# User Driven Model Adjustment via Boolean Rule Explanations

**Elizabeth M. Daly, Massimiliano Mattetti, Öznur Alkan, Rahul Nair**
IBM Research
Dublin, Ireland

## Abstract

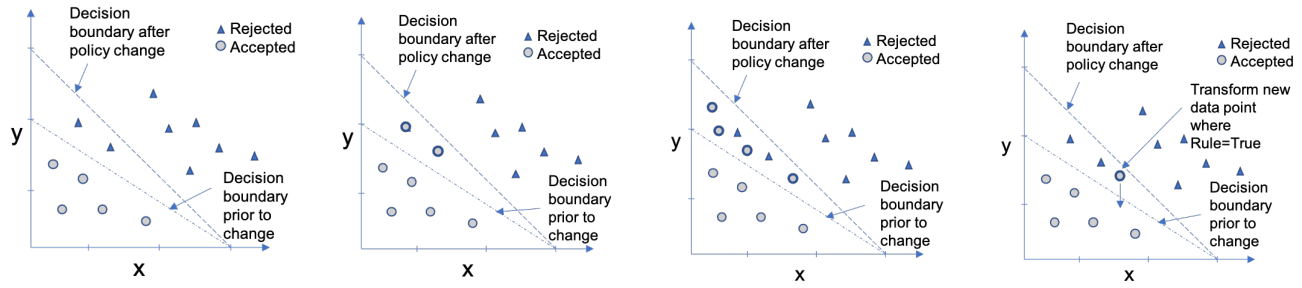
AI solutions are heavily dependant on the quality and accuracy of the input training data, however the training data may not always fully reflect the most up-to-date policy landscape or may be missing business logic. The advances in explainability have opened the possibility of allowing users to interact with interpretable explanations of ML predictions in order to inject modifications or constraints that more accurately reflect current realities of the system. In this paper, we present a solution which leverages the predictive power of ML models while allowing the user to specify modifications to decision boundaries. Our interactive overlay approach achieves this goal without requiring model retraining, making it appropriate for systems that need to apply instant changes to their decision making. We demonstrate that user feedback rules can be layered with the ML predictions to provide immediate changes which in turn supports learning with less data.

## Introduction

AI is increasingly integral in many real world tasks from loan approval to forecasting organizational revenue. While supervised ML tasks such as classification perform well with appropriate historical data, they can not be updated quickly to support more dynamic situations. Retraining can be costly, and more challenging still is obtaining labeled data that accurately reflects the current decision landscape. Consider a loans approval application where there exists a business policy that a loan request should be accepted for any user with $age > 30 \wedge income > 50k \wedge education =$ "$Masters$" and a machine learning model has been trained based on historical data to predict if a user will be approved or not by leveraging many other features other than the three mentioned in the business logic. Now consider a new policy comes into place where the rule changes from age>30 to age>26. Based on historical data the machine learning model may incorrectly reject users between the ages 26-30. Options include relabelling historical data in order to accurately reflect the new decision boundary which is labour intensive. Alternatively, the solution must wait for new data to be collected with the correct labels. In both cases, the model must be retrained after the data has been updated. These approaches can result in a lack of data for some instances' coverage and may be time consuming making it less appropriate for applications where decision processes or policies may change.

In this paper, we investigate the scenario where the data used to train an existing ML model may not reflect the current decision criteria. Additionally, the data may be lacking external knowledge such as policies or policy changes. In order to tackle this challenge, we present a solution that leverages the predictive power of an ML model with user contributed decision rules to enable adjustments online acting as a "patch" or "bandaid" taking immediate affect. Figure 1 illustrates how current ML solutions cope with changing decision boundaries in a simplified linear scenario: the solution is either to relabel existing data, or wait until sufficient new data is available. Active learning has shown promise in reducing the amount of newly labelled data needed, however model retraining is still required and in practise users may need to label many data instances in order to impact the model, which can prove frustrating for the user (Cakmak, Chao, and Thomaz 2010; Guillory and Bilmes 2011). In an online scenario where policies need to have an immediate effect, for example in a spam detection setting, waiting for relabelling and retraining to update the model may not be possible. Our solution attempts to harness the underlying ML model by storing a series of user adjustments in the form of decision rules that can be applied to push future instances to the appropriate decision boundary in order to reflect the user constraints. We present this solution as a user modifiable layer that can support immediate changes and influence over an existing model. Our goal is to use an existing ML model, while permitting users to provide rule based modifications that adjust the final decision making criteria allowing them to adjust the predictions for specified parameters. This is achieved without modifying or updating the underlying ML model but by post-processing the model output through a combination of interpretable Boolean rules and inferred transformation functions which modify the input request to reflect the user feedback.

One question might arise: why not either use a complete rule based system or simply create a post processing filter-




(a) Problem

(b) ML Approach: Relabel and train

(c) ML Approach: Collect new data and train

(d) Interactive Overlay Approach

Figure 1: Approaches to learning new decision boundaries

ing layer. However, a user may not be able to specify and define the entire decision criteria, whereas they can easily contribute corrections or updates that are reflective of a number of key variables. Consider our example for a loan scenario where many different features are factors in the underlying prediction and the only change a user wants to make is to adjust the age requirement from 30 to 26. For a rule system to achieve the same level of accuracy as an ML model the rules may need to become increasingly complex which can hurt interpretability (Lakkaraju, Bach, and Leskovec 2016). In the case of a post processing filter, the solution would not necessarily know that the primary contributing factor for the label was indeed the age and therefore would need to do some post analysis to determine the appropriate label.

To tackle these challenges, the rule-based solution we propose does not try to capture all rules and features but makes a trade off between compact interpretable rules that are as accurate as possible while the underlying decision making is still governed by the ML model. As a result, only minor adjustment on known features or thresholds are needed and understanding the entire feature space or increasingly complex rules is not needed. Our solution provides a mechanism to support post analysis in a generalizable way, by storing user post filtering rules and then adjusting the input instance to determine if the variable the user wishes to adjust is the contributing factor. Additionally, it is important to note that in many circumstances we do not expect the interactive overlay approach to outperform an ML model with access to sufficient correctly labeled data.

In this paper, we aim to address the following research questions: R1) Can we create an explainable interactive overlay that supports modifications to an existing ML model decision process without retraining the model? R2) Can our interactive overlay system with only partial knowledge converge to the same performance as an ML model with full knowledge through rule based modifications? The rest of the paper is organized as follows. We first review related work and then present our solution for supporting modifications to an existing ML model through user feedback. We evaluate our framework with benchmark datasets and finally, we conclude the paper highlighting future lines of research.

## Related Work

Explainability has become an increasing area of interest in the AI community given the number of high stake scenarios where machine learning is employed. The goal of explainable AI is to make the decision making process understandable to a user (Gunning 2017). Many different approaches have been proposed, some of which involve new predictive algorithms where explainability is built in, and others focus on post-hoc explanations agnostic to the underlying algorithm. These techniques include calculating feature importance (Guidotti et al. 2018; Ribeiro, Singh, and Guestrin 2016), finding similar instances from past predictions (Gurumoorthy et al. 2019), identifying what features are present or missing to support the prediction known as contrastive explanations (Dhurandhar et al. 2018) or generating interpretable rule based representations (Dash, Gunluk, and Wei 2018; Ribeiro, Singh, and Guestrin 2018). Yousefzadeh and O'Leary present a mechanism for finding 'flip' points which provides the smallest change to a single continuous feature that would induce a change to the output of the model (Yousefzadeh and O'Leary 2020).

While improvements in explainability and interpretability may assist in allowing the user to trust an AI system, it does not allow the user to correct errors or add domain logic. In reality, many ML solutions deployed in applications require some level of business logic which is either crafted by the data scientist into the solution through data selection or as a set of post processing logic. The need for user control and interpretability has garnered a resurgence in rule based models (Lakkaraju, Bach, and Leskovec 2016; Dash, Gunluk, and Wei 2018). Popordanoska et al. present a soltuion which generates a rule-based system from data that the user may modify which is then used as a rule-based executable model (Popordanoska, Kumar, and Teso 2020). While rule based models have the advantage of being interpretable, in order to achieve coverage the model must add rules to cover increasingly narrow slices which can in turn negatively impact interpretability. Our work is complementary as the main focus is on a user editing rules whereas our goal is to build a framework to allow such modifications to adjust/patch a trained ML model where adjustments may require changes to a small subset of rules or clauses and the remaining deci-

sion process is still governed by the ML model.

Fails and Olson were the first to introduce the term Interactive Machine Learning (IML) (Fails and Olsen Jr 2003; Amershi et al. 2014). They presented an Interactive ML framework where the ML model is intentionally trained quickly and the results are presented to the user, allowing the user to give feedback, explore the impact of their changes and then tune their feedback accordingly.

The most common way for a user to influence or instill knowledge in an AI system is to provide labels and active learning has been leveraged to reduce the load on users by intelligently selecting which data instances to present to the user for annotation. However, users may need to label many instances in order to impact the model, leading to labelling fatigue (Cakmak, Chao, and Thomaz 2010; Guillory and Bilmes 2011).

The work most similar to our own combines active learning with explanations and user corrections. Teso et al. presented a framework for Explanatory Interactive Learning (XIL) (Teso and Kersting 2019). Active learning is employed to select a data instance to present to the user along with the prediction and an explanation based on feature importance. If the prediction is incorrect the user is given the opportunity to relabel the data instance. If the prediction is correct but the explanation is incorrect the user can give feedback on the presented LIME values (Ribeiro, Singh, and Guestrin 2016) and may flag a feature as irrelevant. This information is then used to generate counter example instances that are identical except the irrelevant feature is modified either through randomization or some other strategy. These new data instances are then used as input for future retraining to influence the model. Our work is complimentary to this space; however, we leverage the interpretable value of Boolean rules to allow a user to give much more fine-grained feedback. A similar mechanism of taking user feedback which modifies the input training data is presented in (Schramowski et al. 2020), however both solutions require model retraining for the user changes to be reflected in the model. Our solution enables modifications to take immediate effect, without model retraining, through post processing of data instances.

## Preliminaries

To simplify the discussions, we present our solution considering the binary classification problem, however, the ideas could be extended to multi-class classification problems as well. We consider a binary classification problem from a domain D(X,Y), from which $n$ i.i.d. samples are drawn $(x_i, y_i)$, $i \in \{1, ..., n\}$ with labels $y_i \in \{0, 1\}$. Below, we present the important definitions and terminology that we use to describe our solution.

**Rule**. Following the terminology from previous works (Dash, Gunluk, and Wei 2018; Lakkaraju, Bach, and Leskovec 2016), we define a *rule* $R$ as a tuple $R(e, p)$, where $e$ is a boolean *clause*, and $p$ is the class label assigned to all the instances satisfying $e$. A *clause* is a conjunction of conditions over a subset of features in $X$.

**Condition**. A condition is defined as a triple $\langle variable, comparison\ \ operator, value \rangle$ where the variable represents a feature in $X$ and the comparison operator can be one in the set $\{`=', `\neq', `>', `\geq', `<', `\leq'\}$.

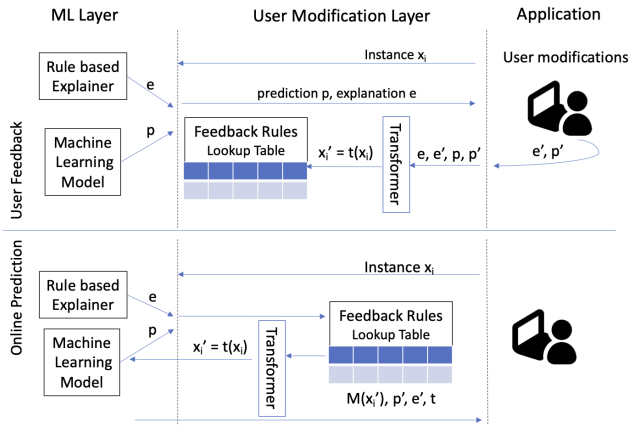



Figure 2: Overview

**Rule Satisfiability**. An instance $x_i$ satisfies a clause $e$ if all the conditions in $e$ are evaluated to $True$ on $x_i$. By extension, we say that an instance $x_i$ satisfies a rule $R(e, p)$ if $x_i$ satisfies the boolean clause $e$ in $R$. Formally, we define a boolean clause as a function $e : X \rightarrow \{0, 1\}$, and an instance $x \in X$ **satisfies** $e \iff e(x) = 1$.

**Feedback Rule**. A *feedback rule* ($FR$) is defined as a triple of the form $FR(R, R', T)$, where $R$ contains the original rule, $R'$ contains the users feedback, which is the modified version of $R$ (with $e'$ and $p'$ indicating the boolean clause and the class label of the modified rule, respectively), and $T$ stores the transformation function.

**Conflicting Rules**. Two rules $R_1(e_1, p_1)$ and $R_2(e_2, p_2)$ are *conflicting* if there exists at least one instance $x_i$ that satisfies $e_1$ and $e_2$ but $p_1 \neq p_2$. There are two types of conflicting rules to be considered. The first is where two rules from the underlying explainer cover the same data instance but link to a different class. For the purposes of this paper we will make the simplified assumption that the rules are *conflict-free*. The second situation where conflicting rules may come into play is where a user modifies a rule which then results in a conflict of other feedback rules. In order to support this scenario a conflict analysis, such as the one presented in (Lindgren 2004), would need to occur to allow the user to understand which other rules would be impacted and potentially need to be updated based on their feedback. For the purposes of this paper however we ensure feedback rules are *conflict-free*.

**Transformation Function**. A transformation function is a function that modifies an input instance $x_i$ which previously satisfied a rule $R$ so as to turn it into an instance $x_i'$ that satisfies a modified rule $R'$.

## Proposed Interactive Overlay

The overall flow of our proposed interactive overlay solution is provided in Figure 2. The overlay includes three layers, namely, *ML Layer, User Modification Layer and Application Layer*. To illustrate the framework the two main pro-

cesses of the solution are also separated out in Figure 2: *processing user feedback* (upper), and *making online predictions* (lower). In order to provide feedback to the system, the user is presented with a response and given the option to make adjustments which are then stored in the Feedback Rules Lookup Table. When making online predictions, the solution generates a response by activating previous user changes through the Feedback Rules Lookup Table in order to influence the current prediction.

The ML Layer contains the ML model and the Explainer. They are assumed to be provided to the interactive overlay. It should be emphasized that our solution is not dependant on any specific ML model. As can be seen from the online prediction phase, once the end-user sends an instance and asks for a response, this request passes directly to the ML Layer. The ML Layer then provides the initial prediction $p$ through calling the ML model along with an interpretable explanations $e$ which is provided by the Explainer [algorithm 1 line 2-3]. Assuming no relevant user feedback is found, the user is then presented with both the prediction and the explanation and given the opportunity to modify $e$ and/or $p$ [algorithm 1 line 13]. These modifications ($e'$ and $p'$) are then stored in the Feedback Rule Lookup Table along with an inferred transformation function $t$. When making an online prediction, the label $p$ is used together with the explanation $e$ for accessing the Feedback Rules Lookup Table [algorithm 1 line 5] and retrieving relevant feedback rules. The input instance $x_i$ is then evaluated against each one of the retrieved feedback rules [algorithm 2 line 2-15]. The instance can either satisfy both, the original explanation rule and the feedback rule, or only one of the two rules [algorithm 2 line 5]. The latter condition occurs when the instance falls in between the decision boundaries of the two rules. If the input instance $x_i$ does not satisfy the feedback rule, $p'$ is set to the other label [algorithm 2 line 6-7]. Finally, if the model prediction $p$ does not match $p'$ the transformation is applied and the new result returned to the user [algorithm 2 line 8-13].

## Explainer and Feedback Rules

Our proposed solution can work with any ML model; however, since users can provide feedback on the explanations themselves, the solution requires both the explanations and feedback to have the same structure. With the increasing need for interpretability there have been a number of works recently that revisit Boolean rule set logic where the rules themselves are the model. This is particularly relevant for high stakes decision making such as in the medical domain or criminal prosecution (Rudin 2019; Angelino et al. 2017; Rudin, Wang, and Coker 2018). These solutions are composed of *if-then* statements that predict the label, and have the advantage that they are completely transparent and interpretable to the user. Drawing from these solutions, we assume that both the explanations and the feedback are in the form of an *if-then statement*, or formally, *Boolean rules*. In order to produce meaningful explanations to support the prediction from an ML model, we leverage the proposed BRCG framework (Dash, Gunluk, and Wei 2018). This framework produces a disjunctive normal form (DNF) representation of a logical formula to predict a class label for an instance, where the class label can be either the ground truth label or the label provided by an ML model. Although BRCG is proposed for binary classification models, it can be generalized to multi-class problems using a *one-vs.-rest* configuration (Bishop 2006). Our solution can use other explainer models, as long as the explanations can be mapped to boolean rules. Given a binary classifier $M(Y|X)$ which predicts Y given X, two sets of rules are generated using BRCG which explain the predictions of the classifier for the two classes. These two rule sets together form the *Explainer Rule Set (ERS)*, which is used by the ML Layer. The feedback given by the users is stored as a *feedback rule* ($FR$). A *feedback rule set (FRS)* is the set of feedback rules that are stored in Feedback Rules LookUp Table, as depicted in Figure 2.

## User Modification Layer

The User Modification Layer is responsible for mapping user feedback into instance transformation functions. The user can modify a clause by adding or removing one or more conditions or modifying existing conditions by changing the comparison operator, the value or both. Given an explainer clause such as $age > 26 \wedge income > 50k \wedge education = \text{“}Masters\text{”}$, examples of possible user modifications are:

**add a condition** $age > 26 \wedge income > 50k \wedge education = \text{“}Masters\text{”} \wedge$ ***occupation = “Sales”***

**delete a condition** ~~***age > 26 ∧***~~ $income > 50k \wedge education = \text{“}Masters\text{”}$

**modify a value** $age > $ **30** $\wedge income > 50k \wedge education = \text{“}Masters\text{”}$

**modify an operator** $age <$ **<** $26 \wedge income > 50k \wedge education = \text{“}Masters\text{”}$

## Instance Transformation

Given a prediction $p$ from a binary classifier $M(Y|X)$ and an explanation clause $e$, there can be 3 possible outcomes (Teso and Kersting 2019): 1) the prediction is correct and the explanation is correct as well (*right for the right reasons*). 2) the prediction is correct but the explanation is not (*right for the wrong reasons*). 3) the prediction is wrong and the explanation is also necessarily wrong (*wrong for the wrong reasons*). Our research focuses on enabling the user to provide feedback in order to correct the model in cases 2 and 3. Whenever the prediction is correct, but the user is not satisfied with the explanation clause, they can modify the clause to correctly reflect the reason behind the prediction (case 2). More formally, given an explanation clause $e$ and a user input correction clause $e'$, we define a function $t : X \rightarrow X$ such that Equation 1 holds.

$$\forall i \in \{1, ..., n\}, e'(x_i) = 1 \iff e(t(x_i)) = 1 \quad (1)$$

Intuitively, the transformation $t$ modifies only the instances that fall between the boundaries defined by $e$ and $e'$. The instances are pushed inside or pulled outside the decision boundaries of the ML model for the target class $p$ depending on whether the user is relaxing or narrowing the original

**Algorithm 1:** GenerateResponse

**Input:** $x_i$: data input instance,
*ERS*: Explainer Rule Set,
*FRS*: Feedback Rule Set
**Output:** ***response***: system response that applies to $x_i$

$response \leftarrow \{\}$
$p \leftarrow QueryMlModel(x_i)$
$explanations \leftarrow explain(x_i, p, ERS)$
**for** *each* $e$ *in* $explanations$ **do**
  $candidate\_fr \leftarrow RetrieveFeedBackRule(e, p, FRS)$
  $response \leftarrow$ ***EvaluateFeedbackRules***(x$_i$, $e$, $p$, $candidate\_fr$)
// no feedback rule matching explanation $e$ and prediction $p$
**if** $response$ *is empty* **then**
  **for** *each* $frs\_entry$ *in* $FRS$ **do**
    $e, candidate\_fr \leftarrow frs\_entry$
    **if** $e$ *not in* $explanations$ **then**
      $response \leftarrow$ ***EvaluateFeedbackRules***(x$_i$, $e$, $p$, $candidate\_fr$)
// no feedback rule matching instance $x_i$
**if** $response$ *is empty* **then**
  $response \leftarrow p, \mathbf{random\_select}(explanations)$
return $response$

**Algorithm 2:** EvaluateFeedbackRules

**Input:** $x_i$: data input instance,
$e$: explanation rule,
$p$: prediction,
$candidate\_fr$: list of feedback rules to search for applicability to $x_i$
**Output:** ***response***: system response that applies to $x_i$

$response \leftarrow \{\}$
**for** *each* $fr$ *in* $candidate\_fr$ **do**
  $R', t \leftarrow fr$
  $e', p' \leftarrow R'$
  **if** $x_i$ *satisfies* $e$ ***or*** $x_i$ *satisfies* $e'$ **then**
    **if** $x_i$ *not satisfies* $e'$ **then**
      $p' \leftarrow GetOtherLabel(p')$
    **if** $p \neq p'$ **then**
      $x'_i \leftarrow t(x_i)$
      $new\_p \leftarrow QueryMlModel(x'_i)$
      $response \leftarrow new\_p, p', e', t.description$
      **if** $new\_p = p'$ **then**
        return $response$
    **else**
      // model is already capturing this rule
      return $response \leftarrow p, p', e'$
return $response$

boundaries defined by $e$. For instance, given as explanation $e$ the clause $age > 26 \wedge income > 50k \wedge education =$ "$Masters$" with a predicted class label $p = 1$, and a user input correction clause $e'$ as $age > \mathbf{30} \wedge income > 50k \wedge education =$ "$Masters$", the function $t$ is defined as if $(age > 26 \wedge age \leq 30)$ then $age = 25$. As the clause $e'$ narrowed the boundaries on the feature age, the transformation is pushing the instances that fall within the interval defined by $e$ ($age > 26$) and $e'$ ($age \leq 30$) outside the decision boundaries of the ML model for class $p = 1$. The *margin* between the new value assigned to the numeric feature and the boundaries defined by $e$ and $e'$ is configurable per feature and by default is set to 1. For a categorical feature the transformation $t$ draws a new value from the feature domain. Given the explanation clause $e$ presented above and a correction clause $e'$ as $age > 26 \wedge income > 50k \wedge education =$ "$\mathbf{Doctorate}$" where the condition on the categorical feature *education* has been changed, the inferred transformation $t$ is described in Algorithm 3.

**Algorithm 3:** Example of a transformation on a categorical feature when the class label is preserved

**Input:** $x_i$: data input instance,
$E$: set representing the domain of feature education
**Output:** $x'_i$: transformed instance

$value \leftarrow x_i.education$
**if** *value* = *"Doctorate"* **then**
  $new\_value \leftarrow$ "$Masters$"
**else if** *value* = *"Masters"* **then**
  $new\_value \leftarrow$ ***random_select***$(E - \{$"$Masters$"$\})$
$x_i.education \leftarrow new\_value$
return $x_i$

The user is also able to correct a model when a prediction is wrong (case 3) by changing the label. Formally, given a clause $e$, a prediction label $p$, a user input correction clause $e'$ and a user label $p'$ where $p' \neq p$, the transformation function $t : X \rightarrow X$ is defined such that Equation 2 holds.

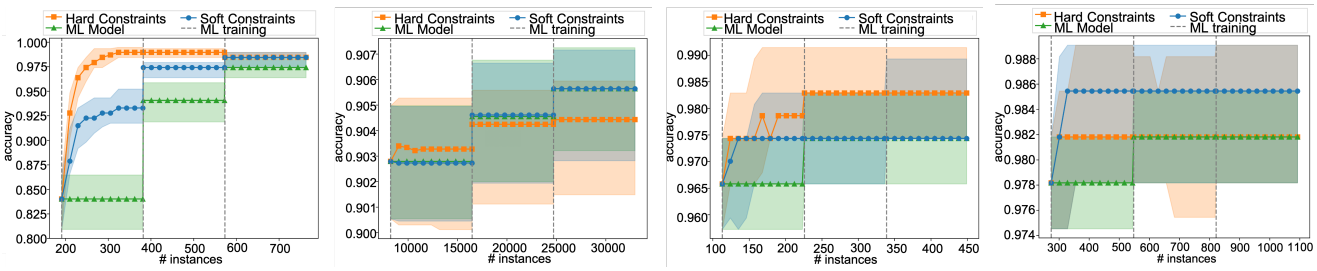
$$\forall i \in \{1, ..., n\}, e'(x_i) = 1 \iff e(t(x_i)) = 0 \quad (2)$$

Intuitively, the transformation $t$ pushes all instances outside the decision boundaries of the ML model for class label $p$. For example, given as explanation $e$ the clause $age > 26 \wedge income > 50k \wedge education =$ "$Masters$", a user correction clause $e'$ as $age > \mathbf{30} \wedge income > 50k \wedge education =$ "$Masters$", and a new predicted class label $p'$ such that $p \neq p'$, the resulting transformation $t$ is shown in Algorithm 4. For pseudo-code documenting each variation of the transformation algorithms please see the supplementary material.

It is important to note that for the purposes of this paper we assume an approximation for the decision boundaries are given by the clauses of the BRCG framework and so the instance transformation uses these decision boundaries to transform the input instance. Other options to consider could be leveraging prior work on detecting 'flip points' for these values (Yousefzadeh and O'Leary 2020).

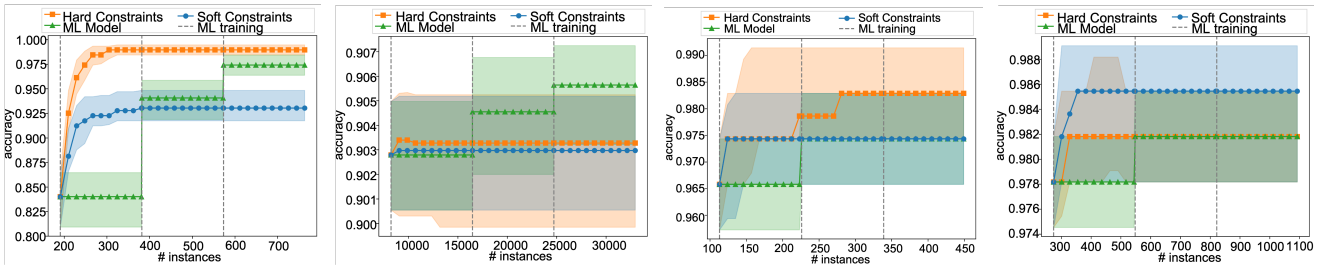
## Response Generation

Figure 2 shows that when an input instance $x_i$ is presented to the system, the prediction label $p$ and the explanation $e$ are




(a) Tic-Tac-Toe (b) Bank-mkt (c) Breast Cancer (d) Banknote

Figure 3: Experiment 1 - Demonstrating Interactive Overlay Approach (Median and 25-75 percentiles)




(a) Tic-Tac-Toe (b) Bank-mkt (c) Breast Cancer (d) Banknote

Figure 4: Experiment 2 - Demonstrating Interactive Overlay Approach with underlying ML model only trained at 20%

**Algorithm 4:** Example of a transformation on a numeric feature when the class label is changed

**Input:** $x_i$: data input instance,
*margin*: margin between the new value and the boundaries
**Output:** $x_i'$: transformed instance

```
value ← x_i.age
if value > 30 then
    new_value ← 26 − margin
else if value ≤ 26 then
    new_value ← 26 + margin
x_i.age ← new_value
return x_i
```

used as a lookup to the Feedback Rules Lookup Table in order to retrieve any Feedback Rules $FR$ [algorithm 1 line 5]. Given our rules are not necessarily non-overlapping, it could be possible that there are multiple explanations for a single instance, in this case we evaluate all explanations. When an instance $x_i$ satisfies at least one of the two rules $e$ and $e'$, and $p'$ differs from the model prediction $p$, then the transformation function $t$ is applied to generate $x_i'$ which is then given as new input to the machine learning model. If the transformation results in a modification of $p$ to $p'$ then the modified result is returned to the user along with the user contributed modification $e'$ and the transformation performed [algorithm 2 line 8-13]. If no user rule results in a modification we return the last seen $FR$ that applied to the instance or whose related explanation $e$ applied to the instance. If no user rules are found at all we return the explanation rule $R$ to the user with the option for the user to provide feedback.

A feedback rule may have decision boundaries that are looser than those defined by the original explanation rule, that is, an input instance that satisfies the feedback rule may not satisfy the original explanation rule. To correctly handle this scenario all remaining feedback rules are evaluated against the instance [algorithm 1 lines 8-11]. One challenge that exists is that rule based explanations may change after model retraining and as a result there may be rules in the Feedback Rule Lookup Table that are no longer returned from the Explainer; however, the system will still need to honour them. The evaluation of all remaining feedback rules accounts for properly handling this scenario as well.

As some domains may require ***hard constraints (hc)*** such as regulatory compliance, we return both the transformed prediction $M(x_i')$ as well as the user input prediction $p'$. If the application requires the constraint to be treated as a hard constraint then $p'$ can be used, if the user feedback is treated as a ***soft constraint (sc)*** then the prediction on the transformed instance is used.

## Experimental Evaluation

In order to address our research questions $R1$ and $R2$ we want to evaluate if we can improve the performance of an underlying machine learning model in between retrains by combining the ML predictions with user contributed modifications to prediction explanations.

## Methodology

To mimic the contributions of a user we employ an oracle based approach where the rule based explainer is trained on 100% of the data, we call this Full Knowledge Rule Set ($FKRS$). This approach is similar to mechanisms used to evaluate active learning approaches (Kulkarni et al. 2018). For the purposes of these experiments the underlying machine learning algorithm used is a logistic regression with 500 iteration limit[1]. Numeric features are pre-processed with StandardScaler and the categorical one with a OneHotEncoder. We provide results for both the *hard constraint* (hc) approach of our solution and the *soft constraint* (sc) approach. Our accuracy measure is the ratio of the number of correct predictions to the total number of samples. We select four well known binary classification benchmarks from the UCI repository[2] TIC-TAC-TOE, BANKNOTE, BANK-MKT and BREAST CANCER.

**Experiment 1:** To evaluate R1 we assume both the overlay solution and the ML algorithm learn on the same data. However, we assume user corrections may be provided to the overlay solution from the oracle $FKRS$. **Train:** The data is divided into 80% for training and 20% for the hold-out test set. The training data is further divided into four slices representing 20% of the data. The ML model is trained on the first 20% and the resulting model is used to train the rule based explainer learnt on the labels provided by the ML model, we refer to the resulting ruleset as the Partial Knowledge Rule Set ($PKRS$). **Online Learning:** The evaluation presents each data instance of the next slice to the overlay solution. Whenever the overlay solution does not find a user feedback rule that applies to a data instance, the explanation clause and prediction label provided by $PKRS$ are compared with those output by the oracle and in the cases where they differ, the oracle output is added as a user feedback rule to the lookup table of the overlay solution. After each slice has been processed the slice is added to the training data and the ML model is retrained. The $PKRS$ is also retrained on the updated ML model. Given we aim to improve performance between model retrains, we calculate the accuracy of the overlay solution every additional $10\%$ of data slice, whereas the ML accuracy is only recomputed after each additional retrain. We repeat the experiment $50$ times, each time using a different shuffle of the whole dataset, therefore producing different slices for training and testing each time. (For the evaluation simulation algorithm please see the Supplemental material).

**Experiment 2:** To evaluate R2 where we need to assess whether the overlay solution can achieve comparable performance when given access to less training data, we repeat experiment 1 above with the exception that the underlying ML model used by the overlay solution is not retrained after the initial training on 20%, whereas the pure ML based solution is retrained on the initial 20% data plus each processed slice.

**Experiment 3:** We compare our solution to an Active Learning approach. The optimal conditions for comparing the approaches is with an ML model whose performance has room for improvement. Hence, for this experiment we trained the model so that its initial performance is relatively low similar to related work (Ghai et al. 2020). We use 20% of the data as hold out set for evaluation and we initially trained the ML model on 2% of the data while the remaining 78% of the data is used as a data pool from which the active learning selector draws the instances to use for retraining. We adopted a low margin strategy (Scheffer, Decomain, and Wrobel 2001) to select the instances to provide to the oracle for labelling. At each iteration a batch of 10 instances is selected from the pool, labelled by the oracle with the ground truth, and then used for retraining the ML model. We repeat these steps 20 times and after each retraining the accuracy of the model is measured against the test set. Our overlay solution is also initialized with a ML model trained on 2% of the data, but this model is never retrained. The batch of instances drawn by the selector are used to test the predictions of the framework and provide corrections in the form of ground truth labels and explanation clauses from $FKRS$. After a batch is processed, the accuracy of the overlay solution is measured against the testing set. As with the previous experiments it is repeated $50$ times each time using a different shuffle of the whole data.

We implement all algorithms in Python using scikit-learn (Pedregosa et al. 2011) and perform the experiments on a cluster of Intel Xeon CPU E5-2683 processors at 2.00GHz with 8 Cores and 64GB of RAM. For the BRCG implementation we use the "Light" version of BRCG (Arya et al. 2019) available in the open source library AIX360[3]. Table 1 shows the details of the dataset used along with the overall accuracy of the BRCG solution in generating boolean rules that reflect the decision processes. As can be seen, the accuracy varies across datasets; however, for the purposes of this paper the goal is not to have a complete rule based system that delivers 100% accuracy but to provide clauses rich enough for a user to contribute modifications and demonstrate that the final predictions reflect those modifications. Table 2 shows an example of DNF rules generated during our experiments, specifically the one that represents the Full Knowledge Rule Set for the Bank Marketing dataset.

| | tic-tac-toe | bank-mkt | b-cancer | banknote |
|---|---|---|---|---|
| # instances | 958 | 45,211 | 569 | 1,372 |
| # features | 9 | 17 | 32 | 5 |
| BRCG Acc | 0.992 | 0.903 | 0.976 | 0.976 |

Table 1: Datasets

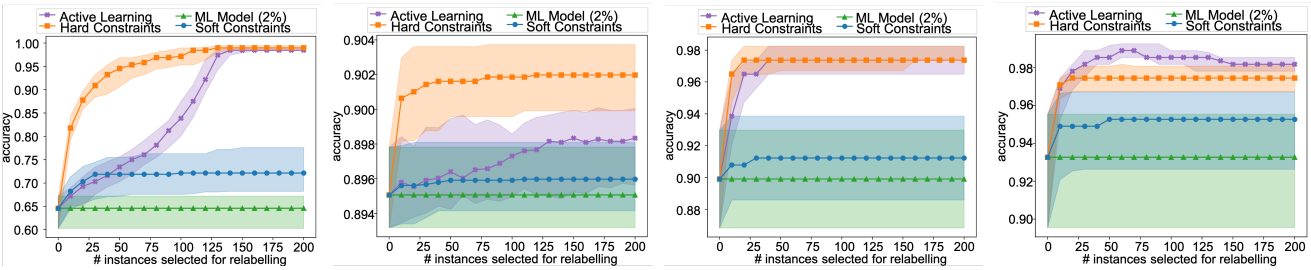
## Results

Figure 3 shows the results for Experiment 1. As can be seen, the SC approach is able to learn user modifications that improve the baseline ML performance between model retrains, meaning we can support online changes without compromising accuracy. In most cases the SC approach performs better

[1]For our 4 test datasets the logistic regression model reaches its max accuracy in less iterations.

[2]https://archive.ics.uci.edu/ml/datasets/

[3]https://github.com/Trusted-AI/AIX360




(a) Tic-Tac-Toe (b) Bank-mkt (c) Breast Cancer (d) Banknote

Figure 5: Experiment 3 - Demonstrating Interactive Overlay Approach vs Active Learning. ML model is trained at 2% and retrained on each batch of 10 instances selected for Active Learning whereas it is never retrained for the Overlay solution

| Class | DNF rule |
|---|---|
| no | $(nr.employed > 5076.20) \vee (poutcome \neq \text{“}success\text{”} \wedge duration \leq 368.00)$ |
| yes | $(duration > 280.00 \wedge emp.var.rate \leq -3.00) \vee (poutcome = \text{“}success\text{”} \wedge nr.employed \leq 5076.20)$ |

Table 2: DNF rules generated with the Light BRCG method for the Bank Marketing dataset. The classification goal is to predict if the client will subscribe (yes/no) a term deposit. According to the UCI data dictionary, *nr.employed* is the number of employees, *poutcome* is the outcome of the previous marketing campaign, *duration* is the last contact duration, in seconds and *emp.var.rate* represents the employment variation rate.

than the HC approach, demonstrating that the rule based approach still benefits from the underlying predictive value of the ML model. Interestingly, for the TIC-TAC-TOE dataset the HC outperforms both the ML and SC approach. This is due to the deterministic nature of the TIC-TAC-TOE domain which is by definition rules based.

Figure 4 shows the results for Experiment 2 where the ML model is retrained after each additional 20% of the data but the ML model leveraged by the SC approach is kept at 20% so any improvement seen is through simulated user feedback rules. The impact is clearly seen on the performance in the TIC-TAC-TOE dataset where the ML model outperforms the SC after the first retrain. For the BANKNOTE data, on the other hand, we see relatively little impact on performance, demonstrating enough information was captured in the first 20% of the data when combined with user modifications.

As can be seen from figure 5, the SC based approach is quickly overtaken by the active learning approach where the model is retrained. This is unsurprising given the ML model leveraged by SC is only trained on 2% of the data and does not benefit from retraining. What is interesting, however, is that the HC approach rapidly outperforms the active learning approach, demonstrating that feedback in rule format can have a higher impact on performance than simple labelling. The quick divergence between SC and HC could potentially be used as a signal to the solution that the underlying ML model is no longer sufficient to capture the current decision processes and trigger a model retrain only when needed.

Our experiments above have demonstrated the ability to support model adjustments to an already trained model. Given the need for user input, our solution most benefits from adjustments to interpretable features. As seen from the tic-tac-toe case, when it is possible to express rules that clearly reflect the underlying decision making process then the overlay can improve solution accuracy. One limitation of soft constraints is seen when the ML model has far too little knowledge and the rule reflected in the model may be too different from the target rule, in which case the hard constraint logic becomes necessary. As a result, we believe this framework is most useful when making adjustments and corrections to existing rules related to interpretable variables e.g. age threshold or adding additional categorical values.

## Conclusion

Given the reliance on AI for important decision making problems, the area of explainability has become a crucial subject of research. Advances in explainability have made it possible to consider opportunities in Interactive Machine Learning (IML) which seek to include the domain experts in the model creation process by allowing them to give feedback, explore the impact and tune their feedback accordingly (Amershi et al. 2014). In this paper we have presented a solution that allows users to provide modifications to interpretable boolean rules. We have demonstrated that these rules can be applied in an online fashion in order to cope with dynamic policy changes. Machine learning models in regulated enterprises, such as finance and healthcare, struggle to encode policy directives because these directives are generally not part of the data fabric. Our approach invites policy experts to encode domain-specific directives and applies appropriate transformations to existing ML decision boundaries at the time they take effect. Once the ML training procedure has caught up with the policy change the decision rule could be promoted from the interaction layer to a constraint in the ML training process (Thomas et al. 2019).

## References


Amershi, S.; Cakmak, M.; Knox, W. B.; and Kulesza, T. 2014. Power to the people: The role of humans in interactive machine learning. *Ai Magazine* 35(4): 105–120.

Angelino, E.; Larus-Stone, N.; Alabi, D.; Seltzer, M.; and Rudin, C. 2017. Certifiably optimal rule lists for categorical data. In *Proceedings of the 23rd ACM SIGKDD Conference of Knowledge, Discovery, and Data Mining (KDD)*.

Arya, V.; Bellamy, R. K.; Chen, P.-Y.; Dhurandhar, A.; Hind, M.; Hoffman, S. C.; Houde, S.; Liao, Q. V.; Luss, R.; Mojsilović, A.; et al. 2019. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv preprint arXiv:1909.03012* .

Bishop, C. M. 2006. *Pattern recognition and machine learning*. springer.

Cakmak, M.; Chao, C.; and Thomaz, A. L. 2010. Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development* 2(2): 108–118.

Dash, S.; Gunluk, O.; and Wei, D. 2018. Boolean decision rules via column generation. In *Advances in Neural Information Processing Systems*, 4655–4665.

Dhurandhar, A.; Chen, P.-Y.; Luss, R.; Tu, C.-C.; Ting, P.; Shanmugam, K.; and Das, P. 2018. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. In *Advances in Neural Information Processing Systems*, 592–603.

Fails, J. A.; and Olsen Jr, D. R. 2003. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, 39–45.

Ghai, B.; Liao, Q. V.; Zhang, Y.; Bellamy, R.; and Mueller, K. 2020. Explainable Active Learning (XAL): An Empirical Study of How Local Explanations Impact Annotator Experience. *arXiv preprint arXiv:2001.09219* .

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2018. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)* 51(5): 1–42.

Guillory, A.; and Bilmes, J. A. 2011. Simultaneous learning and covering with adversarial noise. In *ICML*.

Gunning, D. 2017. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web* 2: 2.

Gurumoorthy, K. S.; Dhurandhar, A.; Cecchi, G.; and Aggarwal, C. 2019. Efficient Data Representation by Selecting Prototypes with Importance Weights. In *2019 IEEE International Conference on Data Mining (ICDM)*, 260–269. IEEE.

Kulkarni, A.; Uppalapati, N. R.; Singh, P.; and Ramakrishnan, G. 2018. An Interactive Multi-Label Consensus Labeling Model for Multiple Labeler Judgments. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Lakkaraju, H.; Bach, S. H.; and Leskovec, J. 2016. Interpretable Decision Sets: A Joint Framework for Description and Prediction. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD 16, 16751684. New York, NY, USA: Association for Computing Machinery. ISBN 9781450342322. doi:10.1145/2939672.2939874. URL https://doi.org/10.1145/2939672.2939874.

Lindgren, T. 2004. Methods for rule conflict resolution. In *European Conference on Machine Learning*, 262–273. Springer.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12: 2825–2830.

Popordanoska, T.; Kumar, M.; and Teso, S. 2020. Toward Machine-Guided, Human-Initiated Explanatory Interactive Learning. *arXiv preprint arXiv:2007.10018* .

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 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*, 1135–1144.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: High-Precision Model-Agnostic Explanations. In *AAAI*, volume 18, 1527–1535.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1(5): 206–215.

Rudin, C.; Wang, C.; and Coker, B. 2018. The age of secrecy and unfairness in recidivism prediction. *arXiv preprint arXiv:1811.00731* .

Scheffer, T.; Decomain, C.; and Wrobel, S. 2001. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, 309–318. Springer.

Schramowski, P.; Stammer, W.; Teso, S.; Brugger, A.; Herbert, F.; Shao, X.; Luigs, H.-G.; Mahlein, A.-K.; and Kersting, K. 2020. Making deep neural networks right for the right scientific reasons by interacting with their explanations. *Nature Machine Intelligence* 2(8): 476–486.

Teso, S.; and Kersting, K. 2019. Explanatory interactive machine learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 239–245.

Thomas, P. S.; da Silva, B. C.; Barto, A. G.; Giguere, S.; Brun, Y.; and Brunskill, E. 2019. Preventing undesirable behavior of intelligent machines. *Science* 366(6468): 999–1004.

Yousefzadeh, R.; and O'Leary, D. P. 2020. Auditing and Debugging Deep Learning Models via Decision Boundaries: Individual-level and Group-level Analysis. *arXiv preprint arXiv:2001.00682* .