SMACE: A New Method for the Interpretability of Composite Decision Systems

Gianluigi Lopardo[1] (✉), Damien Garreau[1], Frédéric Precioso[2], and Greger Ottosson[3]

[1] Université Côte d'Azur, Inria, CNRS, LJAD, France
[2] Université Côte d'Azur, Inria, CNRS, I3S, France
[3] IBM France

Abstract. Interpretability is a pressing issue for decision systems. Many *post hoc* methods have been proposed to explain the predictions of a single machine learning model. However, business processes and decision systems are rarely centered around a unique model. These systems combine multiple models that produce key predictions, and then apply rules to generate the final decision. To explain such decisions, we propose the Semi-Model-Agnostic Contextual Explainer (SMACE), a new interpretability method that combines a geometric approach for decision rules with existing interpretability methods for machine learning models to generate an intuitive feature ranking tailored to the end user. We show that established model-agnostic approaches produce poor results on tabular data in this setting, in particular giving the same importance to several features, whereas SMACE can rank them in a meaningful way.

Keywords: Interpretability · Composite AI · Decision-making.

1 Introduction

Machine learning is increasingly being leveraged in systems that make automated decisions. However, the massive adoption of artificial intelligence in many industries is hindered by mistrust, due to the lack of explanations to support specific decisions [Jan et al., 2020]. Interpretability is deeply linked to trust and, as a result of public concern, has become a regulatory issue. For example, the European guidelines for trustworthy AI[4] recommend that "AI systems and their decisions should be explained in a manner adapted to the stakeholder concerned."

While numerous interpretability methods for single machine learning models exist [Linardatos et al., 2021], in many practical applications, a decision is rarely made by a unique model. In fact, composite AI systems, combining machine learning models together with explicit rules, are very popular, particularly in business settings. Incorporating decision rules is important, for two main reasons. Firstly, *decision rules are crucial for expressing policies that can change (even very quickly) over time.* For example, depending on last quarter's financial

[4] https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

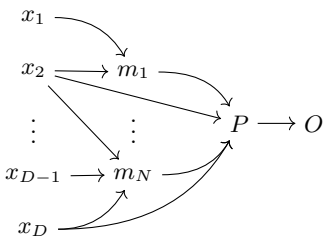

Fig. 1. Structure of a composite decision system with D input features $x_1, \ldots, x_D$, and N models $m_1, \ldots, m_N$. A decision policy P (*i.e.*, a set of decision rules) is finally applied to produce an outcome O. Note that in general both the models and the rules take a subset of input features as input, tough not necessarily the same.

results, a company might be more or less risk-averse and therefore have a more or less conservative policy. Using an individual machine learning model would require to retrain it with new data each time the policy changes. In contrast, with a rule-based system, risk aversion can be managed by changing only a rule. Secondly, *machine learning models are not suitable for incorporating strict rules.* Indeed, while often a policy may represent a soft preference, in many cases we may have strict rules, due to domain needs or regulation. For example, we may have to require that clients' age be over 21 in order to offer them a service. It is typically difficult to account for such strict rules in a machine learning setting.

We focus our study on tabular data, most commonly used in businesses' day-to-day operations, often corresponding to customer records. Our interest in this paper is the interpretability of composite decision-making systems that include multiple machine learning models aggregated through decision rules in the form

$$\texttt{if } \{\texttt{premise}\}\texttt{, then } \{\texttt{consequence}\}.$$

Here, `premise` is a logical conjunction of conditions on input attributes (*e.g.*, age of a customer) and outputs of machine learning models (*e.g.*, the churn risk of a customer); `consequence` is a decision concerning a user (*e.g.*, propose a new offer to a customer). A phone company's policy for proposing a new offer can be

$$\texttt{if age} \leq 45 \texttt{ and churn\_risk} \geq 0.5, \texttt{ then offer } 10\% \texttt{ discount}.$$

On the one hand, a number of additional challenges arise in this framework (see Section 3). On the other hand, there is knowledge we can leverage: we know the decision policy and how the models are aggregated. It is worth exploiting this information instead of considering the whole system as a black-box and being completely model-agnostic. In contrast, we want to be agnostic about the nature of individual models: we call this situation "semi-model-agnostic."

In this setting, we present the *Semi-Model-Agnostic Contextual Explainer* (SMACE), a novel interpretability method for composite decision systems that combines a geometric approach (for decision rules) with existing interpretability

solutions (for machine learning models) to generate explanations based on feature importance. The key idea of SMACE is to agglomerate individual models explanations in a manner similar to that used by the whole decision system. By making the appropriate assumptions (see Section 4.2), we can see a decision system as a decision tree where some nodes refer to machine learning models. In a nutshell, we agglomerate the explanations for each model in a linear fashion, following the structure of this tree. We therefore combine an *ad hoc* method for the interpretability of decision trees, with *post hoc* methods for the models.

Contributions. The main contributions of this paper are

- The description of a new method, SMACE, for the interpretability of composite decision-making systems;
- The Python implementation of SMACE, available as an open source package at https://github.com/gianluigilopardo/smace;
- The evaluation of SMACE *vs* some popular methods showing that the latter perform poorly in our setting.

The rest of the paper is organized as follows. In Section 2, we briefly present some related work on both decision trees and *post hoc* methods for machine learning. Section 3 outlines the main challenges we want to address. In Section 4 the mechanisms behind SMACE are explained step by step; an overview is given in Section 4.3. Finally, we provide an evaluation of our method compared to established *post hoc* solutions in Section 5, before concluding in Section 6.

2 Related Work

A decision policy can be embedded in a decision tree. Small CART trees [Breiman et al., 1984] are intrinsically interpretable, thanks to their simple structure. However, as the number of nodes grows, interpretability becomes more challenging. Alvarez [2004] and Alvarez and Martin [2009] propose to study the partition generated by the tree in the feature space to rank features by importance. A similar approach has been used to build interpretable random forests [Bénard et al., 2021]. We develop a solution inspired by this idea based on the distance between a point and the decision boundaries generated by the tree. The main difference in our setting is that each node can be a machine learning model.

Indeed, we also need to deal with machine learning interpretability. LIME [Ribeiro et al., 2016] explains the prediction of any model by locally approximating it with a simpler, intrinsically interpretable linear surrogate. Upadhyay et al. [2021] extends LIME to business processes, by modifying the sampling. Anchors [Ribeiro et al., 2018] extracts sufficient conditions for a certain prediction, in the form of rules. SHAP [Lundberg and Lee, 2017] addresses this problem from a Game Theory perspective, where each input feature is a player, by estimating Shapley values [Shapley, 1953]. Despite the solid theoretical foundation, there is concern [Kumar et al., 2020] about its suitability for explanations. Labreuche and Fossier [2018] leverages Shapley values to explain the result of aggregation

models for Multi-Criteria Decision Aiding. However, their solution requires full knowledge of the models involved, whereas we want to be agnostic about individual models. SMACE requires feature importance measures, provided for instance by LIME and SHAP (or different approaches as proposed by Främling [2022]).

Overall, perturbation-based methods have some drawbacks and are not always reliable [Slack et al., 2020]. In addition, methods using linear surrogates are not suitable to deal with step functions (*e.g.*, the ones encoded by strict decision rules), which often leads to attributing the same contribution to multiple features. In the case of LIME for tabular data this behavior was pointed out by Garreau and von Luxburg [2020] and Garreau and Luxburg [2020a].

3 Challenges

As mentioned in the previous section, the field of interpretable machine learning has many unresolved issues. When trying to explain a decision that relies on multiple machine learning models, a number of additional problems arise:

- *Rule-induced nonlinearities*: decision rules will cause sharp borders in the decision space. For example: a car rental rule might state "`age` of renter must be above 21". Explanations for a machine learning based risk assessment close to the decision boundary `age` = 21, *e.g.*, must accurately indicate `age` as an important feature.
- *Out-of-distribution sampling*: the decision rules surrounding a machine learning model will eliminate a portion of the decision space. Explanatory methods based on sampling like LIME and SHAP are known to distort explanations because of this (see Section 2).
- *Combinations of decision rules and machine learning*: for a specific decision, a subset of rules triggered and a machine learning-based prediction was generated. How do we compose a prediction based on both sources?
- *Multiple machine learning models*: when multiple models are involved in a decision, we must also be able to aggregate multiple feature contributions. These may be (partially) overlapping and conflicting.

In addition, we want to have two desirable properties: *(1) the contribution associated with a feature must be positive if it satisfies the condition, negative otherwise*; *(2) the magnitude of the contribution associated with a feature must be greater the closer its value is to the decision boundary.*

4 SMACE

We now present SMACE in more details, starting with a thorough description of our setting in Section 4.1 and a discussion of our assumptions in Section 4.2. Section 4.3 contains the overview of the method, with additional details in Section 4.4, 4.5, and 4.6.

4.1 Setting

Let $x \in \mathbb{R}^{Q \times D}$ be the input data, where each row $x^{(i)} = (x_1, \ldots, x_D)^\top \in \mathbb{R}^D$ is an instance and D is the cardinality of the *input features set* F. Let the set $M = \{m_1, \ldots, m_N\}$ be the set of models. We will refer to their outputs $m_1(x), \ldots, m_N(x)$ as the *internal features*, whose values we also denote as $y^{(1)}, \ldots, y^{(N)}$ when there is no ambiguity. The union of input and internal features is the set of $D+N$ *features* to which the decision policy can be applied.

We define $\tilde{x} \coloneqq (x_1, \ldots, x_D, m_1(x), \ldots, m_N(x))^\top$ as the completion of x with the outputs of the N models. Likewise, we call $\xi = (\xi_1, \ldots, \xi_D)^\top$ the example to be explained and $\tilde{\xi} = (\xi_1, \ldots, \xi_D, m_1(\xi), \ldots, m_N(\xi))^\top$ its completion. A decision rule R is formally defined by a set of conditions on the features in the form $\tilde{x}_j \geq \tau$, for some cutoff $\tau \in \mathbb{R}$. Figure 1 illustrates a generic composite decision system.

4.2 Assumptions

The definition of SMACE is based on three assumptions required to frame the setting. Ideas for solving some of their limitations are discussed in Section 6.

Assumption 1 *Decision rules only refer to numerical values.*

This assumption allows us to take a simple geometric approach for the explainability of the decision tree. Note that this does not imply any restriction on the input of the machine learning models, that can still be categorical.

Assumption 2 *Each decision rule is related to a single feature, without taking into account feature interactions.*

For instance, this assumption excludes conditions like `if` $\tilde{x}_1 \geq \tilde{x}_2$. Geometrically, this implies decision trees with splits parallel to the axes, such as CART [Breiman et al., 1984], C4.5 [Quinlan, 1993], and ID3 [Quinlan, 1986].

Assumption 3 *The machine learning models only use input features to make predictions.*

We disregard the cases in which a machine learning model takes as input the output of other machine learning models. We remark that this is a very reasonable assumption that covers most real-world applications. Note that assumptions 1 and 2 refer to the decision rules, while Assumption 3 is the only referring to the machine learning models and does not concern their nature.

4.3 Overview

For each example ξ whose decision we want to explain, we first perform two parallel steps:

- **Explain the results of the models**: for each machine learning model m, we derive the (normalized) contribution $\hat{\phi}_j^{(m)}$ for each input feature j. By default, SMACE relies on KernelSHAP to allocate these importance values;

– **Explain the rule-based decision**: measure the contribution r_j of each feature (that is, each input feature and each internal feature directly involved in the decision policy), through Algorithm 2.

Then, to get the **overall explanations** (see Algorithm 1), we combine these partial explanations. The total contribution of the input feature $j \in F$ to the decision for a given instance is

$$e_j = r_j + \sum_{m \in M} r_m \hat{\phi}_j^{(m)} \,. \tag{1}$$

That is, we weight the contribution of input features to each model with the contribution of that model in the decision rule, and we add the direct contribution of feature j to the decision rule (if a feature is not directly involved in a decision rule, its contribution is zero).

4.4 Explaining the results of the models

We need to attribute the output of each machine learning model to its input values. For instance, this is what KernelSHAP does, and by default SMACE relies on it. In any case, SMACE requires a measure of feature importance for the input features, but not necessarily based on SHAP. Any other measure of feature importance is possible. Given the contribution $\phi_j^{(m)}$ of each input feature j for each machine learning model m we define the normalized contribution as

$$\hat{\phi}_j^{(m)} = \begin{cases} \dfrac{\left|\phi_j^{(m)}\right|}{\sum_{i \in F} \left|\phi_i^{(m)}\right|} \,, & \text{if } \max_{i \in F} \left|\phi_i^{(m)}\right| \neq 0 \,, \\ 0 \,, & \text{otherwise.} \end{cases} \tag{2}$$

Indeed, two models m_k and m_h might give results $y^{(k)}$ and $y^{(h)}$ on very different scales, for instance because they do not have the same unit. In the example above, we may have models computing the churn risk and the life time value. The first value estimates a probability, so it belongs to $[0, 1]$, while the second is the expected economic return that the company may get from a customer, and it could be a quantity scaling as thousands of euros. In general, if m_k predicts the churn risk and m_h predicts the life time value, for a feature j in input to both models, we might expect $\left|\phi_j^{(h)}\right| \gg \left|\phi_j^{(k)}\right|$. In order to have a meaningful comparison between the models, we therefore need to scale the ϕ values and we use as scale factor the sum of the ϕ values for each model. The quantities $\hat{\phi}$ defined by means of Eq. (2) are of the same order of magnitude and dimensionless, so can be aggregated. In addition, $\hat{\phi}$ is defined such that

$$\forall j \in F \,, \forall m \in M \,, \quad 0 \leq \hat{\phi}_j^{(m)} \leq 1 \,.$$

Note that the second part of Eq. (2) is equivalent to taking the convention $\frac{0}{0} = 0$: the denominator is zero if and only if each contribution is zero. The definition

implies that if the model m relies on a single feature j, the latter will have

$$\hat{\phi}_j^{(m)} = 1 \implies r_m \hat{\phi}_j^{(m)} = r_m \,,$$

i.e., the whole contribution of the model m to the decision is attributed to the input feature j, which in fact is the only one responsible for its output.

4.5 Explaining the rule-based decision

In Section 2 we stated that the set of conditions used by a decision system can be interpreted as a CART tree, such as the one in Figure 2, where each split represents a condition on a feature. A first approach to explain the decision of such a tree can be to show the trace followed by the point within the tree to the user. However, the trace does not contain enough information to understand the situation: a large change in some conditions may have no impact on the result, whereas a very small increase in one value may lead to a completely different classification, if we are close to a split value.

In addition, there may be many conditions within a decision rule, and simply listing them all would make it difficult to understand the decision. In fact, each condition is a split in the decision tree and each split produces a decision boundary. The collection of decision boundaries generated by the tree induces a partition of the input space and we call decision surface the union of the boundaries of the different areas corresponding to the different classes. Because of Assumption 2, at each point $z \in S$, the decision surface is piecewise-affine, consisting of a list of hyperplanes, each referring to one feature. By projecting an example point $\tilde{x}$ onto each component j of the surface S, we obtain the point $\pi_j^{(S)}(\tilde{x})$ (see Eq. (3)) at minimum distance that satisfies the condition on the j-th feature (see Figure 2). This distance is a measure of the robustness of the decision with respect to changes along feature j. Conversely, the smaller the distance, the more *sensitive* the decision.

As mentioned in Section 3, we want the method to assign a greater contribution to features with higher sensitivity. In this way, values close to the decision boundary are highlighted to the end user and the domain expert, who will be able to draw the appropriate conclusions. The explainability problem is therefore addressed by studying the decision surfaces generated by the decision tree.

However, to properly compare these contributions, we must first normalize the features. We must then query the models on the training set in order to obtain the values $y^{(1)}, \ldots, y^{(N)}$. We thus apply a min-max normalization on both input features

$$\forall\, i \in \{1, \ldots, Q\}\,, \quad x_j'^{(i)} = \frac{x_j^{(i)} - \min x_j}{\max x_j - \min x_j}\,,$$

and internal features, likewise. In this way, the values of each feature is scaled in $[0, 1]$. For the sake of convenience, we continue to denote the features x_i' and $y'^{(k)}$ as x_i and $y^{(k)}$, but from now on we consider them as scaled.

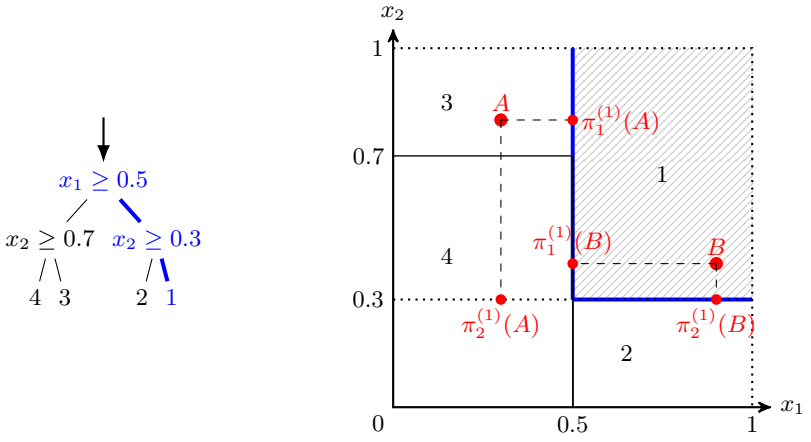

Fig. 2. On the left, a decision tree classifier based on x_1 and x_2. In **blue and bold**, the trace for leaf 1. On the right, the partition it generates. A and B are instance points, classified respectively as 3 and 1. The decision surface for leaf 1 is in **blue and bold**. The dashed lines indicate the distance between the points and the decision boundaries.

Each decision surface S has as many components (hyperplanes) as there are features defining it. For instance, the decision surface for leaf 1 of Figure 2 has two components: h_1 and h_2, along x_1 and x_2, respectively. The projection $\pi_j^{(S)}(x)$ of point x onto h_j is

$$\pi_j^{(S)}(\tilde{x}) \in \arg\min_{z \in h_j} \|\tilde{x} - z\|_2 \,. \tag{3}$$

For instance, let us consider the decision tree of Figure 2 and the partition it generates. Let us say we are interested in leaf 1 (the grid subspace shown in Figure 2) generated by the trace in blue. Example B satisfies both conditions, while A only satisfies the condition on x_2. We also note that the decision for B is very sensitive with respect to changes along axis x_2, while it is more robust with respect to x_1. We compute the contribution r_j of a feature j for the classification of point $\tilde{x}$ in leaf ℓ by means of Algorithm 2 as

$$r_j(\tilde{x}) = \begin{cases} \left|\tilde{x}_j - \pi_j^{(\ell)}(\tilde{x})\right| - 1 \,, & \text{if } \tilde{x}_j < h_j \,, \\ 1 - \left|\tilde{x}_j - \pi_j^{(\ell)}(\tilde{x})\right| \,, & \text{if } \tilde{x}_j \geq h_j \,. \end{cases} \tag{4}$$

We can see that for point A, the feature x_1 has a high negative contribution, since it does not satisfy the condition on it, while x_2 has a positive contribution. Point B satisfies both conditions: both features have positive contributions, but $r_2(B) > r_1(B)$, since the decision is more sensitive with respect to x_2.

4.6 Overall explanations

Finally, once the partial explanations have been obtained, we agglomerate them via Eq. (1). We thus obtain a measure of the importance of features for a specific

Algorithm 1 Overview of `smace`.

function SMACE_EXPLAIN(rule R (set of conditions), list of models M, example to explain $\xi \in \mathbb{R}^{\mathbb{D}}$)
  $\tilde{\xi} \leftarrow \xi$, $\phi \leftarrow \{0\}^N$, $r \leftarrow \{0\}^{D+N}$, $e \leftarrow \{0\}^D$
  for $m \in M$ **do**
    $\hat{\phi}^{(m)} \leftarrow$ EXPLAIN_MODEL(ξ, m) ▷ explain the result of model m (Section 4.4)
    $\tilde{\xi} \leftarrow (\xi_1, \ldots, \xi_D, \ldots, m(\xi))$
  end for
  for $j = 1, \ldots, D + N$ **do**
    $r_j \leftarrow$ RULE_CONTRIBUTION$(R, j, \tilde{\xi})$ ▷ explain the rule-based decision
  end for
  for $j = 1, \ldots, D$ **do**
    $e_j \leftarrow r_j + \sum_{m \in M} r_m \hat{\phi}_j^{(m)}$ ▷ aggregate
  end for
  return e
end function

decision made by a system combining rules and machine learning models. Our measure of importance highlights the most critical features, those therefore most involved in the decision. In this way, a domain expert can analyse a decision by focusing on these features to make her or his own qualitative assessment.

Computational cost. The most computationally expensive step of SMACE is to get explanations for the underlying models. It basically consists in N calls to the explainer on (at most) D input features. For instance, in the case of KernelSHAP, this would be $N \times 1000 \times D$.

5 Evaluation

What makes interpretability even more challenging is the lack of adequate metrics to appropriately assess the quality of explanations. In this section we compare the results obtained with SMACE and those obtained by applying the default implementations of SHAP[5] and LIME[6] on the whole decision system. We first perform a qualitative analysis on simple use cases, where we can get a complete understanding of the decision provided by the system. We show that SHAP and LIME do not satisfy the properties stated in Section 4.5 and we therefore argue that they are not suitable in this context. Finally, we perform a sanity check on aggregate explanations on three different realistic use cases.

[5] https://github.com/slundberg/shap
[6] https://github.com/marcotcr/lime

Algorithm 2 Computing RULE_CONTRIBUTION.

function RULE_CONTRIBUTION(rule R, variable j, example to explain $\tilde{\xi}$)
  $S \leftarrow R$ ▷ projection to the decision surface S generated by R
  $\pi_j^{(S)}(\tilde{\xi}) \leftarrow \arg\min_{z \in h_j} \|\tilde{\xi} - z\|_2$
  if $\tilde{\xi}$ satisfies condition on j **then**
    $r_j \leftarrow 1 - \left|\tilde{\xi}_j - \pi_j^{(S)}(\tilde{\xi})\right|$
  else
    $r_j \leftarrow \left|\tilde{\xi}_j - \pi_j^{(S)}(\tilde{\xi})\right| - 1$
  end if
  return r_j
end function

5.1 Qualitative analysis

The input data consists of 1000 instances, each with three randomly generated components as uniform in $[0, 1]^3$. Note that decision rules on these data generate partitions analogous to those in Figure 2, but in dimension 3.

Rules only Let us first evaluate the case of a decision system consisting of only three simple conditions applied to only three input features. The decision policy contains rule R_1:

$$\texttt{if } x_1 \leq 0.5 \texttt{ and } x_2 \geq 0.6 \texttt{ and } x_3 \geq 0.2 \texttt{ then } 1, \texttt{ else } 0.$$

Note that there are no models, R_1 is based solely on the input data. The method then reduces to the application of Eq. (4), discussed in Section 4.5.

Example with two violated attributes. Take the example to be explained in an arbitrary position with respect to the boundaries: $\xi^{(1)} = (0.6, 0.1, 0.4)^\top$. The decision is 0, since the rule R_1 is not satisfied, indeed the conditions $\xi_1^{(1)} \leq 0.5$ and $\xi_2^{(1)} \geq 0.6$ are violated. We want to know why $\xi^{(1)}$ is not classified as 1 and the contributions of the three features to that decision. The comparison is shown in Table 1. The results of SMACE are computed (Eq. (4)) as

$$\begin{cases} r_1 = |0.6 - 0.5| - 1 = -0.9\,, \\ r_2 = |0.1 - 0.6| - 1 = -0.5\,, \\ r_3 = 1 - |0.4 - 0.2| = 0.8\,. \end{cases}$$

In this case, we see that all the three methods agree in their signs, satisfying property *(1)*. However, SHAP and LIME attribute the same contribution to x_1 and x_2 even though the sensitivities of the values are different. They do not satisfy property *(2)*: the contribution of x_1 should be higher in magnitude than that of x_2, since it is closer to the boundary. This behavior is due to the

Table 1. Example in generic position, three conditions on three input features. LIME and SHAP are producing flat explanations on the variables x_1 and x_2, even if their sensitivities for the decision are very different. SMACE captures this information.

condition	example ($\xi^{(1)}$)	**SMACE**	SHAP	LIME
$x_1 \leq 0.5$	0.6	-0.9	-0.08	-0.21
$x_2 \geq 0.6$	0.1	-0.5	-0.08	-0.21
$x_3 \geq 0.2$	0.4	0.8	0.02	0.04

Table 2. Slight violation on one attribute, conditions on three input features. LIME and SHAP do not highlight the high sensitivities for x_2 and x_3, which are exactly on their respective decision boundary.

condition	example ($\xi^{(2)}$)	**SMACE**	SHAP	LIME
$x_1 \leq 0.5$	0.51	-0.99	-0.29	-0.22
$x_2 \geq 0.6$	0.60	1.00	0.12	0.14
$x_3 \geq 0.2$	0.20	1.00	0.03	-0.20

nonlinearities brought by the decision rules, as mentioned in Section 2. The point is that the sampling is performed in a space away from the boundary, and so by perturbing the example in a small neighborhood, the output does not change.

Slight violation on one attribute. We now consider the specific case where two features are exactly on the decision boundary, while one condition is slightly violated. Let us consider the example $\xi^{(2)} = (0.51, 0.6, 0.2)^\top$. The decision-making system classifies $\xi^{(2)}$ as 0 for a slight violation of the rule on the first attribute. In Table 2 we see that SMACE highlights the slight violation of the rule on x_1.

Simple hybrid system Let us add two simple linear models m_1 and m_2. The models are defined as

$$\begin{cases} m_1(x) = 1x_2 + 2x_3\,, \\ m_2(x) = 700x_1 - 500x_2 + 1000x_3\,. \end{cases}$$

We are interested in rule R_3:

$$\texttt{if}\ x_1 \leq 0.5\ \texttt{and}\ x_2 \geq 0.6\ \texttt{and}\ m_1 \geq 1\ \texttt{and}\ m_2 \leq 600\ \texttt{then}\ 1\,,\ \texttt{else}\ 0\,,$$

and we want to explain the decision for $\xi^{(1)}$. The comparison on the whole system is in Table 3. Again, LIME and SHAP are producing identical results on x_1 and x_2, missing useful information. SMACE disagrees with the other methods on the sign of x_3, correctly giving a negative sign (Property *(1)*). Indeed, the input feature x_3 has a high contribution for the model m_2 and m_2 is not satisfying the condition ($m_2(\xi^{(1)}) = 770 > 600$), so it has a negative contribution.

By analyzing individual explanations, we have shown that SMACE produces meaningful results by assigning each feature a contribution proportional to its distance from the boundary. On the contrary, SHAP and LIME often assign the

Table 3. Simple hybrid system, comparison on the whole decision system. LIME and SHAP both produce the same explanations for features 1 and 2.

example ($\xi^{(1)}$)	**SMACE**	SHAP	LIME
$\xi_1^{(1)} = 0.6$	-1.03	-0.08	-0.19
$\xi_2^{(1)} = 0.1$	-1.73	-0.08	-0.19
$\xi_3^{(1)} = 0.4$	-0.54	0.02	0.09

same contribution to different features, not providing useful information about the relative importance of each feature.

5.2 Sanity check

In the previous section, we showed that SMACE is able to produce meaningful feature attributions. We now demonstrate that SMACE also retains an ability to identify the set of features contributing negatively to a decision, regardless of individual attribution. If a feature contributes negatively, it means it must be moved to meet its condition. Correctly identifying negative features is a desirable property: to change the decision, each of them must be moved.

We consider 100 random instances which do not satisfy the rules (described in the supplementary), from three different datasets, and we apply SMACE, SHAP, and LIME. For each method, we extract the set of negative features. Note that to be sure that the rule will be satisfied, each negative feature should be shifted to a specific value: none of the three methods is giving this information. We then generate 1000 samples by shifting negative features with a local perturbation. The average decision made on these perturbed samples is an indicator of the quality of the explanations provided by each of the three methods.

Cancer treatment A machine learning model is trained to predict whether a breast cancer is benign or malignant from information about its size and structure. An automated decision system is then applied to decide on treatment: if the risk of the tumor being malignant is too high, it proceeds in full reliance on the model. If, on the other hand, the probability is low, but the size and composition of the tumor are suspicious, further investigation is carried out. The decision system consists of 30 continuous *input features* and 1 *internal feature* (coming from the model). We use the *Breast Cancer Wisconsin Data Set*.[7]

In this example, we want to explain *why* the treatment was not proposed, *i.e.*, which input features are negatively contributing to the decision. Given the large number of parameters to be analyzed, it is useful to order them by importance, in order to speed up the investigation by giving the right priorities. The graph at the top left of the Figure 3 shows the comparison. SMACE curve is always above the others: it is better at detecting negative features.

Fraud Detection A financial authority must track mobile money transactions, promptly halting anomalous transactions suspected of fraud. The au-

[7] https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

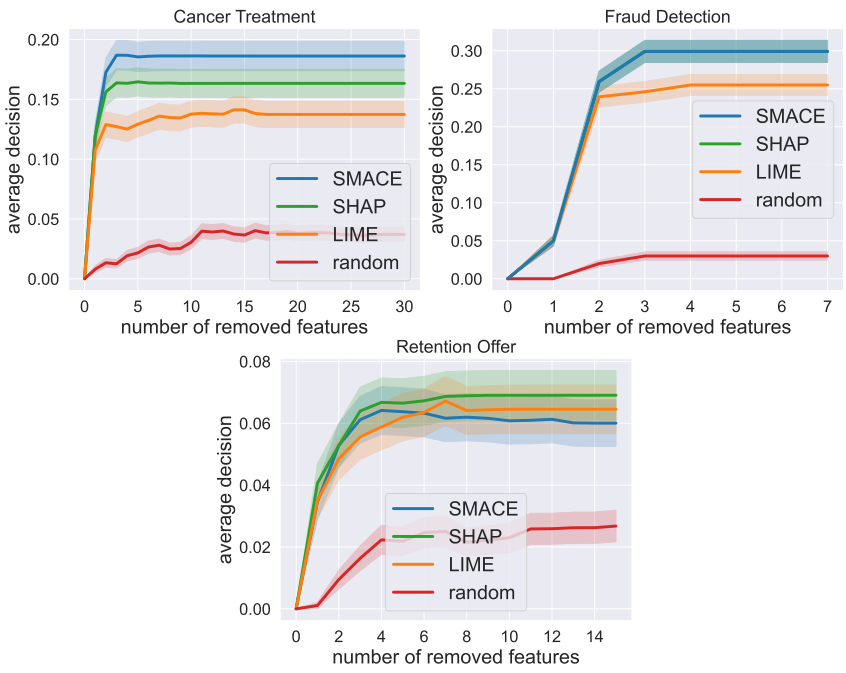

Fig. 3. Comparison of SMACE, SHAP, and LIME on the ability to identify the set of features contributing negatively to a decision, regardless of individual attribution. Correctly identifying negative features is a desirable property: to change the decision, each of them must be moved. When the conditions are not met, the three methods are used to extract the negative features, and we generate perturbed samples around the original values. We then compare the average decision made on the samples.

thority uses a decision-making system to approve or block transactions, according to a *fraud score*, computed through a machine learning classifier, and the amount and balanced involved in the transaction. We use the *Synthetic Financial Datasets For Fraud Detection*[8] As before, we extract and perturb the negative features set for each method.

The graph at the top right of Figure 3 shows that SMACE and SHAP are on par. In this decision system, the conditions based on the input features matter significantly more than the one on the model. This means that SMACE explanations are almost entirely based on Eq. (4) and, consistently with what we saw in Section 5.1, SMACE and SHAP are able to extract the correct set of negative features. However, we remark that SHAP is likely to assign them the same (negative) contribution: SMACE carries more information.

Retention Offer Let us consider a mobile phone company which wants to predict if a customer is going to leave for a competitor, and to decide if a retention offer should be made, while not spending more on retention than the value of

[8] https://www.kaggle.com/ealaxi/paysim1

retaining the customer. The decision policy is based on information about the customer and their subscription (input features), and two models (producing internal features) predicting the *churn risk* (*i.e.*, the likelihood that the customer will cancel their subscription) and the *lifetime value* (*i.e.*, the expected revenue generated by the customer if retained). We use the IBM *Telco Churn* dataset.[9]

In this example, we want to explain *why* a retention offer was not made, in terms of the original input features. In practice, the features that are contributing negatively should be moved to meet the conditions. Note that this use case is characterized by the presence of many categorical input features (see Assumption 1): this is a stress test for SMACE. Figure 3 shows that SMACE is comparable with the state of the art in extracting the right set of negative features: error bars are overlapping. However, it is only a partial measure of quality, since the ranking of features is ignored. As seen in Section 5.1, SMACE is also able to rank these features by sensitivity.

We compared the ability of SMACE, SHAP, and LIME to extract features that are negatively contributing to a decision and should therefore be moved to change it. SMACE is best when applied to the standard context: one or more models and several continuous features (Cancer Treatment). SHAP tends to extract the same set of negative features as SMACE when the impact of models is absent or insignificant (Fraud Detection). SMACE loses performance when many categorical features are involved in the decision: however, the error bars of the three methods are overlapping (Retention Offer). In addition, as seen in Section 5.1, SMACE is also able to rank these features by sensitivity, while SHAP and LIME tend to attribute identical explanations.

6 Conclusion and Future Work

We addressed the problem of explaining decisions produced by a decision-making system composed of both machine learning models and decision rules. We proposed SMACE, to generate feature importance based explanations. Up to the best of our knowledge, it is the first method specifically designed for these systems. SMACE approaches the problem with a projection-based solution to explain the rule-based decision and by aggregating it with models explanations. We finally showed that model-agnostic approaches designed to explain machine learning models are not well-suited for this problem, due to the complications coming with the rules. In contrast, SMACE provides meaningful results by meeting our requirements, *i.e.*, adapting to the needs of the end user.

In future work, we plan to extend SMACE, making it usable in a wider range of applications. A particularly interesting approach to include categorical features in the rules is implemented in CatBoost [Prokhorenkova et al., 2018], a gradient boosting toolkit. The idea is to group categories by *target statistics*, which can replace them. SMACE could also be generalized to more complex model configurations, where some models take as input the output of other

[9] https://github.com/IBMDataScience/DSX-DemoCenter/tree/master/DSX-Local-Telco-Churn-master

models. One natural extension would be to recursively weight the importance of each model with the contribution it brings for other models.

Acknowledgments. This work has been supported by the French government, through the NIM-ML project (ANR-21-CE23-0005-01) and through the 3IA Côte d'Azur (ANR-19-P3IA-0002), and by EU Horizon 2020 project AI4Media (contract no. 951911, https://ai4media.eu/).

Bibliography

Steve T.K. Jan, Vatche Ishakian, and Vinod Muthusamy. AI trust in business processes: the need for process-aware explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13403–13404, 2020.

Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable AI: A review of machine learning interpretability methods. *Entropy*, 23 (1):18, 2021.

Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 1984.

Isabelle Alvarez. Explaining the result of a decision tree to the end-user. In *ECAI*, volume 16, page 411, 2004.

Isabelle Alvarez and Sophie Martin. Explaining a result to the end-user: a geometric approach for classification problems. In *Exact09, IJCAI 2009 Workshop on explanation aware computing (International Joint Conferences on Artificial Intelligence)*, pages p–102, 2009.

Clément Bénard, Gérard Biau, Sébastien Da Veiga, and Erwan Scornet. Sirus: Stable and interpretable rule set for classification. *Electronic Journal of Statistics*, 15(1):427–505, 2021.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "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*, pages 1135–1144, 2016.

Sohini Upadhyay, Vatche Isahagian, Vinod Muthusamy, and Yara Rizk. Extending LIME for business process automation. *arXiv preprint arXiv:2108.04371*, 2021.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30:4765–4774, 2017.

Lloyd S. Shapley. A value for n-person games. *Contributions to the Theory of Games, number 28 in Annals of Mathematics Studies, pages 307–317*, II, 1953.

Elizabeth I. Kumar, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle Friedler. Problems with Shapley-value-based explanations as feature importance measures. In *International Conference on Machine Learning*, pages 5491–5500. PMLR, 2020.

Christophe Labreuche and Simon Fossier. Explaining multi-criteria decision aiding models with an extended Shapley Value. In *IJCAI*, pages 331–339, 2018.

Kary Främling. Contextual importance and utility: A theoretical foundation. In *Australasian Joint Conference on Artificial Intelligence*, pages 117–128. Springer, 2022.

Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

Damien Garreau and Ulrike von Luxburg. Looking deeper into tabular LIME. *arXiv preprint arXiv:2008.11092, v1*, 2020.

Damien Garreau and Ulrike Luxburg. Explaining the explainer: A first theoretical analysis of LIME. In *International Conference on Artificial Intelligence and Statistics*, pages 1287–1296. PMLR, 2020a.

Ross J. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0. URL http://portal.acm.org/citation.cfm?id=152181.

Ross J. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *NeurIPS*, 2018.