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

Gianluigi Lopardo[1,2,3*], **Damien Garreau**[1,2,3], **Frédéric Precioso**[1,2,4], **Greger Ottosson**[2,5]
[1]Université Côte d'Azur
[2]Inria Maasai Project-Team
[3]CNRS, LJAD
[4]CNRS, I3S
[5]IBM France
{gianluigi.lopardo, damien.garreau, frederic.precioso, greger.ottosson}@inria.fr,

Abstract

Interpretability is a pressing issue for decision systems. Many post hoc methods have been proposed to explain the predictions of any machine learning model. However, business processes and decision systems are rarely centered around a single, standalone model. These systems combine multiple models that produce key predictions, and then apply decision rules to generate the final decision. To explain such decision, we present SMACE, *Semi-Model-Agnostic Contextual Explainer*, a novel interpretability method that combines a geometric approach for decision rules with existing post hoc solutions 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 in this framework.

1 Introduction

Machine Learning is increasingly being leveraged in systems that make automated decisions. Nevertheless, a decision is rarely made by a standalone model. One might think that the entire decision process can be handled by an individual machine learning model. This is not always true. In fact, composite AI systems, which combine machine learning and symbolic artificial intelligence, are very popular, particularly in business settings. Incorporating decision rules is still 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 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 one parameter. Secondly, *machine learning models are not suitable for incorporating strict rules*. Indeed, while often a policy may represent only 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. Machine learning relies mainly on probabilistic methods, which makes difficult to accurately adhere to strict deterministic rules. We focus our study on tabular data, which are most commonly used for day-to-day operations by businesses.

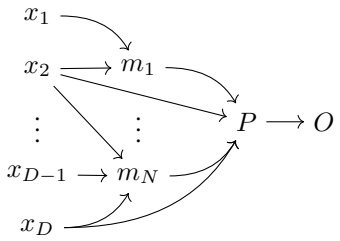

Figure 1: Structure of a composite decision-making system with D input features $x_1, \ldots, x_D$, and N models $m_1, \ldots, m_N$. A decision policy P (*i.e.*, a set decision of rules) is finally applied to produce an outcome O. Note that in general both the models and the rules take a subset of input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

Anyway, the massive adoption of Artificial Intelligence in many industries is hindered by mistrust, mainly owing to the lack of explanations to support specific decisions [Jan *et al.*, 2020]. Interpretability is deeply linked to trust and, as a result of growing public concern, has also become a regulatory issue. In 2016 the European General Data Protection Regulation introduced the concept of "right to explanation" [Wachter *et al.*, 2017], outlining a clear normative direction. Similarly, the United States Federal Trade Commission guidelines[1] recommends that if consumers are denied something of value (*e.g.*, a loan) based on AI, they are entitled to an explanation. However, explanations should be based on information known to the user and not on internally produced values (such as risk scores), the meaning of which and the process that generates them being a priori confidential.

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

`if {premise} then {consequence}.`

Here, `premise` is a logical conjunction of conditions on input attributes (*e.g.*, age of a customer) and outputs of ma-

*Contact Author

[1]https://www.ftc.gov/news-events/blogs/business-blog/2020/04/using-artificial-intelligence-algorithms

chine 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 (simple) example of a phone company's decision policy for proposing a new offer could 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. We present SMACE (*Semi-Model-Agnostic Contextual Explainer*), a novel interpretability method for composite decision systems. The key idea of SMACE is to agglomerate individual model explanations in a manner similar to how the models themselves are agglomerated by decision rules. 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. We therefore combine an ad hoc method for the interpretability of decision trees, with post hoc methods for the machine learning components.

Contributions. The main contributions of this paper are

- The description of a new method, SMACE, for the interpretability of composite decision-making systems;
- The implementation of SMACE in Python package, available at https://github.com/gianluigilopardo/smace;
- The evaluation of SMACE *vs* some popular methods showing that the latter perform poorly in our setting. A deeper evaluation is available on the SMACE repository, where the experiments are reproducible.

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. In Section 5 we show an evaluation of our method compared to established post hoc solutions. Conclusions are finally drawn in Section 6.

2 Related Work

A decision policy can be embedded in a decision tree. Small CART [Breiman *et al.*, 1984] trees 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, in our framework 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. Anchors [Ribeiro *et al.*, 2018] extract 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]. Nevertheless, Kumar *et al.* [2020] believe that they are inherently inadequate for explainability. Upadhyay *et al.* [2021] extend LIME for business processes, by modifying the sampling step. Anyway, perturbation-based methods have some drawbacks and are not always reliable [Slack *et al.*, 2020]. One of the main problems is the instability of explanations [Visani *et al.*, 2020]. Taking a large number of samples should stabilize them, but the speed of convergence seems very low [Garreau and Luxburg, 2020a]. In addition, methods using linear surrogates clearly suffer from nonlinearities (for instance, added by decision rules), often producing flat results. In the case of LIME for tabular data, this behavior was pointed out by Garreau and von Luxburg [2020] and Garreau and Luxburg [2020a]. Since SHAP belongs to the same class of methods as LIME [Lundberg and Lee, 2017], the same phenomena could be at play. Labreuche and Fossier [2018] leverage 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 *additive feature importance measures* [Lundberg and Lee, 2017], provided for instance by LIME and SHAP, to aggregate the explanations. As far as we know, SMACE is the first method specifically designed for such a setting.

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 and nonlinearities 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 border of decision boundary $\texttt{age} = 21$, *e.g.*, must accurately indicate `age` as an important feature.
- *Out-of-distribution sampling*: the business process or decision rules surrounding a machine learning model will eliminate a large 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 decision 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 we have multiple machine learning models involved in a decision, we

also need to be able to aggregate contributions from multiple models (in addition to combining with rules). What is an accurate way of normalizing and aggregating (partially overlapping) feature contributions from multiple models?

4 SMACE

We now present **SMACE**, *Semi-Model-Agnostic Contextual Explainer*: a new interpretability method that combines a geometric approach (for decision rules) with existing interpretability solutions (for machine learning models) to generate feature importance based explanations.

4.1 Notation

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

The composition of the models and decision policy is illustrated in Figure 1.

4.2 Assumptions

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

A1: 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.

A2: 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].

A3: The machine learning models only use input features to make predictions: we disregard the case 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 **A1** and **A2** refer to the decision rules, while **A3** is the only assumption on 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 of its input features j. By default, we rely on KernelSHAP to allocate these importance values fairly;
- **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 assign the output of each machine learning model to its input values. For instance, this is what SHAP does, and by default we rely on the KernelSHAP implementation. In each 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, as long as it gives a signed value. Given the contribution $\phi_j^{(m)}$ of each input feature j for each machine learning model m we define the normalized contribution as

$$\hat{\phi}_j^{(m)} = \begin{cases} \frac{\phi_j^{(m)}}{\max\limits_{i \in F} |\phi_i^{(m)}|} , & \text{if } \max\limits_{i \in F} |\phi_i^{(m)}| \neq 0 , \\ 0 , & \text{otherwise.} \end{cases} \tag{2}$$

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 $|\phi_j^{(h)}| \gg |\phi_j^{(k)}|$. In order to have a sensical comparison between the models, we therefore need to scale the ϕ values and we use as scale factor the maximum ϕ value 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

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

Note that the second part of Eq. (2) is equivalent to taking the convention $\frac{0}{0} = 0$. The definition implies that if model m relies on a single feature j, the latter will have

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

i.e., the whole contribution of m is attributed to feature j.

4.5 Explaining the rule-based decision

In Section 2 we stated that the set of decision rules 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 rule. A first approach to explain its decision may be to give to the user the trace (see Figure 2) followed by the point within the tree. However, the trace does not contain enough information to understand the situation: a large change in some rules 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 A2, 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 feature j (see Figure 3). 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, in our setting 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 conclusions. The explainability problem is therefore addressed by studying the decision surfaces generated by the tree.

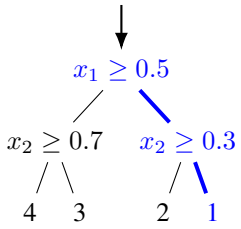

Figure 2: A decision tree classifier based on x_1 and x_2. In **blue and bold**, the trace for leaf 1.

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.

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 3 has two components: h_1 and h_2, along x_1 and x_2, respectively. The projection $\pi_j^{(S)}(x)$

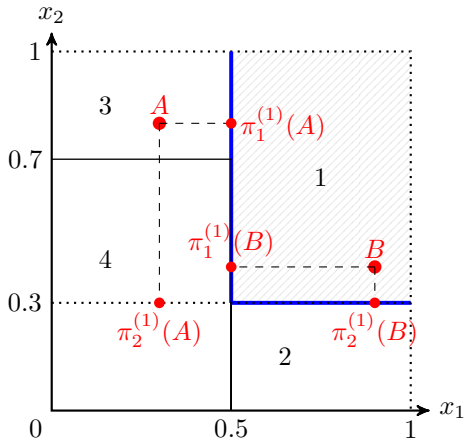

Figure 3: The partition generated by the tree of Figure 2. A and B are instance points, classified respectively as 3 and 1. The decision surface for leaf 1 is in **blue and bold**. The dashed lines indicate the distance between the points and the decision boundaries of leaf 1.

Algorithm 1: Main overview of `smace`.

input rule R (set of conditions), list of models M, example to explain $\xi \in \mathbb{R}^{\mathbb{D}}$
initialize:
$\tilde{\xi} \Leftarrow \xi$, $\phi \Leftarrow \{0\}^N$, $r \Leftarrow \{0\}^{D+N}$, $e \Leftarrow \{0\}^D$;
for $m \in M$ **do**
  // explain the result of model m
  $\hat{\phi}^{(m)} \Leftarrow$ `explain_instance`(ξ, m)
  $\tilde{\xi} \Leftarrow \{\tilde{\xi}, m(\xi)\}$
end
for $j = 1, \ldots, D + N$ **do**
  // explain the rule-based decision
  $r_j \Leftarrow$ `rule_contribution`$(R, j, \tilde{\xi})$
end
for $j = 1, \ldots, D$ **do**
  // aggregate
  $e_j \Leftarrow r_j + \sum_{m \in M} r_m \hat{\phi}_j^{(m)}$
end
return e

Algorithm 2: Computing `rule_contribution`.

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

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 in Figure 3. Let us say we are interested in leaf 1 (the grid subspace shown in Figure 3) generated by the trace in blue. Example B satisfies both conditions, while A only satisfies condition of 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

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

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. 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 .

Note that we designed SMACE to have two desirable properties: *(1) the contribution associated with a feature must be positive if it satisfies the rule, negative otherwise*; *(2) the magnitude of the contribution associated with a feature must be greater the closer its value is to the decision surface*.

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 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.

5 Evaluation

One of the reasons interpretability is challenging is the lack of adequate metrics to measure the quality of explanations. In this section we compare the results obtained with SMACE and those obtained by applying the default implementations of SHAP[2] and LIME[3] on the whole decision system. We first show empirically that SHAP and LIME do not satisfy the properties stated in Section 4.5 and we therefore argue that they are not suitable methods in this context.

The input data consists of 1000 instances, each with three randomly generated components as uniform in $[0, 1]^3$.

5.1 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:

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

[2]https://github.com/slundberg/shap

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

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 is able to capture this information.

condition	$\xi^{(1)}$	**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	$\xi^{(2)}$	**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

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 in generic position. Let us pick the example to explain at generic position with respect to the boundaries:

$$\xi^{(1)} = (0.6, 0.1, 0.4)^\top \,.$$

The decision is 0, since R_1 is not satisfied: the conditions $\xi_1^{(1)} < 0.5$ 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 as

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

In this case, we see that the results of the three methods agree in their signs, with our intuition: property *(1)* is satisfied. 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 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.

Table 3: Simple hybdrid system, explaining the rule-based decision. x_1 is very close to the decision boundary: r_1 is very high, while $r_4 = 0$ since x_4 is not directly involved within the decision policy.

condition	$\xi^{(1)}$	r
$x_1 \leq 0.5$	0.6	-0.90
$x_2 \geq 0.6$	0.1	-0.50
x_3	0.4	0.00
$m_1 \geq 1$	0.6	-0.87
$m_2 \leq 600$	320	0.86

Table 4: Simple hybrid system, comparison on the whole decision system. LIME and SHAP produce flat explanations on x_1 and x_2.

	$\xi^{(1)}$	**SMACE**	SHAP	LIME
x_1	0.6	-0.82	-0.06	-0.19
x_2	0.1	-0.55	-0.06	-0.19
x_3	0.4	0.07	-0.01	-0.03

5.2 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 :

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

and we want to explain the decision for $\xi^{(1)}$. Note that x_1 is not used as input for model m_1 , while x_3 is not (directly) involved in R_3 . Models predict $m_1(\xi^{(1)}) = 0.6$, model $m_2(\xi^{(1)}) = 320$. The rule R_3 is not satisfied. The partial contributions of each feature are shown in Table 3; the comparison on the whole system is in Table 4. Again, LIME and SHAP are producing flat results on x_1 and x_2 , missing useful information. SMACE disagrees with the other methods on the sign of x_3 . However, the value $\hat{\phi}_3^{(1)}$ computed by applying KernelSHAP to m_1 is negative, meaning that x_3 has a negative contribution for the first model (while $\hat{\phi}_3^{(2)} = 0$). In Table 3 we see that the contribution of m_1 to the rule-based decision is also negative: by applying Eq. (1) SMACE returns a negative overall contribution for x_3 , as expected.

More experiments can be found are available on the repository, where SMACE is also applied to more realistic use cases. In particular, we use the churn dataset *DSX Local Telco Churn demo* used by IBM in demo product.[4]

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. It is the first method specifically designed for these systems, to the best of our knowledge. 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. For the sake of reproducibility, SMACE is implemented in a publicly available Python package.

In future work, we plan to extend SMACE by getting rid of some of the assumptions on which it is currently based. The main limitation in our opinion is the restriction to continuous values. A particularly interesting approach to include categorical features 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 can also be generalized to more complex model configurations, where some models take as input the output of other models. The idea is to recursively weigh the importance of each model with the contribution it brings for other models.

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

References

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.

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

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.

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

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.

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

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.

I. Elizabeth 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.

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

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

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

J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

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.

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.

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.

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.

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

Giorgio Visani, Enrico Bagli, Federico Chesani, Alessandro Poluzzi, and Davide Capuzzo. Statistical stability indices for LIME: obtaining reliable explanations for machine learning models. *Journal of the Operational Research Society*, pages 1–11, 2020.

Sandra Wachter, Brent Mittelstadt, and Luciano 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.