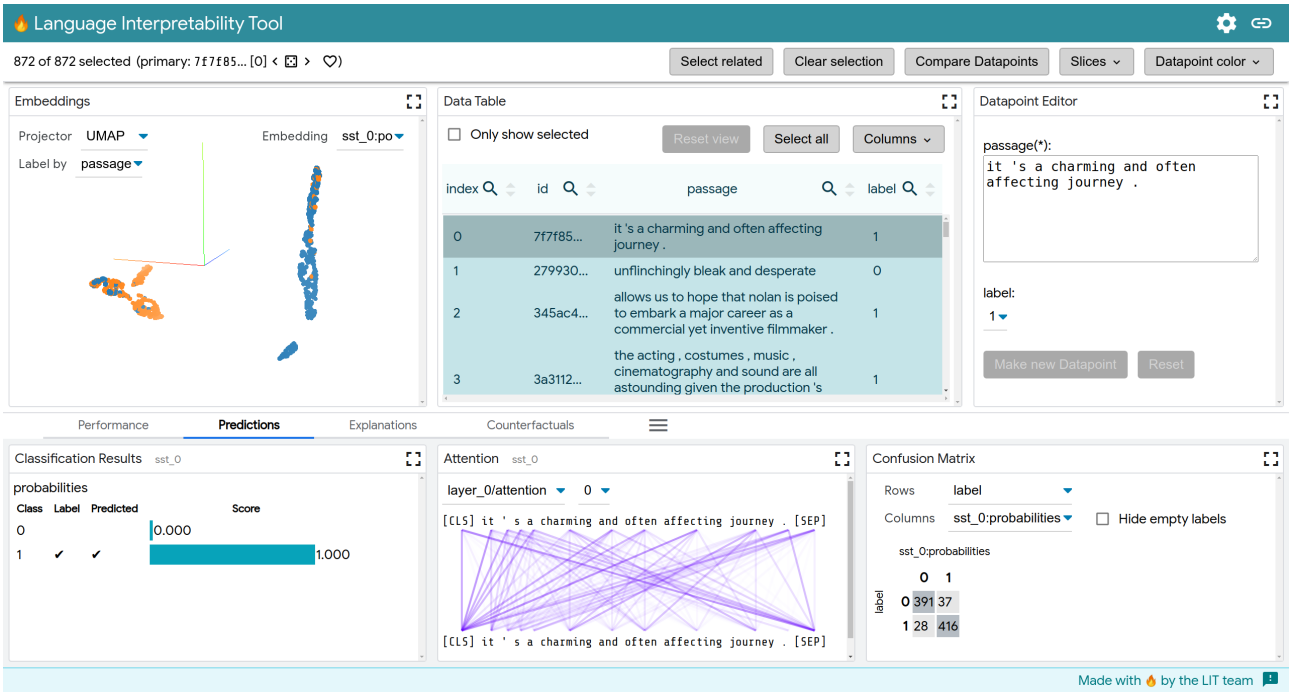


Figure 2. An example view of LIT tool.

Table 1. Properties of effective counterfactual explanation.

Property	Definition
Proximity	An effective counterfactual should be as similar to the original data as possible. This means the distance between the original and counterfactual should be minimized.
Validity	A counterfactual is valid only if it provides a desired class as the predicted classification from the ML system.
Sparsity	An effective counterfactual should minimize the number of different features whose values need to be changed to reach the counterfactual.
Plausibility	A counterfactual that exists as an outlier in the previously seen (training) data, is harder to believe. It is better that a counterfactual lies in a more dense region among the previously seen data.
Discriminative	In terms of explanation, a counterfactual example should be able to show why the counterfactual provides the desired outcome and not the original data.
Actionability	Actionable features refer to the features that the user can change. Thus, an actionable counterfactual only presents changes to the actionable features.
Causality	An effective counterfactual explanation should account for all known causal relationships between features (if a causal graph exists). This is different from causality between the features and the outcome.

Cheng *et al.* [10] conducted interviews with domain experts in their paper where they expressed some enthusiasm about the sub-group analysis feature but not as much for instance level analysis. Their DECE tool also contains a large amount of information and minute details, which can easily overwhelm someone without training. However, the main difference between their approach and the framework proposed in this paper is the goal. Their tool gives the end users

some way to constrain some feature values before the tool searches and presents the counterfactuals. In other words, they allow the users to find actionable counterfactuals with some user constraints. Our **WiXAI** framework is designed to give the end users a tool to explore and understand the classification boundaries and the behavior of the model at an instance level. This allows them to not only find their preferred counterfactuals but also gain more confidence in the system and understand their circumstances better for more informed decision-making.

Prospector [28] is slightly similar in that it aims to be an exploratory tool, but the stated target end-users are ML experts to explore their models and specific decisions. This tool is designed so that ML experts can understand how the model works in certain situations and why certain predictions are made. It also allows the ML experts to check how changing some values might change the prediction. Beyond this, the tool is equipped with dependence plots and other forms of explanations which are mostly useful for, and targeted toward model makers and not as much for end-users (domain experts or decision-subjects).

The goal of XAI systems has always been to ultimately achieve a causal explanation but they fall short of this mark. As Pearl [9] explains, counterfactuals are the "building blocks of scientific thinking, as well as legal and moral reasoning." Counterfactuals also help people identify cause-effect relations as well [39]. There have been many attempts to link CFE with causal explanations [40]-[42], however, it is still far from complete [19].

Chou *et al.* [17] explain how CFE promises to be a promising avenue to moving towards causal understanding as part of XAI. In their paper, Chou *et al.* [17], begin with an optimism about the potential of CFE to move towards causability but ultimately conclude, that the existing literature doesn't contain any approach or algorithm that sufficiently achieves this. They go as far as to claim that the current algorithms provide "spurious correlations rather than cause-effect relationships" which leads to poor and even biased explanations. They also lay out the requirements for an XAI system to promote causability. According to them [17], such a system has to be based on some theory of causality, present explanations in the form of counterfactuals, provide human-centric explanations, allow user interaction with the explanations, and be complemented by contextual or domain-specific knowledge for the end-user.

Building on this work and using Pearl-Woodward approach [43] [44], Baron [34] proposes a standard counterfactual approach to XAI theoretically capable of delivering causal understanding. He goes deep into the three different notions of causal understanding relevant to ML and how utilizing the Pearl-Woodward approach is capable of providing "complete causal certification." For this, he makes use of interventionist causation (explained in the next paragraph) and Pearl-Woodward variables (variables referred to in the Pearl-Woodward approach). Using his proposed system, the causal discovery happens in three steps: identifying ML variables that are individual causes, identifying ML variables that

are partial causes, along with the complex causes that they are part of, and a guarantee that no causes have been missed.

Interventionist Causation means that a variable X is a direct cause of Y, if, when fixing all other variables and changing only the value of X, it is possible to change the value of Y or its probability distribution [44]. While Baron [34] builds on this definition and the Pearl-Woodward approach to define theoretically a framework for a complete causal certification, our approach practically uses the idea of intervention to build a "what-if" framework to promote causal learning. As Wachter *et al.* [18] explains rather than finding the closest counterfactual, it is often more informative to provide several explanations that cover a wider range of diverse counterfactuals. Guidotti [38] also reports that children learn through counterfactual examples and exploring desired "what-if" scenarios. This is the goal of our framework as well: to allow users to explore desired "what-if" scenarios and move from counterfactuals toward causal learning.

3. Visualizing the Model Based on the Sample

The strength of counterfactual explanations comes from users being provided with various counterfactual scenarios, which help users find reasonable recourse, and deduce cause-effect relations from these examples [18]. A more potent method of an explanation framework is to allow the users to explore their desired "what-if" scenarios and gather these relations from observations [17]. However, with large dimensional data, blindly searching "what-if" scenarios can be unfruitful or less productive. As seen in **Figure 1**, explanations are easier to understand visually. Recognizing the model with respect to the sample helps generate counterfactual explanations as well as understand them.

However, unlike **Figure 1**, ML problems have much more than 2 dimensions. Visualizing the N-dimensional model is difficult, especially in a 2-dimensional medium like a computer screen or paper. In this section, we are presenting a method of visualizing the model as it appears from the perspective of the sample in question ($\mathbf{X}$). This is achieved by projecting the model from the data point onto each feature and presenting them one by one as horizontal stacks (seen later in **Figure 3(b)**). Creating these projections follows the concept of intervention (similar to the concept of interventionist causation by Woodward [44]). Intervention can be defined as a change in the value of a feature, while fixing other feature values, thus generating a counterfactual in the purest definition of the word [8]. This visualization of the model shows the users the classification boundaries on each feature and allows them to explore "what-if" scenarios in a more informed manner. As it pertains to a singular data point, these projections give local explanations for the user's sample, rather than trying to generalize the classification model.

In order to understand the model projections, we can consider an ML Classification Model $\mathbf{M}$, being used to predict a user's sample $\mathbf{X}$. A model projection on a specific feature for the given sample would be a series of classifications from

the **M**, when the values for that particular feature are changed according to its range and the rest of the feature values are kept constant. This type of model projection on a feature level can be stacked on top of each other to show all the features, which would give a more complete picture. This form of model projection assumes independence of the features when analyzing individually (meaning: changing one feature doesn't automatically change the value of another feature). Model projections are explained in more detail in the section 3.1.

3.1. Visualizing the Model via Model Projection

Let $\mathbf{X} = \{\theta_1, \theta_2, \cdots, \theta_i, \cdots, \theta_n\}$ be a data point (a sample) with *n* features $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_i, \cdots, \mathbf{x}_n\}$ and **M** be the ML classification Model such that **y** = **M**(**X**) is the classification for the sample **X**.

Now, let the counterfactual $\mathbf{X}_i'$ represent the sample $\mathbf{X}'$ such that: $\mathbf{X}' = \{\theta_1, \theta_2, \cdots, \theta_i', \cdots, \theta_n\}$.

i.e all values of **X** remain the same except for the i^{th} feature.

Then, the model projection on the i^{th} feature can be represented as a series of $\mathbf{y}_i'$ obtained by running $\mathbf{X}_i'$ where the value of $\mathbf{x}_i$ is changed through its range. This implies from $\min(\mathbf{x}_i)$ to $\max(\mathbf{x}_i)$ at a varying interval **v** for numerical features such that the length of $\mathbf{y}_i'$ is constant **r**. Similarly, for categorical features through all possible values of $\mathbf{x}_i$.

Note: We can change the value of **v** *to adjust how many* $\mathbf{y}_i'$ *will represent the projection and ultimately adjust the resolution (***r***) of the projection. Then* **v** *becomes the function of the range of* $\mathbf{x}_i$ *and* **r** *as seen in equation 1.*

$$\mathbf{v} = \frac{\max(\mathbf{x}_i) - \min(\mathbf{x}_i)}{\mathbf{r}} \tag{1}$$

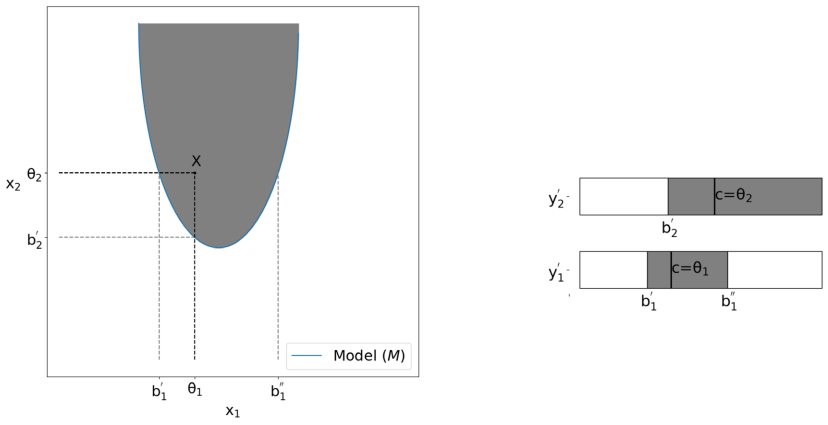
In algorithm 1, we can see the pseudocode for creating Model Projections for the sample **X**. It outputs a list of lists, *y'* where for numerical features, length is equal to the resolution **r** and for categorical features, it is equal to all possible values of $\mathbf{x}_i$. For each feature, the projection is a list of classes that represents a potential "what-if" output class from the sample **X**. To represent the current values, the list also contains '**c**' (current sample to prevent any confusion). To achieve the model projections, we will impute one feature at a time and feed the imputed data point **X**'to the model.

3.2. Understanding How Model Projections Look

As the explanation above and the pseudo-code (given in Algorithm 1) for creating the model projections suggest, model projections are in the form of a series of lists of classifications. Each list represents one feature and can convey the independent impact of the feature within its domain when every other feature is kept fixed. It is also possible to visually look at the classification boundary and take informed steps to move towards a valid counterfactual as will be discussed in the following paragraphs. A visual demonstration of the model projections on simpler 2-dimensional data is shown in Figure 3 and explained subsequently.

Algorithm 1: Creating Model Projections

```
// Model Projections are stored as series of series
// y'_i = [...] for the i^th feature
// y' = [y'_1, y'_2, ..., y'_i, ..., y'_n] represents the Model
   Projections
function Create_Model_Projections(X)
    y' = [ ]
    for i in range(n) do
        y'_i = [ ]
        if type(x_i)==''numerical'' then
            v = (max(x_i) − min(x_i))/r
            for x'_i in range (min(x_i), max(x_i), v) do
                if x'_i == x_i then
                    y'_i.append('c') // c represents the current
                       sample
                else
                    ŷ = M(X'_i)
                    y'_i.append(ŷ)
        else if type(x_i)==''categorical'' then
            for x'_i in all_possible(x_i) do
                if x'_i == x_i then
                    y'_i.append('c') // c represents the current
                       sample
                else
                    ŷ = M(X'_i)
                    y'_i.append(ŷ)
        y'.append(y'_i)
    return y'
```



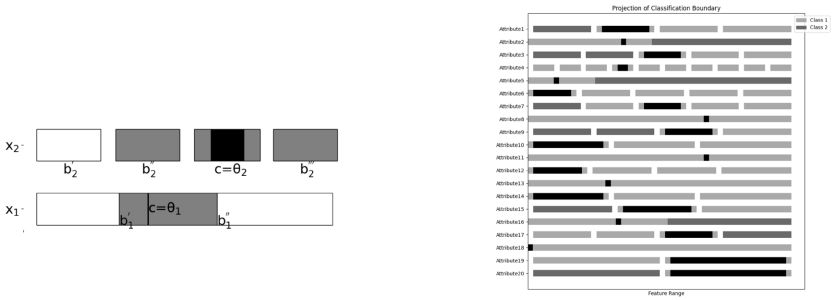
(a) Model Projections on 2D data for $\mathbf{X}$

(b) Model Projections $\mathbf{y}'_1$ and $\mathbf{y}'_2$ for $\mathbf{X}$

Figure 3. Demonstrating the use of Model Projections on 2D data.

In Figure 3(a), we can see $\mathbf{X}$ is a 2-dimensional sample and $\mathbf{M}$ is the binary classification model with the classes being *gray* and *white*. Currently, $\mathbf{X}$ has feature values: $x_1 = \theta_1$ and $x_2 = \theta_2$ and the model projections on feature x_1 and x_2 are represented by y'_1 and y'_2 respectively. For feature x_1 we can see the two boundary points are b'_1 and b''_1. This means if the value of x_1 is changed

such that $x_1 < b_1'$ or $x_1 > b_1''$, the classification is changed from *gray* to *white*. As for feature x_2, the value needs to be changed such that $x_2 < b_2'$ to change the classification. This much information is also obvious in **Figure 3(b)** without having to see the model and its characteristics/shape. In **Figure 4(a)**, we see how the model projections look if there are categorical features. Feature x_2 is categorical with 4 different possible values in this illustration. For categorical features, the model projections are represented as discrete blocks each representing one categorical feature value filled with one color representing the projected class. The current sample is represented by a thick black bar on top as can be seen in the **Figure 4(a)**. The demonstration of this method on a random sample in German Credit dataset [45] is shown in **Figure 4(b)**.



(a) Model Projection with one categorical feature (x_2)

(b) Demonstration of Model Projections on breast cancer data

Figure 4. Demonstration of model projections on a complex dataset.

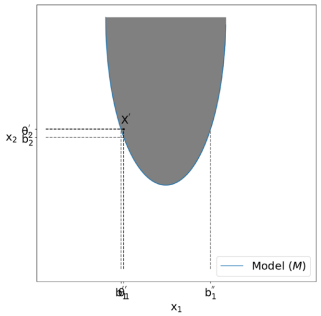
In **Figure 4(b)**, we see an ML classification model on the 20-dimensional German credit dataset [45] projected onto each of the 20 features. 13 of the features are categorical while 7 of them are numerical. The lighter shade represents class 1 (good credit) while the darker shade represents class 2 (bad credit), and the black lines represent the current data point (**c** as used in algorithm 1). This is how the model looks from the sample's perspective. We can see how changing the value for 10 of the features can individually change the predicted class. For the numerical features Attribute2, Attribute5, and Attribute16, increasing the value brings the sample closer to the other class, while for categorical features, some have one feature value that changes the classification while others have multiple. Features like Attribute4 and Attribute8 have a single class model projection, which signifies that changing its value alone cannot change the classification. In this manner, using these model projections, we can guide our search for counterfactuals, even with high-dimensional data.

3.3. Finding Counterfactuals with Model Projections

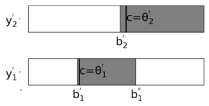
Once we see the projections and understand what they mean, it is not too difficult to figure out how to find valid counterfactuals. The simplest ones would be finding the sparsest counterfactuals with only one feature value changed. In our example, **X** has classification *gray* according to the model **M**. Valid counterfac-

tuals would be crossing the boundaries in the model projections seen in Figure 3(b). Changing the value of x_1 from θ_1 to $< b_1'$ or $> b_1''$ or changing the value of x_2 from θ_2 to $< b_2'$ both result in the change in classification thus creating valid counterfactuals. But as we can see in Figure 3(a), because of the curve of the model **M**, the optimal valid counterfactual would need to change the value of both x_1 and x_2.

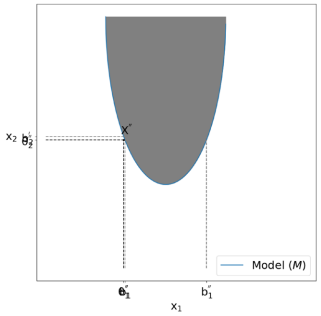
A more complex but also more important method of finding valid counterfactuals using model projections is to make gradual changes and look for changes in the model projections of other features. In our example, we can see from Figure 3(b) that sufficiently decreasing the value of either x_1 or x_2 will create a valid counterfactual but we (the user) can choose which one is more preferable. The choice doesn't matter but the key is to change the value by a small amount and check the projections again. We can see in Figure 5(a), we chose to decrease the value of x_1 slightly and the new projections can be seen in Figure 5(b). While the change needed in x_2 from θ_2 to b_2' was slightly larger, the small change in x_1 from θ_1 to θ_1 has decreased the difference between the two and thus making it possibly easier to change the value of x_2 and find an easier counterfactual. This results in the counterfactual $\mathbf{X}''$ being composed of two steps rather than one step (resulting in a less sparse counterfactual), but at the same time it is more optimal than $\mathbf{X}'$ as is evident in the two Figure 3(a) and Figure 5(c).



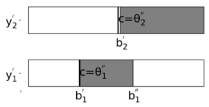
(a) Model Projections on 2D data for $\mathbf{X}'$



(b) New Model Projections $\mathbf{y}_1'$ and $\mathbf{y}_2'$ for $\mathbf{X}'$



(c) Model Projections on 2D data for $\mathbf{X}''$

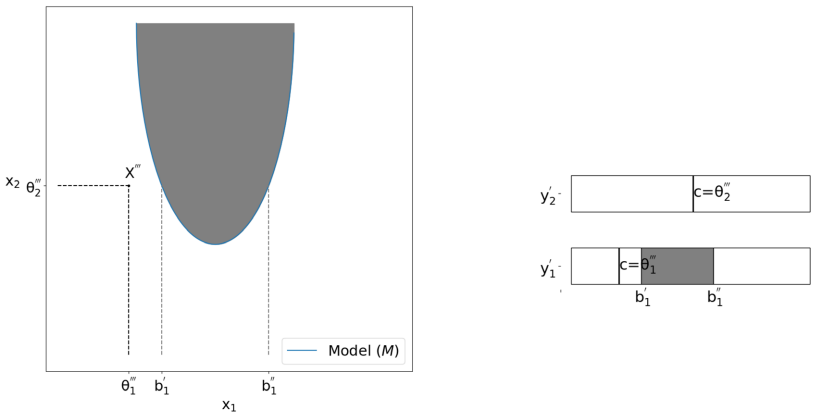


(d) New Model Projections $\mathbf{y}_1'$ and $\mathbf{y}_2'$ for $\mathbf{X}''$

Figure 5. Demonstrating the use of Model Projections on 2D data to find valid counterfactual.

3.4. Using Model Projection to Move towards Causal Understanding

Interventionist feature, similar to interventionist causation [44], is any feature that can change the classification by changing its value while keeping all other feature values fixed. In the case of a multi-class problem, the classification needs to be changed to the desired class. In Figure 6, we can see an example of an interventionist feature x_1 as well as a non-interventionist feature x_2. When looking at the model projections, interventionist and non-interventionist features are easy to find. An interventionist feature will have a multi-class model projection, whereas a non-interventionist feature will have a single-class model projection.



(a) Model Projections on 2D data for $\mathbf{X}'''$

(b) New Model Projections $\mathbf{y}'_1$ and $\mathbf{y}'_2$

Figure 6. Demonstrating the use of Model Projections on 2D data.

Using the visualization of counterfactuals via model projections, and exploring "what-if" scenarios, it is possible to systematically move toward causal understanding. As explained by Baron [34], causal discovery should find features that are whole causes (independent of other features), features that are part of a single cause (related features), and features that do not impact the outcome (non-causes). If we have interventionist features x_i and x_j, we can change the value of x_i without crossing the classification boundary and gather useful inter-feature relations, we can use the following rules to discover related and independent features:

1. If changing the value of x_i from θ_i to θ'_i, such that $\mathbf{M}(\mathbf{X}) = \mathbf{M}(\mathbf{X}')$, we observe change in the model projection for feature x_j (*i.e.* change in $\mathbf{y}'_j$), we can infer that features x_i and x_j are parts of the same cause.

2. If features x_i and x_j are independent causes to the output, changing the value of x_i from θ_i to θ'_i, such that $\mathbf{M}(\mathbf{X}) = \mathbf{M}(\mathbf{X}')$, we should observe no change in the model projection for feature x_j (*i.e.* no change in $\mathbf{y}'_j$).

3. If feature x_i is truly an independent cause, changing its value from θ_i to θ'_i such that $\mathbf{M}(\mathbf{X}) = \mathbf{M}(\mathbf{X}')$, doesn't change model projection ($\mathbf{y}'_j$) $\forall j$:

$1 \leq j \leq n, \quad j \neq i$.

4. If features x_i and x_j are part of the same cause and the closest classification boundary for x_i and x_j lies in the same direction, they are directly related (*i.e.* if b_i and b_j represent the closest boundary for directly related features x_i and x_j, ('c' < b_i) $\Leftrightarrow$ ('c' < b_j) and ('c' > b_i) $\Leftrightarrow$ ('c' > b_j)). If they are inversely related, the closest classification boundary lies in the opposite directions (*i.e.* if b_i and b_j represent the closest boundary for inversely related features x_i and x_j, ('c' < b_i) $\Leftrightarrow$ ('c' > b_j) and ('c' < b_i) $\Leftrightarrow$ ('c' > b_j)).

5. If feature x_k is a non-cause for the model, model projection for x_k (*i.e.* $\mathbf{y}'_k$) will never have more than one class.

In the rules laid out above, we are using "invalid" counterfactuals (counterfactual points that do not change the predicted class) to understand the relation between the features and between the features and the output. The first 2 rules help identify features that are part of the same cause, parts of different causes, and features that are independent causes. If any two features are part of the same cause, changing the value of one of them could influence the math of that cause according to the ML model. This implies that if we observe changes in the classification boundary of another feature while changing the value of a feature, they are parts of the same cause. Similarly, if a feature is an independent cause, changing its value should never change any model calculation for other features. This also implies that for an independent feature, changing its feature value, without changing the output class, will always result in the model projections for all other features remaining unchanged.

The fourth rule specifies how the dependent features are related to each other concerning the output. If the closest classification boundary for two features, part of the same cause, lies in the same direction, they are directly related. If the classification boundaries lie in opposite directions, they are inversely related. This means that if two features are directly related and increasing one's value brings it closer to the other class, increasing the other feature's value also brings it closer to the other class. Similarly, if they are inversely related and increasing one's value brings it closer to the other class, increasing the other feature's value brings it further from the other class. This direct and inverse relation applies to both additive and multiplicative relations at a local level. Finally, the last rule discusses non-causes. For any feature that is not a cause of the output, changing its value will never impact the ML model's math. This means that the model projection for these features will never show a different class.

4. Requirements for an Effective XAI Framework

Even as machine learning becomes more ubiquitous in our daily lives [1]-[4], mass adoption in critical fields like medicine, law, etc. have been met with resistance [17]. ML systems are complex by nature, and most users (domain-experts and decision-subjects) are untrained in ML or complex mathematics. Expecting them to understand the ML models is a tall ask, even with some explanations

from various XAI frameworks. Some of the static explanation methods of AI may even lead to incomplete explanations, which may mislead the users [46]. Without properly understanding the reasons for predictions, it becomes harder to trust ML systems [8]. It is our goal with this framework to allow the users to comprehend the model prediction for their sample more easily, thus improving confidence and trust in the ML system. We explain the reasons for having these framework requirements below:

1) The vast majority of users for an ML system will be decision-subjects or domain-experts. It is neither fair nor practical to expect them to possess any programming or statistical knowledge [47]. It is, therefore, very crucial that the framework is graphically interactive and easy to use without any programming knowledge. For most cases, however, it is reasonable to assume that the users will have some basic understanding of their own sample. For example, if the ML model is being used in a medical setting either by the medical professionals or the patients, it is reasonable that they understand what the sample represents as it pertains to them. In any case, the system can provide feature explanations as a supplement to the explanation framework.

2) The explanations should be dynamic instead of just static like feature ranking or a fixed set of counterfactuals. Being provided with such static explanations puts limitations on the true understanding of the problem by the user and can also diminish confidence in the system. Allowing the users to generate their choice of counterfactual and facilitating the exploration of "what-if"s, will let them intuitively discover the workings of the ML model, especially for their case [17]. Exploration of desired "what-if" scenarios is also how the children learn [38]. Allowing the freedom of exploration also helps users account for their preferences in features that may not be accounted for if calculated via optimization algorithms. Thus, explanation should be given via user exploration of the "what-if" scenarios.

3) The vast majority of the users (both decision-subjects and domain-experts) are not ML or statistics experts. Numerical explanations for them are harder to comprehend. Visual explanations on the other hand are more intuitive and convey the explanations more effectively. Emphasis should be put on providing explanations visually as much as possible. This allows the users to understand the reasoning for the decision better and garner more confidence and trust in the system as a result.

4) As mentioned earlier, static explanations can be misleading or incomplete. These explanations take the form of ranking features, highlighting critical features, providing a calculated set of counterfactuals, etc. To prevent misunderstanding of the model by the users, the explanations should be dynamic. Dynamic explanations are interactive and iterative. Allowing the users to interact with the explanations is very important in gaining confidence [17]. It is also important to make the explanations interactive and iterative to minimize chances of misunderstanding.

5) It isn't always possible for the system to have training data. Oftentimes, ML models that are used are proprietary and the training data used to generate the model is either too large or unavailable. The only guarantee is the availability of the user's data and the model's predictions. An effective explanation framework should be able to function without the training data. However, if the training data is available, the quality of the explanation can be improved.

6) The end goal of any XAI framework is to make the users feel more confident in the system by providing causal explanations. However, without providing a pathway towards causal understanding, it is not possible to achieve high satisfaction [11]. For this reason, an effective XAI framework must be able to lead the users towards causal understanding. This is achievable, if the users are provided with a large number of counterfactuals [17]. Instead of giving static or even calculated explanations, the interactive and user-explored "what-ifs" are more capable of accomplishing this goal.

5. WiXAI Framework Architecture

The **WiXAI** framework inserts itself in between the User-(XAI)-Model dynamic as seen in Figure 7. The box **A** represents the traditional XAI-Model dynamic that the user utilizes, where the user sends their data and receives a prediction with a static XAI explanation. This XAI explanation can take the form of feature ranking, the mathematical impact of features on prediction, a set of calculated counterfactuals [8], etc. It often remains static and even depends on training data being available. In the case of **WiXAI** framework, however, the user—**WiXAI**-model interaction is iterative and thus the explanations are dynamic and do not depend on training data being available. The explanations are provided in the form of a dynamic visualization of the N-dimensional model in 2 dimensions, that the users can interact with.

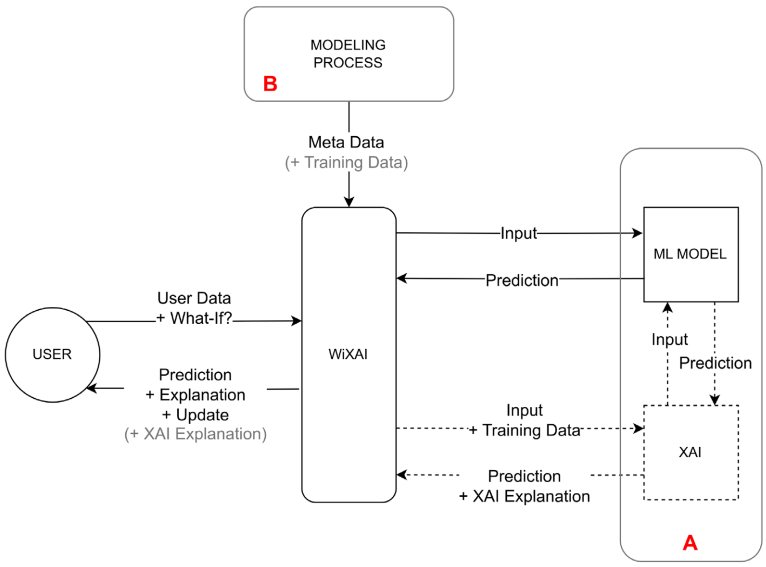


Figure 7. Overview of the WiXAI Framework.

WiXAI Components and Processes

Looking from the outside, **WiXAI** inserts itself into the user-(XAI)-model dynamic and removes any need for the user to be versed in programming. The detailed view, seen in Figure 8, reveals the intricate combination of many components it takes to make it possible. The interaction window and the visualization window are the two components that the users interact with primarily, while the rest of the components are hidden from the user's view. The **WiXAI** framework will take in the user input and provide the prediction as well as dynamic, iterative and visual explanations based on the input. These components and the purpose they serve are explained below:

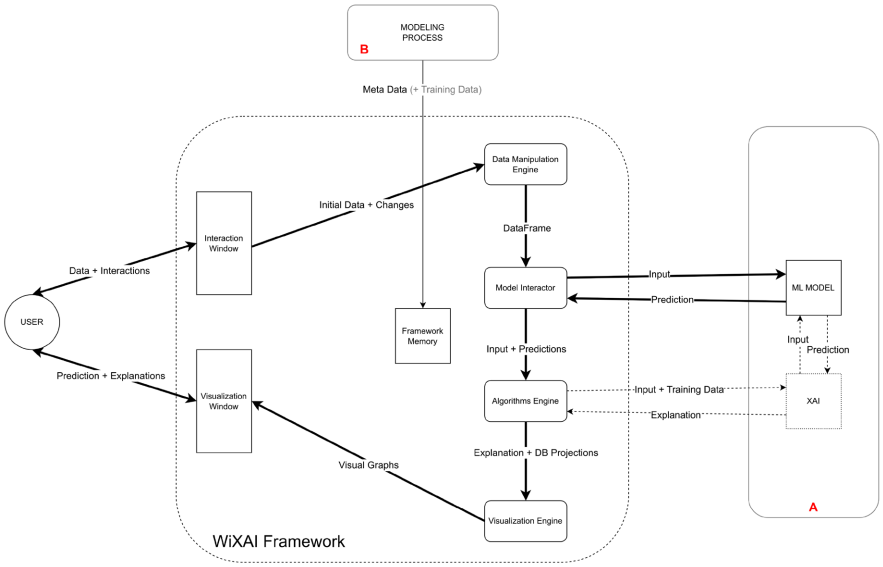


Figure 8. WiXAI Framework Components Sketch.

1) **Interaction Window:** This window is the only way for the users to give input to the framework and thus the ML Model. The inputs are entirely GUI-based and require no knowledge of any programming. This component includes both the initial data input window for the user's initial sample as well as the interaction window (with *sliders* and *drop-down* menus). The users can use these *sliders* and *drop-down* menus to change a value and query a "what-if" scenario. While the initial user sample is only needed once at the start, from that point, the user can make as many small or large changes to the feature values as needed to ask a chain of "what-if" questions. These interactions are done via *sliders* for numerical features and *drop-down* menus for categorical features.

2) **Data Manipulation Engine:** This component deals with creating all the data points (samples) to be tested. From the user's initial input, the main user sample is created based on the metadata stored in Framework Memory. Using the metadata and the user sample, it creates all the data points to generate model projections. The main algorithm for this iterative generation of intervention data points is explained in algorithm 8. All of these iteratively generated "counterfac-

tuals" are converted to a dataframe. The same process is repeated each time the user makes an interaction (in other words makes a new "what-if" inquiry).

3) **Framework Memory:** This component of the **WiXAI** framework stores the important information such as the metadata (with information such as feature names and ranges), the location of the ML model, and other optional information such as the training data itself. The two key information it requires every time is the metadata file, which should contain the feature names, data types and ranges, and the location of the ML Model, so that the Model Interactor can find the ML Model and communicate with it. The optional information of the training data can be used for allowing optional features like testing plausibility, ranking the features based on other XAI methods such as LIME, etc.

4) **Model Interactor:** This component handles all the interactions between the framework and the model. It takes the dataframe generated by and received from the Data Manipulation Engine, sends it to the ML Model and receives the predictions. It makes use of the Framework Memory component to get the location of the ML Model. This is the only component that has any connection with the ML Model, thus very crucial for the framework.

5) **Algorithms Engine:** This is the heart of the **WiXAI** framework which contains all the analysis tools and methods. This component receives the data from the Model Interactor, analyzes them and translates them into visual explanations (in the form of model projections, explained further in section). Using the input and predictions received from the Model Interactor, this component translates the N-dimensional model into its projection on each feature individually. These visual explanations are then sent to the Visualization Engine. If the Framework Memory contains training data as well, it enables additional explanations as well in the form of feature ranking using LIME, analysis of plausibility, etc. These explanations from this component are presented in multiple forms (still expanding), but the two important ones are explained below:

a) **Model Projections:** Model Projections (explained in detail in section) are dependant on the user's sample and simply represent the classification model on a feature level. Looking at each Model Projection, we can find the value (or range of values) in the feature domain where the model gives a certain classification given all other feature values are fixed. At a glance, they help demonstrate how close a sample is to crossing over to the other class.

b) **Ranking Sensitive Features:** Based on the output of the model projections alone, many other analyses can be done and one of them is the ranking of the sensitive features. A feature is said to be more sensitive if the relative change of the feature value required to change the classification is less than that of the other feature. Using the model projections, we can rank the features based on sensitivity, which would make it easier for the users to know which features are likely to produce a different classification, thus giving a higher chance of generating a valid counterfactual.

Using the sample $\mathbf{X}$ and model projections y', Algorithm 2 shows the pseu-

docode for the function Rank_Sensitivity() to output a dictionary of ranked numerical features, categorical features and a list of non-interventionist features. All features are divided between the three lists. If the model projection on a feature doesn't contain the desired class $\hat{y}$, it is considered a non-interventionist feature and appended to that list. For the rest, the features are divided based on datatype (either numerical or categorical). The reason for dividing the numerical and the categorical features is the nature of their values.

Algorithm 2: Ranking Sensitive Features

```
// Input is the sample X and model projections (y')
// y' is the output from Create_Model_Projections()
// y' is a list of lists and ŷ is the desired class
// simply outputs the ordered list of the name of
   features
// based on sensitivity
function Rank_Sensitivity(X, y')
    n_rank = [ ]
    n_sens = [ ] // for numerical features
    c_rank = [ ]
    c_sens = [ ] // for categorical features
    n_i = [ ] // for non-interventionist features
    for i in range(n) do
        if y'_i.contains(ŷ) then
            // for interventionist features
            if type(x_i)==''numerical'' then
                sens_i = 1 - min_distance('c', ŷ) / len(y'_i)
                n_sens.sort_append(sens_i)
                // use the index from n_sens and insert to
                   n_rank
                n_rank.index_append(i)
            else if type(x_i)==''categorical'' then
                sens_i = y'_i.count(ŷ) / (len(y'_i)-1)
                c_sens.sort_append(sens_i)
                // use the index from c_sens and insert to
                   c_rank
                c_rank.index_append(i)
        else
            n_i.append(i) // for non-interventionist features
    output={'numerical':n_rank, 'categorical':c_rank,
     'non_interventionist':n_i}
    return output
```

It is simple to use relative distance from the current sample (**c**) to desired class $\hat{y}$ in model projections for numerical features. However, for categorical features, because there doesn't always exist order among the values, we need a different way to calculate sensitivity. We are defining sensitivity as the probability of reaching the desired class given we change the current feature value. For this, we disregard the current feature value and calculate the probability of reaching the desired class. Because we are assuming we do not have the training data, we will treat each feature value as equally probable. Thus, sensitivity becomes the number of desired classes divided by possible values of the feature-1 (for the

current data point represented by '**c**').

6) **Visualization Engine:** This is the main visualizing engine of the **WiXAI** framework, that receives the explanations from the Algorithms Engine and creates visual graphs. In our implementation, these graphs are created using https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html Pyplot of the Matplotlib library. The model projections are converted into a series of stacked horizontal bar graphs of various colors, where each color represents a different classification of the sample and each bar represents a different feature. An example of these projections can be seen in **Figure 10(a)**.

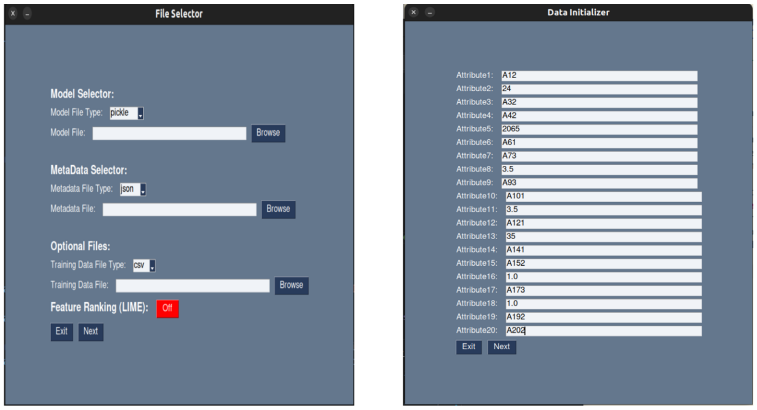
7. **Visualization Window:** This window is the main way the users get visual explanations from the framework. It contains the prediction from the model for their sample but in addition, also contains the projection of the model on each feature for the given sample. This window is dynamic and linked with the interaction window. With each change in the feature values from the user (a "what-if" inquiry), the visualization updates to give the end-user new projections. This window receives the graph from the Visualization Engine and simply displays it for the user to interpret.

6. "What-If" Experiments with WiXAI Framework

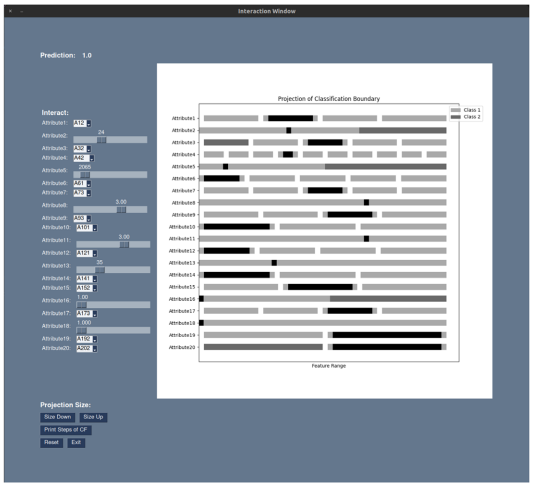
For this paper, we test our framework using the standard German Credit Dataset [45], which contains 20 features and a binary class output. We did not change the names of the features (they are named Attribute1, Attribute2, ..., Attribute20) or the categorical values to keep the look consistent with the original dataset. The features represent status of existing checking account, credit amount, age, purpose, etc. among others. The users can be provided with supplementary information about the features [17] or the names of these values can be changed before training the model, but in this paper, we will refer to them as A_1, A_2, ..., A_{20}. We split the dataset into 70 - 30 for training and testing and saved the entire pipeline. For these experiments, we are using a multi-layer perceptron classifier as the ML method. We also extracted and saved some metadata from the training data such as names, datatype, minimum, and maximum statistics of the features. In the case of categorical features, we simply save the set of possible values for the feature as its range. This is the only information we are considering as a requirement for the **WiXAI** framework to work. As mentioned in section 13, we assume that this metadata is available before using the framework, while the training dataset is optional (and not considered in this paper).

Once the files are generated for testing, we run the framework. In the *file selector window*, we load the model file and the metadata file (seen in **Figure 9(a)**). These files get loaded into the Framework Memory of the **WiXAI** framework. Finally, in the *initialize window*, we personalize the values of the features to create our test sample as seen in **Figure 9(b)**. The selected random sample has a checking account of value between 0 DM and 200 DM, credit duration of 24 months, has paid off past credit, credit request is for furniture/equipment, requested cre-

dit amount is 2065, has less than 100 DM in savings, has current employment for 1 to 3 years, installment amount is 3.5% of disposable income, is a single male, has no other debtors, has been in current residence for 3.5 years, has real estate property, is 35 years old, has other installment plans via a bank, owns current housing, has one existing credit in the bank, is a skilled employee, has one person as liable, has a telephone, and is not a foreign worker. This task normally would be where the users begin their interaction with the **WiXAI** framework, but here we are selecting a random sample (a data point). From there, we (or the end-user) will interact with the framework using *sliders* and *drop-down* options in the *interact window* (seen in Figure 9(c)). Once this window is reached, we will make tweaks to the feature values, track the changes in the projections, and note our findings.



(a) **WiXAI** File Selector Window (b) **WiXAI** Data Initializer Window



(c) **WiXAI** Initial Interact Window

Figure 9. Initializing Windows in WiXAI.

As can be seen in Figure 9(c), the top left section shows prediction to be class 1 (good credit). The value for each feature is displayed and can be interacted with, in the interact section. And, the classification boundaries are projected in

visualization section on the right. The dark and light colors in each feature are representative of the class corresponding to the value of the feature (dark being class 2 and light class 1). The black segment within the projection represents where the value of the feature currently lies (the '**c**' value from algorithm 8). The interactive *slider* works for numerical features (both integer and floating type), and the *drop-down* menu works for categorical features.

6.1. Sample Is Classified. Let's Explain It!

From **Figure 9(c)**, we can see some features are interventionist features: A_2, A_3, A_5, A_{16}, and A_{20}; while the rest are non-interventionist. Based on a similar concept as interventionist causation [44], these interventionist features can change the predicted class by themselves. While all three: A_2, A_5 and A_{16} are numerical features, the scaled distance from the classification boundary and their current values convey their sensitivity. The order from higher sensitivity to lower is as: A_2 (duration of credit in months) > A_5 (Credit amount) > A_{16} (Number of existing credits at this bank).

We also have two categorical features as interventionist: A_3 (Credit history) and A_{20} (foreign worker). We can order these interventionist features based on sensitivity according to the sensitivity ranking algorithm 2 as $A_{20} > A_3$. Apart from the comparison of the features, having the model projections from the initial sample also gives us an idea of how to find the sparsest (one feature) counterfactuals. For A_3: changing to $A_3[1]$, A_{20}: changing to $A_{20}[1]$, or increasing sufficiently the values of A_2, A_5, and A_{16} all produce "valid" counterfactuals. We also have non-interventionist features, which cannot make enough impact to change the classification of the sample while keeping the rest of the feature values fixed.

6.2. Counterfactuals: Understanding Feature Relations

When we want to generate a counterfactual, we simply have to use the interact section of the framework and we can see two examples shown in **Figure 10**. In **Figure 10(a)**, we increase the value for A_{16} (Number of existing credits at this bank) without crossing the classification boundary. This produces an "invalid" counterfactual, as the predicted class has not changed [19]. However, observing the changes in model projections for other features gives us valuable information. We find more features are interventionist by adding on the previous list features: A_1, A_7, A_9, A_{15} and A_{17}. We also observe the boundaries moving for the existing interventionist features. As discussed in section, this implies that these features are observably related and part of the same cause. Apart from the new interventionist features, we also observe some features persist to have a single class projection. While it is not conclusive to say they are globally not related to A_{16} or the output, at an instance level, they do not impact the output enough to change the classification (even with the change in value for A_{16}).

For our second interaction, we change the value of categorical feature A_{19} (Telephone) from $A_{19}[2]$ to $A_{19}[1]$, as seen in **Figure 10(b)**. Here, while observ-

ing similar information as above about interventionist features and sensitivity ranking, we mainly observe that the impact on model projections of A_2, A_5, and A_{16} are minimal. The only real impact we observe is in the projection of A_7 (Present employment since). We can infer from this observation that A_{19} is mostly independent of other features (from the sample's perspective).

(a) **WiXAI** Interaction 1 (b) **WiXAI** Interaction 2

Figure 10. Interacting with the sample in WiXAI.

The new data point we reached by changing the value of A_{16} and A_{19} is a counterfactual, however, because it doesn't yet change the classification, it is considered an "invalid" one. However, as explained by Wachter *et al.* [18], even this "invalid" counterfactual has provided us with meaningful information. We also observe that the classification boundary for all the interventionist numerical features A_2, A_5 and A_{16} lies on the right side of the current value. This means we can increase one, two or all three of their values to reach the counterfactual. The goal of these experiments is to demonstrate causal learning rather than finding simple counterfactuals so we will make other changes.

6.3. More Counterfactuals: Closer to Causal

For the third and fourth interactions, we selected a non-interventionist feature, A_8 (Installment rate in percentage of disposable income), to demonstrate the useful information exploration on single-class projections can provide. We first change the value of A_8 from 3.0 to 1.0 and observe the explanation seen in **Figure 11(a)**. In the figure, we can observe that the classification boundary moves away from all three numerical features A_2, A_5, and A_{16}. This implies, our data point moved away from the classification boundary. Less value for A_8 is a stronger case for Class 1 and the *vice versa* should be the case as well. To test that, we increase the value of A_8 from 1.0 to 3.5 (higher than the original). In **Figure 11(b)**, we observe the changes, which verify our argument. The decision boundaries for all the interventionist numerical features get closer to the current value, and we observe increased Class 2 projections on categorical features: A_3, A_4, A_7, A_9, A_{15} and A_{17}.

(a) **WiXAI** Interaction 3 (b) **WiXAI** Interaction 4

Figure 11. Interacting with the sample in WiXAI Part 2.

Through the observations like above, we (as the end-user) can find feature-feature relations (like relations between A_8, A_2, A_3, A_4, A_5, A_7, etc.) and also the relation of features with the output (example: lower value of A_8 is a stronger case for Class 1 and *vice versa*). In this manner, the users are able to query their desired "what-if" scenarios and with the help of counterfactuals and visual explanations, move towards causal causal understanding.

In **Figure 12(a)**, we change the value of A_{19} back to $A_{19}[2]$ and reaffirm that A_{19} is mostly independent of other features as we do not observe any significant changes in the model projections. Finally, we then change the value of A_{16} from 2.0 to 3.0 and cross over to the other side of the classification boundary and find a "valid" counterfactual (seen in **Figure 12(b)**). We also observe that the black segments in the model projections representing the current values all lie in Class 2 now. This will always be the case as the position of a sample in the N-dimensional feature space is fixed. It can only have one classification and all its feature values have to be in the same class as the sample.

(a) **WiXAI** Data Point 2 (b) **WiXAI** Interaction 4

Figure 12. Interacting with the sample in WiXAI Part 3.

Observing the final model projections in **Figure 12(b)**, we can also see that the interventionist numerical features are A_2, A_8, A_{16} and A_{18}. For A_2, A_8 and A_{16},

the classification boundary lies on the left side of the current value but for A_{18}, it lies on the right. This implies that to change the classification to Class 1 (good credit), we can decrease the value of A_2, decrease the value of A_8, decrease the value of A_{16}, or increase the value of A_{18}, or a combination of the four.

Experimenting with a single sample and trying out "what-if" scenarios on only A_8, A_{16}, and A_{19}, we were able to make a lot of observations. At one point or the other, all features other than A_6, A_{10},..., A_{14}, A_{18} and A_{19} became interventionist. Among these non-interventionist features, we changed the value of A_{19} and observed minimal change in model projections making it practically independent of the other features. We were also able to show that features like A_2, A_5, and A_{16} had a direct relation while A_{18} had an inverse relation between them. We also found some feature-output relations like: increasing A_2, A_5, and A_{16} moves data point towards Class 2; for A_{19}, $A_{19}[1]$ moves closer to class 2 than $A_{19}[2]$ and decreasing A_8 moves further from class 2. Among the 6 counterfactuals we observed, only one of them changed the predicted class, however, we can see how informative were the "what-if" inquiries and the visual explanations.

7. Discussion

We laid out how the **WiXAI** framework allows the users to understand the model and its relation with their sample using visualization of the model using their projections. Users are then able to interact with the sample and explore their desired "what-if" scenarios, generating numerous counterfactuals and gaining more information. Building on the work of Pearl [43], Woodward [44] and Baron [34], we showed how this method of explanation can allow the users (including both domain-experts and decision-subjects) to move towards causal understanding of the system iteratively. Understanding the prediction from the model, helps the users have more trust and confidence in the system, helping the adoption of the ML system.

We also explained 5 requirements for an effective XAI framework in section. Our proposed **WiXAI** framework meets these criteria as it is completely GUI-based, provides visual explanations in the form of "what-if" explorations, and is interactive and iterative. In this paper, we also argue its effectiveness without any training data. We also presented how the exploration of "what-if" scenarios, paired with the visualization of the model, can move the users toward a causal understanding of the system. The counterfactuals generated in the process also meet the qualities of effective counterfactual explanation (explained by Guidotti [38]), as explained in **Table 2**, except for plausibility and preserving causal graphs. This limitation is because we are assuming the training data isn't available and accounting for these without that is not possible. Another property, discriminative, cannot be tested because of its subjective nature.

The **WiXAI** framework also allows the users to understand the feature-feature and feature-output relation (as discussed in section) of the system. It allows the users to discover inter-dependent features that are parts of the same cause, fea-

tures that are part of causes independent of others, and features that are non-causes for the output. Finding all these relations globally requires analysis of a large number of counterfactuals, but at an instance level, **WiXAI** can provide these relations for features that are most relevant for the user.

Our framework also satisfies the requirements laid out by Chou *et al.* [17] for a system to move towards causability and they are explained in the Table 3.

Most counterfactual explanation methods focus on solving the optimization problem while trying to find a balance between the loss functions for proximity, sparsity, actionability, diversity, etc. [19]. These methods generate a fixed number of counterfactuals that are calculated to be the most "appropriate" for the

Table 2. Properties of effective counterfactual explanation method.

CFE Property	Explanation
Proximity and Validity	As the framework shows classification boundaries, finding the closest boundary and crossing it allows the counterfactual to be valid and close to the original sample.
Sparsity	As the users can change one feature value at a time and the model is projected at one feature level, the **WiXAI** framework promotes changes to fewer features.
Plausibility	As we are assuming training data to be optional, plausibility isn't always possible to check.
Discriminative	As Guidotti [38] explains, this property is subjective and not determined in this paper.
Actionability	Counterfactuals are generated completely by the user's perturbation to the existing sample, capturing user preference and thus actionable.
Causality	Just like with plausibility, not always having training data means a lack of any existing causal graphs from the training data.

Table 3. Requirements for promoting causability according to Chou *et al.* [17].

Requirements	Explanation
Requirement 1: the framework needs to be grounded on a formal theory of causality	Our framework is based on the interventionist causation and the system laid by Pearl, Woodward and Baron [34] [43] [44] satisfying the first requirement. We utilize interventionist features (rather than causation), to allow the users to discover causal relations in the system.
Requirement 2: explanations need to be in the form of counterfactuals	Each interaction generates a counterfactual, as well as the visualization of the model and provides explanations in the form of counterfactuals. The entire explanation framework for **WiXAI** is counterfactual-centric.
Requirement 3: explanations need to be human-centric	Since the framework allows the user's "what-if" explorations for generating counterfactuals, the whole **WiXAI** framework is human-centric. It also explains the decision at a local level for the specific user's sample as well as some broader causal relations.
Requirement 4: users should be able to interact with generated explanations	The entire **WiXAI** framework is based on the principle of interaction between the end-user and the system and keeping them in the loop. It is not only incidental but rather carefully designed to provide more control to the end users to promote causal learning.
Requirement 5: XAI systems need to be complemented by contextual and domain-specific knowledge	As mentioned above, the end-users would normally have information about the features as they pertain to them. Otherwise, it is very easy to supplement the framework with domain knowledge and knowledge about the features, what they represent, their ranges, and other information (some of which would also be in the metadata).

user. However, allowing the users to explore any of their desired "what-if" scenarios, with the aid of visual explanations, helps them understand the ML system. It allows them to understand the cause-effect relations better [39], and thus have more confidence in the prediction as well as the explanation. Even the "invalid" counterfactuals that are generated in the process provide valuable insight into how the features and output are related [18].

8. Conclusions

In this paper, we introduced Model Projections, a method of visualizing an N-dimensional classification model in 2 dimensions from the point-of-view of the user's sample. Pairing this with the GUI-based interactive and iterative "what-if" exploration, the **WiXAI** framework allows the users to understand the model prediction better. We demonstrated how the visualization of the model projections helps inform the users in their exploration as it all leads towards a better understanding of the cause-effect relation. Unlike counterfactuals generated via optimization, user-explored counterfactuals generate a large number of "invalid" counterfactuals as well, which provide useful information as discussed in section.

We also presented the requirements laid out for a system that can move towards causal understanding by Chou *et al.* [17], as well as the properties of effective CFE summarized by Guidotti [38]. We argued how the counterfactuals generated via **WiXAI** framework can help the users move towards a more causal understanding of the system. This framework should be useful for both ML experts to inspect the classification boundaries of the model as well as the end-users (both domain-experts and decision-subjects) to have more confidence in the system as a whole.

Future work on the tool can improve upon the explanation when we assume the training data is available. The **WiXAI** framework can also be expanded onto regression problems and more types of data such as text data or images. Making all the framework processes more efficient can also be looked at to deal with very massive models and very high dimensions in data. Instead of freshly calculating the classification from the ML Model **M** for each interaction, it is possible to store the past explanations into a database and decrease the number of **WiXAI**-Model communication. It is also useful to study the cost-benefit of communicating with the Model vs storing past explanations and extracting them in some cases.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Georgiev, P., Bhattacharya, S., Lane, N.D. and Mascolo, C. (2017) Low-Resource

Multi-Task Audio Sensing for Mobile and Embedded Devices via Shared Deep Neural Network Representations. *Proceedings of the ACM on Interactive*, *Mobile*, *Wearable and Ubiquitous Technologies*, **1**, Article No. 50. https://doi.org/10.1145/3131895

[2] Noreen, U., Shafique, A., Ahmed, Z. and Ashfaq, M. (2023) Banking 4.0: Artificial Intelligence (AI) in Banking Industry & Consumer's Perspective. *Sustainability*, **15**, Article 3682. https://doi.org/10.3390/su15043682

[3] Sarker, I.H., Furhad, M.H. and Nowrozy, R. (2021) AI-Driven Cybersecurity: An Overview, Security Intelligence Modeling and Research Directions. *SN Computer Science*, **2**, Article No. 173. https://doi.org/10.1007/s42979-021-00557-0

[4] Pisoni, G. and Díaz-Rodríguez, N. (2023) Responsible and Human Centric AI-Based Insurance Advisors. *Information Processing & Management*, **60**, Article 103273. https://doi.org/10.1016/j.ipm.2023.103273

[5] Goswami, G., Bhardwaj, R., Singh, R. and Vatsa, M. (2014). MDLface: Memorability Augmented Deep Learning for Video Face Recognition. *IEEE International Joint Conference on Biometrics*, Clearwater, 29 September -2 October 2014, 1-7. https://doi.org/10.1109/btas.2014.6996299

[6] Sarker, I.H. (2021) Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, 2, Article No. 160. https://doi.org/10.1007/s42979-021-00592-x

[7] Samek, W., Montavon, G., Lapuschkin, S., Anders, C.J. and Muller, K. (2021) Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications. *Proceedings of the IEEE*, **109**, 247-278. https://doi.org/10.1109/jproc.2021.3060483

[8] Ali, S., Abuhmed, T., El-Sappagh, S., Muhammad, K., Alonso-Moral, J.M., Confalonieri, R., *et al.* (2023) Explainable Artificial Intelligence (XAI): What We Know and What Is Left to Attain Trustworthy Artificial Intelligence. *Information Fusion*, **99**, Article 101805. https://doi.org/10.1016/j.inffus.2023.101805

[9] Roese, N.J. (1997) Counterfactual Thinking. *Psychological Bulletin*, **121**, 133-148. https://doi.org/10.1037//0033-2909.121.1.133

[10] Cheng, F., Ming, Y. and Qu, H. (2021) DECE: Decision Explorer with Counterfactual Explanations for Machine Learning Models. *IEEE Transactions on Visualization and Computer Graphics*, **27**, 1438-1447. https://doi.org/10.1109/tvcg.2020.3030342

[11] Pearl, J. (2019) The Seven Tools of Causal Inference, with Reflections on Machine Learning. *Communications of the ACM*, **62**, 54-60. https://doi.org/10.1145/3241036

[12] Plumb, G., Molitor, D. and Talwalkar, A.S. (2018) Model Agnostic Supervised Local Explanations. *Proceedings of the* 32*nd International Conference on Neural Information Processing Systems*, Montréal, 3-8 December 2018, 2520-2529.

[13] Dhurandhar, A., Chen, P.-Y., Luss, R., Tu, C.-C., Ting, P., Shanmugam, K.-K. and Das, P. (2018) Explanations Based on the Missing: Towards Contrastive Explanations with Pertinent Negatives. *Proceedings of the* 32*nd International Conference on Neural Information Processing Systems*, Montréal, 3-8 December 2018, 590-601.

[14] Mothilal, R.K., Sharma, A. and Tan, C. (2020) Explaining Machine Learning Classifiers through Diverse Counterfactual Explanations. *Proceedings of the* 2020 *Conference on Fairness*, *Accountability*, *and Transparency*, Barcelona, 27-30 January 2020, 607-617. https://doi.org/10.1145/3351095.3372850

[15] Poyiadzi, R., Sokol, K., Santos-Rodriguez, R., De Bie, T. and Flach, P. (2020) FACE: Feasible and Actionable Counterfactual Explanations. *Proceedings of the AAAI/*

ACM Conference on AI, Ethics, and Society, New York, 7-9 February 2020, 344-350. https://doi.org/10.1145/3375627.3375850

[16] Albini, E., Rago, A., Baroni, P. and Toni, F. (2020) Relation-Based Counterfactual Explanations for Bayesian Network Classifiers. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 451-457. https://doi.org/10.24963/ijcai.2020/63

[17] Chou, Y.-L., Moreira, C., Bruza, P., Ouyang, C. and Jorge, J.A. (2021) Counterfactuals and Causability in Explainable Artificial Intelligence: Theory, Algorithms, and Applications. *Information Fusion*, **81**, 59-83. https://doi.org/10.1016/j.inffus.2021.11.003

[18] Wachter, S., Mittelstadt, B. and Russell, C. (2017) Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. *Harvard Journal of Law & Technology*, **31**, 842-887.

[19] Verma, S., Boonsanong, V., Hoang, M., Hines, K.E., Dickerson, J.P. and Shah, C. (2020) Counterfactual Explanations and Algorithmic Recourses for Machine Learning: A Review. arXiv: 2010.10596. https://doi.org/10.48550/arXiv.2010.10596

[20] White, A. and d'Avila Garcez, A. (2019) Measurable Counterfactual Local Explanations for Any Classifier. arXiv: 1908.03020. https://doi.org/10.48550/arXiv.1908.03020

[21] Galhotra, S., Pradhan, R. and Salimi, B. (2021) Explaining Black-Box Algorithms Using Probabilistic Contrastive Counterfactuals. *Proceedings of the* 2021 *International Conference on Management of Data*, Virtual Event China, 20-25 June 2021, 577-590. https://doi.org/10.1145/3448016.3458455

[22] Goyal, Y., Wu, Z., Ernst, J., Batra, D., Parikh, D. and Lee, S. (2019) Counterfactual Visual Explanations. *Proceedings of the* 36*th International Conference on Machine Learning*, May 2019, 2376-2384.

[23] Akula, A.R., Wang, K., Liu, C., Saba-Sadiya, S., Lu, H., Todorovic, S., *et al.* (2022) CX-toM: Counterfactual Explanations with Theory-of-Mind for Enhancing Human Trust in Image Recognition Models. *iScience*, **25**, Article 103581. https://doi.org/10.1016/j.isci.2021.103581

[24] Mertes, S., Huber, T., Weitz, K., Heimerl, A. and André, E. (2022) GANterfactual—Counterfactual Explanations for Medical Non-Experts Using Generative Adversarial Learning. *Frontiers in Artificial Intelligence*, **5**, Article 825565. https://doi.org/10.3389/frai.2022.825565

[25] Kenny, E.M. and Keane, M.T. (2021) On Generating Plausible Counterfactual and Semi-Factual Explanations for Deep Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, **35**, 11575-11585. https://doi.org/10.1609/aaai.v35i13.17377

[26] Thiagarajan, J.J., Thopalli, K., Rajan, D. and Turaga, P. (2022) Training Calibration-Based Counterfactual Explainers for Deep Learning Models in Medical Image Analysis. *Scientific Reports*, **12**, Article No. 597. https://doi.org/10.1038/s41598-021-04529-5

[27] Duong, T.D., Li, Q. and Xu, G. (2023) CeFlow: A Robust and Efficient Counterfactual Explanation Framework for Tabular Data Using Normalizing Flows. *Advances in Knowledge Discovery and Data Mining*, Osaka, 25-28 May 2023, 133-144. https://doi.org/10.1007/978-3-031-33377-4_11

[28] Krause, J., Perer, A. and Ng, K. (2016) Interacting with Predictions. *Proceedings of the* 2016 *CHI Conference on Human Factors in Computing Systems*, San Jose, 7-12 May 2016, 5686-5697. https://doi.org/10.1145/2858036.2858529

[29] Lash, M.T., Lin, Q., Street, N., Robinson, J.G. and Ohlmann, J. (2017) Generalized Inverse Classification. *Proceedings of the* 2017 *SIAM International Conference on Data Mining*, 162-170. https://doi.org/10.1137/1.9781611974973.19

[30] Laugel, T., Lesot, M.-J., Marsala, C., Renard, X. and Detyniecki, M. (2018) Comparison-Based Inverse Classification for Interpretability in Machine Learning. *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations*, Cádiz, 11-15 June 2018, 100-111. https://doi.org/10.1007/978-3-319-91473-2_9

[31] Kanamori, K., Takagi, T., Kobayashi, K. and Arimura, H. (2021) Distribution-Aware Counterfactual Explanation by Mixed-Integer Linear Optimization. *Transactions of the Japanese Society for Artificial Intelligence*, **36**, 1-12. https://doi.org/10.1527/tjsai.36-6_c-l44

[32] Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F. and Giannotti, F. (2018) Local Rule-Based Explanations of Black Box Decision Systems. arXiv: 1805.10820. https://doi.org/10.48550/arXiv.1805.10820

[33] Russell, C. (2019) Efficient Search for Diverse Coherent Explanations. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, Atlanta, 29-31 January 2019, 20-28. https://doi.org/10.1145/3287560.3287569

[34] Baron, S. (2023) Explainable AI and Causal Understanding: Counterfactual Approaches Considered. *Minds and Machines*, **33**, 347-377. https://doi.org/10.1007/s11023-023-09637-x

[35] Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viegas, F. and Wilson, J. (2019) The What-If Tool: Interactive Probing of Machine Learning Models. *IEEE Transactions on Visualization and Computer Graphics*, 26, 56-65. https://doi.org/10.1109/tvcg.2019.2934619

[36] Tenney, I., Wexler, J., Bastings, J., Bolukbasi, T., Coenen, A., Gehrmann, S., *et al.* (2020). The Language Interpretability Tool: Extensible, Interactive Visualizations and Analysis for NLP Models. *Proceedings of the* 2020 *Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 107-118. https://doi.org/10.18653/v1/2020.emnlp-demos.15

[37] Gathani, S., Hulsebos, M., Gale, J., Haas, P.J. and Demiralp, Ç. (2022) Augmenting Decision Making via Interactive What-If Analysis. arXiv: 2109.06160. https://doi.org/10.48550/arXiv.2109.06160

[38] Guidotti, R. (2022) Counterfactual Explanations and How to Find Them: Literature Review and Benchmarking. *Data Mining and Knowledge Discovery*. https://doi.org/10.1007/s10618-022-00831-6

[39] Byrne, R.M.J. (2019) Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 6276-6282. https://doi.org/10.24963/ijcai.2019/876

[40] Mahajan, D., Tan, C. and Sharma, A. (2019) Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers. ar-Xiv: 1912.03277. https://doi.org/10.48550/arXiv.1912.03277

[41] Crupi, R., San Miguel González, B., Castelnovo, A. and Regoli, D. (2022) Leveraging Causal Relations to Provide Counterfactual Explanations and Feasible Recommendations to End Users. *Proceedings of the* 14*th International Conference on Agents and Artificial Intelligence*, Location, Date, 24-32. https://doi.org/10.5220/0010761500003116

[42] Paul, S.K., Firdausi, T.J., Jana, S., Das, A. and Nandi, P. (2021) Counterfactual Causal

Analysis on Structured Data. In: Reddy, V.S., Prasad, V.K., Wang, J. and Reddy, K., Eds., *International Conference on Soft Computing and Signal Processing*, Springer, 187-195. https://doi.org/10.1007/978-981-16-7088-6_16

[43] Pearl, J. (2009) Causality. 2nd Edition, Cambridge University Press. https://doi.org/10.1017/cbo9780511803161

[44] Woodward, J. (2005) Making Things Happen: A Theory of Causal Explanation (Oxford Studies in Philosophy of Science). Oxford University Press.

[45] Hofmann, H. (1994) Statlog (German Credit Data). UCI Machine Learning Repository.

[46] Wang, P.Y., Galhotra, S., Pradhan, R. and Salimi, B. (2021) Demonstration of Generating Explanations for Black-Box Algorithms Using Lewis. *Proceedings of the VLDB Endowment*, **14**, 2787-2790. https://doi.org/10.14778/3476311.3476345

[47] Ming, Y., Qu, H. and Bertini, E. (2019) RuleMatrix: Visualizing and Understanding Classifiers with Rules. *IEEE Transactions on Visualization and Computer Graphics*, **25**, 342-352. https://doi.org/10.1109/tvcg.2018.2864812