Local Rule-Based Explanations of Black Box Decision Systems

Riccardo Guidotti
ISTI-CNR & University of Pisa, Italy
riccardo.guidotti@isti.cnr.it

Anna Monreale
University of Pisa, Italy
anna.monreale@unipi.it

Salvatore Ruggieri
University of Pisa, Italy
salvatore.ruggieri@unipi.it

Dino Pedreschi
University of Pisa, Italy
dino.pedreschi@unipi.it

Franco Turini
University of Pisa, Italy
franco.turini@unipi.it

Fosca Giannotti
ISTI-CNR of Pisa, Italy
fosca.giannotti@isti.cnr.it

ABSTRACT

The recent years have witnessed the rise of accurate but obscure decision systems which hide the logic of their internal decision processes to the users. The lack of explanations for the decisions of black box systems is a key ethical issue, and a limitation to the adoption of machine learning components in socially sensitive and safety-critical contexts. In this paper we focus on the problem of black box outcome explanation, i.e., explaining the reasons of the decision taken on a specific instance. We propose **LORE**, an agnostic method able to provide interpretable and faithful explanations. **LORE** first leans a local interpretable predictor on a synthetic neighborhood generated by a genetic algorithm. Then it derives from the logic of the local interpretable predictor a meaningful explanation consisting of: a decision rule, which explains the reasons of the decision; and a set of counterfactual rules, suggesting the changes in the instance's features that lead to a different outcome. Wide experiments show that **LORE** outperforms existing methods and baselines both in the quality of explanations and in the accuracy in mimicking the black box.

CCS CONCEPTS

• **Information systems** → **Decision support systems**; **Data mining**; **Data analytics**;

KEYWORDS

Explanation, Decision Systems, Rules

1 INTRODUCTION

Popular magazines and newspapers are full of commentaries about algorithms taking critical decisions that heavily impact on our life and society, from granting a loan to finding a job or driving our car. The worry is not only due to the increasing automation of decision making, but mostly to the fact that the algorithms are opaque and their logic unexplained. The main cause for this lack of transparency is that often the algorithm itself has not been directly coded by a human but it has been generated from data through machine learning. Machine learning allows building predictive models which map user features into a class (outcome or decision), obtained by generalizing from a training set of examples. This learning process is made possible by the digital records of past decisions and classification outcomes, typically provided by human experts and decision makers. The process of inferring a classification model from examples cannot be controlled step by step because the size of training data and the complexity of the learned model are too big for humans. This is how we got trapped in a paradoxical situation in which, on one side, the legislator defines new regulations requiring that automated decisions should be explained to affected people[1] while, on the other side, even more sophisticated and obscure algorithms for decision making are generated [16, 37].

The lack of transparency in algorithms generated through machine learning grants to them the power to perpetuate or reinforce forms of injustice by learning bad habits from the data. In fact, if the training data contains a number of biased decision records, or misleading classification examples due to data collection mistakes or artifacts, it is likely that the resulting algorithm inherits the biases and recommends discriminatory or simply wrong decisions[2] [6, 7]. The inability of obtaining an explanation for what one considers a biased decision is a profound drawback of learning from big data, limiting social acceptance and trust on its adoption in many sensitive contexts. Starting from [29] a rich literature has been flourishing on discrimination discovery and avoidance. Some of the ideas developed in that context can be reinterpreted for addressing the more general problem of explaining the logic driving a decision taken by an obscure algorithm, which is precisely the problem tackled in this paper.

In particular, in this paper we address the problem of explaining the decision outcome taken by an obscure algorithm by providing "meaningful explanations of the logic involved" when automated decision making takes place, as prescribed by the GDPR. The decision system can be obscure because based on a deep learning approach, or because of inaccessibility of the source code, or other reasons. We perform our research under some specific assumptions. First, we assume that an explanation is interesting for a user if it clarifies why a *specific decision* pertaining that user has been made, i.e., we aim for *local* explanations, not general, *global*, descriptions of how the overall system works [17]. Second, we assume that the vehicle for offering explanations should be as close as possible to the language of reasoning, that is logic. Thus, we are also assuming that the user can understand elementary logic rules. Finally, we assume that the black box decision system can be queried as many times as necessary, to probe its decision behavior to the scope of reconstructing its logic; this is certainly the case in a legal argumentation in court, or in an industrial setting where a company wants to stress-test a machine learning component of a manufactured product, to minimize risk of failures and consequent industrial liability. On the other hand, we make no assumptions on the specific algorithms used in the obscure classifier: we aim at an *agnostic*

[1] We refer here to the so-called "right to explanation" established in the European General Data Protection Regulation (GDPR), entering into force in May 2018.
[2] www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

explanation method, one that works analyzing the input-output behavior of the black box system, disregarding its internals.

We propose a solution to the black box outcome explanation problem suitable for relational, tabular data, called **LORE** (for **LO**cal **R**ule-based **E**xplanations). Given a black box binary predictor b and a specific instance x labeled with outcome y by b, we build a simple, interpretable predictor by first generating a balanced set of neighbor instances of the given instance x through an ad-hoc genetic algorithm, and then extracting from such a set a decision tree classifier. A *local explanation* is then extracted from the obtained decision tree. The local explanation is a pair composed by (*i*) a *logic rule*, corresponding to the path in the tree that explains why x has been labeled as y by b, and (*ii*) a set of *counterfactual rules*, explaining which conditions should be changed by x so to invert the class y assigned by b. For example, from the *compas* dataset [6, 7] we may have the following explanation: the rule $\{age{\leq}39, race{=}African{-}American, recidivist{=}True\}{\rightarrow}High\ Risk$ and the counterfactuals $\{age{>}40\}$, $\{race{=}Native{-}American\}$.

The intuition behind our method, common to other approaches, such as LIME [30], and Anchor [31] is that the decision boundary for the black box can be arbitrarily complex over the whole data space, but in the neighborhood of a data point there is a high chance that the decision boundary is clear and simple, hence amenable to be captured by an interpretable model. The novelty of our method lies in (*i*) a focused procedure, based on genetic algorithm, to explore the decision boundary in the neighborhood of the data point, which produces a high-quality training data to learn the local decision tree, and (*ii*) a high expressiveness of the proposed local explanations, which surpasses state-of-the-art methods providing not only succinct evidence why a data point has been assigned a specific class, but also counterfactuals suggesting what should be different in the vicinity of the data point to reverse the predicted outcome. We propose extensive experiments to assess both quantitatively and qualitatively the accuracy of our explanation method.

In the rest of this paper, after describing the state of the art in the field of explanation of black box decision models (Section 2), we offer a formalization of the problem by defining the notions of *black box outcome explanation*, *explanation through interpretable models*, and *local explanation* (Section 3). We then define our method **LORE** in Section 4. Section 5 is devoted to the experiments, the set up of which requires the definition of appropriate validation measures. We critically compare local versus global explanations, rule-based versus linear explanations, different types of rule-based explanations with respect to the state of the art, and discuss the advantages of genetic algorithms for neighborhood generation. Conclusions and future research directions are discussed in Section 6.

2 RELATED WORK

Recently, the research of methods for explaining black box decision systems has caught much attention [17]. A large number of papers propose approaches for understanding the *global* logic of the black box by providing an interpretable classifier able to mimic the obscure decision system. Generally, these methods are designed for explaining specific black box models, i.e., they are not black box agnostic. Decision trees have been adopted to globally explain neural networks [9, 22] and tree ensembles [18, 34]. Classification rules have been widely used to explain neural networks [2, 3, 21] but also to understand the global behavior of SVMs [15, 27]. Only few methods for global explanation are agnostic with respect to the black box [19, 24]. In the cases in which the training set is available, classification rules are also widely used to avoid building black boxes by directly designing a transparent classifier [17] which is locally or globally interpretable on its own [23, 25, 39].

Other approaches, more related to the one we propose, address the problem of explaining the *local* behavior of a black box [17]. In other words, they provide an explanation for the decision assigned to a specific instance. In this context there are two kinds of approaches: the model-dependent approaches and the agnostic ones. In the first category most of the papers aim at explaining neural networks and base their explanation on saliency masks, i.e., a subset of the instance that explains what is mainly responsible for the prediction [42, 43]. Examples of salient mask are parts of an image, or words or sentences in a text. On the other hand, agnostic approaches provide explanations for any type of black box. In [30] the authors present LIME, which starts from instances randomly generated in the neighborhood of the instance to be explained. The method infers from them linear models as comprehensible local predictors. The importance of a feature in the linear model represents the explanation finally provided to the user. As a limitation of the approach, a random generation of the neighborhood does not take into account density of black box outcomes in the neighborhood instances. Hence, the linear classifiers inferred from them may not correctly characterize outcome values as a function of the predictive features. We will instead use a genetic algorithm that exploits the black box for instance generation.

Extensions of LIME using decision rules (called anchors) and program expression trees are presented in [31] and [33] respectively. [31] uses a bandit algorithm that randomly constructs the anchors with the highest coverage and respecting a precision threshold. [33] adopts a simulated annealing approach that randomly grows, shrinks, or replaces nodes in an expression tree. The neighborhood generation process adopted is the same as in LIME. Another crucial weak point of those approaches, is the need for user-specified parameters for desired explanations: the number of features [30], the level of precision, the maximum expression tree depth [31]. Our approach is instead parameter-free.

Concerning the counterfactual part of our notion of explanation, [38] computes a counterfactual for an instance x by solving an optimization problem over the space of instances. The solution is an instance x' close to x but with different outcome assigned by the black box[3]. Our approach provides a more abstract notion of counterfactuals, consisting of logic rules rather than flips of feature values. Thus, the user is given not only a specific example of how to obtain actionable recourse (e.g., how to improve application for getting a benefit), but also an abstract characterization of its neighboorhood instances with reversed black box outcome.

To the best of our knowledge, in the literature there is no work proposing a black box agnostic method for local decision explanation based on both decision and counterfactual rules.

[3] If instead of a black box, we are given a machine learning model, this problem is known as the *inverse classification problem* [1].

3 PROBLEM AND EXPLANATIONS

Let us start recalling basic notation on classification of tabular data. Afterwards, we define the *black box outcome explanation problem*, and the notion of *explanation* for which we propose a solution.

Classification, black boxes, and interpretable predictors. A *predictor* or classifier, is a function $b : \mathcal{X}^{(m)} \to \mathcal{Y}$ which maps data instances (tuples) x from a feature space $\mathcal{X}^{(m)}$ with m input features to a decision y in a target space $\mathcal{Y}$. We write $b(x) = y$ to denote the decision y predicted by b, and $b(X) = Y$ as a shorthand for $\{b(x) \mid x \in X\} = Y$. We restrict here to binary decisions. An instance x consists of a set of m attribute-value pairs (a_i, v_i), where a_i is a feature (or attribute) and v_i is a value from the domain of a_i. The domain of a feature can be continuous or categorical. A predictor can be a machine learning model, a domain-expert rule-based system, or any combination of algorithmic and human knowledge processing. We assume that a predictor is available as a software function that can be queried at will. In the following, we denote by b a *black box* predictor, whose internals are either unknown to the observer or they are known but uninterpretable by humans. Examples include neural networks, SVMs, ensemble classifiers, or a composition of data mining, legacy software, and hard-coded expert systems. Instead, we denote with c an *interpretable* predictor, whose internal processing yielding a decision $c(x) = y$ can be given a symbolic interpretation understandable by a human. Examples of such predictors include rule-based classifiers, decision trees, decision sets, and rational functions.

Black Box Outcome Explanation. Given a black box predictor b and an instance x, the *black box outcome explanation problem* consists in providing an explanation e for the decision $b(x) = y$. We approach the problem by learning an interpretable predictor c that reproduces and accurately mimes the *local* behavior of the black box. An explanation of the decision is then derived from c. By *local*, we mean focusing on the behavior of the black box in the neighborhood of the specific instance x, without aiming at providing a single description of the logic of the black box for all possible instances. The neighborhood of x is not given, but rather it has to be generated as part of the explanation process. However, we assume that some knowledge is available about the characteristics of the feature space $\mathcal{X}^{(m)}$, in particular the ranges of admissible values for the domains of features and, possibly, the (empirical) distribution of features. Nothing is instead assumed about the process of constructing the black box b. Let us formalize the problem, and the approach based on interpretable models.

Definition 3.1 (Black Box Outcome Explanation). Let b be a black box, and x an instance whose decision $b(x)$ has to be explained. The *black box outcome explanation problem* consists in finding an explanation $e \in E$ belonging to a human-interpretable domain E.

Definition 3.2 (Explanation Through Interpretable Models). Let $c = \zeta(b, x)$ be an interpretable predictor derived from the black box b and the instance x using some process $\zeta(\cdot, \cdot)$. An explanation $e \in E$ is obtained through c, if $e = \varepsilon(c, x)$ for some explanation logic $\varepsilon(\cdot, \cdot)$ which reasons over c and x.

One point is still missing: which is a comprehensible domain E of explanations? We will define an explanation e as a pair of objects:

$$e = \langle r = p \to y, \Phi \rangle$$

The first component $r = p \to y$ is a decision rule describing the reason for the decision value $y = c(x)$ The second component Φ is a set of counterfactual rules, namely the minimal number of changes in the feature values of x that would reverse the decision of the predictor. Let us consider as an example the following explanation for a loan request for user $x = \{(age{=}22), (job = none), (amount{=}10k), (car{=}no)$:

$$\begin{aligned} e = \langle r = \{&age{\leq}25, job{=}none, amount{>}5k\} \to deny, \\ \Phi = \{(\{&age{>}25, amount{\leq}5k\} \to grant), \\ &(\{job{=}clerk, car{=}yes\} \to grant)\}\rangle \end{aligned}$$

Here, the decision *deny* is due to the age lower than 25, the absence of job and an amount greater than 5k (see component r). For changing the decision instead it is required either an age higher than 25 and a smaller amount, or owning a clerk job and a car (see component Φ). Details are provided in the rest of the section.

In a *decision rule* (simply, a rule) r of the form $p \to y$, the decision y is the *consequence* of the rule, while the *premise* p is a boolean condition on feature values. We assume that p is the conjunction of split conditions sc of the form $a \in [v_1, v_2]$, where a is a feature and v_1, v_2 are values in the domain of a extended with[4] $\pm\infty$. An instance x *satisfies* r, or r *covers* x, if the boolean condition p evaluates to true for x, i.e., if $sc(x)$ is true for every $sc \in p$. For example, the rule $\{age{\leq}25, job{=}none\} \to deny$ is satisfied by $x_0 = \{(age{=}22), (job{=}none)\}$ and not satisfied by $x_1 = \{(age{=}22), (job{=}clerk)\}$. We say that r *is consistent* with c, if $c(x) = y$ for every instance x that satisfies r. Consistency means that the rule specifies some conditions for which the predictor makes a specific decision. When the instance x for which we have to explain the decision satisfies p, the rule $p \to y$ represents *a motivation for taking* a decision value, i.e., p locally explains why b returned y.

Consider now a set δ of split conditions. We denote the update of p by δ as $p[\delta] = \delta \cup \{(a \in [v_1, v_2]) \in p \mid \nexists [w_1, w_2], (a \in [w_1, w_2]) \in \delta\}$. Intuitively, $p[\delta]$ is the logical condition p with ranges for attributes overwritten as stated in δ, e.g. $\{age{\leq}25, job{=}none\}[age{>}25]$ is $\{age > 25, job{=}none\}$. A *counterfactual rule* for p is a rule of the form $p[\delta] \to \hat{y}$, for $\hat{y} \neq y$. We call δ a *counterfactual*. Consistency is meaningful also for counterfactual rules, meaning that the rule is an instance of the decision logic of c. A counterfactual δ describes *what* features to change and *how* to change them to get an outcome different from y. Since c predicts either y or $\hat{y}$, if such changes are applied to the given instance x, the predictor c will return a different decision. Continuing the example before, changing the age feature of x_0 to any value greater than 25 will change the predicted outcome of c from *deny* to *grant*. An expected property of a consistent counterfactual rule $p[\delta] \to \hat{y}$ is that it should be minimal w.r.t. x. Minimality

[4] Using $\pm\infty$ we can model with a single notation typical univariate split conditions, such as equality ($a = v$ as $a \in [v, v]$), upper bounds ($a \leq v$ as $a \in [-\infty, v]$), strict lower bounds ($a > v$ as $a \in [v + \epsilon, \infty]$ for a sufficiently small ϵ). However, since our method is parametric to a decision tree induction algorithm, split conditions can also be multivariate, e..g, $a \leq b + v$ for a, b features (as in oblique decision trees).

Algorithm 1: *LORE*(x, b)

Input : x - instance to explain, b - black box, N - # of neighbors
Output: e - explanation of x

1 $G \leftarrow 10$; $pc \leftarrow 0.5$; $pm \leftarrow 0.2$; // init. parameters
2 $Z_{=} \leftarrow GeneticNeigh(x, fitness^{x}_{=}, b, N/2, G, pc, pm)$ // generate neigh.
3 $Z_{\neq} \leftarrow GeneticNeigh(x, fitness^{x}_{\neq}, b, N/2, G, pc, pm)$ // generate neigh.
4 $Z \leftarrow Z_{=} \cup Z_{\neq}$; // merge neighborhoods
5 $c \leftarrow BuildTree(Z)$; // build decision tree
6 $r = (p \rightarrow y) \leftarrow ExtractRule(c, x)$; // extract decision rule
7 $\Phi \leftarrow ExtractCounterfactuals(c, r, x)$; // extract counterfactuals
8 **return** $e = \langle r, \Phi \rangle$;

is measured[5] w.r.t. the number of split conditions in $p[\delta]$ not satisfied by x. Formally, we define $nf(p[\delta], x) = |\{sc \in p[\delta] \mid \neg sc(x)\}|$ (where $nf(\cdot,\cdot)$ stands for the number of falsified split conditions), and, when clear from the context, we simply write nf. For example, $\{age < 25, job{=}clerk\} \rightarrow grant$ is a counterfactual with two conditions falsified by x_0. It is not minimal if the counterfactual $\{age{>}25, job{=}none\} \rightarrow \hat{y}$, with only one falsified condition, is consistent for c. In summary, a counterfactual rule $p[\delta] \rightarrow \hat{y}$ is a (minimal) *motivation for reversing* a decision value.

We are now in the position to formally introduce the notion of explanation that we are able to provide.

Definition 3.3 (Local Explanation). Let x be an instance, and $c(x) = y$ be the decision of c. A local explanation $e = \langle r, \Phi \rangle$ is a pair of: a decision rule $r = (p \rightarrow y)$ consistent with c and satisfied by x; and, a set $\Phi = \{p[\delta_1] \rightarrow \hat{y}, \ldots, p[\delta_v] \rightarrow \hat{y}\}$ of counterfactual rules for p consistent with c.

This definition completes the elements of the black box outcome explanation problem. A solution to the problem will then consists of: *(i)* computing an interpretable predictor c for a given black box b and an instance x, i.e., defining the function $\zeta(\cdot,\cdot)$ according to Definition 3.2; *(ii)* deriving a local explanation from c and x, i.e., defining the explanation logic $\varepsilon(\cdot,\cdot)$ according to Definition 3.2.

4 PROPOSED METHOD

We propose **LORE** (**LO**cal **R**ule-based **E**xplanations, Algorithm 1) as a solution to the black box outcome explanation problem. An interpretable predictor c is built for a given black box b and instance x by first generating a set of N neighbor instances of x through a *genetic algorithm*, and then extracting from such a set a *decision tree* c. A local explanation, consisting of a single rule r and a set of counterfactual rules Φ, is then derived from the structure of c.

4.1 Neighborhood Generation

The goal of this phase is to identify a set of instances Z, with feature characteristics close to the ones of x, that is able to reproduce the local decision behavior of the black box b. Since the objective is to learn a predictor, the neighborhood should be flexible enough to include instances with both decision values, namely $Z = Z_{=} \cup Z_{\neq}$

[5]Such a measure can be extended to exploit additional knowledge on the feature domains in order not to generate invalid or unrealistic rules. E.g., the split condition $age \leq 25$ appears closer than $age > 30$ for an instance with $age = 26$. However, it is not actionable: an individual cannot lower her age, or change her race or gender.

Algorithm 2: *GeneticNeigh*$(x, fitness, b, N, G, pc, pm)$

Input : x - instance to explain, b - black box, *fitness* - fitness function, N - population size, G - # of generations, pc - crossover probability, pm - mutation probability
Output: Z - neighbors of x

1 $P_0 \leftarrow \{x | \forall 1 \ldots N\}$; $i \leftarrow 0$; // population init.
2 $evaluate(P_0, fitness, b)$; // evaluate population
3 **while** $i < G$ **do**
4 $P_{i+1} \leftarrow select(P_i)$; // select sub-population
5 $P'_{i+1} \leftarrow crossover(P_{i+1}, pc)$; // mix records
6 $P''_{i+1} \leftarrow mutate(P'_{i+1}, pm)$; // perform mutations
7 $evaluate(P''_{i+1}, fitness, b)$; // evaluate population
8 $P_{i+1} = P''_{i+1}$; $i \leftarrow i + 1$ // update population
9 **end**
10 $Z \leftarrow P_i$ **return** Z;

where instances $z \in Z_{=}$ are such that $b(z) = b(x)$, and instances $z \in Z_{\neq}$ are such that $b(z) \neq b(x)$. In Algorithm 1, we extract balanced subsets $Z_{=}$ and $Z_{\neq}$ (lines 2–3), and then put $Z = Z_{=} \cup Z_{\neq}$ (line 4). This task differs from approaches to instance *selection* [28], based on genetic algorithms [36] (also specialized for decision trees [40]), in that their objective is to select a subset of instances from an available training set. In our case, instead we cannot assume that the training set used to train b is available, or not even that b is a supervised machine learning predictor for which a training set exists. Our task is instead similar to instance *generation* in the field of active learning [14], also including evolutionary approaches [10]. We adopt an approach based on a *genetic algorithm* which generates $z \in Z_{=} \cup Z_{\neq}$ by maximizing the following fitness functions:

$$fitness^{x}_{=}(z) = I_{b(x)=b(z)} + (1 - d(x,z)) - I_{x=z}$$

$$fitness^{x}_{\neq}(z) = I_{b(x)\neq b(z)} + (1 - d(x,z)) - I_{x=z}$$

where $d : \mathcal{X}^{(m)} \rightarrow [0,1]$ is a distance function, $I_{true} = 1$, and $I_{false} = 0$. The first fitness function looks for instances z similar to x (term $1 - d(x,z)$), but not equal to x (term $I_{x=z}$) for which the black box b produces the same outcome as x (term $I_{b(x)=b(z)}$). The second one leads to the generation of instances z similar to x, but not equal to it, for which b returns a different decision. Intuitively, for an instance z_0 such that $b(x) \neq b(z_0)$ and $x \neq z_0$, it turns out $fitness^{x}_{=}(z_0) < 1$. For any instance z_0 such that $b(x) = b(z_0)$, instead, we have $fitness^{x}_{=}(z_0) \geq 1$. Finally, $fitness^{x}_{=}(x) = 1$. Thus, maximization of $fitness^{x}_{=}(\cdot)$ occurs necessarily for instances different from x and whose prediction is equal to $b(x)$.

Like neural networks, genetic algorithms [20] are based on the biological metaphor of evolution. They have three distinct aspects. *(i)* The potential solutions of the problem are encoded into representations that support the *variation* and *selection* operations. In our case these representations, generally called chromosomes, correspond to instances in the feature space $\mathcal{X}^m$. *(ii)* A fitness function evaluates which chromosomes are the "best life forms", that is, most appropriate for the result. These are then favored in *survival* and *reproduction*, thus shaping the next generation according to the fitness function. In our case, these instances correspond to those similar to x, according to distance $d(\cdot,\cdot)$, and with the same/different outcome returned by the black box b depending on the fitness function $fitness^{x}_{=}$ or $fitness^{x}_{\neq}$. *(iii)* Mating (called crossover) and mutation

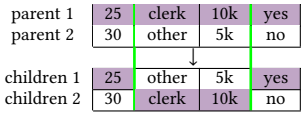


Figure 1: Crossover.

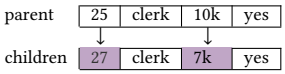


Figure 2: Mutation

produce a new generation of chromosomes by recombining features of their parents. The final generation of chromosomes, according to a stopping criterion, is the one that best fit the solution.

Algorithm 2 generates the neighborhoods $Z_=$ and $Z_\neq$ of x by instantiating the evolutionary approach of [4]. Using the terminology of [10], it is an instance of generational genetic algorithms for evolutionary prototype generation. However, prototypes are a condensed subset of a training set that enable optimization in predictor learning. We aim instead at generating new instances that separate well the decision boundary of the black box b. The usage of classifiers within fitness functions of genetic algorithms can be found in [5, 8, 12, 41]. However, the classifier they use is always the one for which the population must be selected or generated for and not another one (the black box) like in our case. Algorithm 2 first initializes the population P_0 with N copies of the instance x to explain. Then it enters the evolution loop that begins by *selection* of the P_{i+1} population having the highest fitness score. After that, the crossover operator is applied on a proportion of P_{i+1} according to the pc probability, the resulting and the untouched individuals are placed in P'_{i+1}. We use a *two-point crossover* which selects two parents and two crossover features at random, and then swap the crossover feature values of the parents (see Figure 1). Thereafter, a proportion of P'_{i+1}, determined by pm, is mutated and placed in P''_{i+1}. The unmutated individuals are also added to P''_{i+1}. Mutation consists of replacing features values at random according to the empirical distribution[6] of a feature (see Figure 2). Individuals in P''_{i+1} are evaluated according to the fitness function, and the evolution loop continues until G generations are completed. Finally, the best individuals according to the fitness function are returned. Algorithm 2 is run twice, once using the fitness function $fitness^x_=$ to derive neighborhood instances $Z_=$ with the same decision as x, and once using $fitness^x_\neq$ to derive neighborhood instances $Z_\neq$ with different decision as x. Finally, we set $Z = Z_= \cup Z_\neq$.

Figure 3 shows an example of neighborhood generation for a black box consisting of a random forest model and a bi-dimensional feature space. The figure contrasts uniform random generation around a specific instance x (starred) to our genetic approach. The latter yields a neighborhood that is denser in the boundary region of the predictor. The density of generated instances will be a key factor in extracting local (interpretable) predictors.

Distance Function. A key element in the definition of the fitness functions is the distance $d(x, z)$. We account for the presence of mixed types of features by a weighted sum of simple matching coefficient for categorical features, and of the normalized Euclidean distance[7] for continuous features. Formally, assuming h categorical features and $m - h$ continuous ones, we use:

$$d(x, z) = \frac{h}{m} \cdot SimpleMatch(x, z) + \frac{m-h}{m} \cdot NormEuclid(x, z).$$

[6] In experiments, we derive such a distribution from the test set of instances to explain.
[7] reference.wolfram.com/language/ref/NormalizedSquaredEuclideanDistance.html

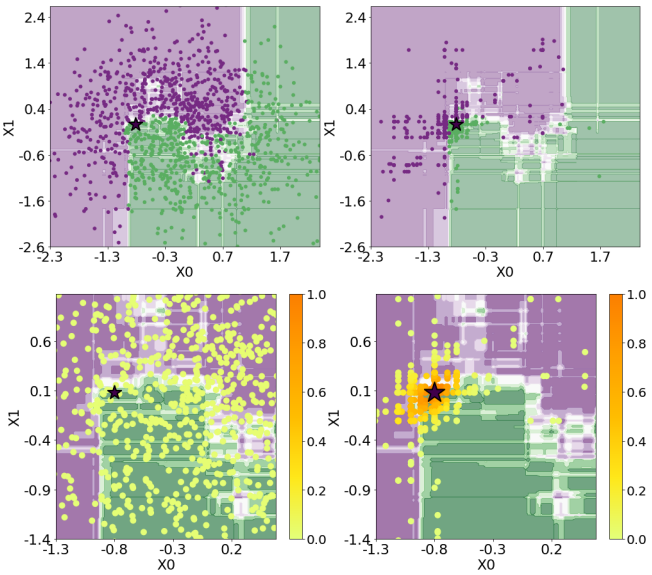


Figure 3: Random forest black box: purple vs green decision. Starred instance x. *(Top)* Uniformly random (left) and genetic generation (right). *(Bottom)* Density of random (left) and genetic (right) generation. (Best view in color).

Our approach is parametric to $d(\cdot, \cdot)$, and it can readily be applied to improved heterogeneous distance functions [26]. Empirical results with different distance functions are reported in Section 5.3.

4.2 Local Rule-Based Classifier and Explanation Extraction

Given the neighborhood Z of x, the second step is to build an interpretable predictor c trained on the instances $z \in Z$ labeled with the black box decision $b(z)$. Such a predictor is intended to mimic the behavior of b locally in the Z neighborhood. Also, c must be interpretable, so that an explanation for x (decision rule and counterfactuals) can be extracted from it. The **LORE** approach considers decision tree classifiers due to the following reasons: *(i)* decision rules can naturally be derived from a root-leaf path in a decision tree; and, *(ii)* counterfactuals can be extracted by symbolic reasoning over a decision tree. For a decision tree c, we derive an explanation $e = \langle r, \Phi \rangle$ as follows. The decision rule $r = (p \rightarrow y)$ is formed by including in p the split conditions on the path from the root to the leaf node that is satisfied by the instance x, and setting $y = c(x)$. By construction, the rule r is consistent with c and satisfied by x. Consider now the counterfactual rules in Φ. Algorithm 3 looks for all paths in the decision tree c leading to a decision $\hat{y} \neq y$. Fix one of such paths, and let q be the conjunction of split conditions in it. Again by construction, $q \rightarrow \hat{y}$ is a counterfactual rule consistent with c. Notice that the counterfactual δ for which $q = p[\delta]$ has not to be explicitly computed[8] – this is a benefit of using decision trees. Among all such q's, only the ones with minimum number

[8] However, it can be done as follows. Consider the path from the leaf of p to the leaf of q. When moving from a child to a father node, we retract the split condition. E.g., $a \leq v_2$ is retracted from $\{a \in [v_1, v_2]\}$ by adding $a \in [v_1, +\infty]$ to δ. When moving from a father node to a child, we add the split condition to δ.

Algorithm 3: *ExtractCounterfactuals*(c, r, x)

Input : c - decision tree, r - rule $(p \rightarrow y)$, x - instance to explain
Output: Φ - set of counterfactual rules for p

1 $Q \leftarrow GetPathsWithDifferentLabel(c, y)$; // get paths with label $\hat{y} \neq y$
2 $\Phi \leftarrow \emptyset$; $min \leftarrow +\infty$; // init. counterfactual set
3 **for** $q \in Q$ **do**
4 $\quad qlen \leftarrow nf(q, x) = |\{sc \in q \mid \neg sc(x)\}|$
5 $\quad$ **if** $qlen < min$ **then** $\Phi \leftarrow \{q \rightarrow \hat{y}\}$; $min \leftarrow qlen$;
6 $\quad$ **else if** $qlen = min$ **then** $\Phi \leftarrow \Phi \cup \{q \rightarrow \hat{y}\}$;
7 **end**
8 **return** Φ;

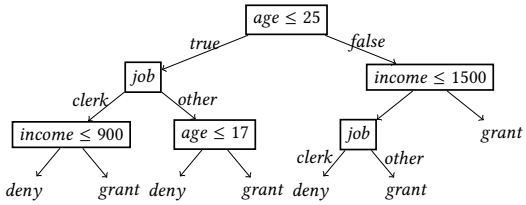


Figure 4: Example decision tree.

of split conditions sc not satisfied by x (line 4 of Algorithm 3) are kept in Φ. As an example, consider the decision tree in Figure 4, and the instance $x = \{(age, 22), (job, clerk), (income, 800)\}$ for which the decision *deny* (e.g., of a loan) has to be explained. The path followed by x is the leftmost one in the tree. The decision rule extracted from the path is $\{age \leq 25, job{=}clerk, income \leq 900\} \rightarrow deny$. There are four paths leading to the opposite decision: $q_1 = \{age \leq 25, job{=}clerk, income > 900\}$, $q_2 = \{17 < age \leq 25, job = other\}$, $q_3 = \{age > 25, income \leq 1500, job = other\}$, and $q_4 = \{age > 25, income > 1500\}$. It turns out: $nf(q_1, x) = 1$, $nf(q_2, x) = 1$, $nf(q_3, x) = 2$, and $nf(q_4, x) = 2$. Thus, $\Phi = \{q_1 \rightarrow deny, q_2 \rightarrow deny\}$.

As a further output, **LORE** computes a *counterfactual instance* starting from a counterfactual rule $q \rightarrow \hat{y}$ and x. Among all possible instances that satisfy q, we choose the one that minimally changes attributes from x according to q. This is done by looking at the split conditions falsified by x: $\{sc \in q \mid \neg sc(x)\}$, and modifying the lower/upper bound in sc that is closer to the value in x. As an example, for the above path q_1, the counterfactual instance of x is $\{(age, 22), (job, clerk), (income, 900 + \epsilon)\}$, and for q_2 is $\{(age, 22), (job, other), (income, 800)\}$.

5 EXPERIMENTS

LORE has been developed in Python[9], using, for the genetic neighborhood generation, the `deap` library [13], and for decision tree induction (the interpretable predictor), the `yadt` system [32], which is a C4.5 implementation with multi-way splits of categorical attributes. After presenting the experimental setup, we report next: *(i)* some analyses on the effect of the genetic algorithm parameters for the neighborhood generation; *(ii)* evidence that the local genetic neighborhood is more effective than a global approach; *(iii)* a qualitative and quantitative comparison with naïve baselines and state of the art competitors.

[9]The source code and datasets will be available at *anonimized url*. The experiments were performed on Ubuntu 16.04.1 LTS 64 bit, 32 GB RAM, 3.30GHz Intel Core i7

5.1 Experimental Setup

We ran experiments on three real-world *tabular* datasets: *adult*, *compas* and *german*[10]. In each of them, an instance represents attributes of an individual person. All datasets includes both categorical and continuous features.

The *adult* dataset from UCI Machine Learning Repository, includes 48, 842 instances with demographic information like age, workclass, marital-status, race, capital-loss, capital-gain etc. The income divides the population into two classes "<=50K" and ">50K".

The *compas* dataset from ProPublica contains the features used by the COMPAS algorithm for scoring defendants and their risk (Low, Medium and High), for over 10, 000 individuals. We considered two classes "Low-Medium" and "High" risk, and we use the following features: age, sex, race, priors_count, days_b_screening_arrest, length_of_stay, c_charge_degree, is_recid.

In the *german* dataset from UCI Machine Learning Repository each person of the 1, 000 entries is classified as a "good" or "bad" creditor according to attributes like age, sex, checking_account, credit_amount, duration, purpose, etc.

We experimented the following predictors as black boxes: support vector machines with RBF kernel (**SVM**), random forests with 100 trees (**RF**), and multi-layer neural networks with 'lbfgs' solver (**NN**). Implementations of the predictors are from the `scikit-learn` library. Unless differently stated, default parameters were used for both the black boxes and the libraries of **LORE**. Missing values were replaced by the mean for continuous features and by the mode for categorical ones.

Each dataset was randomly split into train (80% instances), and test (20% instances). The former is used to train black box predictors. The latter, denoted by X, is the set of instances for which the black box decision have to be explained. In the following, for some fixed set of instances, we denote by Y the set of decisions provided by the interpretable predictor c and by $\hat{Y}$ the decisions provided by the black box b on the same set.

We consider the following properties in evaluating the mimic performances of the decision tree c inferred by **LORE** and of the explanations returned by it against the black box classifier b:

- ***fidelity***$(Y, \hat{Y}) \in [0, 1]$. It compares the predictions of c and of the black box b on the instances Z used to train c [11]. It answer the question: how good is c at mimicking b?
- ***l-fidelity***$(Y, \hat{Y}) \in [0, 1]$. It compares the predictions of c and b on the instances Z_x covered by the decision rule in a local (hence "l-") explanation for x. It answers the question: how good is the decision rule at mimicking b?
- ***cl-fidelity***$(Y, \hat{Y}) \in [0, 1]$. It compares the predictions of c and b on the instances Z_x covered by the counterfactual rules in a local explanation for x.
- ***hit***$(y, \hat{y}) \in \{0, 1\}$. It compares the predictions of c and b on the instance x under analysis. It returns 1 if $y = c(x)$ is equal to $\hat{y} = b(x)$, and 0 otherwise.
- ***c-hit***$(y, \hat{y}) \in \{0, 1\}$. It compares the predictions of c and b on a counterfactual instance of x built from counterfactual rules in a local explanation of x.

[10]https://archive.ics.uci.edu/ml/datasets/adult, https://github.com/propublica/compas-analysis, https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

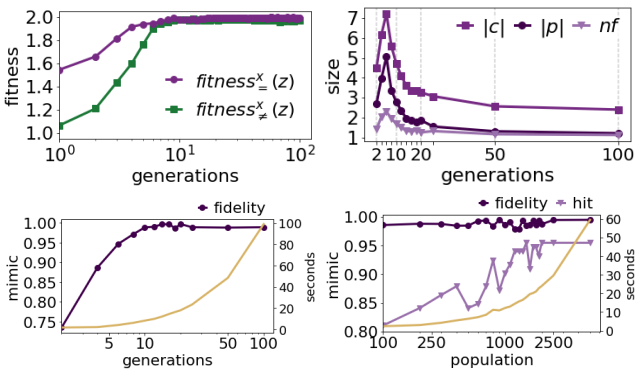


Figure 5: Impact of the number of generations G and of population size N parameters of the genetic neighborhood generation. Bottom plots also report elapsed running times.

Distance	*hit*	*fidelity*	*l-fidelity*	$\|c\|$	$\|p\|$	*nf*
cosine	.938±.24	**.976±.11**	.936±.24	4.4±2.5	**2.1±1.8**	1.9±1.0
minmax	.958±.19	.965±.15	.956±.17	4.5±2.7	2.3±2.3	**1.8±0.9**
neuclid	**.966±.17**	.967±.15	**.963±.19**	**4.3±2.6**	2.2±2.1	1.8±1.0

Table 1: Comparison of distance measures.

We measure the first three of them by the f1-measure [35]. Aggregated values of f1 and hit/c-hit are reported by averaging them over the the set of test instances $x \in X$.

5.2 Analysis of Neighborhood Generation

We analyze here the impact of the number of generations G and size of neighborhood N on the performances of instance generation and on the size complexity of the **LORE** output. We report only results for *german* dataset, since we get similar results for the other ones. The other parameters of Algorithm 2 (probabilities of crossover *pc* and mutation *pm*) are set with the default values of 0.5 and 0.2 respectively [4]. Figure 5 shows in the top plots the value of fitness functions and measures of sizes of local classifier c (decision tree depth), of decision rule (size of the antecedent p), and of counterfactual rules (number *nf* of falsified split conditions). The bottom plots show fidelity (f1-measure) and hit (rate) as well as running times of neighborhood generation. Fixed $N = 1000$, after 10 generations, the fitness function converges around the optimal value (top left), fidelity is almost maximized (bottom left), and also the measures of sizes (top right) become stable and small. We then set G=10 in all other experiments. Figure 5-(bottom right) shows instead that the size N of the neighborhood instances to be generated is relevant for the *hit* rate but not for *fidelity*. By taking into account also the running time (right side scale of the bottom plots), a good trade-off is obtained by setting N=1000.

5.3 Comparing Distance Functions

A key element of the neighborhood generation is the distance function used by the genetic algorithm. A legitimate question is whether the results of the approach are affected by the choice of the distance function adopted (see Section 4.1). For instance, [38] presents considerable differences in their output of counterfactual instance

Dataset	*Method*	*hit*	*fidelity*	*l-fidelity*	*tree depth*
adult	lore	**.912 ± .29**	**.959 ± .17**	**.892 ± .29**	**4.16 ± 0.21**
	global	.901 ± .28	.750 ± .00	.873 ± .27	12.00 ± 0.00
compas	lore	**.942 ± .23**	**.992 ± .03**	**.937 ± .23**	**4.72 ± 2.15**
	global	.902 ± .29	.935 ± .00	.857 ± .29	12.00 ± 0.00
german	lore	**.925 ± .26**	**.988 ± .07**	**.920 ± .26**	**4.95 ± 2.54**
	global	.880 ± .32	.571 ± .00	.824 ± .31	6.00 ± 0.00

Table 2: Local vs global approach.

varying the choice of the distance in their stochastic optimization approach. Table 1 reports basic measures contrasting the *normalized Euclidean* distance adopted by **LORE** with *cosine* and *min-max* distance on *german* dataset. The table does not highlight any considerable difference. This can be justified by the fact that, following instance generation, there are phases, such as decision tree building, that abstract instances to patterns, resulting in resilience against variability due to the distance function adopted.

5.4 Validation of Local Explanations

We now compare our local approach with a global approach, and discuss alternative neighborhood instance generation methods.

Local vs Global Explanations. Extracting a predictor from the neighborhood of an instance is a winning strategy, if contrasted to an approach that builds a single predictor from all instances in the test set, i.e., $Z = X$. In particular, this means that the interpretable predictor will be the same for all instances in the test set. Let compare our approach with such a `global` approach. Table 2 reports the mean and standard deviation values of *hit*, *fidelity*, *fairness* and *tree depth* for each dataset aggregating over the results of the various black boxes. While for *hit* both **LORE** and `global` obtain similar high performances, for the other scores **LORE** considerably overtakes `global`. In particular, the size and depth of the decision tree of the global approach may lead to explanations (decision rules and counterfactuals) more complex to understand than those returned by the proposed local approach **LORE**.

Comparing Neighborhood Generations. After concluding that "local is better than global", we now show that our genetic programming approach improves over the following baselines in the generation of neighborhoods:

- `crn` returns as Z the $k = 100$ instances from X (the test set) that are *closest* to x;
- `rnd` augment the output of `crn` with additional randomly generated instances so that a stratified Z is obtained;
- `ris` starting from the output of `rnd` performs the instance selection procedure[11] *CNN* [28];
- `ros` starting from the output of `rnd` performs a random oversampling to balance the decision outcomes in Z.

Table 3 reports the aggregated evaluation measures over the various black boxes and datasets. **LORE** overtook the performance of all the other neighbors generators. Intuitively, this means that **LORE**'s genertic programming approach contributes more than the other methods in capturing/explaining the behavior of the black box, both for direct and counterfactual decisions. Such a

[11] http://contrib.scikit-learn.org/imbalanced-learn

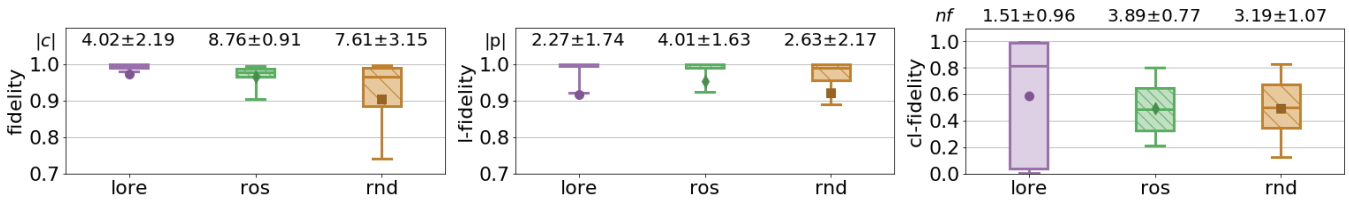


Figure 6: Comparison of neighborhood generations methods.

Method	*hit*	*fidelity*	*l-fidelity*	*c-hit*	*cl-fidelity*
lore	.962 ± .19	**.993 ± .04**	**.959 ± .19**	**.588 ± .42**	**.756 ± .40**
crd	.924 ± .26	.855 ± .23	.894 ± .25	.349 ± .26	.583 ± .48
rnd	.946 ± .22	.904 ± .15	.920 ± .22	.494 ± .24	.712 ± .40
ris	.916 ± .27	.869 ± .05	.870 ± .26	.501 ± .22	.708 ± .39
ros	**.968 ± .17**	.965 ± .03	.953 ± .17	.491 ± .22	.733 ± .34

Table 3: Comparison of neighborhood generations methods.

Dataset	german		compass		adult	
Black Box	lore	lime	lore	lime	lore	lime
RF	**.925 ± .2**	.880 ± .3	**.941 ± .2**	.826 ± .4	**.901 ± .3**	.824 ± .4
NN	.980 ± .1	**1.00 ± .0**	**.987 ± .1**	.902 ± .3	.918 ± .3	**.998 ± .1**
SVM	**1.00 ± .0**	.966 ± .1	**.997 ± .1**	.900 ± .3	.985 ± .1	**.987 ± .1**

Table 4: LORE vs LIME: *hit* scores.

conclusion is reinforced by Figure 6, which shows the box plots of the distributions of *fidelity*, *l-fidelity* and *cl-fidelity*, and some summary data on the size of decision trees ($|c|$), of decision rule premises ($|p|$), and of the number of falsified split conditions in counterfactual rules (nf). **LORE** has the highest mean and median f1-measures (high mimic of the black box), the smallest interquartile ranges (low variability of results), and the lowest complexity sizes. Only for *cl-fidelity* **LORE** has the largest variability, but a median value that is higher than the 90^{th} percentile of the competitors.

5.5 Comparison with the State-of-Art

In this section we compare our approach with the state of the art.

5.5.1 Rules vs Linear Regression for Explanations. We present first a quantitative and qualitative comparison with the linear explanations of LIME[12] [30]. A first crucial difference is that in LIME, the number of features composing an explanation is an input parameter that must be specified by the user. **LORE**, instead, automatically provides the user with an explanation including only the features useful to justify the black box decision. This is a clear improvement over LIME. In experiments, unless otherwise stated, we vary the number of features of LIME explanations from two to ten and we consider the performance with the highest score.

Quantitative Comparison. Table 4 reports the mean and standard deviation of *hit* for each black box predictor and dataset. Moreover, Figure 7 details the box plots of *fidelity* (top) and *l-fidelity* (bottom). Results show that **LORE** definitely outperforms LIME under various viewpoints. Regarding the *hit* score, even when **LORE** is worse than LIME, it has a score close to 1. For RF black box, instead, LIME performs considerably worse than **LORE**. The box plots show that, in addition, **LORE** has better (local) fidelity scores and is more robust than LIME, which, on the contrary, exhibits very high variability in the neighborhood of the instance to explain (i.e., for *l-fidelity*). This can be tracked back to the genetic instance generation of **LORE**. Figure 8 reports a multidimensional scaling of the neighborhood of a sample instance x generated by the two approaches. **LORE** computes a dense and compact neighborhood. The instances generated by LIME, instead, can be very distant from each other and always with a low density around x.

Qualitative Comparison. We claim that the explanations provided by **LORE** are more abstract and comprehensible than the ones of LIME. Consider the example in Figure 9. The top part reports a **LORE** local explanation for an instance x from the `german` dataset. The central part is a LIME explanation. Weights are associated to the categorical values in the instance x to explain, and to continuous upper/lower bounds where the bounding values are taken from x. Each weight tells the user how much the decision would have changed for different (resp., smaller/greater) values of a specific categorical (resp., continuous) feature. In the example, the weight 0.11 has the following meaning [30]: if the duration in months had been higher than the value it is for x, the prediction would have been, on average, 0.11 less "0" (or 0.11 more "1"). A not very easy logic to follow when compared to a single decision rule which characterize the contextual conditions for the decision of the black box. Another major advantage of our notion of explanation consists of the set of counterfactual rules. LIME provides a rough indication of where to look for a different decision: different categorical values or lower/higher continuous values of some feature. **LORE**'s counterfactual rules provide high-level and minimal-change contexts for reversing the outcome prediction of the black box.

5.5.2 Rules vs Anchors for Explanations. A recent extension of LIME is the Anchor[13] approach [31]. It provides explanations in the form of decision rules, called anchors. Rules are computed by incrementally adding equality conditions in the premise, while an estimate of the rule precision is above a minimum threshold (set to 95%). Such an estimation relies on neighborhood generation through pure-exploration multi-armed bandit.

On a qualitative level of comparison, the Anchor approach requires the *apriori* discretization of continuous features, while the decision rule of **LORE** benefits of the capabilities of decision tree to split continuous features. Contrast, for instance, the example rules in Figure 9. Moreover, the approach of Anchor does not clearly extend to compute counterfactuals.

Let us compare now the two approaches on a quantitative level. Figure 10 reports the average precision of decision rules, where the

[12]https://github.com/marcotcr/lime

[13]https://github.com/marcotcr/anchor

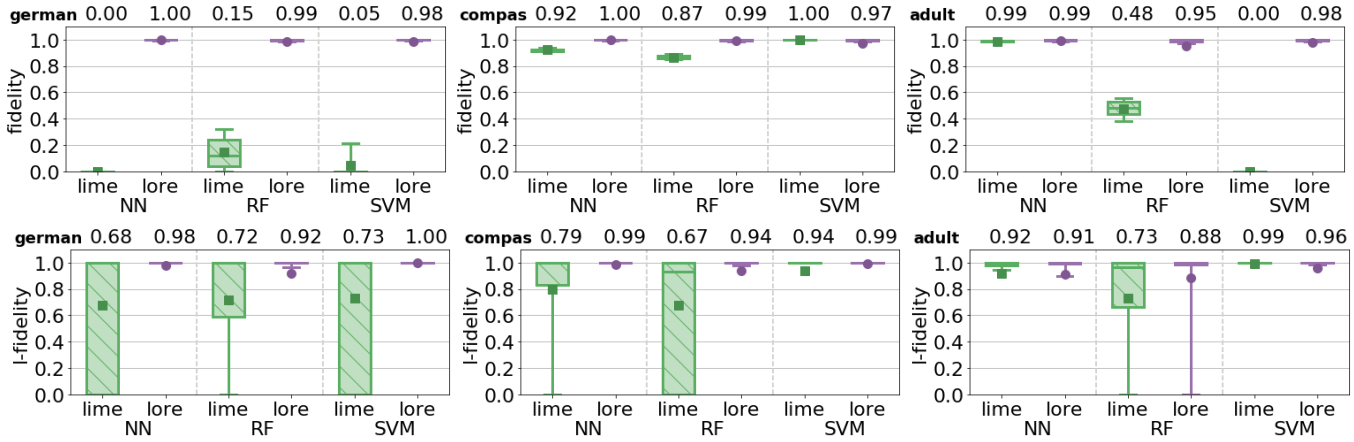


Figure 7: LORE vs LIME: box plots of *fidelity* and *l-fidelity*. Numbers on top are the mean values.

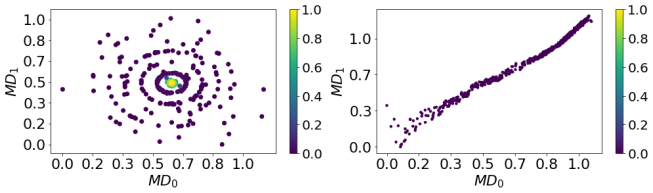


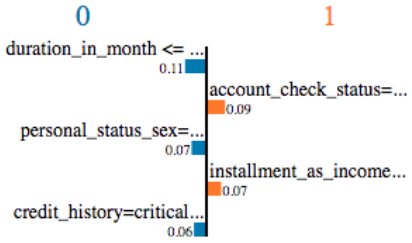
Figure 8: Neighborhoods of LORE (left) and LIME (right).

- LORE

$r =$ *({credit_amount > 836, housing = own, other_debtors = none, credit_history = critical account} ⟶ decision = 0)*

$\Phi =$ *{ ({credit_amount ≤ 836, housing = own, other_debtors = none, credit_history = critical account} ⟶ decision = 1), ({credit_amount > 836, housing = own, other_debtors = none, credit_history = all paid back} ⟶ decision = 1) }*

- LIME



- Anchor

a = *({credit_history = critical account, duration_in_month ∈ [0, 18.00]} ⟶ decision = 0)*

Figure 9: Explanations of LORE, LIME and Anchor.

precision of a rule is the fraction of instances in the neighborhood set that are correctly classified by the rule. Although **LORE** does not require to set the level of precision as parameter, the rule precision is on average high and very similar to that one obtained by Anchor, which is by construction at least 95%. This can be attributed to the performances of the decision tree induction algorithm, and of the instance generation procedure which produces balanced

Dataset	german		compass		adult	
Black box	lore	anchor	lore	anchor	lore	anchor
RF	**.76 ± .15**	.61 ± .15	**.75 ± .12**	.73 ± .14	**.70 ± .15**	.69 ± .15
NN	**.69 ± .18**	.53 ± .21	**.83 ± .13**	.79 ± .16	**.81 ± .12**	.65 ± .16
SVM	**.82 ± .16**	.32 ± .16	**.71 ± .16**	.70 ± .20	**.87 ± .14**	.67 ± .13

Table 5: LORE vs Anchor: Jaccard measure of stability.

neighborhoods Z_- and Z_+. Figure 10 also shows the average coverage of decision rules, where the coverage of a rule is the fraction of instances to explain covered by the rule. As reported in [31], large values of coverage are preferable, since this means that the set of decision rules produced over the instances to explain can be condensed/restricted to a subset of it. **LORE** shows a consistently better coverage than Anchor. Finally, we compare the stability of the two approaches with respect to randomness introduced in the neighboorhood generation. We measure stability using the Jaccard coefficient of feature sets used in the 10 decision rules computed for a same instances in 10 runs of the system. Table 5 reports mean and standard deviation of the Jaccard coefficient. **LORE** has a better stability than Anchor for all datasets and black boxes.

6 CONCLUSION

We have proposed a local black box agnostic explanation approach based on logic rules. **LORE** builds an interpretable predictor for a given black box and instance to be explained. The local interpretable predictor, a decision tree, is trained on a dense set of artificial instances similar to the one to explain generated by a genetic algorithm. The decision tree enables the extraction of a local explanation, consisting of a single rule for the decision and a set of counterfactual rules for the reversed decision. An ample experimental evaluation of the proposed approach has demonstrated the effectiveness of the genetic neighborhood procedure that leads **LORE** to outperform the proposals in the state of the art. A number of extensions and additional experiments can be mentioned as future work. First, **LORE** now works tabular data. An interesting future research direction is to make the method suitable for image and text data, for example by applying a pre-processing step for extracting semantic tags/concepts that may be mapped to a tabular format. Second, another study might be focused on the possibility

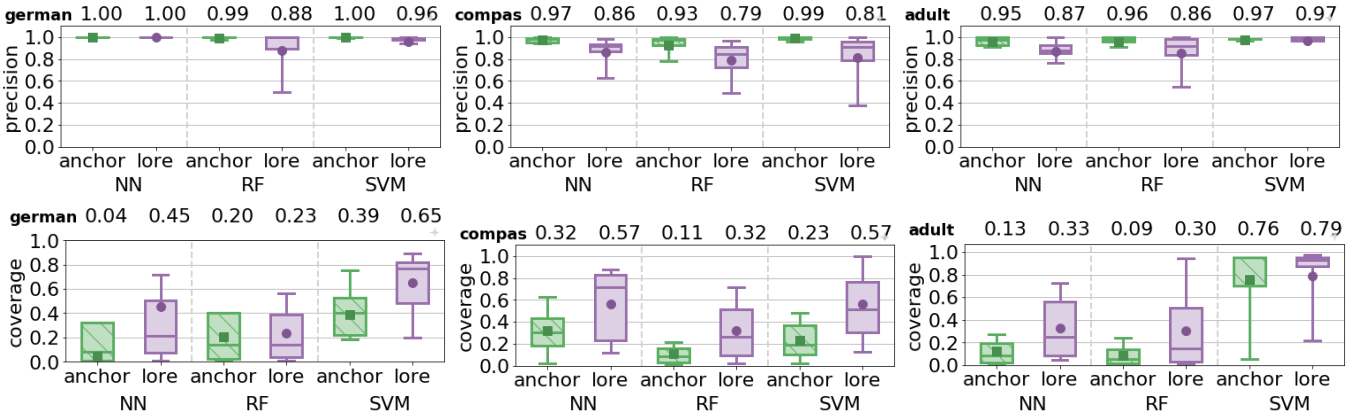


Figure 10: LORE vs Anchor: box plots of *precision* and *coverage*. Numbers on top are the mean values.

to derive a global description of the black box bottom-up by composing the local explanations and minimizing the size (complexity) of the global description. Third, research lab experiments would be useful for evaluating the human comprehensibility of the provided explanations. Finally, **LORE** explanations can be used for identifying data and/or algorithmic biases. After the local explanations are retrieved, it would be interesting to develop an approach for deriving an unbiased dataset for safely training the obscure classifier, or to prevent the black box from introducing an algorithmic bias.

REFERENCES

[1] C. C. Aggarwal, C. Chen, and J. Han. The inverse classification problem. *J. Comput. Sci. Technol.*, 25(3):458–468, 2010.

[2] R. Andrews, J. Diederich, and A. B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowl.-Based Syst.*, 8(6):373–389, 1995.

[3] M. G. Augasta and T. Kathirvalavakumar. Reverse engineering the neural networks for rule extraction in classification problems. *Neural Processing Letters*, 35(2):131–150, 2012.

[4] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary computation 1: Basic algorithms and operators*, volume 1. CRC press, 2000.

[5] S. Baluja. Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Dept Of Computer Science, 1994.

[6] S. Barocas and A. D. Selbst. Big data's disparate impact. *California Law Review*, 104, 2016.

[7] R. Berk, H. Heidari, S. Jabbari, M. Kearns, , and A. Roth. Fairness in criminal justice risk assessments: The state of the art. *arXiv preprint arXiv:1703.09207*, 2017.

[8] J. R. Cano, F. Herrera, and M. Lozano. Stratification for scaling up evolutionary prototype selection. *Pattern Recognition Letters*, 26(7):953–963, 2005.

[9] M. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. In *NIPS*, pages 24–30. MIT Press, 1995.

[10] J. Derrac, S. García, and F. Herrera. A survey on evolutionary instance selection and generation. *Int. J. of Applied Metaheuristic Computing*, 1(1):60–92, 2010.

[11] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608v2*, 2017.

[12] L. J. Eshelman. The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Foundations of genetic algorithms*, volume 1, pages 265–283. Elsevier, 1991.

[13] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, 2012.

[14] Y. Fu, X. Zhu, and B. Li. A survey on instance selection for active learning. *Knowl. Inf. Syst.*, 35(2):249–283, 2013.

[15] G. Fung, S. Sandilya, and R. B. Rao. Rule extraction from linear support vector machines. In *KDD*, pages 32–40. ACM, 2005.

[16] B. Goodman and S. R. Flaxman. EU regulations on algorithmic decision-making and a "right to explanation". volume abs/1606.08813, 2016.

[17] R. Guidotti, A. Monreale, F. Turini, D. Pedreschi, and F. Giannotti. A survey of methods for explaining black box models. *arXiv preprint arXiv:1802.01933*, 2018.

[18] S. Hara and K. Hayashi. Making tree ensembles interpretable. *arXiv preprint arXiv:1606.05390*, 2016.

[19] A. Henelius, K. Puolamäki, H. Boström, L. Asker, and P. Papapetrou. A peek into the black box: exploring classifiers by randomization. *Data mining and knowledge discovery*, 28(5-6):1503–1529, 2014.

[20] J. H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[21] U. Johansson, L. Niklasson, and R. König. Accuracy vs. comprehensibility in data mining models. In *Int. Conf. on Information Fusion*, pages 295–300, vol. 1, 2004.

[22] R. Krishnan, G. Sivakumar, and P. Bhattacharya. Extracting decision trees from trained neural networks. *Pattern recognition*, 32(12), 1999.

[23] H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *KDD*, pages 1675–1684. ACM, 2016.

[24] Y. Lou, R. Caruana, and J. Gehrke. Intelligible models for classification and regression. In *KDD*, pages 150–158. ACM, 2012.

[25] D. M. Malioutov, K. R. Varshney, A. Emad, and S. Dash. Learning interpretable classification rules with boolean compressed sensing. In *Transparent Data Mining for Big and Small Data*, pages 95–121. Springer, 2017.

[26] B. McCane and M. Albert. Distance functions for categorical and mixed variables. *Pattern Recognition Letters*, 29(7):986–993, 2008.

[27] H. Núñez, C. Angulo, and A. Català. Rule extraction from support vector machines. In *Esann*, pages 107–112, 2002.

[28] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez Trinidad, and J. Kittler. A review of instance selection methods. *Artif. Intell. Rev.*, 34(2):133–143, 2010.

[29] D. Pedreschi, S. Ruggieri, and F. Turini. Discrimination-aware data mining. In *KDD*, pages 560–568. ACM, 2008.

[30] M. T. Ribeiro, S. Singh, and C. Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *KDD*, pages 1135–1144. ACM, 2016.

[31] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. AAAI, 2018.

[32] S. Ruggieri. Yadt: Yet another decision tree builder. In *Tools with Artificial Intelligence, ICTAI.*, pages 260–265. IEEE, 2004.

[33] S. Singh, M. T. Ribeiro, and C. Guestrin. Programs as black-box explanations. *arXiv preprint arXiv:1611.07579*, 2016.

[34] H. F. Tan, G. Hooker, and M. T. Wells. Tree space prototypes: Another look at making tree ensembles interpretable. *arXiv preprint arXiv:1611.07115*, 2016.

[35] P.-N. Tan, M. Steinbach, and V. Kumar. Introduction to data mining. 1st, 2005.

[36] C. Tsai, W. Eberle, and C. Chu. Genetic algorithms in feature and instance selection. *Knowl.-Based Syst.*, 39:240–247, 2013.

[37] S. Wachter, B. Mittelstadt, and L. Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2):76–99, 2017.

[38] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law & Technology, Forthcoming*, 2017.

[39] F. Wang and C. Rudin. Falling rule lists. In *AISTATS*, volume 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org, 2015.

[40] S. Wu. *Better Decision Tree from Intelligent Instance Selection*. VDM Verlag, 2009.

[41] S. Wu and S. Olafsson. Optimal instance selection for improved decision tree induction. In *IIE Annual Conference. Proceedings*, page 1. Institute of Industrial and Systems Engineers (IISE), 2006.

[42] K. Xu et al. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057, 2015.

[43] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929. IEEE, 2016.