Technical Report - DIAB-24-02-1

FLARE: Fuzzy Local Agnostic Rule-Based Explanations for Black Box Classifiers

Guillermo Fernandez[1], **Juan A. Aledo**[1], **Jose A. Gámez**[1], **Jose M. Puerta**[1]

Intelligent Systems and Data Mining Lab, Albacete, Spain
{Guillermo.Fernandez,JuanAngel.Aledo,Jose.Gamez,Jose.Puerta}@uclm.es

Abstract. The massive amount of data available in recent years has led to an explosive growth in the field of machine learning. Black box models gain from this, because they can be trained more effectively and provide the best results. In contrast, however, these models are intrinsically less interpretable than other *white box* models, e.g. a decision tree.
With regards to supervised classification problems, where the objective is to assign a class label to a new instance, there are critical applications where interpretability can be a key factor when deciding which model to use or even whether or not to implement a machine learning process. A well-accepted practice to incorporate black box models into these processes is to build a surrogate *explainable* model that can mimic black box behavior in a neighborhood of the instance whose classification has to be explained. For that purpose, white box models can be used to extract an explanation, defined as both a factual and a counterfactual explanation. A factual explanation is a way to justify the classification of the given instance in a certain category, whereas a counterfactual explanation is a way to understand why that particular instance has *not* been classified differently. In the literature, we find that these factual and counterfactual explanations are most often built using crisp classifiers.
In this work, we propose the use of a fuzzy rule-based system which generates factual and counterfactual explanations by mimicking the behavior of a black box model. This will be done by learning a fuzzy decision tree in a neighborhood of the given instance, which will provide a rule-based system. These rules will be used to extract a factual and a counterfactual explanation. Finally, to maintain the semantics of the problem domain, these rules, learned in a neighborhood of the instance, will be mapped to the global fuzzy sets, defined over the entire range of each variable.

Keywords: Explainable Artificial Intelligence, Fuzzy Decision Trees, Counterfactuals, Machine Learning, Rule-Based Systems, Fuzzy Agnostic Explainer

1 Introduction

In recent years, machine learning is making its way into numerous fields, thanks to the enormous amount of data available, which can be used to feed new, increasingly complex models [6, 9, 31]. Because of this large amount of data, some of these models (in particular black box models) are able to very accurately approach previously unaffordable problems.

Increasing model complexity is often accompanied by decreasing interpretability [3], which means that certain decisions cannot be explained. This is unacceptable in certain fields, e.g. critical systems, where it is as important to have an accurate solution so as to be able to explain the decision made by the model. Furthermore, this is also countersigned by the current legislation in Europe, e.g. the *right to explanation* included in the General Data Protection Regulation [23], which affects both humans and artificial intelligence techniques.

In the current landscape of supervised machine learning, the most proficient algorithms are mainly black box algorithms, e.g. Deep Neural Networks and Ensembles. They also happen to be those whose inner workings are most difficult

to understand. It is this problem that has given rise to the field of *eXplainable Artificial Intelligence* (XAI) [4]. Research is being carried out to push for interpretability and explainability in order to better understand the results generated by these processes. A current approach focuses on *opening* these black boxes, that is, on creating ways to explain how these models work. *Agnostic* methods [14, 24, 25] explain the decision made by any classifier, independently of its nature and inner structure. *Local* explanations focus on explaining the classification process for a single instance, while *global* explanations seek to explain or understand the reasoning process of the model as a whole. Much work is being done on generating *explainable* models that can mimic the behavior of the black box in a small region of the feature space, i.e. a neighborhood of the target instance.

This paper presents Fuzzy Local Agnostic Rule-based Explanations, FLARE, a fuzzy agnostic explainer that creates a neighborhood by sampling instances from a known probability distribution and uses a Fuzzy Decision Tree (FDT) [26] to generate a set of rules which allows us to provide factual and counterfactual explanations for the decision made by the black box algorithm. A factual explanation is a justification, usually a set of attribute-value pairs, which attempts to answer the question of *why* the instance has been classified in that category. On the other hand, a counterfactual explanation seeks to provide an answer to the question of *why* the instance *has not* been classified in a different category.

The main contributions of this study are:

- First, we provide a method to generate a neighborhood of the target instance. The method selects an instance for each class value, which it uses to define the features' sub-spaces from which neighbors are sampled.
- Second, we use a fuzzy decision tree trained from that neighborhood to generate the explanations. We generate the factual explanation as proposed in [12], and propose a method to obtain the counterfactual explanation based on that work.
- Third, to obtain semantically meaningful factual explanations, we provide a mapping between the local fuzzy sets restricted to the neighborhood subspace, used in the (locally trained) fuzzy decision tree, and the globally defined linguistic variables.
- Fourth, we empirically compare both the factual and counterfactual explanations against some of the most representative algorithms in the state of the art by using some well-established metrics and datasets.

The rest of the paper is structured as follows. In Section 2, we define the problem at hand. In Section 3, we provide an overview of the state of the art. Section 4 explains the algorithm to generate the explanations, while Section 5 provides an example to illustrate the algorithm step by step. In Section 6, we provide a comparison against some of the outstanding existing methods regarding the metrics considered. Finally, Section 7 presents the conclusions and proposes some future work.

2 Problem Definition

Let us consider a supervised classification problem, consisting of assigning a class c_j belonging to a predetermined set $C = \{c_1, ..., c_m\}$ to an instance $x = (x_1, ..., x_n)$ of values defined over n *input variables* $X_1, X_2, ..., X_n$. We will denote as $dom(X_i)$ the domain in which a variable X_i can take values, as n_{cont} the number of continuous variables, and as n_{disc} the number of discrete variables, $0 \leq n_{cont}, n_{disc} \leq n$, $n_{cont} + n_{disc} = n$.

Let us assume that, associated with each continuous input variable X_i, there is a fuzzy (linguistic) variable $F_i = \{v_{i,1} \ldots, v_{i,k_i}\}$ defined through a Ruspini partition of k_i ordered fuzzy sets (see Figure 2a). We will use v_{i,z_i} to denote both the fuzzy set and its corresponding associated linguistic label indistinctly. Given a value $\delta \in dom(X_i)$, let

$$\mu_i(\delta) = (\mu_{i,1}(\delta), \ldots, \mu_{i,k_i}(\delta))$$

be the vector of membership degrees of δ to the k_i fuzzy sets of F_i. In other words, $\mu_{i,z_i}(\delta)$ is the membership degree of δ to the set v_{i,z_i}. [1]

Let B be a classifier whose decision-making process needs to be explained, e.g. a black box model, learned from a training dataset $TR = \{(x_1^t, \ldots, x_n^t, c^t)\}_{t=1}^T$, where T is the number of instances, and $c^t \in C$ is the classification (category or class label) associated with the t-th instance.

Given an instance x_0, let $B(x_0) = c \in C$ denote that B assigns the class value c to x_0. Our objective is to create an explainer $E(x_0, c, B)$ that is able to return an explanation $e = \langle f, cf \rangle$ for x_0, where f is a factual explanation and cf is a counterfactual explanation.

As the explainer uses a fuzzy decision tree as the surrogate (local) classifier, the explanations are based on the rules extracted from this tree. A rule R of length b is defined as

$$R : l_1 \wedge l_2 \wedge \cdots \wedge l_b \rightarrow c_j.l$$

Here, each $l_p = \langle F_i, v_{i,z_i} \rangle$ is a literal, i.e. an attribute-value pair. For the continuous variables, F_i is a fuzzy variable and v_{i,z_i} is a fuzzy set belonging to that fuzzy variable. For the discrete variables, $F_i = X_i$ and v_{i,z_i} is a value from its domain. Finally, the consequent of the rule is $c(R) = c_j$.

In this work, because fuzzy reasoning is used, more than one rule may be necessary to justify the classification c assigned to x_0, i.e., f may consist of more than one rule [12].

On the other hand, the counterfactual explanation, cf, represents the minimum changes that must be applied to x_0 to modify the class assigned by classifier B. Let x_{cf} be the instance obtained by applying the changes in cf to x_0. Then, to obtain cf, we propose to minimize $d(x_0, x_{cf})$ according to the following distance:

[1] Note that for discrete variables, we can interpret each value as a linguistic label whose associated fuzzy set has membership degree 1 in case the instance takes that value and 0 otherwise.

$$d(x,y) = \frac{n_{cont}}{n} \sum_{i \in n_{cont}} |x_i - y_i| + \frac{n_{disc}}{n} \sum_{i \in n_{disc}} I(x_i \neq y_i) \quad (1)$$

where the function $I(cond)$ returns 1 if $cond$ is true and 0 otherwise. Note that, in the case of numerical variables, not only is the number of modified features counted but the magnitude of the change is also taken into account.

3 Related Works

In this work we focus on *factual and counterfactual explanations*, which are types of *local explanations*, consisting in explaining the decision made for a single instance. Counterfactual explanations date back as far as 1983 [8] and are widely used in fields such as chemistry [30], medicine [22] or justice [11]. In computer science, they have been used for improving automatic code review [7] and debugging machine learning models [1].

There are multiple ways to generate counterfactual explanations. They can be created by using adversarial examples [20], optimization processes [2,5,10,18,29], white box models such as decision trees [12, 14, 16, 27], etc.

Furthermore, our explainer is *model-agnostic*, which means that it is independent of the underlying model used to make the decision. Prominent related methods in the literature include LORE [14], CEM [10], WACH [29] and CEGP [18]. The first is the algorithm that inspired this contribution, while the other three are some of the best-performing counterfactual explanation generation methods according to the study carried out in [13].

LORE (LOcal Rule-based Explainer) [14] is a local agnostic method that provides explanations formed by both factual and counterfactual rules. LORE takes an instance and a black box binary classifier, and creates a balanced neighborhood around the given instance (with half of the neighbors matching the classification of the instance and half having the opposite classification) by means of a genetic algorithm. A decision tree is then learned from the neighborhood obtained, and a factual and a counterfactual rule are extracted from that decision tree. Given the nature of the decision tree used, LORE is only able to generate explanations for problems with binary class labels.

WACH [29] is one of the first approaches in the literature to deal with counterfactual explanations. WACH defines a loss function to be minimized, which is a trade-off between (1) the margin between the class predicted by the counterfactual explanation and the class assigned by the original classifier to the instance to be explained, and (2) the quadratic distance between the given instance and its modification by using the counterfactual explanation. The balance (preference) between these two objectives is controlled by a numerical parameter, and a gradient-descent-based algorithm is used to guide the optimization of the counterfactual explanation until convergence is achieved.

CEM (Contrastive Explanation Method) [10] uses the concepts of *pertinent positives* and *pertinent negatives*. The *pertinent positives* are the minimally sufficient set of features to obtain a particular classification. In contrast, the *pertinent*

negatives are the set of features that need to be modified in an instance to obtain a particular classification. This means that by applying the changes proposed in the *pertinent negatives*, the instance would be classified differently, similarly to a counterfactual explanation. In order to find the pertinent positives and pertinent negatives, two different loss functions are defined. The loss function for the pertinent negatives (counterfactual explanations) is based on a weighted aggregation that combines (1) the margin between the black box prediction of the instance and the modified instance; (2) a regularization term to minimize the changes carried out in the instance; and (3) a term, controlled by an autoencoder, which minimizes the implausibility of the modified instance. By implausibility, the authors in [10] refer to how far (close) the modified instance is with respect to feasible realizations in the problem feature space. The loss function for the case of pertinent positives has a similar structure, but the perturbation applied to the original instance consists of removing features instead of modifying their values, and the same classification must be maintained for both the given and the modified instance.

CEGP (Counterfactual Explanation Guided by Prototypes) [18] uses the loss function introduced by CEM but incorporates the concept of prototype to control the proximity of the perturbed instance (counterfactual explanation) to the data distribution given a class label. Therefore, prototypes are class-label-dependent, and in CEGP, for each class label c, they are computed as the average encoding over the k nearest instances to x_0 with class label c. The autoencoder designed in CEM is also used in CEGP to search for the nearest instances in the latent space. Both CEM and CEGP methods use an iterative gradient-descent-based process to optimize the perturbed instance,i.e., the counterfactual explanation.

4 Methodology

In this paper, we propose Fuzzy Local Agnostic Rule-based Explanations (FLARE), an algorithm that generates a factual and a counterfactual explanation for the classification assigned by a classifier to a given instance. In Algorithm 1, we can see that, aside from the usual inputs received by agnostic explainers, i.e., the instance, the classifier and a dataset, because of its fuzzy nature, FLARE also needs as input a set of fuzzy variables defined for each continuous domain variable. At first, FLARE generates a neighborhood around an instance by randomly sampling the neighbors according to the marginal probability distributions estimated from the dataset. A fuzzy decision tree is then trained for this neighborhood and, from this local model, a factual and a counterfactual explanation are extracted. Finally, FLARE expresses the local factual explanation in terms of the global fuzzy variables.

4.1 Neighborhood

The first step consists in generating a neighborhood N from which the fuzzy decision tree will be trained. This neighborhood must be composed of instances

Algorithm 1 FLARE

Require: training set TR, fuzzy variables F, classifier B, class variable C, $m = |C|$

procedure FLARE(x_0, s)
  $c \leftarrow B(x_0)$
  $N \leftarrow$ CreateNeighborhood(x_0, $\lceil s/m \rceil$, c)
  for all $c_j \in C \wedge c_j \neq c$ **do**
    $TR^j \leftarrow TR^{\downarrow C=c_j}$
    $\bar{x}^j \leftarrow \arg\min_{x \in TR^j} d(x_0, x)$
    $N^j \leftarrow$ CreateNeighborhood($\bar{x}^j$, $\lceil s/m \rceil$, c_j)
    $N \leftarrow N \cup N^j$
  end for
  $F' \leftarrow$ FuzzyPartitioning(N)
  $tree \leftarrow$ FuzzyDecisionTree(F', N)
  $f \leftarrow$ Factual($tree$, x_0)
  $cf \leftarrow$ Counterfactual($tree$, x_0)
  return $\langle f, cf \rangle$
end procedure

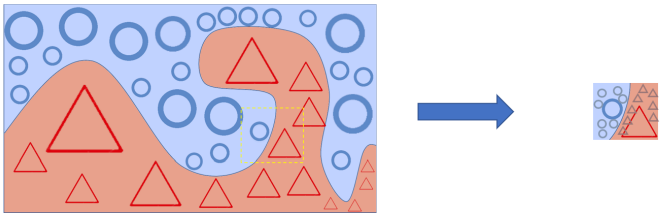


Fig. 1: Complex global decision boundaries can be simple when observed locally.

Algorithm 2 Neighborhood creation

Require: classifier B
  procedure CreateNeighborhood(x, l, c_j)
    $N^j \leftarrow \emptyset$
    while $|N^j| < l$ **do**
      $\hat{x} \leftarrow$ RandomPerturbation(x)
      if $B(\hat{x}) = c_j$ **then**
        $N^j \leftarrow N^j \cup \{\hat{x}\}$
      end if
    end while
    return N^j
  end procedure

similar to x_0 in order to reproduce the local decision boundary of B. The intuition behind using a neighborhood is that local boundaries can be simpler to model than global boundaries, as represented in Figure 1.

We can see in Algorithm 1 that, for each class value c_j, we extract a pivot $\bar{x}^j$ from TR^j (the subset of instances from TR with class value c_j). This pivot is the closest instance in TR^j to x_0 according to Eq. 1. In particular, when $c_j = B(x_0)$, then $\bar{x}^j = x_0$. Using that pivot, we then generate the sub-neighborhood of size $l = \lceil s/m \rceil$ using Algorithm 2.

In order to generate a neighbor $\hat{x}$ by randomly perturbing the pivot $\bar{x}^j$, the process is as follows:

- For each continuous variable X_i, the i-th component of $\hat{x}$ is obtained by sampling from the empirical distribution in the interval

$$[\max(\min(X_i), \bar{x}_i - \epsilon_i), \min(\max(X_i), \bar{x}_i + \epsilon_i)],$$

  where ϵ_i defines the range of the interval.
- For each discrete variable X_i, the i-th component of $\hat{x}$ is obtained by sampling from the multinomial probability distribution $p(X_i)$ estimated from TR.

If $B(\hat{x}) = c_j$, we add $\hat{x}$ to N^j. Otherwise, we discard it. We create each sub-neighborhood N^j by repeating the process until we obtain the desired number l of neighbors, and finally, all the sub-neighborhoods are joined to obtain N.

4.2 Fuzzy Decision Tree

The next step is to build the fuzzy decision tree. In this work, we use the Fuzzy Multiway Decision Tree, FDT, described in [26] with the aggregated vote process for the inference.

The tree is learned from the dataset N and the locally defined set of fuzzy variables F'. These variables are obtained by means of fuzzy entropy-based fuzzy partitioning [26].

Algorithm 3 Factual explanation generation

Require: global fuzzy variables F, local fuzzy variables F'
  procedure FACTUAL(tree, x_0)
    $f \leftarrow \emptyset$
    $f' \leftarrow$ MINROBUSTFACTUAL(tree, x_0) ▷ See [12]
    for all $R'_f \in f'$ **do**
      $R_f \leftarrow \emptyset$
      for all $\langle F'_i, v'_{i,z'_i} \rangle \in R'_f$ **do**
        $v_{i,z_i} \leftarrow M(v'_{i,z'_i}, F_i)$ ▷ See Eq. 2
        $R_f \leftarrow R_f \wedge \langle F_i, v_{i,z_i} \rangle$
      end for
      $R_f \leftarrow R_f \wedge c(R'_f)$
      $f \leftarrow f \cup R_f$
    end for
    return f
  end procedure

4.3 Factual Explanation

A factual explanation is the reason why a particular instance is classified into a certain class value. This typically means that the rule fired by a (crisp) decision tree is the factual explanation, as the path that leads to the classification is unique. This does not apply directly when fuzzy models are used to generate the explanation. There are proposals, such as [27], which consider that using the best rule according to the inference process is sufficient to generate the factual explanation. Other works, such as [12], propose that more than a single rule can be used to generate a factual explanation.

In this work, we use the aggregated vote process to classify the instance with the locally learned FDT, and it is thus appropriate to use multiple rules to explain as well as to classify. Therefore, this algorithm is compatible with factual explanations formed by multiple rules.

In [12], a factual explanation for x_0 with class label c is considered *robust* if, when removing from the set of rules fired by x_0 those with consequent c and not included in the factual explanation, the classification for x_0 using the remaining rules is still c. In particular, the *minimum robust factual* explanation, $mr\text{-}factual$, becomes a good choice because it ensures the *robustness* while keeping the number of rules corresponding to the factual explanations as low as possible. For this reason, we will use the minimum robust factual explanation in our experiments and it is the method referred to in Algorithm 3.

4.4 Factual Explanation Mapping

In our method, we use two different sets of fuzzy variables. FLARE receives the set $F = \{F_1, \ldots, F_n\}$ of *linguistic* fuzzy variables defined over $\{dom(X_1), \ldots, dom(X_n)\}$, for all numerical variable X_i. On the other hand, each time an explanation

is needed for the classification $B(x_0) = c$, a new set of fuzzy variables $F' = \{F'_1, \ldots, F'_n\}$ is learned over $\{dom_N(X_1), \ldots, dom_N(X_n)\}$ with $dom_N(X_i) \subseteq dom(X_i)$, N being the neighborhood of x_0 used to learn the FDT. To obtain meaningful factual explanations, they must be expressed in terms of the linguistic fuzzy variables provided, otherwise, the explanations for different instances would have different semantics. Therefore, we need to establish a mapping $F' \rightarrow F$ to re-write the rules in the factual explanation.

In order to map F' to F, we compute a similarity measure between two fuzzy sets. Given a fuzzy set v_{i,z_i} defined over the variable X_i, take

$$V_{i,z_i} = [a_{z_i}, b_{z_i}] \in dom(X_i), \text{ such that:}$$

$$(1)\ \forall \delta \in [a_{z_i}, b_{z_i}],\ \mu_{i,z_i}(\delta) \geq 0.5$$

$$(2)\ \nexists\ [a'_{z_i}, b'_{z_i}] \supset [a_{z_i}, b_{z_i}] \text{ satisfying (1)}$$

Note that as we use triangular fuzzy sets, V_{i,z_i} corresponds to the α-cut with $\alpha = 0.5$ in v_{i,z_i}.

Then, given two fuzzy sets $v_{i,z_i} \in F_i$ and $v'_{i,z'_i} \in F'_i$, we compute its similarity as

$$S(v_{i,z_i}, v'_{i,z'_i}) = \frac{\text{length}(V_{i,z_i} \cap V'_{i,z'_i})}{\text{length}(V_{i,z_i} \cup V'_{i,z'_i})}$$

Finally, given a variable X_i, we define the function $M(v'_{i,z'_i}, F_i)$ that takes a fuzzy set $v'_{i,z'_i} \in F'_i$ and returns the set $v_{i,z_i} \in F_i$ with the greatest similarity

$$M(v'_{i,z'_i}, F_i) = \arg \max_{v_{i,z_i} \in F_i} S(v_{i,z_i}, v'_{i,z'_i}) \tag{2}$$

We use this function M in Algorithm 3 to denote the mapping between the local and global fuzzy sets to generate a global factual explanation from the local *minimum robust* explanation obtained from the FDT.

4.5 Counterfactual Explanation

Given $B(x_0) = c$, a counterfactual explanation is the minimum set of modifications to be applied over x_0 that modifies the class value assigned by the classifier B. As in [12], we use the rules from the FDT with a class different to c to find possible counterfactual explanations.

Each of these rules R_{cf} proposes a set of changes cf to x_0 that may change the assigned class label. The method ApplyChanges in Algorithm 4 generates a candidate counterfactual from a rule R_{cf} and x_0 as follows:

- In the case of a numerical variable X_i, we consider that a change must be applied if the membership degree is smaller than 0.5. In that case, we need to provide an exact value from the infinite set $dom(X_i)$. In our case, for the literal $\langle F'_i, v'_{i,z'_i} \rangle$, the change is the value δ such that $\mu_{v'_{i,z'_i}}(\delta)$ is maximum[2].

[2] Note that if normalized triangular fuzzy sets are used, this δ value corresponds to the middle point of the triangle, with membership equal to 1

Algorithm 4 Counterfactual explanation generation

Require: classifier B

procedure ApplyChanges(x_0, R)
  $x \leftarrow$ clone x_0
  $cf \leftarrow \emptyset$ ▷ Changes to apply over x_0
  for all X_i continuous $\wedge$ $\langle F'_i, v'_{i,z'_i} \rangle \in R$ **do**
    if $\mu_{i,z'_i}(x_i) < 0.5$ **then**
      $x_i \leftarrow \text{Peak}(v'_{i,z'_i})$
      $cf \leftarrow cf \cup \{\langle X_i, x_i \rangle\}$
    end if
  end for
  for all X_i discrete $\wedge$ $\langle F'_i, v'_{i,z'_i} \rangle \in R$ **do**
    if $x_i \neq v'_{i,z'_i}$ **then**
      $x_i \leftarrow v'_{i,z'_i}$
      $cf \leftarrow cf \cup \{\langle X_i, x_i \rangle\}$
    end if
  end for
  return x, cf
end procedure

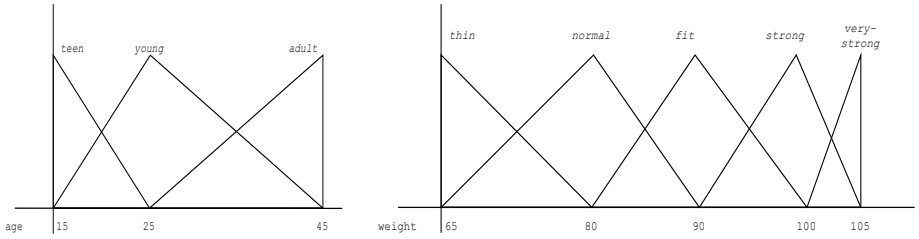
procedure Counterfactual($tree$, x_0, F')
  $rules \leftarrow \text{ExtractRules}(tree)$
  $cfRules \leftarrow \{R_{cf} \mid R_{cf} \in rules, c(R_{cf}) \neq B(x_0)\}$
  $\Psi \leftarrow [\text{ApplyChanges}(x_0, R_{cf}) \mid R_{cf} \in cfRules]$
  $\Psi \leftarrow \text{Sort}(\Psi)$
  for $k \leftarrow 1$ to $|\Psi|$ **do**
    $\langle x_{cf}, cf \rangle \leftarrow \Psi_k$
    if $B(x) \neq B(x_{cf})$ **then**
      return cf ▷ Counterfactual found
    end if
  end for
  return $\emptyset$ ▷ No counterfactual found
end procedure

– In the case of a categorical variable X_i, the change is the value $v'_{i,z'_i} \in dom(X_i)$ in R_{cf}.

Note that ApplyChanges returns the set of changes cf proposed by the rule, as well as the instance x_0 with those changes applied, x_{cf}.

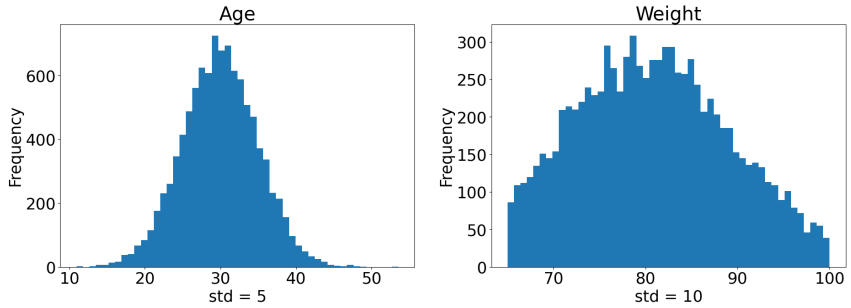
The method Counterfactual in Algorithm 4 starts extracting the set of rules $cfRules$ from the FDT with a class label different to $B(x_0)$. Next, from $cfRules$ we obtain Ψ as the list of candidate counterfactual explanations for x_0 that result from applying ApplyChanges to each R_{cf} in $cfRules$. Then, Ψ is sorted increasingly according to $d(x_{cf}, x_0)$ (Eq. 1). Finally, the algorithm runs over this sorted list, testing whether $B(x_{cf}) \neq B(x_0)$ and, if so, returns cf, i.e., the counterfactual explanation.

5 Illustrative Example



(a) Fuzzy sets for *age*. (b) Fuzzy sets for *weight*.

Fig. 2: Global fuzzy sets defined over the complete domain of the fuzzy variables *age* and *weight*.



(a) *age* histogram. (b) *weight* histogram.

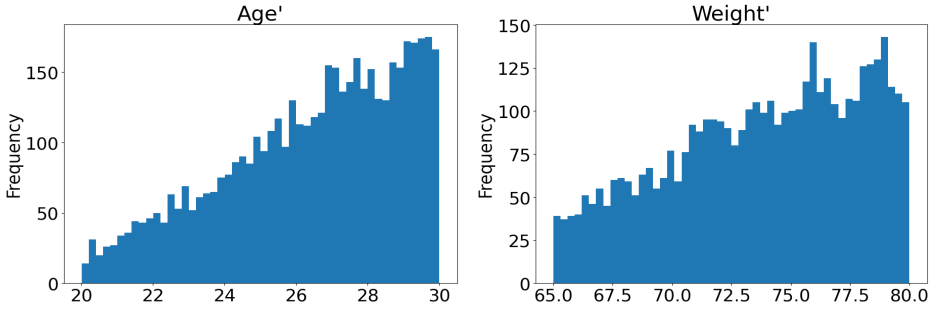
Fig. 3: Histograms extracted from TR for the continuous variables *age* and *weight*.

For illustrative purposes, let us consider a problem where the classification of an instance must be explained. We consider a binary classification problem, $C = \{yes, no\}$ where the following objects are received as inputs: a training set TR defined over variables *age*, *weight*, *hair* and C; a (possibly black box) classifier B already trained using TR; and the fuzzy variables $age = \{young, teen, adult\}$ and $weight = \{thin, normal, fit, strong, very\text{-}strong\}$ (see Figure 2) defined for the numerical domain variables with the same name. Figure 3 shows the histograms for *age* and *weight* obtained from TR.

In this example, we aim to explain the classification of the following instance:

$$x_0 = \{age : 25,\ weight : 70,\ hair : blond\}$$
$$B(x_0) = yes$$

5.1 Neighborhood Generation



(a) *age* projected distribution. (b) *weight* projected distribution.

Fig. 4: Projected probability distributions for N^{yes}.

The first step of the algorithm is to obtain neighborhood N for x_0. As $C = \{yes, no\}$ we need to generate N^{yes} and N^{no} (Algorithm 2).

As $B(x_0) = yes$, the pivot for N^{yes} is x_0 itself. Given the values in x_0 for *age* and *weight* and their estimated standard deviations ($\sigma_{age} = 5$ and $\sigma_{weight} = 10$, shown in Figure 3), which we will use as ϵ for each variable, the sampling intervals are:

$$[\max(\min(age), 25 - 5), \min(\max(age), 25 + 5)] =$$
$$= [20, 30],$$
$$[\max(\min(weight), 70 - 10), \min(\max(weight), 70 + 10)] =$$
$$= [65, 80]$$

for *age* and *weight* respectively. Figure 4 shows the empirical distributions projected over these intervals, from which the neighbor values are sampled. Thus, N^{yes} is formed by individuals such as

$$\{age : 23,\ weight : 68,\ hair : red\}$$
$$\{age : 27,\ weight : 72,\ hair : blond\}$$
$$\{age : 24,\ weight : 75,\ hair : red\}$$
$$\ldots$$

For N^{no}, we start by identifying the pivot $\bar{x}^{no}$ as the closed instance to x_0 in TR^{no} according to the distance defined in (Eq. 1). In our example, that instance is:

$$\bar{x}^{no} = \{age : 27,\ weight : 82,\ hair : brown\}$$
$$B(\bar{x}^{no}) = no$$

with distance

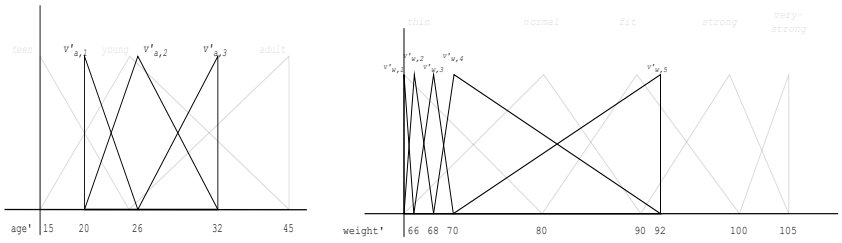
$$d(x_0, \bar{x}^{no}) = \frac{2}{3} \cdot (|25 - 27| + |70 - 82|) + \frac{1}{3} = 9.67$$

By using the same process as for N^{yes}, we get neighbor instances such as:

$$\{age : 25,\ weight : 80,\ hair : brown\}$$
$$\{age : 30,\ weight : 80,\ hair : blond\}$$
$$\{age : 29,\ weight : 77,\ hair : red\}$$
$$\ldots$$

5.2 Training a Fuzzy Decision Tree

Next, we obtain the fuzzy variables $F' = \{age', weight'\}$ to be used in the process of learning an FDT from N. The original fuzzy variables for age and $weight$ cannot be used, as the neighborhood only involves a sub-interval of the original domains. The variables age' and $weight'$ obtained from N by using the fuzzy entropy-based fuzzy partitioning are shown in Figure 5. The FDT learned from N by using F' is shown in Figure 6.



(a) Fuzzy sets for age' (bold) superimposed on the fuzzy sets for age (light).

(b) Fuzzy sets for $weight'$ (bold) superimposed on the fuzzy sets for $weight$ (light).

Fig. 5: Local fuzzy sets superimposed on the original (global) fuzzy sets.

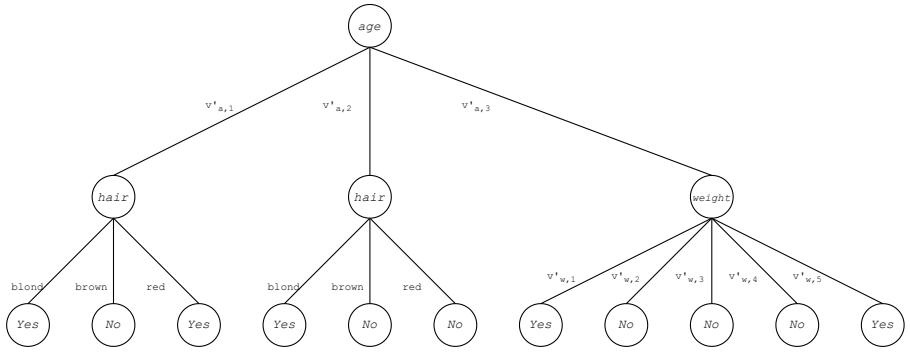


Fig. 6: Local explainer model (FDT) learned using N and F'.

5.3 Factual explanation

From the FDT, we extract the minimum robust factual explanation introduced in [12]:

$$R_f \rightarrow \{age : v'_{age',2} \ \& \ hair : blond\}.$$

We then have to express the factual explanation in terms of the original linguistic fuzzy variables, so we have to map $v'_{age',2}$ to one of the linguistic labels (fuzzy sets) in age. First, we compute the similarities:

$$S(teen, v'_{age',2}) = \frac{\text{length}(teen \cap v'_{age',2})}{\text{length}(teen \cup v'_{age',2})} = 0$$
$$S(young, v'_{age',2}) = \frac{\text{length}(young \cap v'_{age',2})}{\text{length}(young \cup v'_{age',2})} = 0.2$$
$$S(adult, v'_{age',2}) = \frac{\text{length}(adult \cap v'_{age',2})}{\text{length}(adult \cup v'_{age',2})} = 0$$

We then get:

$$M(v'_{age',2}, age) = \arg \max_{v_{i,z_i} \in age} S(v_{i,z_i}, v'_{age',2}) = young.$$

Finally, as $hair$ is a discrete variable, the factual explanation becomes:

$$f \rightarrow \{age : young \ \& \ hair : blond\}.$$

5.4 Counterfactual explanation

For the counterfactual explanation, all the rules from FDT with a different class value from the instance must be extracted, i.e. $R_{cf} \in tree : c(R_{cf}) \neq c(x_0)$.

They are:

$$R_{cf_1} \rightarrow \{age : v'_{age',1} \ \& \ hair : brown\}$$
$$R_{cf_2} \rightarrow \{age : v'_{age',2} \ \& \ hair : brown\}$$
$$R_{cf_3} \rightarrow \{age : v'_{age',2} \ \& \ hair : red\}$$
$$R_{cf_4} \rightarrow \{age : v'_{age',3} \ \& \ weight : v'_{weight',2}\}$$
$$R_{cf_5} \rightarrow \{age : v'_{age',3} \ \& \ weight : v'_{weight',3}\}$$
$$R_{cf_6} \rightarrow \{age : v'_{age',3} \ \& \ weight : v'_{weight',4}\}$$

These rules generate a set of changes that must be applied to x_0. When $\mu_{i,v'_{i,z'_i}}(x_i) < 0.5$, i.e. the membership degree of the instance to that fuzzy set is smaller than 0.5, we apply that change. In those cases, x_i is replaced by the peak of the triangular fuzzy set.

ApplyChanges returns the following modified instances (each x_{cf} being the first component of the elements in Ψ):

$$x_{cf_1} \rightarrow \{age : 20, \ weight : 70, \ hair : brown\}$$
$$x_{cf_2} \rightarrow \{age : 25, \ weight : 70, \ hair : brown\}$$
$$x_{cf_3} \rightarrow \{age : 25, \ weight : 70, \ hair : red\}$$
$$x_{cf_4} \rightarrow \{age : 32, \ weight : 66, \ hair : blond\}$$
$$x_{cf_5} \rightarrow \{age : 32, \ weight : 68, \ hair : blond\}$$
$$x_{cf_6} \rightarrow \{age : 32, \ weight : 70, \ hair : blond\}$$

We then compute the distance (Eq. 1) between x and the counterfactual explanations above:

$$d(x, x_{cf_1}) = \frac{2}{3} \cdot 5 + \frac{1}{3} = 3.66$$
$$d(x, x_{cf_2}) = \frac{1}{3} = 0.33$$
$$d(x, x_{cf_3}) = \frac{1}{3} = 0.33$$
$$d(x, x_{cf_4}) = \frac{2}{3} \cdot (7 + 4) = 7.33$$
$$d(x, x_{cf_5}) = \frac{2}{3} \cdot (7 + 2) = 3$$
$$d(x, x_{cf_6}) = \frac{2}{3} \cdot 7 = 4.66$$

The closest instances are x_{cf_2} and x_{cf_3}. In case of a tie, the counterfactual explanation is selected arbitrarily in the order that the rules have been generated. For this example, we choose the changes proposed by R_{cf_2}, thus selecting as counterfactual explanation

$$cf \rightarrow \{hair : brown\}$$

5.5 Explanation

Hence, our explanation becomes

$$\begin{aligned} e &= \langle f, cf \rangle \\ &= \langle \{age : young \ \& \ hair : blond\}, \{hair : brown\} \rangle \end{aligned}$$

That is, the classification of x_0 as *yes* is supported because the person is *young* and has *blond* hair, while the simplest way to change its classification to *no* is to get *brown* hair.

6 Experimental evaluation

6.1 Datasets

For our experimental study, we chose the four binary (two class labels) datasets described in Table 1, which are widely used in the literature [3].

Table 1: Number of continuous and discrete variables and number of instances per dataset.

Name	Discrete	Continuous	Instances
adult	9	4	48.800
compas	8	4	7.200
fico	0	23	10.400
german	18	3	1.000

6.2 Code, reproducibility and computational resources

The algorithm proposed in this paper is implemented as part of the Teacher [4] library and is completely open-source. It was programmed in Python 3.10, using libraries such as *Pandas* [28] [19], *NumPy* [15] and *scikit-learn* [21] to properly manage the data structures and efficiently generate the explanations. To guarantee reproducibility, all the experiments are also published in a public Github repository [5].

The experiments were executed in a Ryzen 9 3900x@3.8GHz with 32GB of 3600MHz RAM Memory. The results were replicated from [13], where the experiments are open-source, and so were adapted to work with our proposal.

[3] `https://archive.ics.uci.edu/ml/index.php`, `https://www.kaggle.com/datasets`.

[4] `https://github.com/Kaysera/teacher`

[5] `https://github.com/Kaysera/flare-experiments`

6.3 Experimental methodology

For each dataset, we trained a Neural Network [6] and a Support Vector Machine [7] from *scikit-learn* as the classifier to be explained. For validation, *holdout* has been considered by using standard values to create the partition (70% training and 30% test). The models were trained using a randomized search to find suitable hyperparameters for each dataset. Then, the instances to be explained are taken from the test set.

6.4 Factual explanation evaluation

To evaluate the factual explanation generated by FLARE, we use the following two commonly considered metrics:

- *fidelity*: Measures the similarity of the predictions of B and the local explainer for the instances in the neighborhood.
- *hit*: Measures whether $B(x_0)$ is equal to the output of the local model learned from the neighborhood N of x_0.

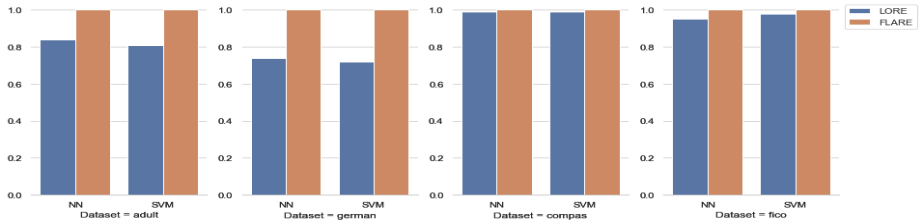


Fig. 7: Ratio of factual explanations found per method.

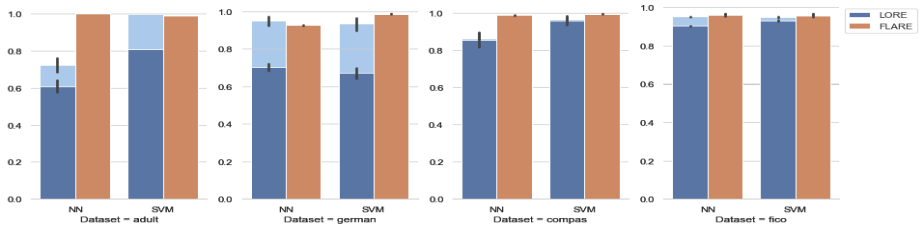


Fig. 8: *Fidelity* of each method.

[6] `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`

[7] `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

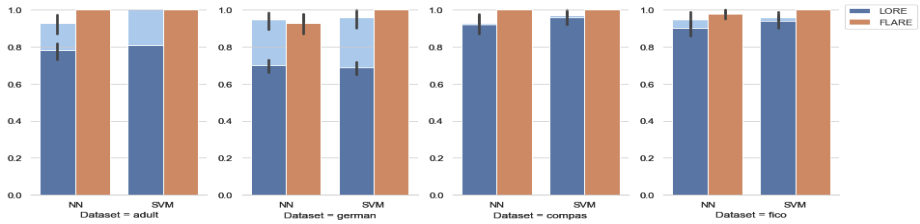


Fig. 9: *Hit* for each method.

In particular, we compared FLARE against LORE, because it also learns a local model on the neighborhood of x_0. Of the rest of the methods considered in this study, only CEM provides something resembling a factual explanation, the *pertinent positives*. However, it directly searches for them by optimizing certain metrics, not by learning a surrogate model. Therefore, fidelity and hit cannot be computed.

In Figure 7 we show the percentage of factual explanations extracted per method. We used the currently available implementation of LORE [8], where, as can be observed, no factual explanation is found in some cases, which seems to be an actual limitation of the implementation, not of the algorithm. For the rest of the metrics, we show the actual value, but also a penalized one by taking into account the percentage of cases in which the factual explanation is not found. In Figures 8 and 9, the actual value can be seen in a lighter shaed and the penalized one is in a stronger color.

In Figure 8, we can see that in most cases, for a given classifier and dataset, the neighborhood generated by FLARE is able to better model the neighborhood while maintaining the same classification as the original accurate classifier for the instance to explain. This results in a higher percentage of factual explanations, which also improves the *hit* as can be seen in Figure 9.

6.5 Counterfactual explanation evaluation

Evaluating counterfactual explanations is currently a highly active field of research. In particular, many metrics are considered to study the different features of the counterfactual explanations. In [13], the most prominent methods in the state of the art are reviewed and compared considering a set of selected metrics. From this set of metrics, we discarded those not applicable to the algorithms here compared, e.g. those generating multiple counterfactual explanations, and selected the subset that allows us to properly evaluate the counterfactual explanations generated by the algorithms considered in these experiments:

– *Proximity dissimilarity*: Measures the proximity between x_0 and x_{cf} as the distance between them (Eq. 1).

[8] `https://github.com/riccotti/LORE`

– *Sparsity dissimilarity*: Measures the length of the cf, that is, the number of features that would have to change in x_0 to change the classification.
– *Implausibility*: Accounts for how close the counterfactual explanation is to the reference population, as measured by the distance (Eq. 1) between x_{cf} and the closest instance in TR, i.e.

$$\min_{x \in TR} d(x_{cf}, x).$$

Note that, in all three metrics, the lower, the better. To evaluate FLARE's capability with respect to counterfactual explanation generation, we compared it against state-of-the-art algorithms CEGP, CEM and WACH [13]. LORE was tested but, in line with [13], the results were not as good as those ones obtained by the other methods under study, and so it was excluded from the counterfactual explanation evaluation. All the algorithms are open-source and implemented in a well-known library [17] so that the experimental evaluation can be reproduced.

Hyperparameters were fine-tuned for every method (see the Appendix for details). For all the optimization-based methods, the number of iterations was limited to 1000 to keep the run time reasonable (several minutes per counterfactual explanation extracted), so it is possible that, under this time limit, the algorithm does not find a counterfactual explanation. The same time restriction was applied to FLARE, which never exceeds it.

Figure 10 represents the number of counterfactual explanations found per method. It is highly dependent on both the different datasets and the black box classifier. However, we can see that, under the limit set, FLARE is the only method that always finds a counterfactual explanation. Note that this also influences the results of the other metrics because, in some cases, the rest of the algorithms have good averages since they only take into account the counterfactual explanations found, which seem to be the less difficult ones regarding the optimization process. Similarly to Section 6.4, all figures have the actual value of the computed metrics in an intense color and a penalty applied based on the ratio of counterfactual explanations in a lighter shade of the same color.

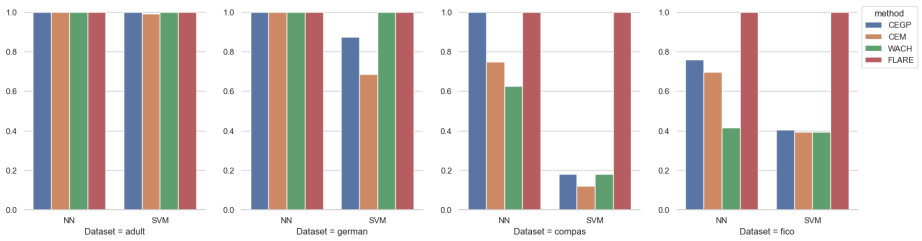


Fig. 10: Ratio of found counterfactual explanations.

Figure 11 shows the results for the proximity dissimilarity. The smaller the distance, the better is the counterfactual explanation. Here we can see that,

despite being highly dataset-dependent, CEM or CEGP are usually the best methods. FLARE, however, is close to both of them, winning in most cases after applying the penalty corresponding to not always finding a counterfactual explanation. As CEM, CEGP and WACH use optimization processes, they sometimes are able to find shorter distances than FLARE, which is limited by the partitions of the fuzzy sets and the greedy criteria we have chosen to define the changes in the continuous variables.

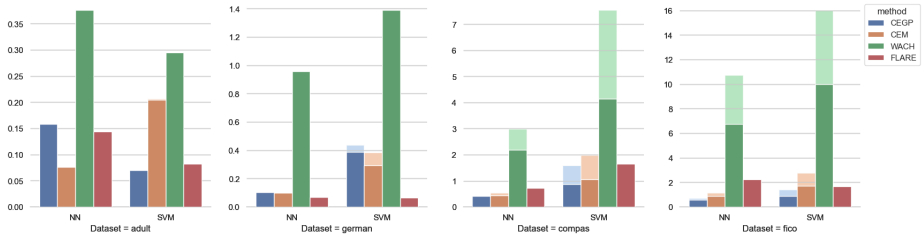


Fig. 11: Proximity Dissimilarity: distance between x_0 and x_{cf}

Figure 12 represents the results of the sparsity dissimilarity. As can be observed, FLARE is systematically better than the alternatives. In this case, the closer instance obtained by CEM, CEGP and WACH entails changes in multiple variables. FLARE, by changing the variables according to the fuzzy sets found, performs bigger changes in each variable of the counterfactual explanation, so needing to change the value of fewer variables.

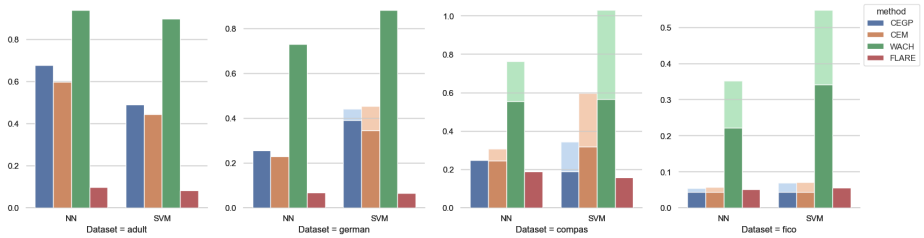


Fig. 12: Sparsity Dissimilarity: number of changes in cf

Figure 13 represents the implausibility. In this case, the best-performing algorithm is WACH. In adult and fico, the rest of the algorithms obtain notably worse results. In german and compas we can observe that CEM and CEGP still perform worse, but FLARE is able to get close or even improve WACH in some cases.

Finally, Figure 14 shows the time spent to find a counterfactual explanation. We can see here that, despite limiting the optimization methods to 1000

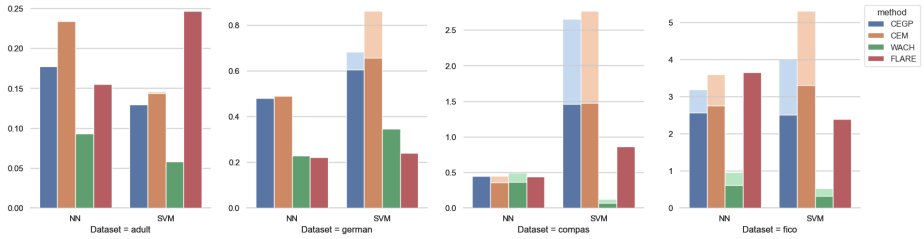


Fig. 13: Implausibility: distance between x_{cf} and the closest instance in TR

iterations, they take several orders of magnitude longer than FLARE to find a counterfactual explanation. It should be noted that all these methods work by finding a counterfactual explanation for each new instance whose classification needs to be explained, i.e. they cannot cache the counterfactual explanations from previous cases to be applied to a new one. Given the nature of the problem, in the real world, an explanation is expected almost immediately, as quickly as a classification. This is why, for a real-world application, immediacy is a crucial factor, and FLARE stands out in this regard.

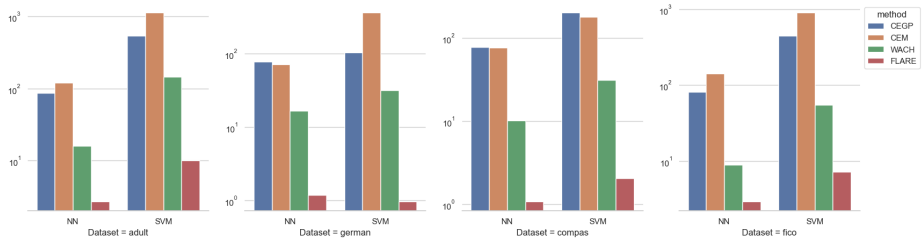


Fig. 14: CPU Time

7 Conclusions

This work proposes FLARE, a novel approach to agnostic explainers by introducing fuzzy logic in the generation process of factual and counterfactual explanations, which provides an interesting alternative to the existing methods in the state of the art.

We define FLARE, give an open-source code implementation of the algorithm, and compare it against state-of-the-art factual and counterfactual algorithms. The factual explanations for FLARE improve upon those of LORE, which inspired the algorithm; while the counterfactual explanations of FLARE are on-par or even better than the current best methods such as CEM, CEGP or WACH.

We also showcase the greatest strength of FLARE against the current trend of optimization-based methods: by using a surrogate model, finding a counterfactual explanation in a near-instant time is guaranteed. Optimization methods sometimes need minutes to find counterfactual explanations, even failing to find one if time or computation constraints (i.e. number of iterations) are imposed. This is a major drawback in a problem that requires a (near) real-time response.

There is room to extend this work. When looking for counterfactual explanations, we aim to study a modification of the greedy behavior of our method to change the value in continuous variables so that smaller changes are generated, thus improving metrics like proximity dissimilarity and implausibility. However, a more in-depth study is required to find ways to avoid a computationally slow optimization process. It would also be interesting to generate natural language explanations from these factual and counterfactual explanations so that they can be presented to a final user, taking advantage of the closeness of fuzzy logic to human language.

Acknowledgements

This work has been funded by the project SBPLY/21/180225/000062 funded by the Government of Castilla-La Mancha and "ERDF A way of making Europe". It is also partially funded by MCIN/AEI/10.13039/501100011033 and "ESF Investing your future" through the projects PID2019–106758GB–C33 and FPU19/02930.

References

1. Abid, A., Yuksekgonul, M., Zou, J.: Meaningfully debugging model mistakes using conceptual counterfactual explanations. In: International Conference on Machine Learning. pp. 66–88. PMLR (2022)
2. Albini, E., Long, J., Dervovic, D., Magazzeni, D.: Counterfactual shapley additive explanations. In: 2022 ACM Conference on Fairness, Accountability, and Transparency. pp. 1054–1070 (2022)
3. Angelov, P.P., Soares, E.A., Jiang, R., Arnold, N.I., Atkinson, P.M.: Explainable artificial intelligence: an analytical review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **11**(5), e1424 (2021)
4. Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al.: Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information fusion **58**, 82–115 (2020)
5. Artelt, A., Hammer, B.: Efficient computation of counterfactual explanations and counterfactual metrics of prototype-based classifiers. Neurocomputing **470**, 304–317 (2022)
6. Chen, T., Guestrin, C.: XGBoost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. pp. 785–794. ACM (2016)

7. Cito, J., Dillig, I., Murali, V., Chandra, S.: Counterfactual explanations for models of code. In: Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice. pp. 125–134 (2022)
8. Clancey, W.J.: The epistemology of a rule-based expert system—a framework for explanation. Artificial intelligence **20**(3), 215–251 (1983)
9. Dai, Z., Liu, H., Le, Q.V., Tan, M.: CoAtNet: Marrying Convolution and Attention for All Data Sizes. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS-2021) (2021)
10. Dhurandhar, A., Chen, P.Y., Luss, R., Tu, C.C., Ting, P., Shanmugam, K., Das, P.: Explanations based on the missing: Towards contrastive explanations with pertinent negatives. Advances in neural information processing systems **31** (2018)
11. Dressel, J., Farid, H.: The accuracy, fairness, and limits of predicting recidivism. Science advances **4**(1), eaao5580 (2018)
12. Fernandez, G., Aledo, J.A., Gamez, J.A., Puerta, J.M.: Factual and counterfactual explanations in fuzzy classification trees. IEEE Transactions on Fuzzy Systems (2022). https://doi.org/10.1109/TFUZZ.2022.3179582
13. Guidotti, R.: Counterfactual explanations and how to find them: literature review and benchmarking. Data Mining and Knowledge Discovery pp. 1–55 (2022)
14. Guidotti, R., Monreale, A., Giannotti, F., Pedreschi, D., Ruggieri, S., Turini, F.: Factual and counterfactual explanations for black box decision making. IEEE Intelligent Systems **34**(6), 14–23 (2019)
15. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (Sep 2020). https://doi.org/10.1038/s41586-020-2649-2, https://doi.org/10.1038/s41586-020-2649-2
16. Kanamori, K., Takagi, T., Kobayashi, K., Ike, Y.: Counterfactual explanation trees: Transparent and consistent actionable recourse with decision trees. In: International Conference on Artificial Intelligence and Statistics. pp. 1846–1870. PMLR (2022)
17. Klaise, J., Looveren, A.V., Vacanti, G., Coca, A.: Alibi explain: Algorithms for explaining machine learning models. Journal of Machine Learning Research **22**(181), 1–7 (2021)
18. Looveren, A.V., Klaise, J.: Interpretable counterfactual explanations guided by prototypes. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 650–665. Springer (2021)
19. Wes McKinney: Data Structures for Statistical Computing in Python. In: Stéfan van der Walt, Jarrod Millman (eds.) Proceedings of the 9th Python in Science Conference. pp. 56 – 61 (2010). https://doi.org/10.25080/Majora-92bf1922-00a
20. Pawelczyk, M., Agarwal, C., Joshi, S., Upadhyay, S., Lakkaraju, H.: Exploring counterfactual explanations through the lens of adversarial examples: A theoretical and empirical analysis. In: International Conference on Artificial Intelligence and Statistics. pp. 4574–4594. PMLR (2022)
21. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)

22. Pfohl, S.R., Duan, T., Ding, D.Y., Shah, N.H.: Counterfactual reasoning for fair clinical risk prediction. In: Machine Learning for Healthcare Conference. pp. 325–358. PMLR (2019)
23. Regulation, G.D.P.: General data protection regulation (GDPR). Intersoft Consulting, Accessed in October 24 **1** (2018)
24. Ribeiro, M.T., Singh, S., Guestrin, C.: "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. pp. 1135–1144 (2016)
25. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-precision model-agnostic explanations. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
26. Segatori, A., Marcelloni, F., Pedrycz, W.: On distributed fuzzy decision trees for big data. IEEE Transactions on Fuzzy Systems **26**(1), 174–192 (2017)
27. Stepin, I., Alonso, J.M., Catala, A., Pereira-Fariña, M.: Generation and evaluation of factual and counterfactual explanations for decision trees and fuzzy rule-based classifiers. In: 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). pp. 1–8. IEEE (2020)
28. pandas development team, T.: pandas-dev/pandas: Pandas (Feb 2020). `https://doi.org/10.5281/zenodo.3509134`, `https://doi.org/10.5281/zenodo.3509134`
29. Wachter, S., Mittelstadt, B., Russell, C.: Counterfactual explanations without opening the black box: Automated decisions and the GDPR. Harv. JL & Tech. **31**, 841 (2017)
30. Wellawatte, G.P., Seshadri, A., White, A.D.: Model agnostic generation of counterfactual explanations for molecules. Chemical science **13**(13), 3697–3705 (2022)
31. Zhang, S., Liu, M., Yan, J.: The diversified ensemble neural network. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems (NeurIPS 2020) (2020)

Appendix: Hyperparameter Tuning

In order to find good hyperparameters for each method, a Random Search was performed for every counterfactual algorithm. For all the optimization methods (CEM, CEGP and WACH), the number of iterations was limited to 1000 to keep the run-time reasonable (less than 15 minutes per counterfactual explanation extracted). The same time constraint is applied to FLARE although it never exceeds a few seconds.

For each algorithm, 100 different combinations of parameters were randomly sampled. Each combination of parameters was then tested by evaluating 10 instances not seen by the classifiers in training. The best combination for each dataset and black box classifier was chosen and used for the experiments in this paper. Note that, with more time and/or iterations, the optimization methods might be able to obtain better results in the experimental evaluation, i.e. more counterfactual explanations found.

More specifically, a configuration of hyperparameter values is selected based on the following criteria:

1. Maximize the number of counterfactual explanations found. Given that we are limiting the number of iterations to 1000, the optimization methods might not find all 10 counterfactual explanations.

2. If several configurations of hyperparameter values find the same number of counterfactual explanations, the dissimilarity with respect to the instance being explained is used to break the tie. In particular, we use a convex combination of the proximity dissimilarity and the sparsity dissimilarity. In case of a tie, we choose the best configuration at random.

Tables 2a, 2b, 2c and 2d show the hyperparameter search space tested for the considered algorithms, as well as the range of options sampled. Tables 3a, 3b, 3c and 3d show the best combinations of the studied parameters found for each algorithm, dataset, and black box classifier. The rest of the parameters for CEM [9], CEGP [10] and WACH[11] were left at their default values provided by the library.

Parameter	Range	Step
κ	[0.01, 0.2]	0.01
β	[0.01, 0.4]	0.01
c_init	[1, 10]	1
c_steps	[10, 50]	1

(a) CEM Random Search Parameters

Parameter	Range	Step
β	[0.01, 0.4]	0.01
c_init	[1, 10]	1
c_steps	[10, 50]	1

(b) CEGP Random Search Parameters

Parameter	Range	Step
tol	[0.01, 0.05]	0.01
λ_init	[0.01, 0.1]	0.01
max_λ_steps	[10, 50]	1
lr_init	[0.01, 0.2]	0.01

(c) WACH Random Search Parameters

Parameter	Values
max_depth	[1,2,3,4,5,inf]
neigh_size	[100,500,1000,1500,2000]
min_examples	[1,3,5,7,10]

(d) FLARE Random Search Params

[9] https://docs.seldon.io/projects/alibi/en/stable/methods/CEM.html
[10] https://docs.seldon.io/projects/alibi/en/stable/methods/CFProto.html
[11] https://docs.seldon.io/projects/alibi/en/stable/methods/CF.html

Classifier	Dataset	κ	β	c_init	c_steps
SVM	adult	0.06	0.33	6	26
	compas	0.06	0.2	3	21
	german	0.02	0.36	9	32
	fico	0.19	0.37	1	38
NN	adult	0.02	0.23	5	37
	compas	0.06	0.2	3	21
	german	0.06	0.23	5	34
	fico	0.06	0.2	3	21

(a) CEM Final Parameters

Classifier	Dataset	β	c_init	c_steps
SVM	adult	0.29	8	32
	compas	0.24	4	30
	german	0.25	4	18
	fico	0.2	5	32
NN	adult	0.36	1	33
	compas	0.06	3	40
	german	0.39	4	38
	fico	0.37	1	38

(b) CEGP Final Parameters

Classifier	Dataset	tol	λ_init	max_λ_steps	lr_init
SVM	adult	0.04	0.5	32	0.12
	compas	0.02	0.5	25	0.01
	german	0.03	0.6	32	0.02
	fico	0.04	0.5	35	0.04
NN	adult	0.04	0.9	18	0.17
	compas	0.03	0.9	27	0.1
	german	0.01	0.6	10	0.04
	fico	0.01	0.6	10	0.04

(c) WACH Final Parameters

Classifier	Dataset	max_depth	neigh_size	min_examples
SVM	adult	3	500	5
	compas	3	500	3
	german	3	500	3
	fico	4	2000	5
NN	adult	3	500	1
	compas	inf	500	1
	german	3	500	10
	fico	4	1500	10

(d) FLARE Final Parameters