POLITECNICO DI TORINO

Master of Science in
Mathematical Engineering

Master Thesis

Explainable AI for business decision-making

Academic supervisor
Elena Maria Baralis (DAUIN)
External supervisors
Damien Garreau (UCA/Inria)
Frédéric Precioso (UCA/Inria)
Greger Ottosson (IBM)

Candidate
Gianluigi Lopardo

September 2021

Summary

Machine Learning (ML) is increasingly being leveraged in business processes to make automated decisions. Nevertheless, a decision is rarely made by a standalone model. Concretely, the reality of business decision services is an orchestration of models, each predicting key quantities for the problem at hand, combined by decision rules to produce the final decision. Applying decision rules on top of ML-based predictions or classifications is typically performed by companies to deliver better conformance, adaptability, and transparency.

Interpretability is a pressing question in these situations. While the field of interpretable ML is full of open challenges in itself, when trying to explain a decision that relies on both business rules and multiple ML models, a number of additional challenges arise. First, the business rules surrounding the models carry non-linearities that cause problems for attribution-based interpretability methods like LIME [Ribeiro et al., 2016] and SHAP [Lundberg and Lee, 2017]. Second, the already transparent business rules represent knowledge that unless exploited will cause problems for sampling-based explanation methods [Jan et al., 2020]. Third, machine learning models with overlapping features will produce conflicting explanation weights. As a result, applying current methods to these real-world decision systems produce unreliable and brittle explanations. In this configuration, there is knowledge that we can exploit to make our explanations process-aware [Jan et al., 2020]. We know which variables are involved in the decision policy and we know its rules. We surmise that it is worth exploiting this information instead of treating the whole system as a black-box and being completely model-agnostic.

In this thesis, we present SMACE - *Semi-Model-Agnostic Contextual Explainer*, a new interpretability method that combines a geometric approach (for business rules) with existing interpretability solutions (for ML models) to generate feature importance based explanations. We show that while LIME and SHAP produce poor results when applied to such a decision system, SMACE provides intuitive feature ranking, tailored to business needs.

Acknowledgements

First of all, I want to thank my supervisors. I thank Professor Elena Baralis, for her willingness to supervise my thesis. Thanks to Damien Garreau, Frédéric Precioso and Greger Ottosson for the constant help, for all that you have taught me. I would especially like to thank Damien. Thank you for your support during these six months, for your dedication and for the passion you have given me. I am happy and proud to keep working with you.

The Maasai team was the best environment I could have wished for doing my thesis, a scientifically stimulating group made up of great and fun people. Thank you all for these months, so happy to have you as colleagues. Thank you for how you welcomed me as soon as I arrived. You managed to give me a great time even in the post-apocalyptic pandemic scenario of empty buildings. Thanks to Giulia, for helping me with the very complicated French bureaucracy. But most of all, thank you for always having fresh beers in your backpack. Thank you Hugo, for teaching me French swear words. And for learning some of them in Italian. Thank you Louis, for organizing the afterworks.

This thesis is for me also an official goodbye to Turin, the city of my heart, and to the Politecnico. I would therefore like to thank all the amazing people I have met over these wonderful years. Thanks to everyone, to my first year classmates, to my friends at JEToP and Visionary. Thank you to my fellow mathematicians: I love the collaborative environment and the discussions we had in and out of class. It has been very sad not to see each other over the past year and a half of online classes.

Needless to say, I couldn't have done it without my friends and my Turin family. Thank you so much to all the people of *Castelo*. I miss our board games, the wine at Arturo's, the San Simone. Thank you Angela and Monica, the best roommates and companions I could have wished for. I miss our talk-filled dinners and talk-free breakfasts like crazy. Thank you Alessandra, companion of a thousand misfortunes. Exam sessions without you are not the same. Thank you Sara, Luca, Marly, Lorenzo, for every second we spent together. Thank you to my favorite *blin blin*, Kevin. You have often been a mentor and reference point for me. Thank you for all your pearls of wisdom.

Many thanks to the best partner in crime. To the best study buddy, literally from the first day to the last day of university. Thank you for bearing my anxieties, for supporting my choices, for always helping me through everything. You're included in just about every thank you above. Thank you, Lucia: the only constant in these five years of variables.

Finally, I want to express my utmost gratitude to my family. Mamma, papà, grazie per il vostro costante sostegno. Grazie per avermi insegnato a perseguire i miei obiettivi. Grazie

per il vostro amore, che ho sentito in ogni istante. Grazie a Maria Antonietta e Alessio, per i nostri discorsi, perché mi aiutate a capire i giovani d'oggi. Grazie ai nonni, per l'appoggio incondizionato. Grazie per tutti i pacchi da giù.

Contents

List of Tables

List of Figures

"*All models are wrong,*
but some models are useful."
[Box, 1976]

Chapter 1

Introduction

Our main interest is the interpetability of decision-making systems that include multiple machine learning models aggregated through decision rules in the form

$$\text{IF } <\text{PREMISE}> \text{ THEN } <\text{CONSEQUENCE}>.$$

PREMISE is a logical conjunction of conditions on input attributes (*e.g.*, age and income 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.*, to make or not to make an offer to a customer). A (simplified) example of a phone company's decision-making policy for proposing a new offer to a customer is

$$\text{IF AGE} \leq 45 \text{ AND CHURN RISK} \geq 0.5 \text{ THEN PROPOSE AN OFFER}.$$

Most state-of-the-art models in machine learning are not interpretable on their own and when several of them are aggregated, explainability is even more challenging. Although very popular in business process automation, the aggregation of machine learning models using business rules is still unexplored in the interpretability literature. Our aim is to propose a method that provides explanations in this setting. In particular, we build a measure of feature importance that responds to some of the open challenges in the business context.

The rest of the chapter is organized as follows. In Section 1.1 we briefly present the historical context of our study, introducing the business setting in which we operate. In Section 1.2 we discuss the motivation for this work, highlighting the business side needs, while in Section 1.3 we set out the direction of the legislation. In Section 1.4 we precise our definition of interpretability. In Section 1.5 we identify the problems we are trying to solve. Finally, we present the structure of the thesis document in Section 1.7.

1.1 A brief historical overview

Business processes are key elements in all industries: they represent the way in which an organisation creates value, by transforming its resources into the final product or service. The way these processes are managed is crucial for a company, which has to maximize its

revenue from limited resources and thus find the most efficient process for its needs. Studies in Business Process Management date back to the first industrial revolution: in 1776 Adam Smith described business processes with the popular example of the production of a pin [Smith, 1827]:

> "One man draws out the wire; another straights it; a third cuts it; a fourth points it; a fifth grinds it at the top for receiving the head; to make the head requires two or three distinct operations; to put it on is a peculiar business; to whiten the pins is another, ..., and the important business of making a pin is, in this manner, divided into about eighteen distinct operations, which, in some manufactories, are all performed by distinct hands, though in others the same man will sometimes perform two or three of them."

In the second half of the twentieth century, several specific business tasks found solutions in early work on artificial intelligence and decision theory. Alan Turing provided many of the theoretical foundations on which these new fields are based, particularly with his studies to model the human brain [Turing, 1950]. Turing worked on the *Entscheidungsproblem* (decision problem) posed by David Hilbert in 1928, thus building what is now known as the "Turing machine": a mathematical model of computation that defines an abstract machine that manipulates symbols according to a table of rules. Decision-making programs are in fact domain-specific expert systems: designed to solve complex problems by reasoning through bodies of knowledge, represented mainly as (deterministic) IF-THEN rules.

Towards the end of the 20th century, the economic environment changed further, driven by technological development, and so did the problems to be managed. For instance, a very popular application of machine learning to business problems is now the automatic reading of documents. LeCun et al. [1998] proposed the use of convolutional neural networks to develop optical character recognition systems for bank checks. The time-consuming work of manually writing down digits and codes from one document to another was thus (partially) automated. Optical character recognition (OCR) systems are now used daily by organisations and businesses with several advantages in terms of time, cost, security and accuracy. Other examples of business processes today are loan applications, insurance claims processing, or the evaluations of whether to propose a retention offer to a customer. In all of these situations, decision-making processes are sets of specific tasks (decisions, predictions, classifications, *etc.*) orchestrated in sequence. Machine learning models are used daily by companies to perform these individual tasks.

1.2 Motivation

Business Process Management is now a growing field dealing with companies operation to discover, model, analyze, measure, improve, optimize, and automate business processes. It has benefited from recent technological advances, especially artificial intelligence (AI). The massive use of the internet and computer tools, together with improvements in the computing power of electronic devices, make it possible today to process huge amounts of data by automated algorithms. The literature is therefore rich of machine learning solutions to reduce

costs and provide better customer experience, for instance with predictive maintenance [Susto et al., 2014] or movie recommendation systems [Reformat and Yager, 2014].

On the other hand, state-of-the-art models are often *black-box* and even a data scientist can have difficulties to explain why a decision-making system made a specific decision. In addition to the complexity of machine learning models, in practice companies rarely make decisions using a single model. Instead, the reality of decision-making services is that of a collection of models, each predicting key quantities for the problem at hand, which are then agglomerated by a decision tree to produce the final decision. In this context, it is crucial for business users to understand why a certain decision has been made and, possibly, to have information simple enough to be able to provide explanations to the customer in a way he or she can understand.

The massive adoption of AI in business automation is hindered by mistrust and risk-aversion [Jan et al., 2020]. This is mainly due to the lack of explanation of specific model predictions, which makes the results difficult to trust for business users, who are typically experts in their domain, but not data scientists. This is the so-called "Last Mile Problem"[1] in data science.

The spread of artificial intelligence in *critical* contexts has rapidly increased the need for transparent models that provide a clearer view of the mechanism that led to a certain decision. In particular, *life-or-death* applications, such as in healthcare or self-driving cars, require interpretability to be able to promptly detect anomalous decisions and ensure that they are taken for the right reasons.

For example, Syed et al. [2019] used deep learning for a diagnostic study to predict enteropathy (a disease of the intestines) and celiac disease (an immune reaction to gluten consumption) in children. A small neural network was trained on duodenal biopsy image data, demonstrating excellent results in terms of accuracy. However, accuracy gives information on the overall performance, but does not give any explanation on an individual decision. To be reliable in such sensitive contexts, an automated system should provide evidence to support its decision. In this way, it can be analyzed by an expert in the field and then challenged or validated.

In contrast, applications in sensitive domains, such as justice, mortgage application, or job application, have often shown discriminatory behaviour towards minority groups, due to bias in the training dataset. Concerns about bias and correctness in machine learning and artificial intelligence in general have in fact motivated many recent studies and debates (see Mehrabi et al. [2021] for a recent survey). Again, having explanations to support a decision can help promptly detect these anomalies, even before the model is deployed in production.

[1] Greger Ottosson. Solving machine learning's "Last Mile Problem" for operational decisions, 2019. Towards Data Science. https://towardsdatascience.com/solving-machine-learnings-last-mile-problem-for-operational-decisions-65e9f44d82b

1.3 Right to explanation

Interpretability is deeply linked to trust and, as a result of growing public concern, has also recently become a regulatory issue. In 2016 the European Union Parliament approved the General Data Protection Regulation (GDPR) [EU, 2016], an important collection of rules concerning data protection and privacy. Recital 71 of the GDPR states

> "The data subject should have the right not to be subject to a decision, which may include a measure, evaluating personal aspects relating to him or her which is based solely on automated processing and which produces legal effects concerning him or her or similarly significantly affects him or her, such as automatic refusal of an online credit application or e-recruiting practices without any human intervention.
>
> [...]
>
> In any case, such processing should be subject to suitable safeguards, which should include specific information to the data subject and the right to obtain human intervention, to express his or her point of view, to obtain an explanation of the decision reached after such assessment and to challenge the decision.
>
> [...]
>
> Automated decision-making and profiling based on special categories of personal data should be allowed only under specific conditions."

The way this article is worded makes it controversial and in fact does not impose a legal obligation, as outlined by Wachter et al. [2017], but outlines a clear direction of regulation. In 2019 the EU High-Level Expert Group on AI presented Ethics Guidelines [EU, 2019] proposing seven keys requirements that AI systems should meet in order to be deemed trustworthy. One of them is Transparency:

> "The data, system and AI business models should be transparent. Traceability mechanisms can help achieving this. Moreover, AI systems and their decisions should be explained in a manner adapted to the stakeholder concerned. Humans need to be aware that they are interacting with an AI system, and must be informed of the system's capabilities and limitations."

In 2021, the European Commission published a proposal for a Regulation "laying down harmonised rules on artificial intelligence" [EU, 2021]. This aims to introduce a requirement for explainability in applications considered to be high risk.

In the United States, the Federal Trade Commission guidelines dictate that if consumers are denied something of value (*i.e.*, a loan) based on AI, they are entitled to an explanation and they state that when assigning risk scores to consumers, the key features affecting said scores ought to be disclosed in rank order of importance [Smith, 2020].

However, it is important to note that there is no agreement on a precise mathematical or legal definition of what an "explanation" is. The research field of interpretability is very young

and the literature is not yet mature enough. The crucial year was 2016, when DARPA (the US Defense Advanced Research Projects Agency) published a call for funding for Explainable Artificial Intelligence (XAI) projects[2]:

> "DARPA is soliciting innovative research proposals in the areas of machine learning and human-computer interaction. The goal of Explainable Artificial Intelligence (XAI) is to create a suite of new or modified machine learning techniques that produce explainable models that, when combined with effective explanation techniques, enable end users to understand, appropriately trust, and effectively manage the emerging generation of Artificial Intelligence (AI) systems. Proposed research should investigate innovative approaches that enable revolutionary advances in science, or systems. Specifically excluded is research that primarily results in evolutionary improvements to the existing state of practice."

1.4 Context

The lack of agreement in the scientific community about what a good explanation should be makes explainability even more challenging, because different needs can lead to conflicting implications. A very popular non-mathematical definition by Miller [2019] is *"Interpretability is the degree to which a human can understand the cause of a decision."* There is also disagreement on the relationship between the terms "explainability," "interpretability," and "explanation." We follow Molnar [2020] and we use "interpretable" and "explainable" interchangeably, while "explanation" refers to the outcome of the explainability of an individual prediction.

State-of-the-art decision-making services in the industry are still largely partially rule-based and therefore we have to take this specific structure into account when trying to explain these systems. In fact, one could think that this trend is disappearing, because the entire decision-making system can be managed by a single machine learning model. We argue that this is not necessarily true, for two main reasons.

Firstly, *decision rules are crucial for expressing policies that can change (even very quickly) over time.* For example, depending on the economic results in the last quarter, a company might be more or less risk-averse and therefore have a more or less conservative policy. With one large machine learning model, it would have to be trained all over again with new data to which the new policy is applied. Instead, with a rule-based system it is possible to manage risk appetite by changing just one parameter.

In addition, *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 application domain needs or regulation. For example, we may have to require the age of a customer to be greater than 18 in order to be able to offer them a certain deal.

[2]DARPA Broad Agency Announcement. Explainable Artificial Intelligence (XAI). https://www.darpa.mil/attachments/DARPA-BAA-16-53.pdf

Machine learning relies mainly on probabilistic methods, which makes it difficult for it to accurately adhere to strict deterministic rules. Applying decision rules after machine-learning based predictions or classifications will typically deliver better conformance, adaptability and transparency.

As we mentioned, different needs and contexts can lead to opposing interpretability requirements. The final identified aim of interpretability is what characterizes a method and its explanations, as we will se in Chapter 2 and in Chapter 4. In the case of feature importance based explainability methods, Bénard et al. [2021b] distinguish between two cases. The first, which we might say is *engineering-oriented*, is to find a small number of variables that maximize the accuracy of a system. This is for instance the case of an optimization model used in an industrial automation process. To make the model more flexible and faster, the engineers would like to use as few variables as possible, while still ensuring good performance. The second possible aim, which we say is more *business-oriented*, is to rank all directly and indirectly influential variables so that they can be analyzed with a fair priority by a domain expert.

These two cases, which may seem similar, require different explanations for each other. For example, if two variables are correlated, one should be neglected in the first case, while it should show up as an influential variable in the second. In fact, in the first case having both variables is wasteful and does not bring additional useful information to the model. In the second case, instead, a good method of interpretability must evidence to the expert of the domain (the *business user*) both the variables in order to be able to validate, to contest or to modify the decision.

Our aim falls completely into the second case, with an additional requirement due to the interaction between decision rules and machine learning. The framework in which we work is characterized by two levels. *Level 1* is the one accessible by the business user, who manages or knows the decision policy (*i.e.*, the decision rules), the data of the customer under consideration, and the values for predicted by the models internally in the company. The end user (*e.g.,* consumer) only perceives *Level 2*: based on the data provided, the company makes a decision, and the end user is the recipient of that decision. Explanations about this decision must be based solely on this data. The end user does not know the internals of the decision-making, such as the machine learning models used by the company and the company does not want to reveal its decision policy.

1.5 Challenges

The field of interpretable machine learning is full of open challenges in itself. To date, it is very difficult to interpret the predictions taken by most models. The methods mainly used for explainability also have several drawbacks in various aspects, that we will discuss further in Chapter 2, and in any case do not always meet the needs of the user.

In addition, when trying to explain a decision that relies in part on multiple machine learning models, a number of other problems arise:

- *Rule-induced non-linearities*: decision rules will cause sharp borders and non-linearities in the decision space. For example: a car rental rule might state "AGE of renter must be

above 25." Explanations for a machine learning based risk assessment close to the border of decision boundary AGE = 25, *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 are known to distort explanations because of this (see Section 2.8.1).
- *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? For example, let us say we predict RISK with a machine learning model, and we then consider the rule IF RISK > 80% THEN DECLINE APPLICATION. How would we combine and present feature importance with the risk rule? Would a RISK of 81% (close to rule boundary) indicate we should scale the machine learning explanations differently from a case where RISK is 99% (far from rule boundary)?
- *Multiple machine learning models*: when we have multiple machine learning models used for a decision, we also need to be able to aggregate attribution weights from multiple models (in addition to combining with rules). What is an accurate way of normalizing and aggregating (partially overlapping) feature weights from multiple models? For example, one classification model predicts CHURN RISK for a customer, which a probability in [0,1]. Another regression model predicts LIFE-TIME VALUE, which is a quantity measured in some currency with values in $(-\infty, +\infty)$. If we apply an explanatory method to both models individually, what is an accurate way of combining these importance weights?

1.6 Contributions

Our interest in this thesis is the interpretability of hybrid decision systems used in business contexts. Our contributions are as follows:

- *A literature review* of interpretability methods, decision systems, and machine learning algorithms used by companies. We identify the methods, systems, and models closest to our framework and analyze their strengths and weaknesses. To the best of our knowledge, no one method is satisfactory for the problem we want to tackle.
- *The proposal of a new method*, SMACE, to address all of the challenges described above by providing local contextual explanations based on feature importance.
- *Implementation* of SMACE in Python package, available on Github at `https://github.com/gianluigilopardo/smace`.
- *Evaluation* of SMACE vs some popular methods showing that the latter perform poorly in our business context (due to complications brought by the business rules), and that

SMACE deliver better on the challenges described in Section 1.5. The evaluation reported in the thesis can be found at https://github.com/gianluigilopardo/smace/evaluation and all experiments are reproducible.

1.7 Structure

The thesis is organized as follows. First, we present some related work and its relation to the problem at hand. In Chapter 2 we present the state-of-the-art of post-hoc methods for the interpretability of individual black-box machine learning models, discussing their limitations and drawbacks. In Chapter 3 we present some studies of rule-based systems that work in contexts similar to ours, but without accounting for machine learning components. In Chapter 4 we list some popular machine learning models that are used in practice on a daily basis in business. They are usually simple models based on tree-ensembles, fast to train, and do not require large training sets. However, in more complex cases, companies develop ad-hoc machine learning methods tailored to their needs: we will see a few examples. SMACE is presented in Chapter 5, where we also show how it works on some small examples. In Chapter 6 we apply SMACE to use-cases of increasing complexity, comparing it to some of the more common interpretability methods. We will highlight how methods that do not tend to account for the structure of the decision system actually fail to capture information and thus fail to provide useful explanations. We draw our conclusions in Chapter 7, discussing the results of this thesis and presenting some points we aim to work on in the future.

Chapter 2

Interpretable machine learning

2.1 Introduction

Artificial Intelligence has entered nearly every scientific field and every economic or social context. Machine learning is the popular framework at the moment, as the algorithms developed within this framework tend to perform better than humans in classification and prediction tasks. Even when they perform just as well or slightly worse, there are still huge advantages when using these methods in terms of speed, reproducibility, and scalability. For example, in the case of OCR mentioned in the previous chapter, a human operator could possibly achieve better accuracy than an automated system. However, if the operator takes about one second to transcribe a word, while the OCR takes one millisecond, the automated solution is preferred. This has allowed machine learning algorithms to be intensively used nowadays in various domains and in many business.

On the other hand, the complexity of the state-of-the-art models makes it impossible to understand why a particular decision has been made. We can distinguish two main situations. In the first, there is no access to the model, so no knowledge of how it works. This is very common when a company buys a model externally. The second situation is when one has total access to the model, but its complexity does not allow the understanding of individual predictions. For example, the popular convolutional neural network AlexNet [Krizhevsky et al., 2012] used in image classification has 60 million parameters.

In some cases, this is not an issue: we just want to know that a system works well. This happens in low-risk problems (like movies recommender) or in well-studied fields (like optical character recognition systems). However, in critical contexts and for different reasons (we may want to ensure the model's fairness, to know why a loan has been refused, *etc.*), interpretation can be crucial: we also care about why a prediction has been made, not only which one. In practice, there is a trade-off between interpretability and accuracy. Explaining machine learning models can help us ensuring fairness, robustness, causality and trust. Depending on the objective and needs, there are different types of methods. In fact, interpretability methods can be used for different purposes, there are methods for explaining complex black-box models, methods for creating white-box (intrinsically interpretable) models, methods

that promote fairness and restrict the existence of discrimination, methods for analysing the sensitivity of model predictions.

A comprehensive literature review is presented in an online book by Molnar [2020], an excellent introduction to the field. The surveys of Guidotti et al. [2018] and Adadi and Berrada [2018] attempt to bring order to the taxonomy by classifying different interpretation methods and them according to the purpose of the explanations. The most up-to-date review paper is by Linardatos et al. [2021], making distinction among the terms "interpretability" and "explainability." A popular definition of interpretability is "the ability to explain or to present in understandable terms to a human," but, as already mentioned, there are no formal mathematical definitions. The subtle difference is that interpretability refers to the relationship between the input and the output, while explainability is related to the understanding of the model's mechanism.

2.2 Taxonomy

Different criteria are generally admitted by the community to classify interpretability methods [Molnar, 2020]:

- *Intrinsic or post-hoc*: intrinsic refers to models that are interpretable thanks to their simple structures (*e.g.*, simple linear models and small decision trees), while *post-hoc* methods analyze models after training.
- *Model-specific or model-agnostic*: the first refers to methods restricted to a particular class of models; the second can be used *post-hoc* to any model seen as a black-box.
- *Local or global*: according to whether explanations apply to individual predictions or to the entire model.
- *Results* of the interpretation: can be statistics, visualizations, example of points, model internals.

Concerning interpretability methods for machine learning models, in this thesis we focus on **post-hoc**, **model-agnostic**, **local** methods, providing feature importance. In fact, we want SMACE to be applicable to an already structured decision-making system whose models are already trained. We want it to be robust to model changes and adaptable to any type of model. We focus on local explanations because we are interested in explaining individual decisions made for a user. Note that we are not completely model-agnostic: we know the graph structure of the decision system and can call the models individually. Model-agnosticity refers only to machine-learning models. We therefore say that SMACE is semi-model-agnostic.

2.3 Overview

Practically, the simplest way to have good explanations is to use intrinsically interpretable models, like linear models with few parameters and small decision trees. We can easily

understand the behaviour of a model obtained by linear regression by looking at the weight of each feature (assuming they are standardized). For instance, let us consider the problem of computing a credit score from historical data with 5 parameters:

- x_1 = character,
- x_2 = capital,
- x_3 = collateral,
- x_4 = capacity,
- x_5 = condition.

Let us image the linear model is $f(x) = 0.1x_1 + 5x_2 + 3x_3 + 10x_4 + x_5$. This easily tells us that that increasing collateral by one unit increases credit score by 3 units.

Likewise, small decision trees like the one in Figure 2.1 produce very simple and short rules of the type IF $x_1 > v_1$ AND $x_2 > v_2$ THEN y. Unfortunately there are drawbacks: in general, intrinsically interpretable models do not achieve a good accuracy, especially if we try to keep them as simple as possible.

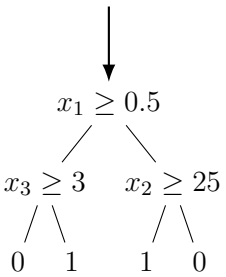

Figure 2.1. A small decision tree for binary classification with three attributes. For example, point $\xi = (0.8, 12, 4)^\top$ is classified as 1 because $\xi_1 = 0.8 \geq 0.5$ and $\xi_2 = 12 < 25$.

For example, we can easily explain a prediction made by a decision tree if it involves five parameters, but generally this will be less accurate than a model with dozens of parameters, which would be difficult to interpret.

Model-agnostic methods are very flexible, both in terms of model, of explanation and of representation. They give *post-hoc* explanations regardless of the model. The point is not only that we can use it with each kind of algorithm, but often multiple models are combined in a machine learning system or several of them are evaluated to solve the same task. Some techniques provide global explanations, meaning that we can understand the global behaviour of a model. The simplest idea to do this is through feature importance: we rank features by measuring the impact on the model's outcome when permuting them. Better methods also keep into account features interaction. Other techniques such as the locally interpretable methods explain individual predictions. This is what LIME [Ribeiro et al., 2016] and SHAP [Lundberg and Lee, 2017] do for example: by perturbing data, they approximate complex

black-box models with intrinsically interpretable models. In Section 2.5 and Section 2.3 we explain them in more detail. Anchors [Ribeiro et al., 2018] also provides local explanations, with the added benefit of taking global coverage and precision into account, as we discuss in Section 2.7. Therefore, anchors provide local explanations that also make sense globally. The drawback is that the method is complex with respect to (the general structure of) LIME: it uses reinforcement learning, graph search algorithm, and optimization methods. All of the above are perturbation-based methods.

Gradient-based methods are an alternative. They analyze the impact that small changes in the input have on the output. As intuitive, there is a deep connection between these two classes of methods. This connection is illustrated by Agarwal et al. [2021], showing that (with proper assumptions) methods of the two different classes converge to the same explanation. However, we do not treat this class: the decision rules make the system piecewise flat the gradient is always zero.

Another general framework are example-based explanations, where instances of the dataset are used to explain the model. An idea of this approach is given by the k-nearest neighbours algorithm. We can explain a prediction by looking at its k closest neighbours: we see that an instance is classified in a certain way because it is similar (in a sense defined by the underlying metric) to other instances that we know to belong to that class.

2.4 User-friendly Explainable AI

Despite today's relative immaturity of methods of explainability, due to the growing demand on the business side, many large companies have started to incorporate them into products and toolkits.

In Arya et al. [2020], IBM presented AI Explainability 360: "an extensible toolkit for understanding data and machine learning models". The paper also gives a taxonomy to make order in the field. The toolkit implements different explainability methods in order to provide the best suitable information to each type of user and according to the phase of the modeling. The increasing use of AI applications and their permeability in all type of industries have caused demand for explanations to spread. Despite the huge work of the community, there is still a big gap between the solutions provided and the needs of the users. This is also due to the fact that there is no common definition of what an explanation should be, because AI *consumers* have a very large variability in their needs and backgrounds. As a response, the tool includes *persona-specific explanations*.

For instance, in the case of a loan application Arya et al. [2020] distinguish three types of *personas*, with different needs:

- *the data scientist*, who must ensure that the model works appropriately before deployment and would ideally like to understand the behavior of the model as a whole, not just on specific loan applicants;
- *the loan officer*, who needs to assess the model's prediction and make final judgement: she wants to understand how and why the model came to that prediction in order to make an informed and trusted decision;

- *the bank customer*, who want to understand the reason for the application result: how and why the decision was made to accept or reject their loan application.

Each of these users is presented with explanations in a different format, generated by different methods.

Chari et al. [2020] analyze future direction for user-friendly explainable AI, stating that researchers in explainable AI have been guided by the capabilities of the models, instead of focusing on the users, who should have the right to understand the results provided by systems. Instead, they should be adapted "to the end user's *understanding, context, and current needs.*"

To go this way, it is worth to look at results of social sciences and psychology and find links with the explainable AI state-of-the-art. By combining the classification of explanations in the two areas, we can provide a more comprehensive taxonomy. The outputs of the explanations can be

- *Case-based*: shows actual (similar) prior cases for which the same apply;
- *Contextual*: refers to information about user, situation and environment;
- *Contrastive*: put in evidence differences between the actual input and output and the event that should have generated the output of interest;
- *Counterfactual*: what results if other inputs would be provided;
- *Everyday*: real-world general (basic) common knowledge;
- Scientific: on a rigorous proof based on the scientific method;
- *Simulation-based*: results generated by an implemented imitation of the system of interest;
- *Statistical*: evidence based on data statistics;
- *Trace-based*: the decision path followed to reach the decision.

Chari et al. [2020] also state that, regardless of approach, good user-centric explanations should be personalized, trustworthy, and context-aware. It is important to point out that "different situations, contexts, and user requirements demand explanations of varying complexities, granularities, levels of evidence, presentations, *etc.*" The choice of the method to use strictly depends on the application: whether the task is classification or regression and if we deal with tabular data, images, or texts. We focus on **tabular data**, that is, data lying in $\mathbb{R}^{n \times d}$. In other words, an orderly arrangement of n rows and d columns. In our notation, each row represents an instance, vector or point. The columns are parameters, attributes, or features. Below we introduce LIME and SHAP, focusing on tabular data. These are two local post-hoc methods that are very popular right now, and are the building blocks of SMACE.

2.5 LIME — Local Interpretable Model-agnostic Explanations

LIME [Ribeiro et al., 2016] stands for Local Interpretable Model-agnostic Explanations and it is a very popular method that includes three different versions depending on whether the data is image, text or tabular. Many extensions and improvements have been proposed. For example, Kovalev et al. [2020] presented SurvLIME: an extension of LIME to survival models. We focus on the standard tabular data version. The general idea is to approximate the black-box model in the neighborhood of the example ξ to explain with an instrinsically interpretable local linear surrogate.

In fact, one can see every complex machine learning model operating on tabular data as a function $f : \mathbb{R}^D \to \mathbb{R}$ and a local surrogate for model f and instance x is defined as

$$\mathrm{e}(x) \in \arg\min_{g \in G} \left\{ L(f, g, \pi_x) + \Omega(g) \right\}, \tag{2.1}$$

where G is the set of linear functions, L a loss function, π_x a proximity measure and $\Omega(g)$ is the complexity of the surrogate model, that we want to keep low. LIME only requires the access to a training set $\mathcal{X}$ and the possibility to query the model.

Let us say we have tabular data in $\mathbb{R}^d$ and we want to explain the instance ξ. The default tabular version performs five sequential steps:

1. **binning**: create p boxes long each coordinate corresponding to the quantiles. This step splits each axis in p ($p = 4$ by default) range, each with equal number of observations and these are the interpretable features. This binning procedure lays the foundation for producing explanations of the form $a < x_1 < b$;

2. **bin sampling**, sample box IDs at random: for each new example i, a vector of boxes IDs $b_i \in \{1, \ldots, p\}^d$ is sampled uniformly random. Let b_j^* be the box on coordinate j containing ξ_j, then the interpretable features are defined as $z_{i,j} = 1 \iff b_{i,j} = b_j^*$.

3. **data generation**: sample truncated Gaussian $x_1, \ldots, x_n \in \mathbb{R}^d$ in the boxes, whose μ and σ parameters are derived from the training set $\mathcal{X}$;

4. **weight**: define positive weights π_i depending on the distance between each x_i and ξ;

5. **surrogate model**: fit a linear model on each z_i with ridge regression:

$$\hat{\beta}_n \in \arg\min_{\beta \in \mathbb{R}^{d+1}} \left\{ \sum_{i=1}^{n} \pi_i (y_i - \beta^\top z_i)^2 + \lambda \|\beta\|^2 \right\}. \tag{2.2}$$

The result is a linear model and the contribution of each coordinate $j \in \{1, \ldots, d\}$ to the prediction is simply given by coefficient $\hat{\beta}_j$.

It is important to note that the choice of the weights π is crucial and it really changes the outcome. Although simple, this method has several shortcomings from various points of view. A comprehensive analysis of this method for tabular data has been made by Garreau

and von Luxburg [2020] in a simplified setting, extended to the default implementation in Garreau and Luxburg [2020a]. Most of the problems with this method are due to sampling, which causes very unstable explanations. This will be discussed in Section 2.8.1.

2.6 Shapley values and SHAP – SHapley Additive exPlanation

The problem of fairly finding features importance in the prediction of a machine learning model can be addressed from a Game Theory perspective. We consider a D-players game where each feature $j \in \{1, \dots, D\}$ is a player and we want to value their contribution. There are 2^D possible coalitions and each coalition S is associated with a characteristic function

$$v : 2^D \to \mathbb{R} \,.$$

The Shapley value [Shapley, 1953] of player j is

$$\phi_j(v) = \sum_{S \subseteq \{1,\dots,d\} \setminus \{j\}} \frac{|S|!(D - |S| - 1)!}{D!} [v(S \cup j) - v(S)] \,. \tag{2.3}$$

The idea is that if player j plays much better than the other, then $v(S \cup \{j\})$ is consistently higher than $v(S)$ and therefore $\phi_j(v) \gg 0$.

The general idea shared by several explanation methods is to locally approximate the original model with an "explanation model," which is simpler to interpret. SHAP (Shapley Additive Explanations) [Lundberg and Lee, 2017] is a framework unifying some of these interpretability methods in the class of *"additive feature attribution methods"* and providing feature importance measures, based on a solid theory.

Let us say we want to explain the prediction $f(x)$ made by model f for instance x. We consider a simplified instance x' and a mapping function $x = h_x(x')$, such that if $x' \approx z'$, then $g(z') \approx f(h_x(z'))$ and $g(x') = f(h_x(x')) = f(x)$. The class of additive feature attribution methods is defined having an explanation models in the form

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \,, \tag{2.4}$$

with $z' \in \{0,1\}^M$, so that ϕ_j is the importance of attribute j. We can rewrite Eq. (2.3) as

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup i}(x_{S \cup i}) - f_S(x_S)] \,, \tag{2.5}$$

where f_S is the model (re)trained on the subset S of the total attributes F. Obviously this is extremely expensive: we would have to train $2^{|F|}$ models. In practice, approximations are used and these characterise a method.

Lundberg and Lee [2017] defines three desirable properties for this class of methods and identify a unique solution with these properties:

- *Local accuracy*: when approximating f in x, the explanation model g must at least equal it in the simplified instance x' corresponding to x: $f(x) = g(x')$;
- *Missingness*: if a feature is missing, its attribute is null: $x'_i = 0 \Rightarrow \phi_i = 0$;
- *Consistency*: if a model changes so that some simplified input's contribution increases or stays the same regardless of the other inputs, that input's attribution should not decrease.

The model in the class of additive feature attribution methods satisfying these properties is

$$\phi_j(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \setminus j)], \tag{2.6}$$

where $|z'|$ is the number of non-zero entries in z, and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x'.

SHAP values is presented as the Shapley values of a conditional expectation function of the original model, meaning $f_x(z') = \mathbb{E}[f(z)|z_S]$. In the case of a linear model in the form $f(x) = \sum_{j=1}^{M} \omega_j x_j + b$, SHAP values are

$$\phi_j(f, x) = \omega_j(x_j - \mathbb{E}[x_j]). \tag{2.7}$$

2.6.1 KernelSHAP

A popular implementation of this approach is the KernelSHAP (SHapley Additive exPlanations) algorithm. It "provides model-agnostic, human interpretable explanations suitable for regression and classification models applied to tabular data." This method is a member of the additive feature attribution methods class; feature attribution refers to the fact that the change of an outcome to be explained (*e.g.*, a class probability in a classification problem) with respect to a baseline (*e.g.*, average prediction probability for that class in the training set) can be attributed in different proportions to the model input features.

For each instance x, KernelSHAP performs 5 steps:

1. **sample coalition**: $z'_k \in \{0,1\}^M$;
2. **get prediction** for each z'_k by applying $f(h_x(z'_k))$;
3. **compute the weight** for each z'_k with the SHAP kernel;
4. **fit** weighted linear model;
5. **return Shapley values** ϕ_k the coefficients from the linear model.

The function $h_x : \{0,1\}^M \to \mathbb{R}^d$ is such that $h_x(z') = z$. It maps 1 to the corresponding value from the instance x that we want to explain and 0 to the values of another instance that we sample from the dataset. The linear surrogate model g of Eq. (2.6) is then obtained by minimizing the loss function

$$L(f, g, \pi_x) = \sum_{z' \in Z} [f(h_x(z')) - g(z')]^2 \pi_x(z') \,. \tag{2.8}$$

Note that this formula is similar to that of LIME. The great difference stands in the weight function:

$$\pi_x(z') = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)}\,.$$

In addition to SHAP, several works have followed this approach. Lipovetsky and Conklin [2001] applied Shapley values to explain multiple regression models, while Štrumbelj and Kononenko [2014] analyzed them from a sensitivity analysis perspective. Datta et al. [2016] introduced the family of Quantitative Input Influence measures to capture the degree of influence of inputs on outputs of systems. Merrick and Taly [2019] perform an in-depth study of various Shapley-value-based model explanation methods, showing how the formalization of the underlying game has a strong impact on explanations. Frye et al. [2020] exploit Shapley values to incorporate causal knowledge into model-agnostic explainability. Aas et al. [2021] extend the KernelSHAP method to handle dependent features.

2.7 Anchors: High-Precision Model-Agnostic Explanations

Anchors [Ribeiro et al., 2018] is a model-agnostic local explainability method for Text and Tabular data that extracts *sufficient* conditions for a certain prediction, in the form of rules. Like LIME, this method is also perturbation based, but while LIME locally approximates a black box model with a linear function, Anchors looks for high precision rules that better explain a prediction. An *anchor* is an IF-THEN rule that "that sufficiently anchors the prediction locally – such that changes to the rest of the feature values of the instance do not matter."

Formally, let us consider the black-box model $f : X \to Y$, and let us say we want to explain the prediction for instance $x \in X$, *i.e.*, $f(x)$. A rule A is a set of predicates such that $A(x) = 1$ if all the predicates are satisfied for x. Let $\mathcal{D}(\cdot|A)$ be the conditional distribution when A applies, such that, for each sample z from $\mathcal{D}(\cdot|A)$, $A(z) = 1$. A is therefore an anchor for x if $A(x) = 1$, and A is a sufficient condition for $f(x)$ with high probability. Mathematically, we say that A is an anchor if

$$\mathbb{E}_{\mathcal{D}_x(z|A)}[\mathbb{1}_{f(x)=f(z)}] \geq \tau \quad \text{and} \quad A(x) = 1\,, \tag{2.9}$$

where the hyperparameter τ is desired level of precision. For example, $\tau = 0.9$ means that we require the anchor to have an accuracy of at least 90%. Once τ is fixed, the method searches for the anchor with the higher coverage, aiming to find rules that apply to a preferably large part of the model's input space. Formally, the coverage is defined as $cov(A) = \mathbb{E}_{\mathcal{D}_{(z)}}[A(z)]$.

In general, there is a trade-off between accuracy and coverage. A good explanation needs to be locally accurate, but in order to make predictions about the behavior of the model, you also need to have more general knowledge, hence more coverage.

As an Example of explanation, using the popular dataset *Titanic - Machine Learning*

from Disaster[1], Anchors gives

IF `sex=female`
AND `class=first`
THEN PREDICT `survived=true`
WITH PRECISION 97%
AND COVERAGE 15%.

The usability of the method is finally evaluated in Ribeiro et al. [2018], with a series of experiments. The principle guiding this evaluation is that a good explanation should make the user able to accurately predict the behavior of the model on new, unseen instances.

Anchors is an innovative method, but it has a number of shortcomings in common with other perturbation-based methods. It requires a tuning step for choosing parameters, which can lead to different explanations. The choice of the desired precision, is very delicate, because it can generate too complex explanations with low coverage, or explanations that are too trivial.

2.8 Problems with existing methods

The problem of interpretability of machine learning models is far from being solved and even the most common methods still have many critical issues. Knowing their limitations and shortcomings is important as we build our approach on the basis of these methods.

2.8.1 Problems with perturbation-based methods

Slack et al. [2020] highlight problems in perturbation-based methods in "Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods." Post-hoc methods based on input perturbation such as LIME and SHAP are very popular, however, the paper shows that these methods are not reliable. It proposes a framework which could be used by an adversary to hide the bias of its model from customers or regulators.

The main idea behind this framework is that instances generated using perturbation could be out-of-domain (for instance, a negative age value) or out-of-distribution (*e.g.*, many ultracentarians). This is often the case, and makes it possible to distinguish between original and perturbed instances and act differently on them. Let us say that real world data $\mathcal{X}$ on which the classifier f could be applied follows a distribution $\mathcal{P}$. The adversary could design an adversarial classifier that applies the original biased classifier on instances sampled from $\mathcal{P}$ and an unbiased classifier ψ on instances that do not. So that, the adversarial classifier e takes the form

$$\tilde{f}(x) = \begin{cases} f(x) & \text{if } x \text{ was drawn according to } \mathcal{P}\,, \\ \psi(x) & \text{otherwise}\,. \end{cases}$$

[1] Titanic - Machine Learning from Disaster. https://www.kaggle.com/c/titanic

To create such a classifier, we need a way to decide whether a given instance x comes from $\mathcal{P}$ or not. A classifier `is_OOD` is build, such that `is_OOD(`x`)=True` if $x \in \mathcal{X}$ and it is true if x is out-of-distribution (OOD). This classifier is trained on the combined dataset $\mathcal{X} \cup \mathcal{X}_p$, where $\mathcal{X}$ is the real-word dataset and $\mathcal{X}_p$ is generated by perturbing data in $\mathcal{X}$ and we add a class label indicating whether the instance comes from the original data or it is a perturbation sample. The unbiased classifier ψ is build on synthetic *uncorrelated features* that have zero correlation with the sensitive attributes (race, gender).

Both LIME and SHAP can get fooled with this approach, even if there are some differences in the behaviour due to their internal structure. LIME is very vulnerable to these attacks once the `is_OOD` is well performing, while SHAP starts responding to attacks with a less accurate classifier, but then the explanations deteriorate more gradually. The effectiveness of this approach is based on the ability to distinguish between real-world and sampled instances, which is by no means simple. However, if this classification is well performing, the adversarial classifier completely hide the bias.

Another major problem with perturbation-based methods is that of instability. With each use, explanations can change, even in contradictory ways. This is due to the random nature of sampling. Even though taking a large number of samples should stabilize things [Garreau and Luxburg, 2020a], it is not always doable (larger computational cost), and the speed of convergence seems very low. Visani et al. [2020] analyzed this problem in LIME, providing a solution to alleviate it based on stability indicators.

2.8.2 Specific problems with LIME

Normalization problem. As detailed above, LIME explains any machine learning model by approximating it locally with an instrinsically interpretable model. The default implementation uses a linear regression model. In the one-box version ($p = 1$), the importance of an attribute is given by the beta coefficient of the respective regressor. However, we claim that the results provided by the original implementation[2] user interface are not what one would intuitively expect from reading the paper [Ribeiro et al., 2016] and especially seeing Figure 3 therein. We show with a simple example that this is due to an internal standardization of the data.

Let us consider a simple linear model in the form $y = \beta_0 + \beta_1 x$, with $\beta_0 = 0$ and $\beta_1 = 30$, *i.e.*, $y = 30x$. We generate a uniform random sample in [0,1] of size 1000. Let us say we want to explain the prediction for the example $\xi = 0.42$. The model output is

$$y = \beta_0 + \beta_1 \xi = 30\xi = 12.6\,.$$

Fitting the data with linear regression from the `sklearn` library we obtain

$$\begin{cases} \texttt{Intercept:} & \hat{\beta}_0 = 0.0\,, \\ \texttt{Coefficient:} & \hat{\beta}_1 = 30\,, \end{cases}$$

[2]LIME: Explaining the predictions of any machine learning classifier. https://github.com/marcotcr/lime

so that $y = \hat{\beta}_0 + \hat{\beta}_1 \xi = 12.6$. `sklearn` manages to derive the intercept and the coefficient of the model. We expect the same results from LIME.

LIME user interface with the `verbose=True` option returns

$$\begin{cases} \texttt{Intercept:} \quad 15.02\,, \\ \texttt{Prediction\_local:} \quad 12.6\,, \\ \texttt{Right:} \quad 12.6\,. \end{cases}$$

`Prediction_local` gives the right result, but what does `Intercept` represent? We go deeper using attributes of the object `Explanation`:

$$\begin{cases} \texttt{Intercept:} \quad \hat{\beta}_0 = 15.02\,, \\ \texttt{Coefficient:} \quad \hat{\beta}_1 = 8.64\,. \end{cases}$$

The model would be $y = \hat{\beta}_0 + \hat{\beta}_1 \xi = 18.65$. This is not what expected: we show that it is due to the fact that the data is scaled.

We standardize the original data and fit a linear regression with `sklearn` as above, showing that this gives the same results that LIME gives with the original data.

We scale the data as follows:

$$\tilde{x_i} = \frac{x_i - \mu_x}{\sigma_x}\,,$$

where μ_x is the mean value of the dataset and σ_x is the standard deviation. We now fit scaled data $\tilde{x}$ with a `sklearn` linear regression model:

$$\begin{cases} \texttt{Intercept:} \quad \hat{\beta}_0 = 15.02\,, \\ \texttt{Coefficient:} \quad \hat{\beta}_1 = 8.65\,. \end{cases}$$

We recover: $y = \hat{\beta}_0 + \hat{\beta}_1 \xi = 18.65$, *i.e.*, the same results obatined by applying LIME to the original data x.

Plain sampling problem in 2D. Let us consider a two-dimensional problem, so we use LIME to explain a model with only two features. The training set points are as shown in the left panel of Figure 2.2, where clearly the data is not uniform in space $[0,1]^2$. Applying LIME with two boxes ($p = 2$), the quantiles correspond with the median value along the two axes (middle panel of Figure 2.2). In each of the four quadrants thus generated, LIME will sample according to a normal distribution of the same size, regardless of the initial data densities within them. The right panel of the Figure 2.2 shows the density of points generated by LIME: it is markedly different from the initial distribution. This sampling problem can greatly distort explanations; it queries the model in regions of the input space in which it could be poorly trained.

Bandwidth cancellation. We mentioned earlier that the choice of weights π in Eq. (2.5) is critical because it can change the results. The default is

$$\pi_i = \exp\left(\frac{-\|\mathbb{1} - z_i\|^2}{2\nu^2}\right)\,,$$

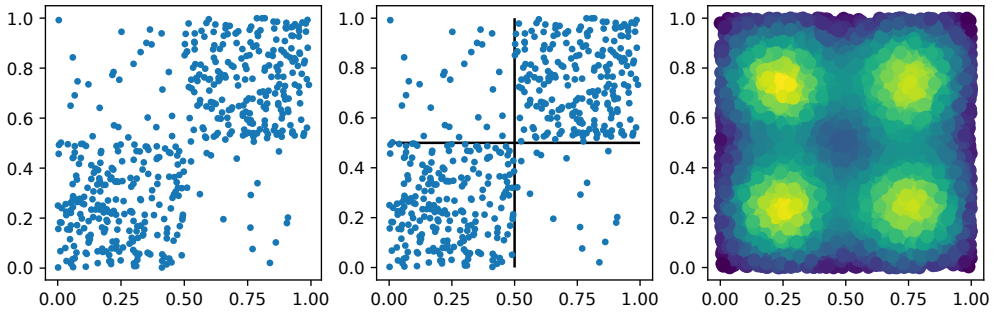

Figure 2.2. LIME plain sampling problem in 2D. LIME will sample within the boxes indistinctly, despite a clear inhomogeneous pattern. The figure shows the $p = 2$ boxes version. *Left panel*: training data, *middle panel*: quantiles on each feature, *right panel*: density of the LIME sampling.

where ν is the bandwidth, free parameter of LIME. For tabular data, the default value is $\nu = \sqrt{0.75d}$, where d is the number of attributes. However, Garreau and von Luxburg [2020] show that the explanations change a lot as a function of this parameter. This change can lead to the phenomenon of "bandwidth cancellation": as the parameter changes, a coefficient can be unexpectedly cancelled; it can also change in magnitude and even sign in counter-intuitive ways.

2.8.3 Specific problems with Shapley values

Kumar et al. [2020] consider the Shapley values to be inadequate in themselves for explainability. Game-theoretic approaches have indeed become popular in Explainable AI thanks to their desirable mathematical properties and their formulation of feature importance. Shapley value is a method for additively attributing value among players of a cooperative game: in this setting, features are the players and the game is the prediction of the model in hand. Several methods have been implemented to approximate Shapley values and they are used to provide feature importance; the paper makes a classification according to the value function involved. SHAP, for instance, belongs to the class of *conditional methods*, as its value function is the conditional expected model output on a data point when only features in S are known:

$$v_{f,x}(x) = \mathbb{E}[f(X)|X_S = x_S] = \mathbb{E}_{X_{\bar{S}}|X_S}[f(x_S, X_{\bar{S}})]\,,$$

where $\bar{S}$ is the set of feature not presents in S. Other methods use ideas from causal inference to simulate intervention on features not in S. These are the *interventional methods* and their value function is defined as

$$v_{f,x}(x) = \mathbb{E}_{\mathcal{D}}[f(x_S, X_{\bar{S}})]\,,$$

where the distribution $\mathcal{D}$ is derived from the product of the marginal distributions of the features in $\bar{S}$. However, the paper shows that Shapley-value-based explanations for feature importance "fail to serve their desired purpose in general." Criticism is presented on two fronts: one of mathematical formalism and one that looks at the explanations from a human-centric position.

Mathematical issues. The different behaviour of these two classes is shown through the *indirect influence* debate: choosing between one class or the other will lead to completely different results. In general, interventional methods induce an *out-of-distribution* problem, as shown by Slack et al. [2020]. On the other hand, conditional methods require additional modeling of features interrelations. For example, it is not clear if two statistically related features should be considered as separate players or not: a proposed solution is to express this relation by means of causal knowledge and attribute all the importance to the "ancestor" feature. The problem is that these methods are very sensitive with respect to the amount of prior knowledge which is infused. In addition, the *additivity constraints* used to define Shapley values is only considered "mathematically convenient" and it is not an innocuous property: Shapley value is conceptually limited for non-additive models. Furthermore, the algorithms used to estimate Shapley values are only an approximation, although they are becoming increasingly efficient (see Covert and Lee [2021] for recent advances).

Human-centric issues. The mathematical formulation of the Shapley values does not guarantee certain desirable properties of the explanations. While one can interpret these values to provide contrastive explanations, they fail to make the user understand how to achieve a desired result (counterfactual explanations): observing that a feature has a large positive influence does not necessarily imply that increasing its value will lead to a better result, especially for non-linear cases. One more point of Kumar et al. [2020] is that surveys show that data scientist and Shapley value users do not really understand what these values represent, so "over-trusting and misusing" Shapley tools.

2.9 Extending LIME for Business Process Automation

Business processes can be very complex as they combine multiple models and rules in different patterns to get a decision. The most common techniques in explainable AI such as LIME and SHAP cannot be applied directly to business automation systems, as their complexity leads to virtually all of the problems discussed above and thus to misleading explanations. As seen in Section 2.8.2, in LIME many problems are due to local neighborhood sampling, which is done by perturbing features independently. Feature independence is never satisfied in practice, and in a business process this is certainly not the case, as there are features that are calculated or estimated based on others (*e.g.*, through machine learning models). The feature independence assumption causes LIME to generate samples which are *out of distribution* (see Figure 2.2): improbable or even impossible and thus lead to meaningless explanations.

For instance, let us consider the simplified loan approval scenario presented by Upadhyay et al. [2021], where the features CREDIT and RISK have a correlation coefficient $\rho = -0.9$. LIME would sample the two features independently, generating applicants with CREDIT and RISK both very high (or very low), thus querying the model in spaces where it is not (or not enough) trained. Upadhyay et al. [2021] propose to extend LIME sampling by taking into account features correlations. The approach of this extension is to formulate the model to be explained in such a way that it is highly accurate on samples that conform to the real

distribution $\mathcal{D}$ and random on samples that do not conform to $\mathcal{D}$. Let x be an instance of data and y be the output of the model on it. In a two variables case the formulation of the extended model M is therefore

$$M(x) = \begin{cases} y & \text{if } p(x) \geq 0.01\,, \\ \mathcal{B}(0.5) & \text{if } p(x) < 0.01\,, \end{cases} \tag{2.10}$$

where $\mathcal{B}(0.5)$ is the Bernoulli distribution with parameter 0.5 and p is the density function of the original distribution $\mathcal{D}$. By locally approximating the extended M model, LIME can produce more meaningful and accurate results on real data.

This process-aware approach has been shown to provide better results than the standard LIME and, because it achieves fixed accuracy with less samples, it is also more computationally efficient. On the other hand, it requires the full relationship between the features to be known. In addition, it should be considered that when the link between variables is given by a machine learning model, the correlation is unlikely to capture sufficient information.

Chapter 3

Rule-based decision systems

Artificial Intelligence spread into corporate environments in the early 1980s due to the commercial success of expert systems. These are domain-specific decision systems designed to emulate the ability of a human expert, mostly through IF-THEN rules [Russell and Norvig, 2002]. The branch of artificial intelligence based on human-readable representations of *symbolic* problems is the Symbolic AI. Decision systems then evolved into business rule management systems: software programs used to manage and monitor (even complex) business operations. Business rules give logic to these systems, allowing the business policy for a given task to be infused into an automated system.

Symbolic AI was the dominant paradigm of artificial intelligence research from the mid-1950s until the late 1980s, when it was eventually supplanted by Statistical Learning. The data-hungry framework of Machine Learning seems to provide practical solutions to business problems and indeed the literature is rich of business applications, with promising models for predictions, decision-making, *etc.* However, not a lot of machine-learning innovation are adopted in business context.

Business automation processes are often built by combining models and rules in sequence or in complex graphical structures. If we apply interpretability methods to the entire system, using all of the variables involved in the decision-making policy (policy variables), we completely neglect this process and can obtain meaningless results. A "process-aware" method could keep into account the process structure and overcome these issues, by augmenting explanation methods with causal relationships, by knowing the admissible values for each attribute, *etc.*

The approach of applying interpretability methods to policy variables is defined by Jan et al. [2020] as *Level 1*. *Level 2* brings process-awareness, including knowledge of the business process, business rules, models and all features. The simplest case of a decision system in our context is a set of rules that apply to certain known attributes. To interpret at *Level 1* a specific decision made by such a system, we can look at the set of rules that are satisfied by the instance under consideration. We call this set the *trace* and in the case of a decision tree it corresponds to the path followed by a point from the first node to the leaf. An example of trace is reported in Figure 3.1.

To have trust in business processes based on artificial intelligence models, companies need

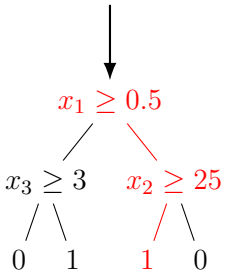

Figure 3.1. A small decision tree for binary classification with three attributes. The trace in the decision tree for point $\xi = (0.8, 12, 4)^\top$, predicted as 1, is shown in red. It represents the path that the point follows within the tree.

process-aware explanations. In fact, decision-making systems can be extremely diverse, in purpose, structure and complexity. Below we show two studies applied to two different frameworks of decision-making systems.

3.1 A study of explanation generation in a rule-based system

Decision theory and artificial intelligence have common roots and, more and more, a lot of intersections. However, rule-based systems are a field of study in itself: the thesis from El Mernissi [2017] does not deal with machine learning algorithms or Explainable AI methods but it focuses on IBM Operational Decision Manager, market leader in the sector. As these systems are needed to be used real-time, computational constraints must be kept into account, for the system itself and also for the explanatory method.

"Business Rule Management System" (BRMS) refers to series of tools to formulate business policies in automated processes. Decision making systems must provide good performances, but also the need for explainability is growing: businesses want transparent and flexible systems that can efficiently and reliably satisfy their needs.

A rule typically takes form a IF-THEN expression. However, a rule-based system is typically made up of several of these expression, which makes it difficult to be interpretable. Different ways to classify explanation exist, according to the temporal context (when it is needed: *during* or *after* the decision process), the type of the question (*how*, *why*, *why not*, *what*, *etc.*), the content type, and so on. El Mernissi [2017] presents a method which try to exploit causal rules that may actually exists among the features in the industrial context of the application. This approach is not applicable in our setting, because it is not designed to take machine learning models into account. However, some insights may be useful and be extended to our case.

IBM BRMS. In IBM Operational Decision Manager, decisions are entirely taken automatically. The system stores the trace of this process in a database and then it should show

these information in a representative way. There are two main problems in this context:

- the process involves a large number of decisions, so we need to filter the trace and take only the most useful (in some sense) information;
- explanations have to be provided real-time (or very quickly), so there are strong constraints on the additional computational effort needed.

What is more, the explanation system must be as general as possible, avoiding project-specific requirements, for maintainability purposes. But, at the same time, it should take into account the type of user and its needs. Explanations here are so intended to be *feedbacks*, as they came after the whole decision process, *justification* and *terminological knowledge*, as we need causality and information about concepts and methods used. They need to answer to both qualitative and quantitative questions and to different users, which can be business, engineers, end-users.

Causal model for rule-based systems. Business rules are statements in the form IF <PREMISES> THEN <CONSEQUENT>, where *premises* is a "disjunction of conjunction of conditions," and the *consequent* is a set of actions:

$$\text{IF } (c_1 \text{ AND } \ldots \text{ AND } c_m) \text{ OR } \ldots \text{ OR } (c_1 \text{ AND } \ldots \text{ AND } c_m) \text{ THEN } (a_1, \ldots, a_n)\,.$$

Once rules are defined, the rule engine execute them, according to the algorithm used, the priorities among rules and the ruleflow. Then, we need to construct the trace of the process, providing enough information to show causal dependencies: we must know which rules have been triggered, which parameters were used in that rules, what were their consequences and at which order these rules have been triggered.

Extract causal relations can help reducing the trace and providing useful understandable information to the user. This causal structure guarantees to keep all the information needed to provide a complete causal model. After that, we can find the "minimal decision trace" by removing from the original trace all the nodes that have no path to a rule modifying an output parameter.

3.2 Explaining Multi-Criteria Decision Aiding Models with an Extended Shapley Value

Decision-making systems can be affected by the presence of conflicting criteria. Multi-Criteria Decision Aiding (MCDA) is a sub-field which attempts to propose solution for this kind of issues, mainly based on hierarchical rules priority. Labreuche and Fossier [2018] studied this context as summarized here. In order to trust the system, a decision-maker has three main needs:

- *Interpretability*: a general knowledge of the model behaviour, *e.g.*: feature importance;
- *Explicability*: deeper analysis to understand the relationship between input and output;

- *Sensibility Analysis*: ask what changes in the data changes the results the most.

How these three requirements are met characterizes the system's explainability, which is crucial for users trust. However, most users do not look for complete and formal explanations, like a mathematical proof. Instead, they need quick and simple explanations, even if they are incomplete. For this purpose, an influence index is designed as an extension of the Shapley values [Shapley, 1953], that we discussed in Section 2.6, to measure features impact.

Model and Notations. A MCDA model represents a Decision Maker (DM) over a set of criteria $N = \{1, \dots, n\}$, each associated to an attribute X_i, where $i \in N$. We define $X = X_1 \times \cdots \times X_n$ the set of alternatives. The preference of the DM over X are represented

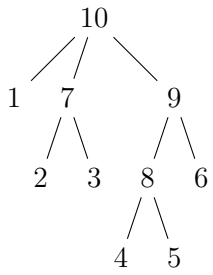

Figure 3.2. Example of MCDA from Labreuche and Fossier [2018]. Each node represents a decision criteria.

by a utility function $U : X \to \mathbb{R}$, decomposable in

$$U(x) = H(u_1(x_1), \dots, u_n(x_n)) \quad \forall x \in X. \tag{3.1}$$

To reduce the number of criteria, which can be large, it is common to hierarchically decompose the aggregation function H. Let M_T be the set of nodes of the tree T, $Ch_T(\ell)$ the set of the children of node ℓ, $N_T \subseteq M_T$ the set of leaves of T, $s_T \in M_T$ the root node of T, τ_T the set of subtrees having the same root node, $T_{[j]}$ the particular subtree when a node $j \in M_T$ becomes a leaf.

For instance, in the example of Figure 3.2, $M_T = \{1, \cdots, 10\}$, $Ch_T(8) = \{4,5\}$, $N_T = \{1,2,3,4,5,6\}$, $s_T = \{10\}$ and so on.

We assume that the leaves of T define exactly the criteria: $N_T = N$. Two cases are considered: *flat organizations*, where $M_T = N \cup s_t$ (*i.e.*, no feature interactions) and *coalition structures*, where leaves are at depth 2.

The definition of Shapley values given by Eq. (2.3) in Game Theory has four useful property: additivity, null player, symmetry, efficiency.

Axiomatic Properties. The aim is to identify the attributes carrying the main contributes in order to reduce the number of criteria. We need an index $I_i(x, y, T, U) \in \mathbb{R}$, measuring the influence of attribute i in the difference of scores $U(y) - U(x)$. The axiomatic properties to be satisfied are:

1. *Restricted Values*: $I(x, y, T', U)$ depends only on $\{U(y_S, x_{-S}), S \subseteq N_{T'}\}$. We measure the influence of attributes in x and y by analyzing the consequence of going from x_i to y_i, for each attribute i, keeping all the others as fixed.

2. *Null Attribute*: if an attribute $i \in N_{T'}$ is null for x, y, U, then $I_i(x, y, T', U) = 0$. Attribute i is null when $U(y_{S\cup\{i\}}, x_{(-S\cup\{i\})}) = U(y_S, X_{-S})$, meaning that changing value x_i to y_i has no impact on the utility.

3. *Restricted Equal Treatment*: if $k, l \in N_{T'}$ have the same parent and $k \sim l$, then $I_k(x, y, T', U) = I_l(x, y, T', U)$. $k \sim l$ when $U(y_{S\cup\{k\}}, x_{(-S\cup\{k\})}) = U(y_{S\cup\{l\}}, x_{(-S\cup\{l\})})$, meaning that attribute k is as desirable as l.

4. *Additivity*: $I(x, y, T', U + U') = I(x, y, T', U) + I(x, y, T', U')$, so that the influence for a sum of two utilities is equal to the sum of the influences for the two utilities.

5. *Generalized Efficiency*: $I_{s_T}(x, y, T', U) = U(y) - U(x)$ and

$$\forall l \in M_{T'} \backslash N_{T'}, \quad \sum_{i \in Ch_{T'}(l)} I_i(x, y, T', U) = I_l(x, y, T', U).$$

6. *Consistency with Restricred Tree*: let $i, j \in M_{T'}$ such that $i \notin Desc_{T'}(j)$ and $j \notin Desc_{T'}(i)$, then $I_i(x, y, T', U) = I_i(x^{T'_{[j]}}, y^{T'_{[j]}}, T'_{[j]}, U_{T'_{[j]}})$. Note that this condition is only valid when i and j do belong to the same part of the tree.

Expression of the Influence Index. The influence index is based on Shapley value. The difficulty to measure the contribution of each attribute in $U(y) - U(x)$ is that x and y are in general different on all the attributes. So, we change one attribute at a time according to an ordering π and measure the influence:

$$\delta_\pi(i) := U(y_{S_\pi(i)}, x_{-S_\pi(i)}) - U(y_{S_\pi(i)\backslash\{i\}}, x_{-S_\pi(i)\backslash\{i\}}), \tag{3.2}$$

where $S_\pi(\pi(k)) = \{\pi(1), \ldots, \pi(k)\}$. The influence index of i is the average value of $\delta_\pi(i)$ on all orderings π: this is exactly the Shapley value ϕ_i of $v : 2^N \to \mathbb{R}$, with

$$v(S) = U(y_S, x_{-S}) - U(x)\,.$$

Note that not all orderings are possible and the admissible ones have to be well defined. The problem is that the computational cost of this calculation is exponential with the number on nodes.

This approach is indeed suitable for a particular structure of decision-making system. However, it requires a priori knowledge of how each node works and how all values are combined. In our case, the decision rules do not necessarily involve all variables directly. Above all, we seek a solution to the case where models are black-boxes.

The interpretability of hybrid decision systems seems to be understudied. To the best of our knowledge, there is no solution to our problems in the literature.

Chapter 4

Machine Learning in business context

Deep learning is now the-state-of-the-art in artificial intelligence for specific and complicated tasks: neural networks achieve outstanding performance in various areas [Schmidhuber, 2015]. Nevertheless, there are some caveats. First, it requires a large amount of data. For example, the success of neural networks in computer vision is largely due to the availability of large datasets such as ImageNet, which now has more than 14 million hand-annotated images. Most companies, especially small and medium-sized ones, don't have that amount of data at their disposal, making it difficult to rely on neural networks for day-to-day tasks. We note that this might change: Kadra et al. [2021] show that by applying an optimal combination of regularization techniques, a small neural network can compete with state-of-the-art machine learning models on tabular data. Second, even if there is enough data, a great computing capacity is required to process it. Neural networks are very expensive in the training phase and, consequently, so are the validation of parameters and the choice of the right architecture. This means higher costs for hardware equipment and more effort on the software. Third, this usually comes with longer development cycles than traditional models. In companies, time can be crucial and translates into higher costs, or even least opportunities.

For these reasons, neural networks are not always suitable for the needs of companies, especially small and medium-sized ones. In their daily work they often have little tabular data available and need predictions in several small problems instead of a large system, where deep learning would have been more appropriate. Simple, established models, often tree-based, are preferred. As simple as they are, decision trees are still very popular due to their good properties (they are fast, non-parametric, no demanding preprocessing), including interpretability. We will discuss this in Section 4.1. Tree ensembles models, such as random forest, are used particularly often in these contexts: they are a way to capitalize on the benefits of decision trees by reducing their tendency to overfit. Boosting is perhaps the approach that has contributed most to their popularity. For instance, XGBoost is a scalable machine learning system for tree boosting, widely recognized in a number of machine learning and data mining challenges [Chen and Guestrin, 2016]. The system is available as an open

source package and is popularly used in business settings [Brownlee, 2019]. Brief overviews of Boosting and XGBoost can be found in Section 4.2 and Section 4.3. Other times, machine learning models are modified to better fit the specific needs of the application area. Many models are built ad-hoc to ensure interpretability: this is the case of Sobol-MDA and SIRUS, which we will discuss in Section 4.4 and Section 4.5.

4.1 Explaining the result of a Decision Tree

Decision trees [Breiman et al., 1983] are widely used in decision-making, providing good results in different domains. In order to evaluate the result of a tree, users usually look at the trace, that is the path followed in the tree, with the list of the tests passed by the case. However, at the end of the process it is not that easy to understand the relevance of a result: a small change in one parameter could completely change the result, while (relatively) strong changes in several values could leave the result, and even the trace, intact. Alvarez [2004] propose the following for explaining the result of a decision tree to the end-user.

The point is that the trace does not take into account the sensitivity of the decision tree to small changes. The paper presents a geometric approach, studying the position of the instance point in the partition generated by the decision tree in the feature space. In particular, this analysis focuses on the distance between the point and the decision surface, defined as the union of the boundaries of the different areas corresponding to the different classes. This approach allows to evaluate decision tree tests that are more sensitive to small changes, and these tests can also be ordered according to their sensitivity.

In the case of linear decision trees, the decision surface consists in pieces of hyperplanes, so binary linear decision tree tests consist in computing the distance h of a case point P to a hyperplane H. Tests are in the form $h(P, H) > 0$, so the area classified by each leaf is the intersection of halfspaces and the decision surface Γ is piecewise affine, so that at each point $y \in \Gamma$, the decisions surface is defined by a list of hyperplanes L. Assuming we work with a complete metric space E, we consider the projection of each point $x \in E$ on the decision surface $p(x) \in \Gamma$. In this point, the decision surface is defined by the unique list of hyperplanes $L(x) = (H_i)_{i \in I}$, such that $p(x) \in \cap_{i \in I} H_i$. The sensitive test of a point x is defined as the tests $h(x, H)$ verified by x, for each hyperplane $H \in L(x)$. This definition satisfies good properties of uniqueness, robustness, ordering relation [Alvarez, 2004] and it is even more robust in the case of decision trees defined by hyperplanes with axis-parallel normal vector (the NDT case). In the latter, the number of sensitive tests for a point is at most equal to the number of attributes d and each test is on exactly one attribute, so it is very easy to understand.

The list of tests grows as the dimension d, and if this is large, it difficult to visualize all of them. We need to prune the list in order to provide understandable information to the user: this is very straightforward since sensitive tests can be easily ordered according to the distances: Let $T(H_1)$ and $T(H_2)$ be two sensitive tests of point x. We say that $T(H_1)$ is more sensitive than $T(H_2)$ if and only if the margin of $T(H_1)$ is greater than the margin of $T(H2)$ ($T(H_1) \succeq T(H_2) \iff d(x, H_1) \geq d(x, H_2)$). By choosing a metric, the distance to the decision boundary can be easily shown for each result as a measure of its relevance and

we can extract and sort the list of attributes actually relevant for the tests.

Alvarez and Martin [2009] extended this idea in classification problem, being either machine learning or deterministic, based on the fact that the "decision boundary always exists, even implicitly," with the restriction to the numerical state space, since a metric must be defined. Let us consider the decision $c(x)$ made for a point x of the input space E. The method defines the robustness s of $c(x)$ as the distance d such that each point in $B(x, d) = \{\xi : \|x - \xi\| \leq d\}$ (*i.e.*, the closed ball centered at x with radius d) belong to the same class of x:

$$s(x) = \max\{d \geq 0,\ \forall y \in B(x, d),\ c(y) = c(x)\}\,.$$

We can also define a sensitive move as the smallest vector v such that the the robustness at $x + v$ is zero:

$$m(x) \in \arg\min_{\delta \in E}\{\|\delta\|,\ s(x + \delta) = 0\}\,.$$

This method is not directly applicable to our case either, because we have nodes composed of machine learning models. However, we will use ideas from this approach for partial explanations about the decision tree generated by the rules component.

4.2 Boosting

The idea of boosting approach [Schapire, 1990] in machine learning for classification problems is to create highly accurate predictions by combining several low-accurate rules. A wide and strong theory has been developed around boosting and some implementations have found a great success in practice, too. One example of these is AdaBoost, presented in the seminal work of Freund and Schapire [1997]. *Wisdom of the crowds* is the basis concept behind boosting, which takes a large amount of its foundations in the game theory field.

Boosting assumes the availability of a *base* or *weak* classifier that can be called repeatedly. This classification algorithm has to be at least better than making random predictions (*weak learning assumption*). A boosting algorithm learns by repeatedly calling the base learning algorithm, but if this always works with the same data and in the same way, we cannot expect anything interesting. The key idea is to choose training sets for the base classifier in such a way to infer something new each time it is called.

As an example, let us consider AdaBoost (Algorithm 1) from Schapire and Freund [2013], assuming a two-classes $(-1, +1)$ classifier. At each round t, it uses the distribution D_t over data, so that the weight assigned to example i at round t is $D_t(i)$. At $t = 0$, all weights are equals, then, on each round, the weights of incorrectly classified examples are increased so that the importance of harder examples is bigger for the classifier. The weighted error $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$ represents the chance for the classifier of h_t misclassifying a random example selected according to D_t, which is the sum of the weights of the misclassified examples. For what we said about the "weak learner," we do not expect the error ε_t to be zero, neither very close to it. Considering the two-class example, a random classifier has an error probability of $\frac{1}{2}$, so we just need the assumption that the error of a weak classifier is bounded away from $\frac{1}{2}$.

Algorithm 1: AdaBoost algorithm [Schapire and Freund, 2013]

Input: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$;
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$;
for $t = 1, \ldots, T$ **do**
  Train weak learner using distribution D_t;
  Get weak hypothesis $h_t : \mathcal{X} \to \{-1,1\}$;
  Aim: select h_t to minimalize the weighted error: $\varepsilon_t = \mathbb{P}_i[h_t(x_i) \neq y_i]$;
  Choose $\alpha_t = \frac{1}{2}\log(\frac{1-\varepsilon_t}{\varepsilon_t})$;
  Update, **for** $i = 1, \ldots, m$ **do**

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \begin{cases} e^{-\alpha_t}, & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t}, & \text{if } h_t(x_i) \neq y_i \end{cases} = \frac{D_t(i) \cdot \exp(-\alpha_t y_i h_t(x_i))}{Z_t};$$

    where Z_t is a normalization factor so that D_{t+1} is a distribution.
  end
end
Output: final hypothesis:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

AdaBoost stand for *"Adaptive Boosting algorithm,"* because it adjusts and adapts to the errors ε_t, in the sense that weak learners are tweaked in favor of those instances misclassified by previous classifiers. It turns out to have good performances in minimizing both the training data and the generalization error and it has good algorithmic characteristics, too: it has no parameters to tune, except for the number of rounds T. The consistency of AdaBoost has been intensively analyzed by Bartlett and Traskin [2007], providing an optimal range for the number of iterations.

XGBoost is an evolution of AdaBoost, and is much more used in practice thanks to its popular open-source software library.

4.3 XGBoost

XGBoost is a scalable gradient boosting system presented by Chen and Guestrin [2016]. It stands for "eXtreme Gradient Boosting." It is a scalable, fast and well-performing tree boosting system, widely used in the data science community. It is designed for optimizing computational resources as it performs different optimization improvements which make it better than other boosting technique.

Gradient boosting involves a loss function, a weak learner, and a model to add new weak learners to reduce the loss function. The procedure builds iteratively a sequence of approximations $f_t : \mathbb{R}^d \to \mathbb{R}, \quad t = 0,1,\ldots,T$ in a greedy fashion. The final prediction for example x_i is then computed as $\hat{y}_i = \sum_{t=0}^{T} f_t(x_i)$. The gradient descent procedure is used to update parameters in the direction of negative gradient. The loss function to minimize at

step t is in the form

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} \left\{ \ell(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) \right\} + \Omega(f_t)\,, \tag{4.1}$$

including a classification loss ℓ and a regularization function Ω that penalizes the complexity of the model. XGBoost calculates the value of the loss function for $f_t(x_i)$ using second-order Taylor approximations, that is

$$\ell(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) \approx \ell(y_i, \hat{y}_i^{(t-1)}) + \frac{\partial \ell(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} f_t(x_i) + \frac{1}{2} \frac{\partial^2 \ell(y_i, \hat{y}_i^{(t-1)})}{\partial (\hat{y}_i^{(t-1)})^2} f_t^2(x_i)\,. \tag{4.2}$$

Renaming the first and second derivatives respectively as g_i and h_i, we can reformulate:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left\{ \ell(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right\} + \Omega(f_t)\,. \tag{4.3}$$

Note that the term $\ell(y_i, \hat{y}^{(t-1)})$ does not depend by f_t, is a constant term independent of any choice of function. We can therefore define a new loss function

$$\tilde{\mathcal{L}}^{(t)} := \sum_{i=1}^{n} \left\{ g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right\} + \Omega(f_t)\,, \tag{4.4}$$

and pick $f_t(x_i) \in \arg\min_{f_t} \tilde{\mathcal{L}}^{(t)}$.

The main types of improvements [Brownlee, 2019] to apply to avoid overfitting and increase performances are

- Tree Constraints: number of estimators, maximum depth, number of leaves, number of observation per split and minimum improvement to loss.
- Shrinkage (Weighted updates): predictions are added sequentially and each contribution is weighted to slow down the learning, so that more trees are required. This allows the contribution of each weak learner to be small and allows the model to continue learning. However, the smaller the learning rate, the more the trees: we have to find a trade-off.
- Random Sampling: stochastic gradient boosting makes the trees less correlated by sub-sampling either the rows and the columns of the training set.
- Penalized learning: standard regularization methods, like L_1 or L_2, are used to penalize the complexity of the model.

Some of the additional innovation included in XGBoost are:

- "Sparse-aware" implementation: making it able to deal with missing values, zero entries or inconsistent instances, *e.g.* by letting trees to take a default direction while encountering them.

- "A block structure" for parallel learning: the system is designed in such a way to optimize the memory using and to facilitate parallel computing.
- Continued Training so that further boosting of an already fitted model on new data is allowed.

XGBoost is a complex algorithm that combines decision trees and it provides several measures of feature importance based on the splitting criteria of the underlying trees. However, these measures are not fully satisfactory: for example, they all show importance in absolute terms, without giving information on the sign of the contribution of each. Depending on the application domain, the requirements of interpretability are different. As mentioned, ad-hoc interpretable methods are often constructed. We now see two tree-based examples of these.

4.4 Sobol-MDA

The most popular variable importance measure for random forest is the Mean Decrease Accuracy (MDA): it measures the decrease of accuracy in the prediction when the values of an input variable are permuted to break its relation to the output. However, several implementations of this idea exists and Bénard et al. [2021b] prove that the different implementations actually have different limit behaviours when $n \to \infty$ and are therefore estimating different things.

They specify two main final aims of variable importance:

- find a small number of variables with a maximized accuracy *(objective 1)*;
- rank all influential variables to focus on further exploration with domain experts *(objective 2)*;

and they state that different methods should be used according to the case. For instance, if two variables are correlated, one must be disregarded in the first case, while it should result as an influential variable in the second. However, it is proved that the most popular implemented versions of MDA are not well suited for the first case when there is interaction and dependence among the features.

The paper also identifies the unnormalized Total Sobol Index [Sobol, 1993] as the desired theoretical counterpart of MDA. The Total Sobol Index of variable $X^{(j)}$ for the model m, with $Y = m(X)$ is defined as

$$ST^{(j)} = \frac{\mathbb{E}[\mathrm{Var}(X)|X^{(-j)}]}{\mathrm{Var}(Y)}\,. \tag{4.5}$$

This quantity gives the proportion of the explained output variance lost when variable $X^{(j)}$ is removed from the model.

In theory, this would require re-training a new model each time a variable is removed. This is too onerous and in fact unfeasible in practice. Removal is usually simulated by permuting the values of the variable within the corresponding column in the entry table. This breaks the relationship of the variable with both the output and the other features.

However, this procedure adds a component due to permutation to the asymptotic result, thus fooling the MDA. To solve this, Bénard et al. [2021b] proposes a method to calculate the Solol-MDA that approximates the Total Sobol Index without relying on permutation.

This is achieved by ignoring splits on the variable j under consideration, making each tree predict the quantity $\mathbb{E}[m(X)|X^{(-j)}]$ to finally recover the unnormalized Total Sobol index $\mathbb{E}[\mathbb{V}(m(X)|X^{(-j)})]$. Geometrically, considering the partition generated by a tree on the input space, this is analogous to projecting the point X into the subspace span by $X^{(-j)}$.

As mentioned above, this method works in the case of measuring the importance of variables (*objective 1*). For our purpose, *objective 2*, the paper mentions Shapley values as a good theoretical measure. In fact, we follow this way.

4.5 SIRUS - Stable and Interpretable RUle Set

SIRUS (Stable and Interpretable RUle Set) [Bénard et al., 2021a] is another tree-based classification model based on random forests, improved to be more interpretable. Like neural networks, random forests can be considered black-boxes, since due to the high number of operations involved in their mechanism, it is not easy to explain their prediction.

As discussed above, in machine learning there is a clear trade-off between accuracy and explainability. SIRUS achieves the same accuracy as random forests, but provides explanations that are very stable with respect to perturbations in the form of simple rule IF CONDITION ON x THEN RESPONSE, ELSE DEFAULT RESPONSE.

These rules are extracted and pruned according to the probability that a certain node is contained in a random tree. The higher the probability, the more robust and therefore important the rule. For a binary classifier this is mainly achieved by the following steps:

1. **Rule generation**: SIRUS relies on the original implementation of random forests [Breiman, 2001]: each tree in the forest is grown with an algorithm that partition the input space. The modification is that SIRUS select the splits among the empirical q-quantiles (by default, $q = 10$). This slight modification is central to the method: all trees in the forest will choose for each coordinate the best split among the same q admissible ones. In the end, for each tree, a rule is associated with each path.

2. **Rule selection**: the most important rules are more shared. The proportion of trees in the forest that share the same rule is an estimate of the probability p that that rule belongs to a tree. The rules that have a probability greater than a certain threshold p_0 (the only parameter of the model) are then selected.

3. **Rule post-treatment**: some of the selected rules are redundant: the paths associated with them overlap. If two rules are linearly dependent, the one with the higher frequency is removed.

4. **Rule aggregation**: the set of rules extracted so far defines a rule classification model, which associates 1 with a certain point x if the probability of the rules exceeds a certain threshold, 0 otherwise.

This model has many desirable properties. In particular its stability is extensively analyzed by Bénard et al. [2021a] and the accuracy of its predictions is shown in a wide range of classical datasets.

The SIRUS approach is to build an intrinsically interpretable model that provides a few simple rules and also achieves good accuracies. This method therefore operates in contexts other than our own, but the ideas used for specific tasks are interesting and applicable.

None of the methods presented here are satisfactory in our setting. We therefore move to the description of a new method, adapted to business needs and built ad-hoc for this setting.

Chapter 5

SMACE – Semi-Model-Agnostic Contextual Explainer

5.1 Introduction

In our framework, there are a few (typically two or three) machine learning models computing quantities based on input data, which are then agglomerated by means of decision rules. An example is shown in Figure 5.1. These rules may refer both to the outputs of the machine learning models and to the input values. In this configuration, there is knowledge that we can exploit to make our explanations process-aware, as discussed at the beginning of Chapter 3. We know which variables are involved in the decision policy and we know its rules. It is worth exploiting this information instead of treating the whole system as a black-box and being completely model-agnostic. In fact the direct application of model-agnostic methods produces poor results, as we show in Chapter 6.

In this chapter, we present SMACE - *Semi-Model-Agnostic Contextual Explainer*. It provides local post-hoc explanations based on feature importance. We built this method to meet the challenges presented in the previous chapters for such a decision system that combines any type of machine learning model. Specifically, SMACE is a method for feature importance that provides two levels of explanation, for the different users involved in the decision-making process. The first level, which is useful for the business user, must provide a ranking of importance for all the variables used, whether they are input attributes or values calculated in-house. This is useful, for example, to the sales representative, who has access to and knowledge of company policies. By interpreting the process, the business user can explain, modify, override or validate the specific decision. The second level is necessary for the end customer. She does not have access to the internal policy rules, nor to the way in which decision-making processes are managed. It therefore requires explanations based solely on information that she is aware of, *i.e.*, input features, such as her personal details or service usage values.

In the following example we present a simple but realistic use case, typical of a business

decision problem: business management is often a question of balancing risk and expected return.

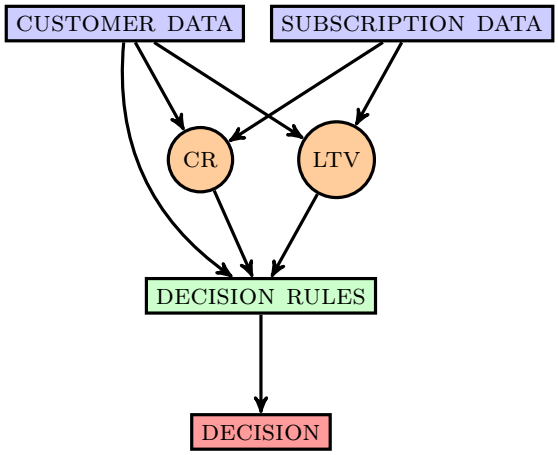

Figure 5.1. Retention offer use-case. Structure of a decision system with two data sources (in blue) and two machine learning models (in orange). The system takes as input the customer's personal data (CUSTOMER DATA) and their subscription data (SUBSCRIPTION DATA). The models CR and LTV predict the churn risk and the lifetime value, respectively. The arrows indicate the passage of information. Both models take both data sources as input, but while CUSTOMER DATA is also used directly in DECISION RULES, SUBSCRIPTION DATA is not. This means that the information contained in the second data source is used exclusively to calculate new values on which decision rules are applied.

Example 5.1.1 ***Retention offer***
A mobile phone company 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 retaining the customer. The decision process relies on different components:

- *Input features, describing:*
  - *the customer,* i.e., *age, income,* etc.*;*
  - *the current subscription,* i.e., *the number of monthly minutes and text messages used,* etc.*;*
- *Two Machine Learning models trained on past customers data:*
  - *XGBoost Classifier, predicting the churn risk* CR, i.e., *the likelihood that the customer will drop out for another company;*
  - *XGBoost Regressor, predicting the life time value* LTV *of the customer,* i.e., *the expected revenue generated by the customer if it is maintained.*

There are three types of retention offers: special, super, and basic. If a user is not eligible for any of them, he/she gets no offer.

These components, shown in Figure 5.1, are finally aggregated by means of decision rules:

```
if CR ≥ 0.5 then
  if LTV ≥ 300 then
    DECISION ← 1: make super offer.
  else
    DECISION ← 2: make basic offer.
  end if
else
  if LTV ≥ 750 then
    DECISION ← 3: make special offer.
  else
    DECISION ← 4: no offer.
  end if
end if
```

Therefore, depending on the estimates of the propensity to churn and the lifetime value, the company decides whether to make an offer to the customer and, if the case, the type of the offer.

In this example, it is fundamental for the sales representative to know, even roughly, *why* this prediction was made. It is also crucial to have explanations that refer to the input data, so that the customer can understand them. Indeed, it may not be informative to provide explanation to the user by showing her some internal values predicted with machine learning and, as mentioned in Section 1.4, it may soon became a legal requirement.

The final agglomeration by means of rules allows the company to express the business policy and they are necessary because policies may quickly change over time. In the example above, according to the economic performances of the company in the last quarter, it may require higher or lower level of *propensity to churn* for a customer to receive a retention offer. What is more, while in some cases these decision rules represent just soft preferences, they are often hard rules, potentially due to regulation (for instance, AGE $\geq$ 18 may be required by law to get a certain offer). Therefore, they cannot be easily incorporated into machine learning models; it is challenging for these to adhere strictly to deterministic policies and rules.

SMACE is a new explanatory method that combines a geometric approach with existing interpretability solutions to generate feature importance based explanations. In the rest of the chapter we present SMACE step-by-step, taking Example 5.1.1 further for the whole Section 5.2.

5.2 Definition of the explanatory method

5.2.1 Assumptions

Before describing SMACE, we present the main hypotheses, underlying their limitations. The method is based on three assumptions, which limit possible applications but are necessary

for operation. Ideas for solving in future work some of these limitations are discussed in Chapter 7.

A1: Decision rules only refer to numerical values. This restricts the area of applicability of SMACE significantly; categorical values may be very important. For instance, in the Example 5.1.1, a decision policy might take into account the payment method of a customer in order to apply a certain offer. However, this assumption allows us to take a simple geometric approach for the explainability of the decision tree. Since it is generally not straightforward to assign a metric to categorical variables, we restrict to decision policies that do not use them. 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 variable, without keeping into account variable interactions. For instance, this assumption excludes rules like IF CR $\geq$ LTV. Geometrically, this implies decision trees with the splits parallel to the axes, similar to CART trees [Breiman et al., 1983].

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. This case would induce a graph structure, to which SMACE would be theoretically extensible, but would make the explanations less stable. Note that this is a very reasonable assumption that covers most real-world applications that we know of.

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.

5.2.2 Notation

Let $x \in \mathbb{R}^D$ be input data, where D is the number of input features and let $f^{(1)}, \ldots, f^{(N)}$ (N is typically 2 or 3) be the machine learning models. We will refer to their outputs $f^{(1)}(x), \ldots, f^{(N)}(x)$ as the *running features*, whose values we also denote $y^{(1)}, \ldots, y^{(N)}$ when there is no ambiguity.

The union of numerical input features and running features $(x_1, \ldots, x_D, y^{(1)}, \ldots, y^{(N)})$ defines the set of the $D + N$ *variables* to which the decision rules are applied.

A decision rule R is formally defined by a set of conditions, and each condition is a triple *(variable, operator, cut-off)*, where operator may be $>, <, \geq$ *or* $\leq$ and *cut-off* is the threshold to be applied. For example, if we apply to CHURN RISK (CR) the condition IF CR ≥ 0.5, this is defined by (CR, $\geq$, 0.5).

5.2.3 Overview

The intuition behind this approach is that we need to agglomerate the explanations of individual models with a mechanism similar to that in which the model themselves are agglomerated. Making the above assumptions, we can see a decision-making system as a linear decision tree

where some of the nodes refer to machine learning models. We therefore combine an ad-hoc method for the interpretability of the (intrinsically interpretable) decision tree, with a model-agnostic method for the machine learning components. To do this, for each case at hand, we first perform two parallel steps:

- **Explain the results of the models**: for each machine learning model $f^{(k)}$, we derive the importance $\hat{\phi}_j^{(k)}$ that each of its input features j has in the prediction. By default, we rely on Shapley values [Shapley, 1953] to allocate these importance values fairly;
- **Explain the rule-based decision**: measure the contribution $r_j^{(1)}$ or $r_k^{(2)}$ of each variable (that is, each input feature j and each model $f^{(k)}$) actively involved in the decision policy. We use a geometric approach extending the work of Alvarez [2004] for the explainability of decision trees.

Then, to get the **Overall explanations**, we combine these partial explanations in order to fairly attribute them to the input features by weighing the importance of these features for a model with the importance of the output of that model in the decision rules. The total contribution of the input feature j to the decision for a given instance as

$$e_j = r_j^{(1)} + \sum_{k=1}^{N} r_k^{(2)} \hat{\phi}_j^{(k)} . \tag{5.1}$$

5.3 Explaining the results of the models

In the process of explaining the decision rules, we considered the input and output values of the models at the same level, because the decision policy does not distinguish between them. However, as discussed above, explanations based on running features may not be informative, as neither the end user nor the business user necessarily knows their meaning or the process that generates them. In parallel, we compute the importance of each variable involved in the business rules, included running features. We need to "project" these importance measures to their input features. To put it differently, we need a way to fairly distribute the importance of the output of a machine learning model among its input values: this is what the Shapley values [Shapley, 1953] do, as discussed in Section 2.6 and Section 3.2.

We rely on the Kernel SHAP implementation, discussed in Section 2.6. So, we get the importance $\phi_j^{(k)}$ (given by Eq. (2.7)) of each input feature j for each machine learning model $f^{(k)}$.

The problem is to compare these importance measures. Two models k and h, in fact, 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 5.1.1, before normalization, we have models computing CHURN RISK and LIFE TIME VALUE. The first value is a probability, so it belong 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 dollars. In general, if $f^{(k)}$ predicts the CHURN RISK and $f^{(h)}$ predicts the LIFE TIME VALUE, for a certain variable j as 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. We use as scale factor the empirical standard deviation $\tilde{\sigma}_k$ of the model output in the dataset of size n:

$$\tilde{\sigma}_k = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i^{(k)} - \bar{y}^{(k)}\right)^2}.$$

Subsequently, we define the contribution of input feature j for model $f^{(k)}$ as

$$\hat{\phi}_j^{(k)} = \begin{cases} \frac{\phi_i^{(k)}}{\tilde{\sigma}_k} & \text{if } \tilde{\sigma}_k \neq 0\,, \\ 0, & \text{otherwise}\,. \end{cases} \tag{5.2}$$

The quantities $\hat{\phi}$ are thus of the same order of magnitude and dimensionless and can be aggregated. Note that the empirical standard deviation $\tilde{\sigma}_k$ is zero if and only if model k predicts the same value for each item i in the dataset. In that case, we would also get $\phi_i^{(k)} = 0 \quad \forall i$, which implies $\frac{\phi_i^{(k)}}{\tilde{\sigma}_k} = \frac{0}{0}$ and we adopt the convention $\frac{0}{0} = 0$.

As shown in Lundberg and Lee [2017], almost all perturbation-based method belong to the same class and they can all be used as well. For instance, the LIME approach is very close to SHAP. On the other hand, some studies are critical of the use of these methods, as discussed in Section 2.8.1 and Section 2.8.3, thus we prefer SHAP as a default.

Anyway, SMACE requires a feature importance measure for all of the input features, but not necessarily relying on SHAP, nor LIME. Any other method of obtaining measures of significance is possible.

5.4 Explaining the rule-based decision

We stated in Chapter 3 that the collection of decision rules used by a decision system can be interpreted as a decision tree classifier. A first, naive approach to explain its decision may be to analyze the trace followed by the point within the tree. However, as discussed in Section 4.1, 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 critical value. In addition, there may be many rules in a decision-making system, and simply listing them all would make it difficult to understand the decision correctly.

We present a method to explain rule-based decisions that provides features importance through a geometric approach inspired by Alvarez [2004] and Alvarez and Martin [2009]. In fact, each decision rule 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. In the case of linear decision trees, the decision surface S is piecewise-affine, consisting in pieces of hyperplanes. At each point $y \in S$, the decision surface consists of a list of hyperplanes, each referring to one variable. By projecting an instance point x onto S, it is thus possible to obtain the list of variables that actively

contribute to its classification. The explainability problem is therefore addressed by studying the decision surface generated by the tree.

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

$$x'_i = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}\,, \quad \forall\, i \in \{1, \dots, D\}\,,$$

$$y'^{(k)} = \frac{y^{(k)} - \min(y^{(k)})}{\max(y^{(k)}) - \min(y^{(k)})}\,, \quad \forall\, k \in \{1, \dots, N\}\,.$$

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

We can now define the distance between two points x and ξ as the Euclidean distance

$$d(x, \xi) = \|x - \xi\|_2\,.$$

The projection $p^S(x)$ of point x onto the subspace S is

$$p^S(x) \in \arg\min_{z \in S} \|x - z\|_2\,.$$

The distance between a point x and a subspace S is

$$d(x, S) = \min_{\xi \in S} \|x - \xi\|_2 = d(x, p^S(x)) = \|x - p^S(x)\|_2\,.$$

We call $S^{(\ell)}$ the decision surface of leaf ℓ and $S_j^{(\ell)}$ its j-th component.

Clearly, if a point x falls into leaf ℓ, we have $p^{(\ell)}(x) = x$: the projection is already in the decision space of ℓ, so its projection is the point itself. Let L be the set of leaves and let us define the *active leaf* and *active boundary* for point x and leaf ℓ as

$$\hat{\nu} = \arg\min_{\nu \in L \setminus \ell} \|p^{(\nu)}(x) - x\|_2\,, \quad \text{and}$$

$$b^{(\ell)}(x) = p^{(\hat{\nu})}(x)\,.$$

For instance, the decision tree and the partition it generates in Example 5.1.1 (after normalization) for two customers A and B are shown in Figure 5.4.

Customer A has CR $= 0.6$ and LTV $= 0.4$, while customer B has CR $= 0.6$ and LTV $= 0.75$ (values are scaled in [0,1] for fair comparison). Both points are classified as 1, so they are going to receive an offer of the same type. They belong to the same leaf and their trace, defined by CR ≥ 0.5 AND LTV ≥ 0.3, is exactly the same. However, the sensitivity of their classification is different and an end-user wold not consider the two decisions in the same way, despite the same outcome. Both A and B are indeed very sensitive to changes in CR (along the horizontal axis), while B is much more robust with the respect to A to changes in LTV (vertical axis).

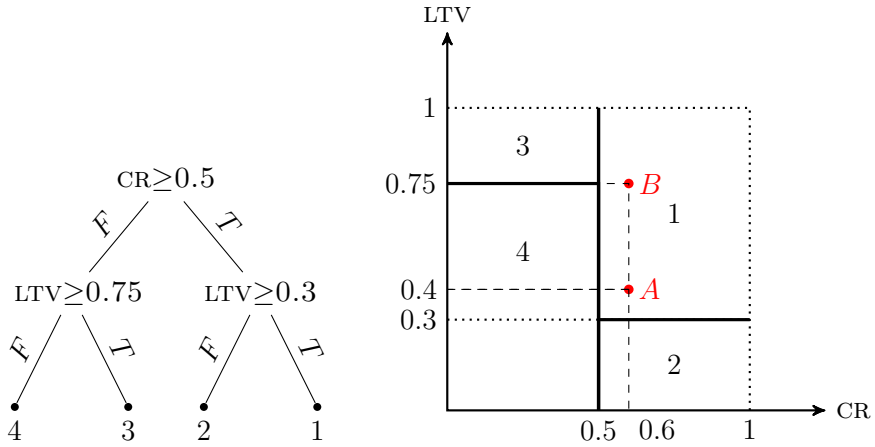

Figure 5.2. A decision tree for the assignment of telephone offers. CR and LTV are the expected churn risk and customer lifetime value, respectively. On the right, the partition generated by the tree in space (CR, LTV). A and B are instance points of two distinct customers eligible for the same offer since they both belong to leaf 1 (CR ≥ 0.5 and LTV ≥ 0.3). However, the sensitivities of their decisions are very different.

Their sensitivity with respect to CR is the same, since $d(A, S^{(1)}_{\text{CR}}) = d(B, S^{(1)}_{\text{CR}}) = 0.6 - 0.5 = 0.1$, they are very different with respect to LTV, indeed $d(A, S^{(1)}_{\text{LTV}}) = 0.4 - 0.3 = 0.1$ and $d(B, S^{(1)}_{\text{LTV}}) = 0.75 - 0.3 = 0.45$. For the classification of point B, we state that the horizontal axis is more important than the vertical one, while with respect to A they have the same importance. What is more, even if the two points have the same sensitivity with respect to CR, moving them to the left by, say, 0.2 would end up in different leaves: A would be classified as 1 and receive no offer, while B would be in class 2.
We compute the contribution $r_j^{(\ell)}$ of a variable j for the classification of point x in leaf ℓ as

$$r_j^{(\ell)}(x) = \begin{cases} -(1 - |x_j - p_j^{(\ell)}(x)|) & \text{if } x_j \notin \ell\,, \\ 1 - |x_j - b_j^{(\ell)}(x)| & \text{if } x_j \in \ell\,, \end{cases} \tag{5.3}$$

where $p_j^{(\ell)}(x)$ and $b_j^{(\ell)}(x)$ are the j-th components of respectively the projection and the active boundary of x on leaf ℓ.

In the case detailed above, this means

$$\begin{cases} r^{(1)}_{\text{CR}}(A) = 1 - |0.6 - 0.5| = 1 - 0.1 = 0.9\,, \\ r^{(1)}_{\text{LTV}}(A) = 1 - |0.4 - 0.3| = 1 - 0.1 = 0.9\,; \end{cases}$$

$$\begin{cases} r^{(1)}_{\text{CR}}(B) = 1 - |0.6 - 0.5| = 1 - 0.1 = 0.9\,, \\ r^{(1)}_{\text{LTV}}(B) = 1 - |0.75 - 0.3| = 1 - 0.45 = 0.55\,. \end{cases}$$

Table 5.1. Contribution of CR and LTV for the classification of customers A and B.

Customer A	value	projection	boundary	contribution
CR	0.60	0.60	0.50	0.90
LTV	0.40	0.40	0.30	0.90
Customer B	**value**	**projection**	**boundary**	**contribution**
CR	0.60	0.60	0.50	0.90
LTV	0.75	0.75	0.30	0.55

Table 5.1 shows the contribution of each variable in the classification of our two customers. We can see that for Customer A, the variables CR and LTV have the same contribution: to change the classification we have to move one or the other by the same amount. Instead, for Customer B, the contribution of CR is higher: the minimum shift along CR to change the classification is much lower than that along LTV.

5.5 Overall explanations

Finally, once the partial explanations have been obtained, they can be agglomerated. We define the total contribution of the input feature j to the decision for a given instance as

$$e_j = r_j^{(1)} + \sum_{k=1}^{N} r_k^{(2)} \hat{\phi}_j^{(k)} , \tag{5.4}$$

where the quantities involved are

- $r_j^{(1)}$: the contribution of input feature j for the business policy computed in Eq. (5.3);
- $r_k^{(2)}$: the contribution of running feature k for the business policy computed in Eq. (5.3);
- $\hat{\phi}_j^{(k)}$: the (normalized) importance of the input feature j for model $f^{(k)}$ derived from Eq. (5.2).

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 variables, those therefore most involved in the decision. In this way, a business user can analyse a decision by focusing on these variables to make his or her own qualitative assessment.

Computational complexity. The SMACE algorithm consists of solving a convex optimization problem and applying SHAP to machine learning models. First, the rule to be explained is read, scrolling through the list of its conditions. The cost is then linear in the number of conditions, which can be at most 2 times the number of variables. The convex

optimization problem is then constructed: we rely on the `cvxpy`[1] library, using the `ECOS` [Domahidi et al., 2013] solver. Next, a call is made to SHAP for each of the N models involved in the decision policy. Naming D the number of input features, KernelSHAP needs by default $1000D$ calls to the model to be explained, corresponding to the number of samples used to estimate shap values.

5.6 Side information

Note that these explanations are based on the distance between a point x and the decision boundary for a certain leaf. This allows three types of question to be answered:

1. Why the point fell in a certain leaf *(contextual explanation)*;
2. Why the point did not fall in a certain other leaf *(contrastive explanation)*;
3. Which leaves are closest to the point.

Let us consider again now the partition in Figure 5.4. Both A and B are classified as 1 and above we answered the first question, by providing the contribution of each variable to this classification. Let us consider a new customer C, having CR $= 0.3$ and LTV $= 0.4$ and so being classified as 4. We wonder now why those three customers are not classified as 3

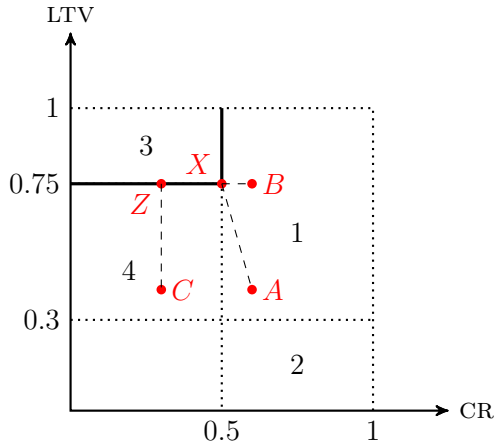

Figure 5.3. Partitions generated by a decision tree in space (CR, LTV). The point X is the projection on the decision boundary of leaf 3 for points A and B, while Z is the projection of point C.

(question of type 2), or, equivalently, why they do not belong to the subspace of the domain

[1] CVXPY library, https://www.cvxpy.org/index.html.

defined by $[0,0.5] \times [0.75,1] \in [0,1]^2$. In Figure 5.6, $X = p^{(3)}(A) = p^{(3)}(B)$ is the projection onto the decision surface of class 3 for both A and B, while the point $Z = p^{(3)}(C)$ is the projection of C.

By applying Eq. (5.3) to A, B and C for leaf 3, we get

$$\begin{cases} r^{(3)}_{\text{CR}}(A) = -(1 - |0.6 - 0.5|) = -0.9\,, \\ r^{(3)}_{\text{LTV}}(A) = -(1 - |0.4 - 0.75|) = -0.65\,, \end{cases}$$

$$\begin{cases} r^{(3)}_{\text{CR}}(B) = -(1 - |0.6 - 0.5|) = -0.9\,, \\ r^{(3)}_{\text{LTV}}(B) = -(1 - |0.75 - 0.75|) = 0\,, \end{cases}$$

$$\begin{cases} r^{(3)}_{\text{CR}}(C) = 1 - |0.3 - 0.5| = 0.8\,, \\ r^{(3)}_{\text{LTV}}(C) = -(1 - |0.4 - 0.75|) = -0.65\,. \end{cases}$$

Table 5.2. Contribution of CR and LTV for customers A, B and C to the classification in leaf 3.

Customer A	value	projection	boundary	contribution
CR	0.60	0.50	0.50	−0.90
LTV	0.40	0.75	0.75	−0.65
Customer B	**value**	**projection**	**boundary**	**contribution**
CR	0.60	0.50	0.50	−0.90
LTV	0.75	0.75	0.75	0.00
Customer C	**value**	**projection**	**boundary**	**contribution**
CR	0.30	0.30	0.50	0.80
LTV	0.40	0.75	0.75	−0.65

For customer A, both CR and LTV contribute negatively to the classification in 3: both components are outside the decision boundaries. This means that you have to move them both to reach leaf 3. CR has the same behaviour for customer B, while LTV has a null contribution: it is exactly on the decision boundary. In the case of customer C, LTV makes a negative contribution, CR positively contribute: is inside the boundary.

To answer the questions of the third type, for each leaf $\ell \in L$ we use the formula:

$$r^{(\ell)}(x) = \begin{cases} -\|x - p^{(\ell)}(x)\|_2, & \text{if } x \notin \ell\,, \\ \|x - b^{(\ell)}(x)\|_2, & \text{if } x \in \ell\,. \end{cases} \tag{5.5}$$

Note that if $x \in \ell$, Eq. (5.5) measures the robustness of the classification. If $x \notin \ell$, it tells how much point x should be moved to reach class ℓ. For customers A, B and C, results are shown in Table 5.3.

Table 5.3. Distance of each leaf from points A, B and C.

A	class	robustness	B	class	robustness	C	class	robustness
	1	0.10		1	0.10		1	-0.20
	2	-0.10		2	-0.45		2	-0.22
	3	-0.10		3	-0.10		3	-0.35
	4	-0.36		4	-0.10		4	0.20

Chapter 6

Evaluation of SMACE

What makes interpretability in these contexts even more 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 SHAP and LIME on the whole decision system. 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*. We now show empirically that SHAP and LIME do not satisfy these properties and we therefore argue that they are not suitable methods in this context. We will use both the default version of LIME, with $p = 4$ boxes, and a version with $p = 1$ developed by us to overcome the problem discussed in Section 2.8.2. The latter will be indicated as LIME-1. For completeness, we also report the explanations obtained with Anchors, although, as mentioned in Section 2.7, it is not a feature importance method. When a rule is not satisfied, we expect Anchors to identify only one of the conditions in the decision rule. In that case, each in fact is deterministically sufficient. On the other hand, when the rule is satisfied, Anchors should identify a subset of the conditions, which jointly provide some precision.

The input data used for Section 6.1 and Section 6.2 consist of 1000 instances $x^{(1)}, \ldots, x^{(1000)}$, each with three randomly generated components x_1, x_2, x_3 as uniform in [0,1].

6.1 Rules only

Let us first evaluate these properties in the case of a decision system consisting of only three simple conditions applied to only three input features. The decision-making system consists of the rule

IF $x_1 \leq 0.5$ AND
$x_2 \geq 0.6$ AND
$x_3 \geq 0.25$
THEN TRUE
ELSE FALSE.

Note that there are no models, the rule is based solely on the input data. The method then reduces to the application of Eq. (5.3), discussed in Section 5.4.

Random example. Let us pick an example at random:

$$\xi = (0.30, 0.15, 0.09)^\top .$$

The decision-making system classifies ξ as FALSE, since conditions $\xi_2 < 0.6$ and $\xi_3 < 0.25$ are violated. We want to know why ξ is not classified as TRUE and the contributions of the three variables to that decision.

Table 6.1. Evaluation in the case of three conditions on three input features.

variable	boundary	example	**SMACE**	SHAP	LIME	LIME-1
x_1	≤ 0.50	0.30	0.802	0.063	0.177	0.021
x_2	≥ 0.60	0.15	−0.546	−0.102	−0.213	−0.082
x_3	≥ 0.25	0.09	−0.842	−0.102	−0.213	−0.054

The comparison is shown in Table 6.1. The results of SMACE are computed as

$$\begin{cases} r_1^{(1)} = 1 - |0.5 - 0.30| = 0.80\,, \\ r_2^{(1)} = -(1 - |0.6 - 0.15|) = -0.55\,, \\ r_3^{(1)} = -(1 - |0.25 - 0.09|) = -0.84\,. \end{cases}$$

In this case, we see that the results of the four methods agree in their signs. However, SHAP and LIME attribute the same contribution to x_2 and x_3 even though the sensitivities of the values are different.

In the case of LIME, this behavior is well studied by Garreau and von Luxburg [2020]. The point is that the sampling is done in a space away from the boundary, and so by perturbing the example in a small neighborhood, the output does not change. Since SHAP belongs to the same class of methods as LIME [Lundberg and Lee, 2017], the same happens with it. SMACE and LIME-1 manage to capture the sign, but the latter does not satisfy property *(2)*: the contribution of x_3 should be higher than that of x_2, since it is closer to its boundary. In particular, one attribute at a time is perturbed, so whenever you need to move more than one to change the results, LIME and SHAP fails.

Anchors give

```
Anchors:   x2 ≤ 0.52
Precision:  1
Coverage:   0.50.
```

In fact, this is a sufficient condition to *not* satisfy the decision rule and get deterministically (with precision 1) FALSE as result. The coverage is 0.50 because we generated data as uniform in $[0,1]^3$: almost half of the data points have x_2 smaller than 0.52.

Example exactly on the decision boundary. Let us now compare the results in the case of an example exactly on the decision surface, so where the sensitivity is maximum:

$$\xi = (0.5, 0.6, 0.25)^\top .$$

This is a borderline case that can happen in the real way. A company might reserve a special offer for customers under the age of 26. If a user is exactly 26 years old, this information should stand out. The decision-making system classifies ξ as TRUE. All values are extremely sensitive to small variations and all variables are equally critical to the decision. Table 6.2 shows that SMACE assigns a high uniform value to all variables, while the others assign different values and in LIME there are negative contributions even if all conditions are satisfied.

Table 6.2. Evaluation in the case of three conditions on three input features, with example on decision boundary.

variable	boundary	example	**SMACE**	SHAP	LIME	LIME-1
x_1	≤ 0.50	0.50	1.00	0.355	−0.196	0.334
x_2	≥ 0.60	0.60	1.00	0.370	0.114	−0.335
x_3	≥ 0.25	0.25	1.00	0.135	−0.197	−0.312

Anchors give

`Anchors:` $x_2 > 0.52$ `and` $x_1 \leq 0.75$
`Precision:` 0.43
`Coverage:` 0.38 .

The anchors try to minimize the number of attributes within the explanations, but sometimes this turns out poorly. In this case, the conditions on x_1 and x_2 are necessary but not sufficient. However, working on the parameters could yield more accurate results.

Slight violation on one attribute. In the example above, applying a small variation on an attribute changes the result. Let us consider:

$$\xi = (0.51, 0.6, 0.25)^\top$$

The decision-making system classifies ξ as FALSE for a slight violation of the rule on the first attribute. In Table 6.3 we see that SMACE highlights the slight violation of the rule on x_1.

Anchors give

`Anchors:` $x_3 \leq 0.25$
`Precision:` 1
`Coverage:` 0.25 .

Violation on two attributes. Let us now analyze the case where the conditions on two variables are violated. This means that in order to bring the result into TRUE it is necessary

Table 6.3. Evaluation in the case of three conditions on three input features, with example that slightly violates the rule on the first attribute.

variable	boundary	example	**SMACE**	SHAP	LIME	LIME-1
x_1	≤ 0.50	0.51	−0.99	−0.290	−0.214	0.357
x_2	≥ 0.60	0.60	1.00	0.115	0.124	−0.338
x_3	≥ 0.25	0.25	1.00	0.035	−0.215	−0.349

to move two variables simultaneously.

$$\xi = (0.6, 0.1, 0.8)^\top$$

From Table 6.4, the behaviour of SHAP is particularly interesting. No change on x_1 or x_2 alone can change the outcome and therefore the two variables have equal importance for it.

Table 6.4. Evaluation in the case of three conditions on three input features, with example that violates two conditions.

variable	boundary	example	**SMACE**	SHAP	LIME	LIME-1
x_1	≤ 0.5	0.6	−0.90	−0.078	−0.193	0.053
x_2	≥ 0.6	0.1	−0.50	−0.078	−0.201	−0.110
x_3	≥ 0.25	0.8	0.45	0.017	0.059	−0.003

Anchors give

`Anchors:` $x_2 \leq 0.52$
`Precision:` 1
`Coverage:` 0.5 .

Violation on each attribute. The above case also applies with the example:

$$\xi = (0.65, 0.59, 0.14)^\top$$

The comparison is shown in Table 6.5. SHAP distributes the importance evenly over the three variables. SMACE captures information on the sensitivity of each value.

Anchors give

`Anchors:` $x_3 \leq 0.25$
`Precision:` 1
`Coverage:` 0.25 .

6.2 Simple hybrid system

To evaluate the complete method, we now add two simple linear models $f^{(1)}$ and $f^{(2)}$, considering the example $\xi = (0.6, 0.48, 0.1)^\top$.

Table 6.5. Evaluation in the case of three conditions on three input features, with example that slightly violates each condition.

variable	boundary	example	**SMACE**	SHAP	LIME	LIME-1
x_1	≤ 0.50	0.65	-0.85	-0.047	-0.193	0.219
x_2	≥ 0.60	0.59	-0.99	-0.047	0.120	-0.150
x_3	≥ 0.25	0.14	-0.89	-0.047	-0.197	-0.178

Case 1: The functions are defined as

$$\begin{cases} f^{(1)}(x) = -3x_1 + 1x_2 + 2x_3\,, \\ f^{(2)}(x) = +700x_1 - 500x_2 + 1000x_3\,. \end{cases}$$

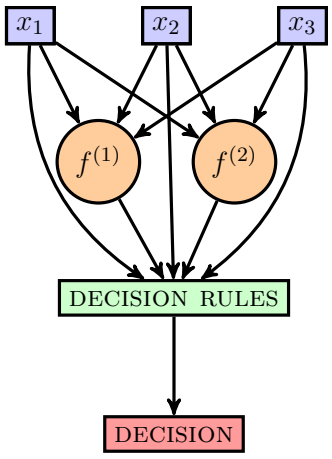

Figure 6.1. Decision-making system structure for Case 1.

Note that the coefficients of the three variables are non-zero for both functions. We now apply a simple rule to all variables, both input features and model outputs:

IF $x_1 \leq 0.5$ AND
$x_2 \geq 0.6$ AND
$x_3 \geq 0.25$ AND
$f^{(1)}(x) \geq 1$ AND
$f^{(2)}(x) \leq 600$
THEN TRUE
ELSE FALSE.

Considering the example ξ, we obtain $f^{(1)}(\xi) = -1.12$ and $f^{(2)}(\xi) = 280$. Not all conditions are met and we want to know the contributions of the three input features, also taking into account their impact on the $f^{(1)}$ and $f^{(2)}$. Following Section 5.2, let us start by applying

SHAP to explain the models. Recall that SHAP approximates Eq. (2.7). The expected values for model $f^{(1)}$ are

$$\begin{cases} \phi_1^{(1)} = -3 \cdot (0.6 - 0.5) = -0.3\,, \\ \phi_2^{(1)} = 1 \cdot (0.48 - 0.5) = -0.02\,, \\ \phi_3^{(1)} = 2 \cdot (0.1 - 0.5) = -0.8\,. \end{cases}$$

For model $f^{(2)}$ we get

$$\begin{cases} \phi_1^{(2)} = 700 \cdot (0.6 - 0.5) = 70\,, \\ \phi_2^{(2)} = -500 \cdot (0.48 - 0.5) = 100\,, \\ \phi_3^{(2)} = 1000 \cdot (0.1 - 0.5) = -400\,. \end{cases}$$

The values obtained by applying KernelSHAP and scaling by empirical standard deviations $\sigma_1 \approx 1.1065$ and $\sigma_2 \approx 382.9165$ are shown in Table 6.6.

Table 6.6. KernelSHAP values for Case 1 and Case 2.

variable	$\phi^{(1)}$	$\phi^{(2)}$
x_1	-0.207	0.134
x_2	-0.058	0.081
x_3	-0.748	-1.040

Secondly, we get the contribution of each variable involved in the rule as discussed in Section 5.4. The results obtained are in Table 6.7.

Table 6.7. Contribution of each variable for the rule in Case 1.

variable	boundary	example	contribution
x_1	≤ 0.50	0.60	-0.900
x_2	≥ 0.60	0.48	-0.880
x_3	≥ 0.25	0.1	-0.850
$f^{(1)}(x)$	≥ 1	-0.88	-0.595
$f^{(2)}(x)$	≤ 600	280	0.840

However, we want to attribute these contributions to the input variables, since we generally do not know the models involved in the decision-making system. We therefore apply the aggregation formula according to Eq. (5.4). Comparison with SHAP and LIME in Table 6.8

Anchors give

`Anchors:` $x_3 \leq 0.25$
`Precision:` 1
`Coverage:` 0.24 .

Table 6.8. Overall contribution of input features for Case 1.

variable	example	**SMACE**	SHAP	LIME	LIME-1
x_1	0.60	−0.664	−0.027	−0.046	0.060
x_2	0.48	−0.778	−0.027	0.037	−0.042
x_3	0.01	−1.278	−0.027	0.026	−0.050

Case 2. Now consider the same example ξ as above and the same two models $f^{(1)}$ and $f^{(2)}$, but change the rule to

IF $x_2 \geq 0.6$ AND
$x_3 \geq 0.25$ AND
$f^{(1)}(x) \geq 1$ AND
$f^{(2)}(x) \leq 600$
THEN TRUE
ELSE FALSE.

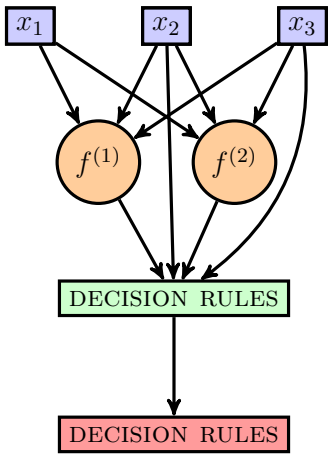

Figure 6.2. Decision-making system structure for Case 2.

i.e., we remove the rule on the first attribute.

The models and the example do not change: the KernelSHAP values are exactly the same as in the Table 6.6. The contributions of the variables to the rule are listed in Table 6.9, which is the same as the previous table except for x_1 that has zero contribution as it is not directly involved in the rule. Finally, we report the overall contribution of input features in Table 6.10 and we see that the contribution of x_1 is the average of its importance to the models weighted by the models contributions to the decision, *i.e.*

$$e_1 = r_1^{(1)} + r_1^{(2)}\hat{\phi}_1^{(1)} + r_2^{(2)}\hat{\phi}_1^{(2)} = 0 + (-0.595)\cdot(-0.207) + 0.84\cdot 0.134 \approx 0.236\,.$$

Table 6.9. Contribution of each variable for the rule in Case 2.

variable	boundary	example	contribution
x_1	/	0.600	0.000
x_2	≥ 0.6	0.48	-0.880
x_3	≥ 0.25	0.1	-0.850
$f^{(1)}(x)$	≥ 1	-0.88	-0.595
$f^{(2)}(x)$	≤ 600	280	0.840

Table 6.10. Overall contribution of input features for Case 2.

variable	example	**SMACE**	SHAP	LIME	LIME-1
x_1	0.60	0.236	-0.027	-0.089	0.059
x_2	0.48	-0.778	-0.027	-0.086	-0.042
x_3	0.01	-1.278	-0.027	-0.086	-0.049

Anchors give

$$\begin{aligned} &\texttt{Anchors:} \quad x_3 \leq 0.25 \\ &\texttt{Precision:} \quad 1 \\ &\texttt{Coverage:} \quad 0.24\,. \end{aligned}$$

Case 3. Lastly, we reuse the rule from **Case 1** and change the models:

$$\begin{cases} f^{(1)}(x) = 1x_2 + 2x_3\,, \\ f^{(2)}(x) = -500x_2 + 1000x_3\,. \end{cases}$$

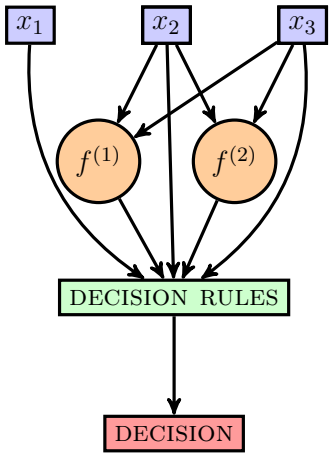

Figure 6.3. Decision-making system structure for Case 3.

Note that models do not use x_1. KernelSHAP values are in Table 6.11: $\hat{\phi}_1^{(1)} = \hat{\phi}_1^{(2)} = 0$. Variables contribution are exactly as in Table 6.7, while the final aggregation is in Table 6.12. In the latter we see that the final contribution of variable x_1 is exactly the same as that obtained for the rule. The variable x_1 was in fact not involved in the models.

Table 6.11. KernelSHAP values for Case 3.

variable	$\phi^{(1)}$	$\phi^{(2)}$
x_1	0.000	0.000
x_2	−0.097	0.095
x_3	−1.257	−1.228

Table 6.12. Overall contribution of input features for Case 3.

variable	example	**SMACE**	SHAP	LIME	LIME-1
x_1	0.60	−0.900	−0.047	−0.179	0.129
x_2	0.48	−0.746	−0.047	−0.173	−0.137
x_3	0.10	−0.341	−0.047	−0.175	−0.152

Anchors give

`Anchors:` $x_3 \leq 0.25$
`Precision:` 1
`Coverage:` 0.24 .

6.3 Real-world use case

Now we apply SMACE to the more realistic, though simple, retention offer use case of Example 5.1.1. We maintain the structure of Figure 5.1, where the two data sources *Customer data* and *Subscription data* are shown in Table 6.13 and Table 6.14, respectively.

In this section we use the realistic churn dataset *DSX Local Telco Churn demo* used by IBM in demo products.[1] It contains information about the customers of a telephone company. Categorical features are present in the dataset. Recalling what was expressed in Section 5.2.1, we cannot address the case where categorical variables are directly present in the decision policy, but they do not pose a problem when used as input to machine learning models. The goal is to analyze customer behavior and apply a retention policy, based on predicting churn risk and lifetime value. As described in the Example 5.1.1, from the available dataset, we train an XGBoost Classifier to predict churn risk (CR) and an XGBoost Regressor to predict lifetime value (LTV). Note that as CR we use the churn risk likelihood obtained via

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

Table 6.13. Personal customer features for Retention offer use case.

feature	type
ID	categorical
Gender	boolean
Status	categorical
Children	boolean
Est. Income	numerical
Car Owner	numerical
Age	numerical

Table 6.14. Subscription features for Retention offer use case.

feature	type
LongDistance	numerical
International	numerical
Local	numerical
Dropped	boolean
Paymethod	categorical
LocalBilltype	categorical
LongDistanceBilltype	categorical
Usage	numerical
RatePlan	categorical

the `predict_proba` function of XGBoost. This is a value is a value in [0,1] and since default threshold is set to 0.5, the condition CR ≥ 0.5 (respectively, CR < 0.5) equals CHURN $== 1$ (respectively, CHURN $== 0$).

We apply a simple rule to easily visualize the results:

IF AGE ≤ 50 AND
LTV ≥ 500 AND
CR ≥ 0.5 AND
USAGE ≥ 200 AND
LOCAL > 200
THEN MAKE OFFER .

Let us consider as example customer ξ, whose input values are shown in Table 6.15. Models predict CR(ξ) $= 0.35$ and LTV(ξ) $= 908.71$. The decision is *not* to make a retention offer to the client: conditions on CHURN RISK, USAGE and LOCAL are not satisfied.

The (normalized) SHAP values for CR and LTV are shown in Table 6.16 and Table 6.17. We note that the input feature rankings for the two models are very different from each other. Indeed, the attribute AGE is the most important for the prediction of LIFETIME VALUE, while

Table 6.15. Input values of customer ξ for Retention offer use case.

feature	value
Gender	M
Status	M
Children	2
Est. Income	29616
Car Owner	N
Age	49.43
LongDistance	29.78
Intenational	0
Local	45.5
Dropped	0
Paymethod	CH
LocalBilltype	FreeLocal
LongDistanceBilltype	Standard
Usage	75.29
RatePlan	2

Table 6.16. SHAP values for CHURN RISK model and customer ξ in Retention offer use case.

feature	$\phi^{(\text{CR})}$
Status	−0.42
Children	−0.40
LongDistance	0.21
Paymethod	0.14
LocalBilltype	0.08
LongDistanceBilltype	0.07
RatePlan	−0.06
Gender	0.05
Est. Income	0.05
Usage	−0.05
Car Owner	0.04
International	−0.03
Dropped	0.02
Local	0.02
Age	0.02

it is the least important for the CHURN RISK model. Rule contribution are in Table 6.18. Obviously, only the variables directly involved in the rule make a direct contribution (positive or negative). For all others, the first-level contribution is zero. Regarding the rule, AGE is the most important value, and is in fact the most sensitive one: the customer's age is 49.43

Table 6.17. SHAP values for LIFETIME VALUE model and customer ξ in Retention offer use case.

feature	$\phi^{(\text{LTV})}$
Age	−0.16
LongDistance	0.10
International	−0.05
Usage	0.04
Paymethod	−0.04
Status	−0.04
Est. Income	0.03
LocalBilltype	0.02
Dropped	−0.02
Gender	0.01
Local	0.01
Car Owner	−0.01
LongDistanceBilltype	0.01
Children	0.01
RatePlan	0.00

Table 6.18. Rule contribution for customer ξ in Retention offer use case.

feature	contribution
Age	0.99
CR	−0.84
LTV	0.78
Usage	−0.66
Local	−0.53

and on this attribute the rule is ≤ 50. Both CHURN RISK and LIFETIME VALUE have a very strong impact, the former negatively, the latter positively. The overall contribution reported in Table 6.19 highlights AGE as the most relevant input feature, with a positive impact. The three input features directly involved in the decision rule turn out to be the three most important variables overall in this case. This is because policy decision-making involves few attributes, and they all turn out to be very important for the specific customer at hand. In contrast, the two models take more features as input and the contribution to the model is more homogeneously shared among them, especially for the LIFETIME VALUE model.

The experiments reported in Section 6.1 and Section 6.2 confirm that the main interpretability methods are not suitable for the specific framework of our interest. The decision rules add non-linearities that cause LIME and SHAP to fail. In particular, we have seen that in several cases these provide flat explanations, without distinguishing between different features, even when their sensitivities with respect to the decision boundary are very different. As mentioned above, for LIME this behavior is known and has been analyzed by Garreau

Table 6.19. Overall Contribution of input features for customer ξ in Telco use-case.

feature	contribution
Age	0.85
Usage	−0.58
Local	−0.55
Children	0.34
Status	0.32
Paymethod	−0.16
LongDistance	−0.10
RatePlan	0.06
LongDistanceBilltype	−0.05
LocalBilltype	−0.05
Dropped	−0.04
Car Owner	−0.04
Gender	−0.04
Est. Income	−0.01
International	−0.01

and von Luxburg [2020]. Regarding the application of Anchors to this framework, we note that in the case where a rule is not satisfied it manages to identify with sufficient accuracy a condition, thus giving meaningful results. However it does not give any information about the contribution of other features. In the case in which the rule is satisfied, instead, we believe that Anchors is not very reliable: when the parameters change the boundaries vary significantly and in any case it is not always able to reach the required precision. On the other hand, SMACE provides a measure of feature importance aligned with operational interests, pursuing the goal of ranking variables according to contribution.

Chapter 7

Conclusion

7.1 Discussion

In this thesis, we focused on methods for the interpretability of business decision-making systems. Traditionally, expert systems based solely on decision rules in the form

IF <PREMISE> THEN <CONSEQUENCE>

have been used in these contexts. Since the late 1990s, Machine Learning is increasingly being leveraged in business processes to make automated decisions. Indeed, despite the advancement of machine learning models, it is still useful and often necessary to manage them with decision rules, instead of entrusting the entire decision to a specific model. Decision rules are crucial for expressing policies that can change over time. Using machine learning alone could result in retraining the model every time the policy is updated.

Considering that automated decision-making systems are used every day in virtually every industry, from medicine to finance, from sports to business, understanding *why* a certain decision was made has become crucial. We mentioned in Section 1.2 that as a result of public concern, many institutions are working on regulating artificial intelligence, having interpretability as one of the central points. Although very popular in business process automation, this framework is understudied in the Explainable AI literature. As far as we know, there is no method available to address our setting. We performed an exhaustive literature review on interpretability methods (Chapter 2) for machine learning and on business rule management systems (Chapter 3). As shown in Section 2.8, interpretability methods for machine learning still present many problems and thus we are not always able to appropriately explain a prediction. We also discussed in Section 1.5 that in addition to this, decision rules bring additional difficulties, for instance, by causing non-linearities that cause problems for attribution-based interpretability methods like LIME [Ribeiro et al., 2016] and SHAP [Lundberg and Lee, 2017]. As a result, applying current methods to business decision systems produce unreliable and brittle explanations: we provided examples in Chapter 6. We exploited knowledge about the structure of such decional systems: the variables involved in policy decision-making and the conditions applied to them. We believe it is worth to exploit

this information instead of treating the whole system as a black-box and being completely model-agnostic.

In Chapter 5 we presented SMACE - *Semi-Model-Agnostic Contextual Explainer*, a new interpretability method that combines a geometric approach inspired by Alvarez [2004] (for business rules) with existing interpretability solutions (for machine learning models) to generate feature importance based explanations. SMACE provides two levels of explanation, for the different users involved in the decision-making process. The first, useful for the business user, provides a ranking of importance for all the variables used, whether they are input attributes or values calculated in-house. This is useful, for example, to the sales representative, who has access to and knowledge of company policies. By interpreting the process, the business user can explain, modify, override or validate the specific decision. The second level is necessary for the end customer. He or she does not have access to the internal policy rules, nor to the way in which decision-making processes are managed. The Evaluation reported in Chapter 6 shows that while LIME, SHAP and Anchors produce poor results when applied to such a decision system, SMACE provides intuitive feature ranking, tailored to business needs. SMACE has been implemented in a Python package available in `https://github.com/gianluigilopardo/smace`, where all the experiments reported in the Evaluation are present and reproducible.

7.2 Future work

7.2.1 Including categorical variables

We recognize as a major limitation of SMACE that it cannot deal with categorical variables within decision rules. A categorical variable takes on discrete values within a set of *categories,* which are not comparable to each other in terms of distance and are often not sortable. In future work we may get rid of the **A1** assumption (see Section 5.2.1). In fact, there are solutions in the literature to be able to treat categorical variables as numerical. A popular solution is *one-hot encoding*, that consists in adding a new binary variabile (that is, a new column) for each category. This is a very straightforward, but problematic technique: it easily leads to an unsustainable number of new variables. A particularly interesting approach to this problem is included in CatBoost (Categorical Boosting) [Prokhorenkova et al., 2018], a gradient boosting toolkit. The idea is to group categories by *target statistics*, so that any categorical value x_k^i of categorical variable i for instance k can be replace by a numerical value $\hat{x}_k^i$. The commonly used target statistic is the expected target value y in each category: $\hat{x}_k^i = \mathbb{E}[y|x^i = x_k^i]$. In this way we overcome all the problems arising from categorical variables and in fact using target statistics seems to be the most efficient way to treat categories with minimum information lost.

7.2.2 Investigating the link between Anchors and LIME

In Chapter 2 we discussed LIME [Ribeiro et al., 2016] and Anchors [Ribeiro et al., 2018]. Conceptually, there is a relationship between these two methods: the former approximates any

complex model with a linear one, while the latter does so with a rule composed of conditions on attributes. But how are the explanations provided by the two methods related? Using the original implementations[1] with the default parameters, the features that are most important to LIME are not always those extracted from the Anchors. However, we believe these can be derived using a sparsity inducing penalty: we want to find a regularization function to apply to LIME that ties the two methods together.

7.2.3 Extending LIME for hybrid decision-making systems

We showed in the Evaluation that LIME standard with $p = 4$ boxes is particularly unsuitable for interpreting decisions made by systems using rules. As mentioned above, this is largely due to the nonlinearities added by the rules. In addition, the relative position of the rule-defined areas to the boxes identified by LIME affects the explanations. This behavior is easy understandable by Figure 10 in Garreau and von Luxburg [2020]. By default LIME identifies boxes with quantiles. In our case, for all variables directly involved in decision making, we already know a partition of the space that can be used to define boxes. By doing so, within each box, the problematic rule-induced linearity is no longer present: the decision boundaries correspond to LIME boxes. We believe that modifying the binning step in this way can make LIME extensible to our framework.

[1] LIME: `https://github.com/marcotcr/lime`, Anchors: `https://github.com/marcotcr/anchor`.

Bibliography

Kjersti Aas, Martin Jullum, and Anders Løland. Explaining individual predictions when features are dependent: More accurate approximations to Shapley values. *Artificial Intelligence*, 298:103502, 2021.

Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, 6:52138–52160, 2018.

Sushant Agarwal, Shahin Jabbari, Chirag Agarwal, Sohini Upadhyay, Zhiwei Steven Wu, and Himabindu Lakkaraju. Towards the unification and robustness of perturbation and gradient based explanations. *arXiv preprint arXiv:2102.10618*, 2021.

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.

Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilovic, et al. AI explainability 360: An extensible toolkit for understanding data and machine learning models. *Journal of Machine Learning Research*, 21(130):1–6, 2020.

Peter L. Bartlett and Mikhail Traskin. Adaboost is consistent. *the Journal of machine Learning research*, 8:2347–2368, 2007.

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, 2021a.

Clément Bénard, Sébastien Da Veiga, and Erwan Scornet. MDA for random forests: inconsistency, and a practical solution via the Sobol-MDA. *arXiv preprint arXiv:2102.13347*, 2021b.

George E. P. Box. Science and statistics. *Journal of the American Statistical Association*, 71 (356):791–799, 1976.

L. Breiman, J. Friedman, R. Olshen, and C. J. Stone. Classification and regression trees. 1983.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Jason Brownlee. XGBoost with Python. *Machine Learning Mastery*, 2019.

Shruthi Chari, Daniel M Gruen, Oshani Seneviratne, and Deborah L McGuinness. Directions for explainable knowledge-enabled systems. *arXiv preprint arXiv:2003.07523*, 2020.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Ian Covert and Su-In Lee. Improving kernelshap: Practical shapley value estimation using linear regression. In *International Conference on Artificial Intelligence and Statistics*, pages 3457–3465. PMLR, 2021.

Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*, pages 598–617. IEEE, 2016.

Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In *2013 European Control Conference (ECC)*, pages 3071–3076. IEEE, 2013.

Karim El Mernissi. *A study of explanation generation in a rule-based system*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2017.

EU. Reform of EU data protection rules, 2016. URL https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN.

EU. Ethics guidelines for trustworthy AI, 2019. URL https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai.

EU. Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts, 2021. URL https://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1623335154975&uri=CELEX%3A52021PC0206.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Christopher Frye, Colin Rowat, and Ilya Feige. Asymmetric shapley values: incorporating causal knowledge into model-agnostic explainability. *Advances in Neural Information Processing Systems*, 33, 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.

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

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

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.

Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Regularization is all you need: Simple neural nets can excel on tabular data. *arXiv preprint arXiv:2106.11189*, 2021.

Maxim S. Kovalev, Lev V. Utkin, and Ernest M. Kasimov. SurvLIME: A method for explaining machine learning survival models. *Knowledge-Based Systems*, 203:106164, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25: 1097–1105, 2012.

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.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

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

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54 (6):1–35, 2021.

Luke Merrick and Ankur Taly. The explanation game: Explaining machine learning models with cooperative game theory. 2019.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

Christoph Molnar. *Interpretable machine learning*. 2020.

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

Marek Z. Reformat and Ronald R. Yager. Suggesting recommendations using pythagorean fuzzy sets illustrated using netflix movie data. In *International conference on information processing and management of uncertainty in knowledge-based systems*, pages 546–556. Springer, 2014.

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.

Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

Robert E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

Robert E. Schapire and Yoav Freund. Boosting: Foundations and algorithms. *Kybernetes*, 2013.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61: 85–117, 2015.

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.

Adam Smith. *Book I., Of the Causes of Improvement in the Productive Powers of Labour and of the Order According to Which Its Produce Is Naturally Distributed among the Different Ranks of the People*. 1827.

Adam Smith. Using artificial intelligence and algorithms., 2020. URL https://www.ftc.gov/news-events/blogs/business-blog/2020/04/using-artificial-intelligence-algorithms.

I. M. Sobol. Sensitivity estimates for nonlinear mathematical models. *Math. Model. Comput. Exp*, 1(4):407–414, 1993.

Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

Gian Antonio Susto, Andrea Schirru, Simone Pampuri, Seán McLoone, and Alessandro Beghi. Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3):812–820, 2014.

Sana Syed, Mohammad Al-Boni, Marium N. Khan, Kamran Sadiq, Najeeha T Iqbal, Christopher A Moskaluk, Paul Kelly, Beatrice Amadi, S Asad Ali, Sean R Moore, et al. Assessment of machine learning detection of environmental enteropathy and celiac disease in children. *JAMA network open*, 2(6):e195822–e195822, 2019.

Alan M. Turing. Computing machinery and intelligence. *Mind 49*, pages 433–460, 1950.

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.