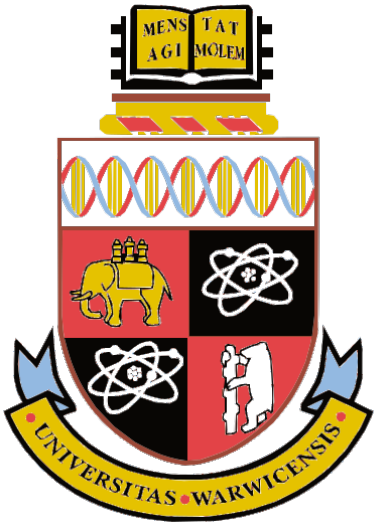

Explainable Artificial Intelligence (XAI)

by

Alexandre Duval

MA4K9 Scholarly Report

Submitted to The University of Warwick

Mathematics Institute

April, 2019

Abstract

This paper examines Explainable Artificial Intelligence (XAI) and sheds light on making machine learning models more interpretable. The current inability of humans to explain these models creates a barrier to the adoption of machine learning that becomes extremely problematic given the range of opportunities it creates for the society. To tackle this issue, this paper positions itself as a scientific guide for interpretability, filling in a major gap in the literature. More precisely, after setting the foundations for this concept, it provides a unified mathematical view of the current main explanation methods, which is vital to truly comprehend both the purpose and functioning of the technique utilised. This rigorous theoretical approach is then completed with an application on a customer churn use case for a telecom company. Its analysis illustrates how to implement, interpret and ideally combine these methods to obtain the best possible understanding of a real-world problem. Finally, to achieve an even more complete comprehension, this paper emphasises the need to search for some alternative approaches to the problem. In this case, survival analysis is chosen, applied and its theory expanded.

Acknowledgements

For my Master degree of Advanced Study in Mathematical Sciences, I conducted this research to gain more insight into a field I am passionate about: Explainable Artificial Intelligence. I would like to thank my supervisor Dr Sebastian Vollmer for giving me the opportunity to carry out this assignment as well as for his great guidance and constant support.

Contents

1 Introduction

Despite the fact that Explainable Artificial Intelligence (XAI) has become an area of interest for quite some time and has recently received much attention, there still exists a few discordances concerning the definition of some concepts that we use throughout this research. To avoid bewilderment due to ambiguity, we shed light on their meaning.

1.1 Definition

There is a general confusion between the terms machine learning (ML) and artificial intelligence (AI). Based on some articles that I have been reading, most individuals refer to AI and ML as equivalent words and utilize them reciprocally. While the majority does not know the distinction between them, some deliberately overlook their differences to create hyper excitement for advertising and sales purposes. Indeed, many organizations find it more profitable to use the vague term AI, which has a lot of baggage and exudes a mystic aura instead of being more specific about what kind of technologies they employ.

On the one hand, machine learning is a branch of Artificial Intelligence, simply defined by Tom M. Mitchell as "the study of computer algorithms that improve automatically through experience" [1]. It is a paradigm shift from 'normal programming' where all instructions must be explicitly given to the computer to 'indirect programming' that takes place through providing data. On the other hand, the term artificial intelligence refers to a broader concept. Andrew Moore defines it as "the science and engineering of making computers behave in ways that, until recently, we thought required human intelligence" [2]. In contrast to ML, AI is a moving target and its definition changes as its related technological advancements turn out to be further developed. This specificity was clarified by Zachary Lipton when he stated AI "is aspirational, a moving target based on those capabilities that humans possess but which machines do not" [3].

An XAI is an artificial intelligence whose actions can be easily understood by humans. It implies dealing with interpretable machine learning (iML) models, whose behaviour and predictions are understandable to humans. It also contrasts with the concept of 'black box' in ML, where even their designers cannot explain why the model arrived at a specific decision. Since those models usually yield the best results, the aim of iML is to produce more explainable models while maintaining the same level of performance. This suggests the existence of a limited trade-off between accuracy and interpretability [4, 5] or equivalently between completeness and interpretability, where the former refers to describing the operation of a system in an accurate way [6].

Interpretability is here defined as the degree to which humans can understand the cause of a decision [7]. In other words, it is the degree to which a human can consistently predict the model result [8]. Like T. Miller, this paper uses both the terms 'interpretable' and

'explainable' interchangeably. Nevertheless, a distinction is made between these terms and 'explanation'. The latter is post-hoc interpretability and is used for explications of individual predictions. It refers to numerous ways of exchanging information about a phenomenon; in this case the functionality of a model or the rationale for a decision [7].

The last precision brought by this paper regards the terms algorithm and model. The former is a set of mathematical instructions or rules to be followed in order to solve a targeted problem. The latter refers to well-defined computations resulting from an algorithm that learns from training data and outputs some value whose nature depends on the task it aims to perform (classification, regression, recommendation, etc.). It implies finding out the parameters in the equation of the algorithm used.

1.2 Motivation

I believe Explainable AI is the key to shaping the future that we want to live in. Knowing why it is needed not only shows why I am choosing this research subject but also why I wish to enter the AI community afterwards. As a result, I would like to place a special emphasis on this section.

1.2.1 Biases

The last decade has witnessed the rise of ubiquitous opaque decision systems, which exploit sophisticated ML models to predict individual information. The predictive models built are able to map user features to an outcome thanks to a learning phase. This learning process is made possible by the digital traces that people leave behind them while performing everyday activities. This enormous amount of data may contain human biases and prejudices and is used to train our decision model, which may inherit such biases, possibly leading to wrong and unfair decisions. For instance, Kate Crawford showed in her famous article 'Artificial Intelligence's White Guy Problem' [9] examples of ML applications including image categorization, jobs advertising, delivery service and many more that were discriminating against some communities such as black people, the poor or women.

The concerns are timely. Algorithmic decision-making mediates more and more of our interactions, influencing our social experiences, the news we see, our finances and our career opportunities. While machine learning models penetrate critical areas like medicine, the criminal justice system or financial markets, the inability of humans to understand them becomes seriously problematic [10, 11]. Although these models are usually conceptualized with the best intentions, we often fail to understand how they are making decisions and consequently fail to measure or improve their outcomes. We therefore propagate a simplified version of reality, often to the detriment of many, usually marginalized people of our complex actual reality. These models, called 'Weapons of Maths Destruction' (WMD) by

Cathy O'Neil [12] share the following characteristics: opaque, scale and damage. They must be avoided at all cost if we want to prevent harmful and unfair decisions.

1.2.2 Legal requirements

In addition to being increasingly present in our daily lives, autonomous machines and black-box algorithms start making decisions previously entrusted to humans. Hence, it becomes necessary for these mechanisms to explain themselves. In response to this reality, the European Parliament has adopted the General Data Protection Regulation (GDPR) in 2018. It introduces for the first time a social right to explanation for all individuals when automated decision-making takes place. More precisely, it states that "the data subject shall have the right not to be subject to a decision based solely on automated processing". The key here is 'solely', it means that if an organization wants to use automated decision-making, say to process loan applications, a human must be able to retrace the steps that led the model to the final decision before confirming or vetting it. Although everyone agrees on the GDPR necessity, it remains a huge open scientific challenge. Without the technology to explain black boxes, the right to explanation remains a dead letter.

1.2.3 Safety and industrial liability

The risk of creating and using decision systems that we do not really understand not only impacts ethics but also safety and industrial liability. Companies increasingly market services and products by embedding machine learning components, often in safety-critical industries such as self-driving cars, robotic assistants, and personalized medicine. An inherent risk of these components is the possibility of inadvertently making wrong decisions, learned from spurious correlations in the training data. For instance, the model can learn to recognise an object in a picture by the properties of the background or lighting due to a systematic bias in training data collection. Another risk arises because deep neural networks (DNN) can be easily fooled into misclassifying inputs with no resemblance to the true category by making some targeted imperceptible alterations to the pixels [13, 14, 15]. Such instances are called adversarial examples and can be created to make targeted errors with zero-knowledge, meaning without internal model information and access to training data. Using interpretability methods to understand which features are important and how they affect prediction is a good way to analyse the weaknesses of the model. One can then reinforce it and make it more robust to this issue by training it on some adversarial examples [6, 16].

In general terms, it is not possible for companies in the business world to trust their products without understanding and validating the underlying rationale of their machine learning components. Strong comprehension of the model is always necessary when it

comes to taking important decisions based on its output, which is why interpretability is compulsory for the development of AI into some core areas of the professional world. Explanation technologies should thus be of great help to companies for creating safer and more trustable products as well as better managing any possible liability they may have.

1.2.4 Research and futuristic scenarios

As a consequence, explanation is at the heart of a responsible and open data science across multiple industry sectors and scientific disciplines. The AI community has, through 'The Future of Life Institute', identified what should be the main research directions and decided to respect some key principles. Among them, we find the notions of failure transparency[1] and judicial transparency[2] [17] that both reveal the importance granted to interpretability.

Some principles of the list aim in the long run at regulating the race towards Artificial General Intelligence[3] that we (humans) are involved in and that can end in a fascinating broad range of aftermath scenarios. In his book Life 3.0, M. Tegmark [18] offers an interesting overview of the most plausible ones. It includes as core idea a superintelligence coexisting peacefully with humans, an AI preventing superintelligence and the extinction of the human race. Although there is no real consensus concerning what situation is ideal, the best way to stay master of our destiny is to align AI with our goals. To do so, M. Tegmark suggests following three unsolved problems: make AI learn your goals, make AI adopt your goals and make AI retain your goals. XAI hence constitutes an obvious first step towards solving these problems and in more general terms, insuring the survival of the human race.

1.3 Desiderata of an interpretable model

At present, explainability has no formal technical meaning. One aim of this paper is to propose more specific definitions. Before delving into them, we must ask what the real-world objectives of interpretability research are. We have already seen several reasons why XAI must be used and use-cases where it should be applied but have still not stated clearly what it should bring.

Some papers motivate interpretability by suggesting it to be prerequisite for trust [19, 20]. But what is trust? Is it simply confidence that a model will perform well? If so, a sufficiently accurate model should be demonstrably trustworthy and interpretability would serve no purpose. Trust might then denote confidence that the model will perform well with respect to the real objectives and scenarios, even when the training and deployment

[1] if an AI system causes harm, it should be possible to ascertain why

[2] any involvement by an autonomous system in a judicial decision making should provide a satisfactory auditable by a competent human authority

[3] AGI: ability to accomplish any cognitive task at least as well as humans

objectives diverge. In other words, the confidence that it makes the right decision for the right reasons, which underlines its ability to generalize well to unfamiliar situations. This implicitly suggests that we should be able to validate how a model takes decisions, justifying, for instance, the effect and importance of certain key features.

While the ML model objective might be to reduce error, its real-world purpose is to provide useful information to human decision makers [21]. The most obvious way to do so is via its output. However, explanation methods can convey additional information to the human decision-maker by shedding light on the inner workings of a model (coefficients, feature interaction, marginal effect of a feature on output). This is even possible without accessing model parameters. For instance, a diagnosis model might provide intuition to a human decision-maker by pointing to similar cases in support of a diagnostic decision.

At present, politicians, journalists and researchers have expressed concern that we must produce interpretations for the purpose of assessing whether decisions taken automatically by algorithms conform to ethical standards [22]. Indeed, ethics is often forgotten due to performance research but this is now likely to change since the new GDPR enforces that individuals affected by algorithmic decisions have a right to explanation. This aims at eliminating biases and decision-making that violates ethical norms.

Finally, the notions of control, accountability and safety are motivations for interpretability. We desire the ability to rapidly identify and correct mistakes in critical situations, the ability to debug and audit the model, the ability to guard against hacking and deliberate manipulation of learning and reward systems, the ability to find who is responsible for each aspect of the AI in case of a problem (self-driving car), and so on.

1.4 Research objectives

This paper cannot tackle everything XAI involves and therefore focuses on Explainable Data Science. In particular, it details from a mathematical perspective the theory of the main explanation methods before exploring an application on a customer churn use-case. Hence, it positions itself as a reference guide to interpretability, which the literature cruelly lacks. Indeed, these techniques are never dealt together, descriptions are scattered all over the web and are structured as a code along with some intuitive explications. The rigorous scientific approach proposed in this paper will solve this problem by laying strong theoretical grounds for all these methods. The case study investigated is then necessary to illustrate how to implement them and how, when combined together, do they enable a wide and reliable understanding of a real-world problem. To achieve a more complete comprehension, this paper also applies survival analysis to approach the problem from a different angle, emphasising particularly the time component. For this task to be conducted properly, I first need to introduce some context. This paper thus starts by providing an accurate overview of the notion of interpretability.

2 Interpretability

This section presents a detailed overview of interpretability, insisting respectively on the taxonomy, interpretation methods, dimensions of interpretability and the explanations provided. These ideas are key prerequisites for the rest of the project, they should be harnessed before going further.

2.1 Taxonomy

Interpretation methods broadly fall into two categories: transparency and post-hoc explanations. Informally, transparency is the opposite of opacity or blackboxness. It connotes some sense of understanding of the mechanism by which the model works. We can consider transparency at the level of the entire model, at the level of individual components[4] and at the level of the training algorithm [23]. Complete transparency implies that for a model to be fully understood, a human should be able to take the input data together with the model parameters, and in reasonable time step, go through every calculation required to produce a prediction. Interpretability is thus achieved by restricting the complexity of the model, meaning both its size and the computation power required. This itself infers that transparency requires dealing with intrinsically interpretable models like decision tree or sparse linear model, which are naturally explainable by their simple structure. A clarification concerning transparency and accuracy of the main machine learning algorithms is brought in Appendix A.

Note however that humans exhibit none of these forms of transparency while they are, by definition, interpretable. They are rather opaque entities explaining their decision after-the-fact and thus justify the existence of post-hoc methods for explainability.
Similarly to humans, this type of method does not try to elucidate precisely how the model works but aims instead to confer via a diverse range of output useful information concerning some other aspects of the model. This is possible because these techniques intervene after model training, which also make them applicable to the whole range of models, from intrinsically transparent to opaque ones. One advantage of such approach is that it allows the interpretation of complex opaque models, which usually yield greater performance. The difficulty engendered by such task has led various researchers to delve into the topic and find a multitude of interpretation techniques. Some common approaches include natural language explanations, visualisations of learned representations, local explanations and explanations by example [23, 24]. This will be the subject of the next subsection. Finally, note that although useful organizationally, the division of interpretation methods is not absolute. Some post-hoc analysis techniques attempt to uncover the significance of various of parameters, an aim we group under the heading of transparency.

[4] each part of the model (input, parameter, and calculation) admits an intuitive explanation

Before looking at explanation methods in more depth, a precision must be brought concerning model-specific and model-agnostic interpretation tools. While the former are limited to specific model classes, model-agnostic tools can be used on any machine learning model and are usually post hoc. Their agnostic nature means they cannot have access to any model internals like weights or structural information. These methods usually operate by analysing feature input and output pairs; modifying the input of the machine learning model and measuring changes in the output. Finally, a further distinction can be made between local and global explanations. Does the interpretation method explain a single prediction (local) or the entire model behaviour (global)? [25]

2.2 Methods

We now turn to explanation methods and present in this section the different outcomes of interpretability, which serve to roughly differentiate them. More detailed descriptions can be found in [4], [25], and were particularly useful to help me narrow down the scope of methods that I wish to investigate for the case study.

- Feature summary statistic (global, model-agnostic). Many interpretability techniques provide a kind of summary statistic of how each feature affects model predictions. It can be feature importance measures or statistics about the interaction strength between features.
- Feature summary visualisation (model-agnostic): some feature summaries can only be visualised and not meaningfully be placed in a table. They aim to determine qualitatively what a model has learned. For instance, arbitrary curves called Partial Dependence Plots (PDP) show the marginal effect of a feature on the outcome, for both linear and non-linear relationships. Individual Condition Expectation (ICE) and Accumulated Local Effects (ALE) are two important variants of PDP.
- Model internals (global; model specific). The interpretation of intrinsically interpretable models, such as the weights of linear models or the learned tree structure of decision trees, falls under this category. It is an efficient way of pointing out what causes a certain outcome. The lines are blurred between model internals and feature summary statistic. In linear models for example, the weights are both model internals and summary statistics for the features.
- Intrinsically interpretable model (model-agnostic). This is a little circular, but one solution to interpreting black box models is to approximate them (either globally or locally) with an interpretable model. The surrogate model itself constitutes the explanation but can be further explained by feature summary visualisation or feature summary statistics. LIME is the most popular local method.
- Text explanations. Humans often justify decisions verbally. Similarly, we might train

one model to generate predictions and a separate model to generate an explanation [26]. For example, one convolutional neural network to learn representations coupled with a recurrent neural network language model that generate captions.

- Data point. This category includes all methods that return existing or newly created data points to make a model interpretable. Influential Instances, for example, spots datapoints that considerably change the parameters or predictions of the model. Counterfactuals, to explain a prediction, find a similar data point where some targeted changes have been made to feature values of the instance regarded such that the predicted outcome changes in the desired way. Adversarial examples, that we mentioned earlier constitute the opposite of counterfactuals as they shift predictions to desired wrong outcome while minimizing the changes applied to the original instance. A last good example-based method is the identification of prototypes and criticisms, which are respectively instances well and not well represented of the data.

2.3 Dimensions

Now that we have seen what the different types of explanation methods are, we need to choose between them to select the one that best apply to the situation faced. To do so, one can identify a set of dimensions to be taken into consideration and that characterize the explainability of the model [4], [27]. To only name a few:

- Type of data: is it tabular data, a text, an image? Each type presents a different level of interpretability to a human and is handled differently by ML algorithms.
- Time limitation: how much time is the user allowed to spend on understanding the explanation. User time availability is strictly related to the scenario where the predictive model is used.
- Nature of user expertise: what is the background knowledge and expertise of the user? Domain experts may prefer sophisticated explanations over more intuitive and opaquer ones while it may just be the opposite for a user lambda.
- Type of problems faced: to get the desired information, do we need to explain the model, explain the outcome, inspect the black box internally or give a transparent solution. If we desire a complete understanding of a model, we should try to combine them. This involves mixing different interpretation methods.

2.4 Explanations

In previous parts, we have seen some core elements that guide us given the situation faced towards the optimal explanation method to provide the user with a good understanding

of how the model behaves and how predictions are made. This section answers how do we want our explanation to be and thus investigates what a good explanation is.

Explanations, and more broadly epistemology, causality, and justification, have been the focus of philosophy for millennia, making a complete overview of the field unfeasible [24]. From this, two main types of explanations can be distinguished in ML according to their completeness, meaning the degree to which the entire causal chain and necessity of an event can be explained [28]. This is often expressed as the difference between 'everyday' and 'scientific' explanations. Miller argues that 'everyday explanations' address why particular things occurred [7] (why did a customer churn?) while 'scientific explanations' regroup general scientific phenomena and philosophical questions (why have we not been contacted by alien life yet?). I focus on 'everyday' explanations because those are relevant to interpretable machine learning, and more precisely to the case study. But even for this type of explanation, the functions used to make decisions may well be too complex for humans to comprehend. Hence, it may not be possible to completely understand the full decision-making criteria or rationale. Under these constraints, what we humans see as a good explanation becomes an open question. Humanities research, and in particular Miller(2017)'s huge survey of publications on explanations, can help us find out. Implications for iML then follow easy and should help us create satisfying explanations.

Explanations are contrastive [29]: humans usually do not ask why a certain prediction was made, but rather why this prediction was made instead of another. For instance, when your loan application is rejected, you are interested in the factors of your application that would need to change so that it got accepted.
Explanations are selected: people do not expect explanations to cover the actual complete list of causes of an event, they only want to know about the main ones. Abnormal causes, meaning those having a small likelihood but that happened anyway, are particularly interesting to look at because removing them often change the outcome significantly.
Explanations are social [24]: both the content and type of explanation need to be adapted to the profile of the receiver and to the social context.
Explanations are truthful: they appear to be true in reality.
Explanations are coherent with prior beliefs of the explainee [30]. Humans tend to ignore information that is not coherent with their prior beliefs. This is called confirmation bias.
Explanations are general and probable. A cause that can explain a lot of events is very general and could be considered as a good explanation. For instance, 'the bigger a house the more expensive it is', is a very general and good explanation of why houses are expensive.

As a final remark, I wish to stress that there is no real consensus on what interpretability in machine learning is accurately. Similarly, it is not clear how to measure it. There is an initial research and an attempt to formulate some approaches to its evaluation, but I will not detail that here. You can refer to Doshi-Velez and Kim [31] for more details.

3 Customer churn analysis for a telecom company

In this section, I describe the use-case chosen to illustrate an application of interpretation methods. It serves as a complement to their theoretical definition and will be studied throughout the rest of the paper. Before delving into the model implementation phase, I define what customer churn analysis is and depict the dataset as well as the algorithm used. This paper focuses solely on a 'normal' data science application for now. The proper explainability part will be attacked in the next chapter.

3.1 Customer churn

In simple terms, customer churn, also called customer attrition or customer defection, is defined to be the loss of clients for a company. Since acquiring new customers is more expensive than retaining existing ones, churn prevention can be regarded as a popular way of reducing the company's costs. Indeed, it has been proven that a small change in retention rate generates a considerable impact on revenues [32]. This statement follows rather logically; earning business from new customers means working leads all the way through the sales funnel, utilising marketing and sales resources throughout the process. Customer retention, on the other hand, is generally more cost-effective as you have already earned the trust and loyalty of existing customers. Because customer attrition will absolutely result in loss of incomes, churn prediction has become a major task for firms to remain competitive. Telecom companies are no exception and, similarly to all firms dealing with long term clients, use customer attrition analysis as one of their key business metrics. By monitoring churn rate[5], they are equipped to determine their customer retention success rates and identify strategies for improvement [33].

Companies usually make a distinction between voluntary churn and involuntary churn [34]. The former refers to a decision by the customer to switch to another company or service provider whereas the latter occurs due to circumstances such as a customer's relocation, fraud or death. In most applications, including this paper, involuntary reasons for churn are excluded from the model. Analysts tend to concentrate on voluntary churn, whose causes are usually related to the company-customer relationship, which firms control, such as how billing interactions are handled or how after-sales help is provided.

Predictive analytics use churn prediction models to assess customers' propensity of risk to churn. Since these models generate a small prioritised list of potential defectors, they are effective at focusing customer retention marketing programs on the subset of the customer base who is most vulnerable to churn. Effective customer churn management for companies therefore requires building comprehensive and accurate models, allowing to

[5] number of customers that churned by the end of the month divided by the number of customers there was at the start of the month

distinguish churners from non-churners as much as possible.

This paper looks at fictitious customers who churned during the last month in the telecommunication company named Telco. The dataset was taken on IBM's website; it contains 7043 observations and 21 variables. Each row represents a unique client and each column contains information about his account (contract, payment method, etc.), his demography (gender, age, etc.) and his services (phone, multiple lines, internet, etc.). The exhaustive list of variables used in this analysis is included in Appendix C.

3.2 Machine learning algorithm

Conducting a reliable analysis induces first of all great performance. Referring to Appendix A, choosing neural networks or boosting methods seems straightforward. Today, deep learning commands greater attention because of its presence at the heart of computer vision, which is trendy given the possibilities it suggests for AI. Its state-of-the art performances, sometimes better than humans, in some very difficult unstructured domains equally contributed to its success. However, in addition to being targeted by many researchers already, neural networks are opaque and need a large amount of data and training time. Boosting methods, on the other hand, are more interpretable while also enabling to reach state-of-the art results for classification and regression problems with tabular data. They are more useful than neural nets in the regime of limited training data, little training time and little expertise for parameter tuning. Because of this, they are ubiquitous in Kaggle solutions and widely used by firms in the professional world. Coupling all these reasons with the motivation part in Chapter 1, this paper decides to incorporate a boosting algorithm.

Before seeing precisely which algorithm is used, let me quickly clarify the concept of boosting. It is an ensemble technique in which the predictors are not made independently, like bagging, but sequentially. It employs the logic in which the subsequent (weak) predictors learn from the mistakes of the previous (strong) predictors [6] [35]. There exist different types of boosting methods. The one we are interested in, which is also the most performant, is called eXtreme Gradient Boosting (XGBoost). Why is it called this way? Very simply because it is an improved version of Gradient Boosting (GBM), which uses gradient descent to minimise the loss function when adding new weak predictors[7]. If you are not familiar with GBM, you should refer to Appendix B for more information.

XGBoost uses a more regularized model formalization to control over-fitting and employs a number of tricks that make it faster and more accurate than traditional gradient boosting. We count among them an approximate algorithm for split finding, a sparsity-aware algorithm to handle sparse data, a block structure for parallel learning or even a weighted

[6] weak and strong refer to a measure of how correlated the learners are to the actual target variable
[7] decision trees for XGBoost

quantile sketch for approximate tree learning. XGBoost also provides insights on cache access patterns, data compression and sharding to build a scalable tree boosting system. More insights those improvements is available in the XGBoost paper [36]. Understanding XGBoost is essential to our model explainability study.

3.3 Model implementation and results

This paper only mentions the results obtained by the final model derived. Note that it contains 150 trees of maximal depth 3 after being tuned. For those who are interested in the way this model was constructed, the main data pre-processing, feature engineering and model implementation steps are detailed in Appendix C. A full version of the code (in Python) is available on the Github link [37].

Since we have used an oversampling technique named SMOTE to deal with our unbalanced dataset, cross-validation results become meaningless for the training test. Concerning the test set, the area under the ROC curve is 0.756 and the accuracy score 0.776. These results are satisfying although we could have hoped for an even better performance for this kind of task. Despite an undeniable improvement caused by the use of SMOTE, the model still struggles to spot customers who churn, which appears clearly in Table 1. The influence of SMOTE on the false positive (FP) - false negative (FN) trade-off constitutes a core part of the analysis provided in Appendix C.

The most important fact to bear in mind from this subsection is that given its good performance, the model can meaningfully be used by the company, which makes the interpretability analysis relevant.

	Predictive class	
True class	churn	non-churn
churn	271 (TP)	110 (FN)
non-churn	204 (FP)	822 (TN)

	Precision	Recall	f1-score
churn = 0	0.88	0.80	0.84
churn = 1	0.57	0.71	0.63
weighted avg	0.80	0.78	0.78

Table 1: Confusion matrix (left) and other performance metrics (right)

4 Interpretation methods

As mentioned in the motivation part, characterisations of interpretation methods are scattered all over the web and always consist of a coding part along with a small textual intuition. This section directly tackles this limitation by proposing a meticulous mathematical definition of the most preeminent model-agnostic explanation techniques. Furthermore, in the literature, no link is made between them. This is rectified by this paper's case study, which not only serves as an implementation and interpretation guide but also as a way to link them together and evaluate their complementarity.

In the following subsections, I assume there are N instances in the dataset, let the feature space $(x_1, ..., x_p)$ be p-dimensional and denote by $\hat{f}(x)$ the ML model predicted probability for a customer to churn at the end of the month; where $\hat{f} : \mathbb{R}^p \longrightarrow [0, 1]$ and x is the datapoint representing the customer. Concerning attributes, let $x_j^{(i)}$ be the value of feature x_j for the i^{th} instance. Upper case X_k is used to emphasise the random variable accounting for the value distribution of the feature x_k.

4.1 Partial Dependence Plot (PDP)

It shows the marginal effect of a feature on the prediction of a machine learning model. PDP is a global method as it considers all instances before plotting the relationship between the target and the chosen features [38].

Graphical rendering of the value of $\hat{f}(x)$ as a function of its arguments provide a comprehensive summary of its dependence on the joint values of the input variables $x = (x_1, ..., x_p)$. However, it is limited to low dimensional arguments. Since this is usually not the case, it is useful to view the partial dependence of the ML model predictions $\hat{f}(x)$ on a selected small subset $x_s \subseteq x$ of the input variables, usually of dimension one or two. In this case, x_s denotes the features of interest for which we want to know the effect on model predictions. Denote by x_c its complement in x, so that $x_s \cup x_c = x$.

In principle, $\hat{f}(x)$ depends on variables in both subsets, meaning $\hat{f}(x) = \hat{f}(x_s, x_c)$. We would like to condition $\hat{f}(x)$ on the specific value of x_c so that it could be considered as a function depending solely on x_s, interactions with other variables included. This can be achieved by marginalising the model output over the distribution of x_c. If the dependence on specific values of x_c is not too strong, the partial function $\hat{f}_s$

$$\hat{f}_s(x_s) = \mathbb{E}_{X_c}[\hat{f}(x)] = \mathbb{E}_{X_c}[\hat{f}(x_s, X_c)] = \int \hat{f}(x_s, x_c) p_c(x_c) dx_c = \int \hat{f}(x_s, x_c) dP_c(x_c)$$

$$\text{where } p_c(x_c) = \int p(x) dx_c \text{ is the marginal probability density of } x_c$$

and P_c its marginal distribution,

represents the partial dependence of $\hat{f}(x)$ on x_s. The above definition infers that PDP calculates the expectation over X_c, which comes back to averaging predictions when x_s is fixed and x_c varies over its marginal distribution $P_c(x_c)$.

Since the true marginal distribution is not known, the partial function $\hat{f}_s(x_s)$ is estimated using Monte Carlo Method by:

$$\hat{f}_s(x_s) = \frac{1}{N}\sum_{i=1}^{N}\hat{f}(x_s, x_c^{(i)})$$

where $x_c^{(i)}$ represents the actual feature values of the attributes we are not interested in for an instance $x^{(i)}$. In simple terms, we take all our data instances, keeping the values of x_c as they are and forcing the features x_s to a certain grid value, and average the predictions for this manipulated dataset.

Algorithm .1 Partial Dependence Plot for 1 feature

Input: feature of interest x_j, which takes m different values in the dataset X containing N instances. p-dimensional feature space.
for l=1 to m **do**
  $x_{j_l} \leftarrow l^{th}$ distinct value of x_j
  for i = 1 to N **do**
    $x_j^{(i)} \leftarrow x_{j_l}$
    $\hat{f}_i \leftarrow \hat{f}(x_1^{(i)}, ..., x_p^{(i)})$
  end for
  $\hat{f}^l \leftarrow \frac{1}{N}\sum_{i=1}^{N}\hat{f}_i$
end for
Draw graph of $\hat{f}^l$ against x_{j_l}

Concerning categorical features, the partial dependence is very easy to calculate: just force all data instances to the same class. For a classification task, such as the case study investigated in this paper, the model can output probabilities and the partial dependence function simply displays the probability of belonging to a certain class (here 'churn') given different values of x_s. If we consider the specific case of regression trees, as in XGBoost, the partial dependence of $\hat{f}(x)$ on x_s can be evaluated without reference to the data itself. For each specific values of the variables, a weighted transversal of the tree is performed. At the root of the tree, a weight value of 1 is assigned. For each non-terminal node visited, if the split variable is in the target subset x_s, the appropriate left or right daughter is visited and the weight is not modified. The part not visited is disregarded. If the node split is a member of the complement subset x_c then both daughters are visited and the current weight is multiplied by the fraction of training observations that went left or right respectively at the node. Each terminal node visited during the transversal is assigned the current value of the weight. Once the tree transversal is completed, the value of $\hat{f}_s(x_s)$ is the corresponding weighted average of the $\hat{f}(x)$ values over those terminal nodes visited

during the transversal. For a collection of M regression trees, simply average the results of individual trees.

Note that we are using sets S and C here instead of simple indices because it is possible to plot the partial dependence of several features with respect to the predicted outcome. However, with strictly more than two features of interest, the plot becomes very difficult to interpret.

This method is intuitive, simple to implement and involves clear causal interpretation. However, it has a few defaults worth mentioning as they allow to introduce ICE and ALE:

- Assumption of independence between features (solved by ALE)
- Heterogenous effects might be hidden as we take the average (solved by ICE)

4.2 Individual Conditional Expectation (ICE)

ICE is equivalent to PDP for local expectations, and therefore enables to see heterogeneous behaviour. For some chosen features of interest x_s, ICE plots a line per instance representing how the instance's prediction changes when we modify x_s while all other features x_c are left unchanged [39]. More formally, for each instance in $\{(x_s^{(i)}, x_c^{(i)})\}_{i=1}^N$, define the curve $\hat{f}^{(i)}$ to be the plot of $\hat{f}(x_s, x_c^{(i)})$ against x_s while $x_c^{(i)}$ is kept fixed.

In other words, the process is exactly the same as PDP except that we compute predictions for each instance separately, meaning without averaging. So, instead of plotting a unique line representing the marginal effect of some features on the outcome, averaged over all observations, it plots exactly one line for each data point.

Since there are usually many instances in the dataset, the plot quickly becomes crowded, which is why ICE are often plotted on a randomly selected subset of the data points. There exists two other ways of improvement:

- centered ICE: Since it can be hard to see if ICE curves differ between individuals as they often start at different predictions, it is helpful to center the curves at a certain point x^* in x_s, called anchor point, and only display the difference $\hat{f}^{(i)} - \hat{f}(x^*, x_c^{(i)})$ in the predicted response. Anchoring the curves at the lower end of the feature is a good choice.
- derivative ICE: look at the individual derivatives of $\hat{f}$ with respect to x_s instead of the predicted response $\hat{f}$.

4.3 Accumulated Local Effects (ALE)

ALE is an unbiased and faster alternative to PDP that solves the independence problem. Its objective is to visualize and understand the 'main effects' dependence of $\hat{f}(x)$ on a

predefined group of predictors (usually one or two). To put it simply, ALE describes how features influence the prediction of a machine learning model on average. [40]

This paper focuses on first order and second order ALE effects, meaning the effect of each feature and each pair of features on model predictions. Higher order effects are difficult to interpret and are therefore rarely used. The theory stays the same but notation becomes tedious.

We first define ALE main effects $f_{j,ALE}(x_j)$ of a feature x_j, for any $j \in \{1, ..., p\}$, assuming that $\hat{f}$ is differentiable. Let $X_{\setminus j} = (X_k : k = 1, ..., p; k \neq j)$ and $z_0 = (z_{0,1}, ..., z_{0,p})$ be the approximate lower bounds for each element in $(X_1, ..., X_p)$. For $k = 1, ..., p$, each $z_{0,k}$ is slightly lower than the smallest observation $\min\{x_k^{(i)} : i = 1, ..., N\}$. This choice is not very important and only affect vertical translation of the ALE plot of $\hat{f}_k(x_k)$ versus x_k. Below is defined the first order uncentred ALE:

$$\tilde{f}_{j,ALE}(x_j) = \int_{z_{0,j}}^{x_j} \mathbb{E}[f^j(X_j, X_{\setminus j}) \mid X_j = z_j] dz_j \tag{1}$$

$$= \int_{z_{0,j}}^{x_j} \int p_{\setminus j|j}(x_{\setminus j} \mid z_j) f^j(z_j, x_{\setminus j}) dx_{\setminus j} dz_j \tag{2}$$

where $f^j(x_j, x_{\setminus j}) = \dfrac{\partial \hat{f}(x_j, x_{\setminus j})}{\partial x_j}$ is the partial derivative of $\hat{f}(x)$ with respect to x_j.

The ALE main effects of x_j are simply a centred version of $\tilde{f}_{j,ALE}(x_j)$, meaning it has mean 0 with respect to the marginal distribution of X_j. That is,

$$f_{j,ALE}(x_j) = \tilde{f}_{j,ALE}(x_j) - \mathbb{E}[\tilde{f}_{j,ALE}(x_j)] = \tilde{f}_{j,ALE}(x_j) - \int p_j(z_j) \tilde{f}_{j,ALE}(z_j) dz_j$$

The function $f_{j,ALE}(x_j)$ can be interpreted as Accumulated Local Effects in the following sense. We calculate the local effect $f^j(x_j, x_{\setminus j})$ of x_j at $(x_j = z_j, x_{\setminus j})$ and average it across all values of $x_{\setminus j}$, with weight $p_{\setminus j|j}(x_{\setminus j} \mid z_j)$. We then accumulate (integrate) this averaged local effect over all values of z_j from $z_{0,j}$ up to x_j, which gives us the influence of x_j on the prediction. Finally, subtracting its average from the result centres the ALE plot so that the average effect over the data is zero.

Note that we average changes in predictions, not predictions itself. Indeed, ALE calculates the gradient conditional on x_j and integrates over $x_{\setminus j}$ to estimate the expected change. Here, the derivative isolates the effect of the feature of interest, blocking the action of correlated features. Also, note that the use of the conditional probability density $p_{\setminus j|j}$ instead of the marginal probability density $p_{\setminus j}$ used for PDP avoids the extrapolation[8] required in PDPs.

[8] extension of a graph by inferring unknown values from trends in the known data

If the function $\hat{f}$ is not differentiable, as it is the case for tree-based methods, three changes are operated in (1):

- replace the first order derivative $f^j(x_j, x_{\setminus j})$ by finite differences for some appropriate discretization of X_j.
- replace the expectation by its corresponding sample average across all $x_{\setminus j}^{(i)}$, for which $x_j^{(i)}$ falls into the same discrete cell as z_j
- replace outer integral by corresponding summation over same discretization used when calculating finite differences.

In fact, all ALE are estimated like this, even for differentiable $\hat{f}$. Indeed, we would like ALE main effects functions $f_{j,ALE}(x_j)$ to be estimated with the same training data that was used to fit $\hat{f}(x)$, as opposed to the former definition and expression via a distribution function for X. This is further detailed in the following paragraph.

Let $\{N_j(k) = (z_{k-1,j}, z_{k,j}] : k = 1, 2, ..., K\}$ be a sufficiently fine partition of the same sample range as $\{x_j^{(i)} : i = 1, ..., N\}$ into K intervals. $z_{k,j}$ is usually chosen as the $\frac{k}{K}$ quantile of the empirical distribution of $\{x_j^{(i)} : i = 1, ..., N\}$, with $z_{0,j}$ just below the smallest observation and $z_{K,j}$ as the largest observation. Let $n_j(k)$ be the number of training observation that falls into the k^{th} interval $N_j(k) = (z_{k-1,j}, z_{k,j}]$ so $\sum_{k=1}^{K} n_j(k) = N$. Lastly, let $k_j(x)$ be the index of the interval in which x falls, meaning the index k such that $x \in (z_{k-1,j}, z_{k,j}]$.

$$\tilde{f}_{j,ALE}(x_j) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{\{i:x_j^{(i)} \in N_j(k)\}} [\hat{f}(z_{k,j}, x_{\setminus j}^{(i)}) - \hat{f}(z_{k-1,j}, x_{\setminus j}^{(i)})]$$

$$f_{j,ALE}(x_j) = \tilde{f}_{j,ALE}(x_j) - \frac{1}{N} \sum_{i=1}^{N} \tilde{f}_{j,ALE}(x_j^{(i)})$$

Let's a bit detail the process. To estimate local effects for a single numerical feature x_j, we divide its range into several intervals (instead of integrating over z), such that each instance is included in a single interval depending on its value $x_j^{(i)}$. For each instance, we compute the model prediction for the specific values $x_j^{(i)} = \textit{endpoint} = z_{k,j}$ and $x_j^{(i)} = \textit{startpoint} = z_{k-1,j}$ of the interval and regard the difference in predictions. This procedure approximates the gradients and also works for models without gradients. It is computed by the right-hand side of the equation, which represents the *effect* that x_j has for an individual instance in a certain interval. We add up the effects of all instances within an interval $N_j(k)$ via the right sum and average them over the number of instances in this interval. The result indicates the average difference in predictions for this interval (covered by the term *local*). Finally, the left-hand sum means that we *accumulate* the average effects across all intervals prior to $k_j(x)$, where x_j is included. For example, the

ALE of a feature value that lies in the third interval is the sum of the effects of the first, second and third intervals. The second equation just represents the centring step. Instead of the expectation for a continuous random variable, we take the average of the uncentred effects $\tilde{f}_{j,ALE}(x_j^{(i)})$ over all data points. The final result $f_{j,ALE}(x_j)$ is the ALE main effect.

This paper now sheds light on second order effects. The logic is exactly the same, we simply consider a pair of features (x_j, x_l) for $j, l \in \{1, ..., p\}$ and $j \neq l$. As a consequence, we will not detail this part too much. Set $X_{\setminus\{j,l\}} = (X_k : k = 1, ..., p; k \neq j, l)$. We split our analysis according to whether $\hat{f}$ is differentiable. For non differentiable $\hat{f}$, we operate the same changes as for first order effects, which makes this case equivalent to the estimation of ALE main effects, that will be detailed a bit later in this section. Hence, we assume that $\hat{f}$ is differentiable. z_0 is taken as it was defined previously. The second order uncentred effects are given by

$$\tilde{\tilde{f}}_{\{j,l\},ALE}(x_j, x_l) = \int_{z_{0,l}}^{x_l} \int_{z_{0,j}}^{x_j} \mathbb{E}[f^{\{j,l\}}(X_j, X_l, X_{\setminus\{j,l\}}) \mid X_j = z_j, X_l = z_l] dz_j dz_l \tag{3}$$

$$= \int_{z_{0,l}}^{x_l} \int_{z_{0,j}}^{x_j} \int p_{\setminus\{j,l\}|\{j,l\}}(x_{\setminus\{j,l\}} \mid z_j, z_l) f^{\{j,l\}}(z_j, z_l, x_{\setminus\{j,l\}}) dx_{\setminus\{j,l\}} dz_j dz_l \tag{4}$$

where $f^{\{j,l\}}(x_j, x_l, x_{\setminus\{j,l\}}) = \dfrac{\partial^2 \hat{f}(x_j, x_l, x_{\setminus\{j,l\}})}{\partial x_j \partial x_l}$ is the second order partial derivative of $\hat{f}(x)$ with respect to x_j and x_l.

The ALE second order effects of x_j, x_l, denoted $f_{\{j,l\},ALE}(x_j, x_l)$, is the same as $\tilde{\tilde{f}}_{\{j,l\},ALE}(x_j, x_l)$ but 'doubly centred'. This way, it has 0 mean with respect to the marginal distribution of $\{X_j, X_l\}$ and the ALE main effects of x_j and x_l on $f_{\{j,l\},ALE}(x_j, x_l)$ are both 0. The centring step is accomplished by substracting from $\tilde{\tilde{f}}_{\{j,l\},ALE}(x_j, x_l)$ its uncentred main effects:

$$\tilde{f}_{\{j,l\},ALE}(x_j, x_l) = \tilde{\tilde{f}}_{\{j,l\},ALE}(x_j, x_l) - \int_{z_{0,j}}^{x_j} \mathbb{E}\left[\frac{\partial \tilde{\tilde{f}}_{\{j,l\},ALE}(X_j, X_l)}{\partial X_j} \mid X_j = z_j\right] dz_j$$
$$- \int_{z_{0,l}}^{x_l} \mathbb{E}\left[\frac{\partial \tilde{\tilde{f}}_{\{j,l\},ALE}(X_j, X_l)}{\partial X_l} \mid X_l = z_l\right] dz_l$$

The above equation can be simplified similarly to the first order case, using (4) and conditional probability density. The complete centring is accomplished by taking

$$f_{\{j,l\},ALE}(x_j, x_l) = \tilde{f}_{\{j,l\},ALE}(x_j, x_l) - \mathbb{E}[\tilde{f}_{\{j,l\},ALE}(x_j, x_l)]$$
$$= \tilde{f}_{\{j,l\},ALE}(x_j, x_l) - \int \int p_{\{j,l\}}(z_j, z_l) f_{\{j,l\},ALE}(z_j, z_l) dz_j dz_l$$

In this subsection, we describe how the estimation of ALE main effects is defined. We partition the $\{x_j, x_l\}$ space into a grid of K^2 rectangular cells obtained as the cross product of individual one dimension partitions, that is $\{N_{\{j,l\}}(k,m) = N_j(k) \times N_l(m) = (z_{k-1,j}, z_{k,j}](z_{m-1,l}, z_{m,l}]$ for $k, m = 1, 2, ..., K\}$. Let $n_{\{j,l\}}(k,m)$ be the number of training observation that falls into the cell $N_{\{j,l\}}(k,m)$, so $\sum_{k=1}^{K}\sum_{m=1}^{K} n_{\{j,l\}}(k,m) = N$. Lastly, let $k_j(x_j), k_l(x_l)$ be respectively the index of the interval $N_j(k), N_l(k)$ in which x falls. First compute an estimate of $\tilde{\tilde{f}}_{\{j,l\},ALE}(x_j, x_l)$, then $\tilde{f}_{\{j,l\},ALE}(x_j, x_l)$ and finally $f_{\{j,l\},ALE}(x_j, x_l)$.

$$\tilde{\tilde{f}}_{\{j,l\},ALE}(x_j, x_l) = \sum_{k=1}^{k_j(x_j)}\sum_{m=1}^{k_l(x_l)} \frac{1}{n_{\{j,l\}}(k,m)} \sum_{\{i:x^{(i)}_{\{j,l\}} \in N_{\{j,l\}}(k,m)\}} \Delta_f^{\{j,l\},k,m}(x^{(i)}_{\setminus\{j,l\}})$$

$$\begin{aligned}\Delta_f^{\{j,l\},k,m}(x^{(i)}_{\setminus\{j,l\}}) &= [\hat{f}(z_{k,j}, z_{m,l}, x^{(i)}_{\setminus\{j,l\}}) - \hat{f}(z_{k-1,j}, z_{m,l}, x^{(i)}_{\setminus\{j,l\}})] \\ &\quad - [\hat{f}(z_{k,j}, z_{m-1,l}, x^{(i)}_{\setminus\{j,l\}}) - \hat{f}(z_{k-1,j}, z_{m-1,l}, x^{(i)}_{\setminus\{j,l\}})]\end{aligned}$$

$\Delta_f^{\{j,l\},k,m}$ is the second order difference of $f(x_j, x_l, x_{\setminus\{j,l\}})$ evaluated at $(x_j, x_l, x_{\setminus\{j,l\}})$.

We now estimate $\tilde{f}_{\{j,l\},ALE}(x_j, x_l)$ by

$$\begin{aligned}\tilde{f}_{\{j,l\},ALE}(x_j, x_l) = \tilde{\tilde{f}}_{\{j,l\},ALE}(x_j, x_l) &- \sum_{k=1}^{k_j(x_j)} \frac{1}{n_j(k)} \sum_{\{i:x_j^{(i)} \in N_j(k)\}} [\tilde{\tilde{f}}(z_{k,j}, x_l^{(i)}) - \tilde{\tilde{f}}(z_{k-1,j}, x_l^{(i)})] \\ &- \sum_{m=1}^{k_j(x_l)} \frac{1}{n_l(k)} \sum_{\{i:x_l^{(i)} \in N_l(k)\}} [\tilde{\tilde{f}}(x_j^{(i)}, z_{m,l}) - \tilde{\tilde{f}}(x_j^{(i)}, z_{m-1,l})]\end{aligned}$$

$$\begin{aligned}\tilde{f}_{\{j,l\},ALE}(x_j, x_l) = \tilde{\tilde{f}}_{\{j,l\},ALE}(x_j, x_l) &- \sum_{k=1}^{k_j(x_j)} \frac{1}{n_j(k)} \sum_{m=1}^{K} n_{\{j,l\}}(k,m)[\tilde{\tilde{f}}(z_{k,j}, z_{m,l}) - \tilde{\tilde{f}}(z_{k-1,j}, z_{m,l})] \\ &- \sum_{m=1}^{k_l(x_l)} \frac{1}{n_l(k)} \sum_{k=1}^{K} n_{\{j,l\}}(k,m)[\tilde{\tilde{f}}(z_{k,j}, z_{m,l}) - \tilde{\tilde{f}}(z_{k,j}, z_{m-1,l})]\end{aligned}$$

Finally, we are able to estimate ALE main effects for a pair of features:

$$\begin{aligned}f_{\{j,l\},ALE}(x_j, x_l) &= \tilde{f}_{\{j,l\},ALE}(x_j, x_l) - \frac{1}{N}\sum_{i=1}^{N} \tilde{f}_{\{j,l\},ALE}(x_j^{(i)}, x_l^{(i)}) \\ &= \tilde{f}_{\{j,l\},ALE}(x_j, x_l) - \frac{1}{N}\sum_{k=1}^{K}\sum_{m=1}^{K} n_{j,l}(k,m)\tilde{f}_{\{j,l\},ALE}(z_{k,j}, z_{m,l})\end{aligned}$$

In conclusion, ALE constitutes a much better tool than PDPs. They are faster to compute, deal with correlated features and their interpretation is clear (centred at 0 and the value at

each point is the difference to the mean prediction). However, unlike PDPs, they are not accompanied by ICEs. Moreover, in addition to being a bit more difficult to understand theoretically, an optimal choice of the interval is difficult to reach.

4.4 Global surrogate model

It involves training an interpretable model to approximate as closely as possible the predictions of a black box model. Conclusions about the black box model are drawn by interpreting the surrogate model. [25]

Perform the following steps to get a surrogate model:

1. Choose a dataset. This could be the same dataset that was used for training the black box model, a subset or a new dataset from the same distribution. Calculate the predictions $\hat{f}(x)$ of the black box model.
2. Choose an interpretable model (linear model, decision tree, logistic regression, etc.), train it on the chosen dataset along with the predictions $\hat{f}(x)$ instead of y. This is the surrogate model.
3. Measure how well the surrogate model replicates the predictions of the black box.
4. Interpret the surrogate model to draw conclusions about the black box functioning.

4.5 Local surrogate model (LIME)

Local Interpretable Model-agnostic Explanation (LIME) is one of the most popular interpretability technique. It was created to help the user trust a machine learning model, as clearly suggests the title of the original paper 'why should I trust you?' [41]. The idea behind LIME is pretty simple. For a particular instance, it tries fitting an interpretable model to a perturbed dataset to explain why this single prediction was made by the black box model. More precisely, the process is the following:

1. Select the instance of interest for which you want to have an explanation of its black box prediction.
2. Perturb the dataset and get the black box predictions for these new points.
3. Weight the new samples according to their proximity to the instance of interest.
4. Train a weighted, interpretable model (whose complexity is limited in advance to K features) on the dataset with the variations.
5. Explain the prediction by interpreting the local model.

This is expressed by the following formula:

$$\text{explanation}(x) = arg\min_{g \in G} L(\hat{f}, g, \pi_x) + \Omega(g)$$

For an instance x, the explanation is defined as the model $g \in G$, taking only K features as input with $K \leq p$, where G is the class of (potentially) interpretable models. $\Omega(g)$ is a measure of complexity for the model g, aimed to rule out complex models from the interpretable class G. $L(\hat{f}, g, \pi_x)$ measures how unfaithful g is in approximating $\hat{f}$ in the locality around x defined by π_x; where $\hat{f} : \mathbb{R}^p \longrightarrow \mathbb{R}$ is the predicted probability of belonging to a class and π_x is a proximity measure between instances z and x. As g is chosen depending on both L and Ω, LIME tries to minimize how close the explanation is to the prediction of the original model $\hat{f}$ while the model complexity $\Omega(g)$ is kept low (favour fewer features). This ensures a trade-off between interpretability and local fidelity. In practice, LIME only optimizes the loss part, the user has to determine the complexity.

As a result, the output of LIME is a list of explanations reflecting the contribution of each feature to the prediction of a data sample. It is usually presented as textual or visual artifacts that provide a qualitative understanding of the relationship between the instance's components. Note that the learned model should be a good approximation of the ML model predictions locally, but it does not have to be a great global approximation. This kind of accuracy is called local fidelity. It gives a good idea of how reliable the interpretable model is for explaining the predictions of the black box in the neighbourhood of the instance of interest.

At this stage, it is still unclear how the perturbed dataset is obtained and how is $L(\hat{f}, g, \pi_x)$ approximated given that we can make no assumption about $\hat{f}$ since we want a model agnostic tool.

We distinguish tabular data from text and image. For the latter, supposing that we want to explain an instance x, new instances z are formed by removing uniformly at random some components of the instance of interest. For a text, we delete some words or bag-of-words (using a binary variable) while for images, we disregard superpixels [9] (colour them in grey). It is then not difficult to weight each new perturbed instance z according to its proximity with x as z is obtained from x by 'switching off' a random number of interpretable components. The more you switch off, the further the perturbed instance is from the original one.

Concerning tabular data, it is slightly different and more complicated. New samples are created by perturbing each feature individually, drawing from a normal distribution with mean and standard deviation taken from the feature's statistics. Hence, unlike texts and images, we need access to the entire dataset. Note that samples are not really taken around the instance of interest but around the training data's mass centre. This problem is solved by weighting each new sample by its proximity to x using a smoothing kernel[10] to determine how large is the neighbourhood around x. If the kernel width is small, only new instances that are close to x have some influence on the output. Otherwise,

[9] group of interconnecting pixels of the same colour

[10] function that takes two data points as input and output a proximity measure

all instances matter. The lime package in Python uses the exponential smoothing kernel $k(x, y) = \exp(-\frac{\|x-y\|}{2\sigma^2})$ and the default width σ is $0.75 \times N$. Choosing the optimal kernel width, meaning the neighbourhood around the instance of interest, is the main unsolved problem of this method.

Global understanding

LIME is a local method that enables trust in predictions but it does not provide trust in the model, which requires a global approach. The original LIME paper proposes a global understanding of the model by explaining a set of individual instances (still model agnostic). The main difficulty is to select judiciously the set V of B instances proposed to the user, where B characterises the total number of instances that the user is ready to consider in order to understand the model. This is called the pick step. It involves explaining all instances of the model and picking a diverse (non-redundant) set of explanations representing how the black box behaves globally. This is achieved by the submodularity pick algorithm.

Algorithm .2 Submodular pick (SP)

Input: dataset X and budget B (maximum number of instances considered).
for $x^{(i)} \in X$ **do**
  $W_i \leftarrow explain(x^{(i)})$ using LIME.
end for
for $j \in \{1, ..., K\}$ **do**
  $I_j \leftarrow \sqrt{\sum_i^N \mid W_{ij} \mid}$ compute feature importance
end for
$V \leftarrow \{\}$
while $\mid V \mid < B$ **do**
  $V \leftarrow V \cup arg\max_i c(V \cup \{i\}, W, I)$ greedy optimization
end while
return V

To understand this algorithm, we need to introduce some concepts. W is the $N \times K$ explanation matrix, which gives the local importance of each feature (or component, since we consider all type of data) used to build to local surrogate model around an instance, for each instance of the dataset. It is obtained using LIME repeatedly. I_j is the global importance of a feature (or component) in the explanation space. Intuitively, we want I_j to be large for features that explain many different instances.

As mentioned earlier, we want to avoid selecting instances with a similar explanation. Hence, we pick and present instances that cover the important features but without being too redundant. This definition of non-redundant coverage of the features is formalised in the equation $c(V, W, I) = \sum_{j=1}^{K} \mathbb{1}_{\{\exists i \in V : W_{ij} > 0\}} I_j$, where coverage is defined as the set function c that computes the total importance of the features that appear in at least one instance of the set V.

The final aim is to find the set V that achieves the highest maximal coverage. To do so, we operate greedily, meaning we add iteratively the instance presenting the highest marginal coverage gain, defined as $c(V \cup \{i\}, W, I) - c(V, W, I)$, to V. This is expressed by

$$Pick(W, I) = arg\max_i c(V, W, I) = arg\max_i \sum_{j=1}^{K} \mathbb{1}_{\{\exists i \in V : W_{ij} > 0\}} I_j$$

4.6 Shapley Value

This method comes from Cooperative Game Theory. It describes how to fairly distribute the total gains of a game to the players depending on their respective contribution, assuming that they all collaborate. In this paper, we extend this theory to explain our model predictions by assuming that each feature x_j for a given instance $x^{(i)}$ is a player in a game where the prediction is the payout. This yields a local method that tells us how much does each feature truly contributes towards the prediction for an instance $x^{(i)}$ compared to the average prediction. This is obtained by considering for a single data point the average marginal contribution of a feature to the prediction when added to any possible group of features, where the order in which they are added to the coalition matters. [42, 43]

Since Shapley value is a local method that works identically for every instance, we can focus without loss of generality on any single data point $x^{(i)}$(for all $i \in \{1, ..., N\}$) in the rest of the section. Consider a subset S of $X = \{x_1^{(i)}, x_2^{(i)}, ..., x_p^{(i)}\}$ and let $\hat{f}(x^{(i)})$ denote the machine learning model prediction for the instance $x^{(i)}$. Note that the above term prediction refers to the predicted probability of churn in our classification application. We define the characteristic function $v : \mathcal{P}(X) \longrightarrow \mathbb{R}$ where $\mathcal{P}(X)$ denotes the power set of X and v satisfies $v(\{\}) = 0$. It associates with each coalition of features $S \in \mathcal{P}(X)$ an expected real valued prediction for the instance $x^{(i)}$ that the features in S can additionally obtain through 'cooperation', and then distribute among themselves. Supposing that $S = \{x_{k_1}^{(i)}, ..., x_{k_s}^{(i)}\} \subseteq X$, the prediction for all feature values $x_k^{(i)} \in S$ needs to be marginalised over those not in S. The change in model prediction is then obtained by subtracting from this quantity the average prediction, which yields the contribution of the coalition S to the prediction $\hat{f}(x^{(i)})$. More specifically,

$$\begin{aligned} v(S) &= \mathbb{E}(\hat{f}(X_1, ..., X_p) \mid X_{k_1} = x_{k_1}^{(i)}, ..., X_{k_s} = x_{k_s}^{(i)}) - \mathbb{E}(\hat{f}(X_1, ..., X_p)) \\ &= \int \hat{f}(x_1^{(i)}, ..., x_p^{(i)}) d\mathbb{P}_{x_k^{(i)} \notin S} - \mathbb{E}(\hat{f}(X_1, ..., X_p))) \end{aligned}$$

Note that we actually do multiple integration as we integrate for each feature in S^c. Let's take a concrete example where there are four features: $\{x_0, x_1, x_2, x_3\}$ and where

$S = \{x_1^{(i)}, x_3^{(i)}\}$. Then

$$v(S) = v(x_1^{(i)}, x_3^{(i)}) = \int_{\mathbb{R}} \int_{\mathbb{R}} \hat{f}(X_0, x_1^{(i)}, X_2, x_3^{(i)}) d\mathbb{P}_{X_0,X_2} - \mathbb{E}(\hat{f}(X_0, ..., X_3))$$

As mentioned above, we are not interested in how much a coalition S contributes to the prediction but in how much each feature truly does. This first induces the necessity to isolate each feature's effect. It can be achieved using a value operator $\phi : v \longrightarrow (\phi_1^{(i)}, ..., \phi_p^{(i)})$ where each $\phi_j^{(i)}$ denotes the fair contribution of the feature x_j to the prediction of the machine learning model (for all $j \in \{1, ..., p\}$). Secondly, it involves defining fairness, which L.Shapley characterises by the four axioms below:

Axiom 1 (Efficiency)**.** $\sum_{j=1}^{p} \phi_j = v(X)$
Axiom 2 (Symmetry)**.** *If* $\forall S \in \mathcal{P}(X)$ *and* $x_k, x_j \notin S$,
$v(S \cup \{x_k\}) = v(S \cup \{x_j\})$ *then* $\phi_k(v) = \phi_j(v)$
Axiom 3 (Dummy)**.** *If,* $\forall S \in \mathcal{P}(X)$, *and* $x_j \notin S, v(S \cup \{x_j\}) = v(S)$, *then* $\phi_j(v) = 0$
Axiom 4 (Additivity)**.** *For all pair of games* $v, w : \phi_j(v) = \phi_j(w)$
where the game is defined by $(v + w)(S) = v(S) + w(S) \quad (\forall S)$.

Using all these concepts, a formula for the Shapley value can be derived via Theorem 1.
Theorem .1. *There exists a unique solution that satisfies the Axioms 1-4 and this is the Shapley value* $\phi^{(i)}$*, whose equation is given for all* $j \in \{1, .., p\}$*by*

$$\phi_j^{(i)}(v) = \sum_{S \subseteq \{x_1^{(i)}, ..., x_p^{(i)}\} \setminus \{x_j^{(i)}\}} \frac{\mid S \mid !(p - \mid S \mid -1)!}{p!} (v(S \cup \{x_j^{(i)}\}) - v(S))$$

This captures the average marginal contribution of $x_j^{(i)}$ to the prediction $\hat{f}(x_j^{(i)})$for a coalition S, denoted by $(v(S \cup \{x_j^{(i)}\}) - v(S))$ and computed over all possible ordered coalitions S of features built up from the empty set. Indeed, for a given S, the marginal contribution of $x_j^{(i)}$ is weighted by the product of $\mid S \mid$, the numbers of ways in which the set S could have been formed prior $x_j^{(i)}$ is added, and $(p - \mid S \mid -1)!$, the number of ways to arrange the remaining features, which is then averaged over all possible orderings $p!$ of the features. Finally, we sum over all possible subsets $S \in X \setminus \{x_j^{(i)}\}$, which comes back overall to considering the order in which features are added to the coalition. Note that the feature of interest is treated specifically: it is not included in S and is excluded from its complement.

This formula can be rewritten as follows:

$$\phi_j^{(i)}(v) = \frac{1}{p} \sum_{S \subseteq \{x_1^{(i)},...,x_p^{(i)}\} \setminus \{x_j^{(i)}\}} \binom{\mid p \mid -1}{\mid S \mid}^{-1} (v(S \cup \{x_j^{(i)}\}) - v(S))$$

$$= \frac{1}{\text{all possible coalition sizes}} \sum_{\text{coalitions excluding } x_j^{(i)}} \frac{\text{marginal contribution of } x_j^{(i)} \text{ to coalition}}{\text{number of coalitions this size}}$$

All possible coalitions of features have to be evaluated, with and without the feature of interest for calculating the exact Shapley value $\phi_j^{(i)}$. For more than a few features, the exact solution to this problem becomes intractable, because the number of possible coalitions increases exponentially by adding more features. Strumbelj et al. (2014) [44] suggest an approximation with Monte-Carlo sampling:

$$\phi_j^{(i)} = \frac{1}{M} \sum_{m=1}^{M} (\hat{f}(x_{+j,m}^{(i)}) - \hat{f}(x_{-j,m}^{(i)}))$$

For each m, $\hat{f}(x_{+j,m}^{(i)})$ is the prediction of a new instance which is constructed using two data points, namely $x^{(i)}$ and z, where z is a random instance of the dataset. Indeed, each component of $x_{+j,m}^{(i)}$ is drawn at random from either $x_j^{(i)}$ or z, except the value for x_j, which is taken from $x^{(i)}$. We construct $x_{-j,m}^{(i)}$ identically except that this time, the value of x_j for $x_{-j,m}^{(i)}$ is taken from z. This approximation, for simplicity, does not capture the average marginal contribution of feature x_j by averaging over all the different sequences according to which the grand coalition could be built up from the empty coalition. Instead, it simply uses a difference between two predictions based on a random assignment of values and repeats this process M times, for a fixed number of M, before taking the average of this prediction difference.

To build an approximate Shapley value estimation algorithm, it might be useful to consider the Shapley value under a new angle. Another way to see it is as an operator that assigns an expected marginal contribution to each feature for a particular instance with respect to a uniform distribution over the set of all permutations $\pi(X')$ from $X' = \{1, ..., p\}$ to X'. Specifically, let $\psi \in \pi(X')$. For each feature of interest x_j we denote by $Pre^j(\psi) = \{x_k : \psi(k) < \psi(j)\}$ the set of features preceding x_j in the permutation ψ. The marginal contribution of x_j with respect to that permutation ψ is $v(Pre^j(\psi) \cup \{x_j\}) - v(Pre^j(\psi))$. Now, if each of $p!$ different permutations $\psi \in \pi(X')$ are randomly chosen with equal probability, then the average marginal contribution of $x_j^{(i)}$ is:

$$\phi_j^{(i)} = \frac{1}{p!} \sum_{\psi \in \pi(X')} v(Pre^j(\psi) \cup \{x_j\}) - v(Pre^j(\psi))$$

Algorithm .3 Approximate Shapley value Estimation Algorithm

Input: number of iterations M, instance of interest x, feature of interest index j, data matrix X, p-dimensional feature space and machine learning model $\hat{f}$

for $m = 1$ to M **do**

choose a random permutation ψ of feature values, from $\{1, ..., p\}$ to $\{1, ..., p\}$

choose a random instance $z \in X$

for $k = 1$ to p **do**

$$x_k^{+j} \leftarrow \begin{cases} x_k & \text{if } x_k \in Pre^j(\psi) \cup \{x_j\}. \\ z_k, & \text{otherwise.} \end{cases} \quad \textit{and} \quad x_k^{-j} \leftarrow \begin{cases} x_k, & \text{if } x_k \in Pre^j(\psi). \\ z_k, & \text{otherwise.} \end{cases}$$

end for

$x^{+j} \leftarrow (x_1^{+j}, ..., x_p^{+j})$ and $x^{-j} \leftarrow (x_1^{-j}, ..., x_p^{-j})$

$\phi_{j,m} \leftarrow \hat{f}(x^{+j}) - \hat{f}(x^{-j})$

end for

$\phi_j \leftarrow \frac{1}{M} \sum_{m=1}^{M} \phi_{j,m}$

Output: Shapley value for the feature x_j at the i^{th}-instance.

We have now reached the end of this subsection and the end of the theory concerning interpretation methods. Before moving on to their application on a real-world example, I invite those who struggled to understand Shapley value to have a look at Appendix D, where a summary in simpler terms is provided.

4.7 Application

Explaining carefully how the analysis of this use-case is conducted and how all results are obtained is made impossible by the complexity of the task together with the limited length of the paper. Again, you can refer to the Github link to access the full analysis and coding files. Here, I would like to propose a sort of more general guide to interpretability. This involves describing and comparing what each method allows to do along with the degree of importance of the information provided, the objective being to identify the combination of methods that yields the best possible understanding of the problem. I shall start with four basic techniques that are already widely used in Data Science, namely data visualisation, feature interaction, feature contribution and feature importance. Their use and perception are facilitated by the choice of XGBoost instead of ANN and the regularisation applied to limit the maximal depth of each tree. I shall then continue with the advanced model-agnostic tools detailed in previous subsections. Note that the order in which methods are regarded matters.

Data visualisation should be the first one to be considered as it is key to understand the data and the essence of the problem regarded. In particular, a clearer idea of the

existing relationship between customer churn and a predictor's value can be achieved through a multitude of graphs representing different aspects of the data. This provides a good intuition concerning what covariates are great predictors for customer churn. While data visualisation naturally includes feature correlation, it often does not propose **feature interaction**, which helps determine if the effect of a predictor on churn could, in fact, be dependent on another predictor's value. Checking both the correlation matrix and Friedman's H-statistic (see Appendix D) is often necessary to refine our understanding of the data and adapt correspondingly the future interpretation of our results.

Then, **feature contribution**, although it not explicitly stated, involves getting general insights on both the true structure and the inner workings of the model. In this case, it consists in plotting some trees used by XGBoost. Despite the short depth of each tree that make them easily interpretable, the functioning of the algorithm and the total number of trees utilised make it impossible to clearly understand how the model works. However, viewing a subsection of those trees enables to capture some core interactions between combinations of predictors and churn. In this case, we mean associations of covariates with specific values that lead to a particular probability of churn. For instance, if a customer has a month-to-month contract and a fiber optic internet service but is not subscribed to online security, its probability of churn is likely to be rather high. On a similar note, if spurious correlations are spotted, our model might be wrong and we might need to improve it. Another benefit of feature contribution is that observing which variables are often used for a split takes our understanding of the model's inner workings further as we are now aware of the covariates that matter when computing a prediction. For instance, *Contract_Month-to-month* was chosen as the first split in many trees, which suggests its high importance in our classification problem.

This notion is taken up in a more rigorous manner by 'weight' feature importance, which is the default choice to measure **feature importance** in XGBoost. However, it presents some clear limits, which is why two other ways to measure feature importance for XGBoost are also regarded, namely 'gain' and 'cover' as well as the model agnostic method called permutation feature importance. I assume that you know how each of them works but if you need a general reminder on feature importance, since it is a basic but essential method, I provide some key knowledge in Appendix D. Combining these four techniques and the data visualisation part allows to draw more accurate conclusions concerning feature importance. Doing it rigorously and weighting these methods by their reliance yield a more objective ranking.

Now that we know what features are important in our model, the next step consists logically in finding out what their effect on the probability of churn is. This is achievable using **Partial Dependence Plot** and **Individual Conditional Expectation** together, or alternatively **Accumulated Local Effects**. Having already investigated feature importance allows focusing directly on the most essential variables, which is quite useful

when the feature space is large. Note that for these methods, as for LIME and Shapley value, we are considering the predicted probability of churn computed by XGBoost before the final classification decision.
Let us first focus on PDP and ICE, whose analysis is absolutely essential to understand in more depth how does the model globally behaves. Indeed, PDP yields extremely rewarding information about the model functioning via a rather accurate measure of how each predictor influences the probability of churn (on average) as its value changes. Coupling it with ICE allows spotting heterogenous behaviour among instances, which improves further our understanding of each covariate's effect. The study of two-dimensional PDP attains a similar target by procuring knowledge concerning interaction effects. In the customer churn example, this approach is also key to avoid confusion regarding the categorical predictor $PaymentMethod$. Although each of the four ways of payment is very significant, studying attentively their PDP enables to spot that their combined effect often cancels out, making three of these dummies nearly insignificant to predict churn. Note that to be able to notice that, it is implicitly suggested that multiple-category categorical variables need to be investigated both as a single covariate and split into several dummies.
ALE yields nearly identical results and should be viewed as an alternative to ICE and PDP. It should be preferred to ICE and PDP if there exist strong correlations between covariates according to the feature correlation part.

I would not recommend to use a **global surrogate model** to improve the understanding of the inner workings of the model as it often entails an important trade-off between precision and simplicity. However, it can still turn out to be precious for the company. Since the surrogate model for XGBoost is obviously a Decision Tree, this approach can help the company target different strategies (marketing, pricing, other) to retain customers with respect to a relevant population segmentation. Distinct policies could indeed be applied for each leaf or group of leaves of the Decision Tree, which should be formed by taking into account the number of customers regarded, their common characteristics and the corresponding probability of churn at each leaf. This analysis necessitates an already good knowledge of the model if the company wants to target distinct relevant strategies per cluster in order to retain customers. Such approach could definitely help the company increase its revenues.

Until now, I have been mainly preoccupied with explaining the model functioning from a global perspective. Only the previous method focuses on groups of customers rather than on the entire model. Let me now delve into even more 'local' approaches and tackle the explanation of single predictions.

The most famous method for this type of assignment is **LIME**, which principally aims to answer the question "why should a prediction be trusted?" via extremely intuitive and easily interpretable results. An important distinction that needs to be made is whether to apply it to the test set or the training set.

When applied to the former, LIME especially serves in the building phase of the model as a complement to previous approaches to verify our findings and check model significance. Although LIME can only explain one instance at a time, SP-LIME selects a few instances (representative of the dataset) to look at in order to get a global overview of how the model behaves. For each instance, we check whether the prediction is consistent with the customer's profile and the results of previous analyses concerning the effect of each covariate on churn. In this case, if the company wishes, the interpretation of the model could be made clearer by deleting two of the four payment method dummies. This builds on the ALE analysis, where it was first shown that including all four variables sort of confuses the comprehension of their overall effect on churn. The company needs solely to accept to lose a tiny bit of performance in exchange for this gain of simplicity.
When applied to the test set, LIME can be used by the company in real time for diverse purposes. A potential application we could think of arises when a customer calls Telco's help desk. The employee could advise him to subscribe to new offers in accordance with the LIME explainer output with the aim of diminishing his predicted probability of churn. For instance, we could encourage the customer to switch to a longer contract if he has been here for a few months already with a month-to-month contract; or to change from electronic check to credit card payment; or to subscribe to additional services via a discount. This potential utility is comparable to the one of the global surrogate model approach but the range of application of these personalised recommendations slightly differ. As a result, both methods could be relevantly applied together. On the same note, remember that LIME can limit the number of features used to build the surrogate model, which along with feature importance and feature interaction, can help us produce more selective explanations. Finally, note that similar to the training set case, formulating these personalised recommendations requires prior knowledge about the model and in particular about each covariate's effect on churn.

The main limit of LIME is that each covariate's effect has no true signification, it is rather intuitive, which introduces the need for a more scientific method such as **Shapley value**. It also aims to explain single predictions and achieve very similar results to LIME. However, Shapley value has the ability to do so in a more scientific and meticulous way, based on a strong theory. In a word, it assigns to each feature its fair contribution towards the predicted probability of churn, making the explanation extremely meaningful. This ability can turn out to be essential in some applications, such as the acceptation of loan demands in the banking field, which now require a 'right to explanation' by the GDPR.
Similarly to LIME, it can either be applied on the training or test set given the desired utility mentioned in the LIME paragraph, and can be generalised to a more global scale approach, meaning where results apply for the entire dataset. Unlike LIME, it offers a wide range of possibilities that encompasses the role of feature importance, PDP and ICE (or ALE) and feature interaction. More precisely, as PDP and ICE, Shapley value is able to create dependence plots giving rewarding insights about the change in the probabil-

ity of churn when the feature of interest's value varies. Although slightly less intuitive than them, it yields identical results and thanks to some special options and visualisation tools, it enables a deeper analysis of both heterogeneous effects and interaction effects between covariates. It is also possible to obtain both an intuition and a measure of feature importance, where the latter is a sort of more complex version of permutation feature importance[11]. Here again, results are relatively consistent with the previous feature importance analysis. Lastly, unlike LIME, it is optimised to trace through XGBoost trees to find the Shapley value estimate.

As a result, from all conclusions drawn in previous parts, it seems that a good understanding of the model is obtained by starting with data visualisation and feature contribution. Then, we could either implement feature importance and interaction, ICE, PDP and LIME or choose to use Shapley value instead. While the latter is more adequate for cases of crucial importance where the 'right to explanation' applies, the former is more suitable for our customer churn use case, where we prefer something more intuitive. Lastly, note the global surrogate model could turn out to be useful. Therefore, in both cases the analysis provides

- an understanding of both the data and the essence of the problem
- an intuition about the model's structure and behaviour
- an importance measure and the exact effect of a feature on the probability of churn
- many potential applications. In this use-case, it includes insights for targeted strategies to make customers stay or real-time personalised offers.
- the possibility to check model significance and improve it from a global and local perspective.

5 Survival analysis

Although the approach followed in Chapter 4 is often perfectly sufficient, the time aspect involved in customer churn analysis often requires particular attention. Indeed, we have been focused on predicting if a customer would churn at the end of the month, which is an immediate problem. However, it is just as important for companies to consider long term factors such as when, why and with what probability is a customer going to churn. Indeed, having knowledge about customers' life cycle, the optimal time to intervene and the ideal way to deal with each type of customer is crucial for companies to plan effective long term strategies for decreasing the customer churn rate. Such useful information can be extracted via survival analysis, which is presented below. Since the theory is wide, I cannot

[11] see Appendix D

cover it entirely. This paper therefore emphasises its core aspects, developing especially the material of interest for conducting properly a general case-study. [45, 46, 47, 48] were the main resources utilised for this section.

5.1 Survival Analysis Theory

Survival analysis is a collection of statistical procedures for data analysis, for which the outcome variable of interest is time until an event occurs. Traditionally, only a single event occurs for each subject, after which the organism or mechanism is dead or broken. This is in contrast with approaches that model the probability of an event, such as regression or boosting. The term survival analysis came into being from initial studies, where the event of interest was death. Now, its scope has become wide; encompassing events like criminal recidivism, stock market crash, equipment failure or customer churn.

For simplicity, I will adopt the terminology of survival analysis, referring to the event of interest as 'death', the waiting time as 'survival time' and a unit of interest as an 'individual'.

5.2 The Survival function and the Hazard function

Let T be a non-negative continuous random variable representing the waiting time until the occurrence of an event. Let $f(t)$ be its associated probability density function and $F(t) = \mathbb{P}(T < t)$ its cumulative distribution function, giving the probability that the event has occurred by duration t. It will often be convenient to work with the complement of the c.d.f, the survival function:

$$S(t) = \mathbb{P}(T \geq t) = 1 - F(t) = \int_t^\infty f(t)dt \tag{5}$$

which gives the probability of being alive just before duration t, or more generally, the probability that the event of interest has not occurred by duration t.

Let μ denote the mean or expected value of T, that is the expectation of life. By definition, one would calculate μ multiplying t by the density $f(t)$ and integrating, so

$$\mu = \int_0^\infty tf(t)dt = \int_0^\infty S(t)dt$$

The last equality is obtained by integrating by parts and making use of the fact that $-f(t)$ is the derivative of $S(t)$, which has boundary conditions $S(0) = 1$ (since the event has not occurred by time 0) and $S(\infty) = 0$ (everyone dies eventually).

For a given dataset, the survival function is most commonly approximated by the Kaplan-

Meier estimator, that is

$$\hat{S}(t) = \prod_{i:t_i \leq t} (1 - \frac{d_i}{n_i})$$

where t_i is the time (at which there is at least one death), d_i the number of deaths observed at time t_i and n_i the number of individuals at risk at time t_i. Note that $\hat{S}(t) = 1$ on $[0, t_1)$ and $(1 - \frac{d_i}{n_i})$ provides the conditional probability to surviving past time t_i given survival up to that time.

An alternative characterization of the distribution of T is given by the hazard function, or instantaneous rate of occurrence of the event, defined as

$$\lambda(t) = \lim_{dt \to 0} \frac{\mathbb{P}(t \leq T < t + dt \mid T \geq t)}{dt}$$

The numerator of this expression is the conditional probability that the event will occur in the interval $[t, t + dt)$ given that it has not occurred before. The denominator is the width of the interval. Dividing one by the other yields a rate of event occurrence per unit of time. Taking the limit as the width of the interval goes down to zero, we obtain an instantaneous rate of occurrence.
The conditional probability in the numerator may be written using Bayes' rule as the ratio of the joint probability $\mathbb{P}(t \leq T < t + dt, T \geq t) = \mathbb{P}(t \leq T < t + dt)$ to the probability $\mathbb{P}(T \geq t)$. The former may be written as $f(t)dt$ for small dt, while the latter is $S(t)$ by definition. Simplifying by dt and passing to the limit gives the useful result

$$\lambda(t) = \frac{f(t)}{S(t)} \tag{6}$$

which some authors give as a definition of the hazard function. In words, the rate of occurrence of the event at duration t equals the density of events at t, divided by the probability of surviving to that duration without experiencing the event.
Since $-f(t)$ is the derivative of $S(t)$ from equation (5), this can be written as

$$\lambda(t) = -\frac{d}{dt} \log S(t)$$

If we now integrate from 0 to t and use the boundary condition S(0) = 1, we can solve the above expression to obtain a formula for the probability of surviving to duration t as a function of the hazard at all times up to t, also called the cumulative hazard function Λ

$$S(t) = \exp(-\int_0^t \lambda(x)dx), \quad \text{where} \quad \Lambda(t) = \int_0^t \lambda(x)dx \tag{7}$$

You may think of $\Lambda(t)$ as the sum of the risks of death you face going from duration 0 to t, that is the expected number of deaths that have occurred by time t. It can be approached

by the Nelson-Aalen estimator.

These results show that the survival and hazard functions provide an alternative but equivalent characterizations of the distribution of T. Given the survival function, we can always differentiate to obtain the density and then calculate the hazard using Equation (6). Given the hazard, we can always integrate to obtain the cumulative hazard and then exponentiate to obtain the survival function using Equation (7).

The simplest possible survival distribution is obtained by assuming a constant risk over time. So the hazard is $\lambda(t) = \lambda$ for all t. The corresponding survival function is $S(t) = \exp(-\lambda t)$. This distribution is called the exponential distribution with parameter λ. The density may be obtained multiplying the survivor function by the hazard to obtain $f(t) = \lambda \exp(-\lambda t)$. The mean turns out to be $\frac{1}{\lambda}$. This distribution plays a central role in survival analysis, although it is probably too simple to be useful in applications in its own right.

5.3 Censoring

The second distinguishing feature of the field of survival analysis is an essential element called censoring. It happens when there are incomplete observations; meaning we have some information about individual survival time, but we do not know the survival time exactly. Failure to take censoring into account can produce serious bias in estimates of the distribution of survival time and related quantities. To understand how does this occur, let's come back to the way survival data is recorded. Subjects are observed for a certain period of time during which the time of the event of interest is registered. However, it is possible that the event cannot be registered because it does not occur during the period of observation. Three forms of censoring can be distinguished:

- right censoring: the event time is larger than the period of observation. In other words, a subject leaves the study before an event occurs, or the study ends before the event has occurred.
- left censoring: the event occurs before the period of observation. This is rarely met.
- interval censoring: the exact event time is not available, and is only considered to be in a certain interval.

Right censoring is the most common case and can be of type I (a fixed time for the end of the study is set) or of type II (the study ends when a fixed number of events amongst subjects has occurred). A simple generalisation of these terms yields respectively fixed censoring and random censoring. For the former, each subject has a maximum observation time, fixed in advance. Concerning random censoring, the censoring time of each subject is independent of its lifetime. We observe the minimum of the censoring and lifetimes as well as an indicator variable that tells us whether observation terminated by death or by censoring.

5.4 Cox proportional hazard model

Up to this point, we have been concerned with a homogeneous population, where the lifetimes of all units are governed by the same survival function $S(t)$. We now introduce the third distinguishing characteristic of survival models, that is the presence of a vector of covariates that may affect survival time.

In this approach, we are particularly interested in survival models that incorporate a regression component, since these regression models can be used to examine the influence of explanatory variables on the event time. In particular, we focus on the most popular family of models, introduced by Cox in 1972, which investigate the effects of explanatory variables directly on the hazard function. The simplest member of the family is the proportional hazards model, which is constructed on several key assumptions. First, the survival function is an exponential function. Second, survival times between distinct individuals in the sample must be independent. Third, there should be a linear relationship between the log hazard and each covariate. Lastly, the hazard ratio[12] for any two compared groups is constant throughout the study period. This last requirement, called the proportional hazards assumption, is extremely important. Formulated differently, it states that for any two subjects $x^{(i)}$ and $x^{(j)}$, the expression below is constant over time.

$$\frac{\lambda_i(t)}{\lambda_j(t)} = \frac{\lambda_0(t)\exp(x^{(i)})}{\lambda_0(t)\exp(x^{(j)})} = \frac{\exp(x^{(i)})}{\exp(x^{(j)})}$$

Using the results derived at the end of 5.1.2, the hazard at time t for the i^{th}individual is assumed to be

$$\lambda_i(t \mid x^{(i)}) = \lambda_0(t)\exp(x_1^{(i)}\beta_1 + ... + x_p^{(i)}\beta_p) = \lambda_0(t)\exp(x^{(i)} \cdot \beta) \tag{8}$$

In this model $\lambda_0(t)$ is a baseline hazard function, that is an unspecified non-negative function describing how the risk of the event (at time t) changes over time at baseline levels of covariates, meaning $x^{(i)} = 0$. Since it can take any form, the Cox model is semi-parametric and has a significant advantage over parametric survival models, which are restricted by the shape of a particular distribution. The hazard rate at time t is thus the product of the baseline hazard at time t and the effect parameters, $\exp(x^{(i)}\beta)$, representing the relative risk (a proportionate increase or reduction in risk) associated with the covariates' values $x^{(i)}$. To put it differently, the covariates increase or decrease the hazard function by a constant relative to the baseline hazard function. Note that the increase or reduction in risk is the same at all durations t because covariates must have the same effect on the hazard at any point in time.

[12]ratio of the hazard rates corresponding to the conditions described by two levels of an explanatory variable

Integrating on both sides from 0 to t of (8) yields the cumulative hazards

$$\Lambda_i(t \mid x^{(i)}) = \Lambda_0(t) \exp(x^{(i)}\beta)$$

which are also proportional. If we refer to (7), changing signs and exponentiating yields the survival functions

$$S_i(t \mid x^{(i)}) = S_0(t)^{\exp(x^{(i)}\beta)} \tag{9}$$

where $S_0(t) = \exp(\Lambda_0(t))$ is a baseline survival function. Thus, the effect of the covariate values $x^{(i)}$ on the survival function is to raise it to a power given by the relative risk $\exp(x^{(i)}\beta)$.

The estimation of the parameter β is based on the partial likelihood function and is performed without specifying or even estimating the baseline hazard. Since individuals are assumed to be independent, the partial likelihood $L(\beta)$ can be written as a product of several likelihoods L_i, where $L_i(\beta)$ denotes the likelihood of death at time t_i of an individual with values say $x^{(i)}$ given survival up to time t_i. This involves computing the probability that the individual i dies at time t_i divided by the probability that someone dies at time t_i among all individuals at risk, denoted by $R(t_i)$. The term partial likelihood is used because the likelihood formula considers probabilities only for the subjects who have died during the study ($i \in C$), and does not explicitly consider probabilities for those subjects who are censored. Hence, the partial likelihood may be written as

$$L(\beta) = \prod_{i \in C} L_i(\beta) = \prod_{i \in C} \frac{\lambda_i(t_i \mid x^{(i)})}{\sum_{j \in R_{t_i}} \lambda_j(t_j \mid x^{(j)})} = \prod_{i \in C} \frac{\exp(x^{(i)}\beta)}{\sum_{j \in R_{t_i}} \exp(x^{(j)}\beta)}$$

The estimated coefficients of the Cox model provide the linear, additive effects of the covariates on the log-hazard scale. Indeed, taking logs, the model separates clearly the effect of time from the effect of the covariates. This yields a simple additive model

$$log\lambda_i(t \mid x^{(i)}) = \alpha_0(t) + x^{(i)} \cdot \beta$$

where $\alpha_0(t) = \log \lambda_0(t)$ is the log of the baseline hazard. As in all additive models, we assume that the effect of the covariates x is the same at all times t. Although the sign of the coefficients is interpretable, their magnitudes are not so easily interpreted.

5.5 Extended cox model

So far we have considered explicitly only covariates that are fixed over time. The local nature of the proportional hazards model, however, lends itself easily to extensions that

allow for covariates that change over time. When we allow such extension, the separation of duration and covariate effects is not so clear, and on occasion, it may be difficult to identify effects that are highly collinear with time. Also, the calculation of survival functions is a little bit more complicated because we need to specify a path or trajectory for each variable. Equation (9) does not apply.
Time-varying covariates can be introduced in the context of accelerated life models but it is not so simple and has rarely been done in applications. One way to address this problem is lag covariates.

The model may also be generalized to allow for effects that vary over time. Usually, the form of time dependence of the effects must be specified parametrically in order to be able to identify the model and estimate the parameters. Obvious candidates are polynomials on duration, where $\beta(t)$ is a linear or quadratic function of time. Although this extension allows for great generality, by allowing regression coefficients to be dependent on time, we violate the assumption of proportional hazards. This implies losing the simple separation of time and covariate effects. Indeed, calculation of the survival function in this model is again somewhat complicated by the fact that the coefficients are now functions of time, so they do not fall out of the integral. The simple equation (9) does not apply anymore.

The foregoing extensions to time-varying covariates and time-dependent effects may be combined to give the most general version of the hazard, that is

$$\lambda_i(t, x_i(t)) = \lambda_0(t) \exp(x^{(i)}(t)\beta(t))$$

where $x^{(i)}(t)$ is a vector of time-varying covariates representing the characteristics of individual i at time t, and $\beta(t)$ is a vector of time-dependent coefficients, representing the effect that those characteristics have at time or duration t.

An alternative to incorporating interactions with time is to divide the data into strata based on the values of one or more covariates. The regression parameters are assumed to be the same across the strata, but a different baseline hazard may exist for each stratum. Stratification is useful for dealing with violations of the proportional hazard assumption, which can be detected using Schoenfeld residuals.

To check the other assumptions of the Cox model, in particular the linear functional form of covariates to express the log-hazard and the exponential form of the survival function, the use of Martingale residuals is preferred. Furthermore, a slightly transformed version of those residuals, called Deviance residuals, can be used to assess outliers[13] in the dataset. These potential problems do not appear in my use-case and will not be developed.

[13] the survival function predicts one event too early or too late

5.6 Application

The analysis conducted builds on previous subsections. Again, due to its complexity and the limited length of the project, I directly emphasise the main results. But this time, they are a bit more related to the use-case since the approach is itself more specific. They are presented as answers to some crucial questions that the company must solve to deal optimally with the customer churn problem. The full analysis is accessible via the following Github link [37].

Do we have an accurate understanding of the customer life cycle ? How does the likelihood of churn change over time ? How long do customers stay in your business on average?

The visualisation of subsets of the censored data is a great tool to understand the presented use-case and to look for interesting patterns in the customer life cycle. Here, we simply notice that a large proportion of customers churn at the very beginning of the study and in a smaller extent at the very end. Hence, a basic approach would suggest the company to focus its efforts to retain customers who have just joined the company or who have been here for a bit more than five years. On a similar note, we shall use the estimated cumulative distribution function to access the risk of churn accumulated up to time t. Its interpretation yields similar conclusions.

If we were asked to estimate the average lifetime of customers, and we naively decided to exclude right-censored individuals, we obtain 17.98 months. However, it is clear that we would be severely underestimating the true average lifespan. Furthermore, if we instead simply took the mean of all observed lifetimes, including right-censored instances, it yields 32.42. But once again, we would be underestimating the true average lifespan. We conclude that this quantity cannot be computed accurately for now. The use of the Cox regression model will make it possible, as we will see later on.

How can the company improve its services to minimise the customer retention rate ? Should we distinguish between several types of customers? Should we treat them separately?

These questions can be answered by observing the survival curves of the population when segmented with respect to a categorical variable. Below is displayed a summary of the possible improvements the company could make to increase the retention rate.

- It should focus its efforts on customers' first 10 months with the company, only for individuals presenting at least one of the following attributes: paperless billing, no dependents, no partner, no subscription to additional services (tech support, online backup, etc.), a fiber optic internet service and paying via electronic check.
- As it seems inevitable that month-to-month contractors will have a lower survival rate than longer term contractors, the company should aim to make these customers

switch to a yearly based contract. For yearly based contractors, attention should be paid around the fourth year of tenure.

- The company should try to sell its additional services, which appears to be great since customers purchasing them show a much higher probability of survival.
- It should try to improve the electronic payment check method, the quality of the internet service (especially fiber optic) and the treatment of senior citizens whose survival probability decreases with high constant rate across time.

If we do not wish to proceed attribute by attribute but under a more general framework, then we can infer from this analysis the presence of two main types of customers, roughly characterised as follows.

- Type I: has partners and dependents, long-term contract, paper billing and is subscribed to additional services.
- Type II: no partner nor dependents, month-to-month contract, no subscription to services and paperless billing.

Each group has some defining characteristics, which implies that they could and probably should be treated separately. For instance, Type I customers naturally stay a bit longer and bring back more money to the company than Type II customers. It therefore might be relevant to consider different strategies for each group, including marketing and pricing, when trying to retain customers. Moreover, the company is not bound to improving its global churn rate. It can choose to focus its churn prevention efforts on long-term high-value customers (Type I) with low survival. Delving further into Customer Lifetime Value analysis will definitely help develop efficient pricing strategies with respect to customers segment, which will not obviously result in a much better customer retention rate but will definitely lead to higher revenues.

This cohort analysis only represents a limited use case of the full potential of survival analysis because we are using it for the aggregated level of the data. There exists more advanced forms of survival analysis, such as the Cox proportional hazard regression model, that broaden the range of possibilities. In particular, it can be used to study survival on an individual basis, based on the estimated effects of covariates.

How does the survival probability of a particular individual evolve across time ? Is there a real-time version of this indicator taking into account how long he has been in the company for ? What about life expectancy ? Can we discover the optimal intervention point to apply the strategy elaborated to make him stay ?

The final Cox proportional hazard regression model is an extended Cox model, which has stratified the variables *TechSupport*, *Contract_One year*, *Contract_Two year* and *TotalCharges* because they were not satisfying the proportional hazards assumption [14].

[14] Python struggles to deal with time-varying covariates so $TotalCharges$ was not converted as such

Before answering the first question, it is important to mention that the output of the final Cox regression gives more accurate information on the covaritates' impact on churn. I will just mention a few significant ones.
The covariate *InternetService_Fiber optic* multiplies by 1.86 the hazard function for customers having purchased it compared to those who have not, making them 86% more likely to churn. Regarding covariates having an opposite effect, there is a -0.34 decrease in the expected log of the relative hazard for each customer having no internet compared to those who have it, holding all other covariates constant. This is equivalent to a 40% decrease in the expected hazard for customers with no internet service, which is traduced by being 40% less likely to churn. Lastly, a one unit increase in monthly charges yields a 2.7% increase in the probability of churn. A more accurate view of continuous features' impact on churn can be achieved by plotting general survival functions (all individuals) when the customer base is segmented with respect to some pre-defined values of the covariate.

Using the coefficient on each covariate along with the attributes of a particular customer yields the survival function of this customer. Its study is a very useful application of the Cox model as it conveys essential information concerning how long a customer will stay in the business and about the optimal intervention point to make a customer stay. Indeed, knowing the estimated survival probability (or churn probability) at any point in time would help the company target and time the ideal strategy for retaining customers. The company could typically pick a median, a quartile or a percentile as a best guess for a subject's churn time and act just before it according to the chosen strategy. The statistics choice can be refined based on experience.
The up-to-date version of this analysis consists of evaluating the survival function of censored individuals given their up-to-date lifetime in the company, which involves the use of conditional probabilities. It yields a real-time analysis for existing customers and therefore contains precious information for the company.
Since the Cox regression model is able to estimate individual survival curve for every customer, including censored observations, we gain access to a life expectancy measurement for each individual. It can easily be traduced into a global life expectancy, that is 47.02 months, representing the average survival time of a customer in the company. This information might be useful to plan long-term strategies. Note that the result is perfectly consistent with the two previous underestimated computations of life expectancy.

6 Conclusion

After showing how essential understanding the machine learning models governing our world is and mentioning the literature's lack of unified scientific view to answer the problem, this paper positions itself as a reference guide to interpretability. In addition to providing a key overview of this concept, it presents a rigorous mathematical definition of the main existing interpretation methods. It also shows how to implement, interpret and combine them on a real-world example (see Github link).

This task is conducted to guide future researches and to help companies get more easily and efficiently a better understanding of a given problem. Indeed, I have shown that combining in a specific way different interpretation techniques enables a good comprehension of the data, the inner workings of the model and the problem. As we still are at an early stage of interpretability, this is already quite promising.

Although already advanced, this understanding can be completed by studying the problem from a new angle. It often requires investigating new approaches and really depends on the case study considered. In this paper, we chose the mathematical modelling technique named survival analysis to emphasise the time component, which is essential to the problem regarded but slightly neglected by the boosting classification model. This original approach provided the company with some brand new key information, especially concerning the customer life cycle, giving it the opportunity to refine its strategy to retain customers.

I would like to emphasise the use of Python for this project. Despite being probably the most widely used programming language for DataScience, it has until now rarely been employed for interpretability. As a result, most limits of the use-case analysis coincide with Python's limitations, which I could not always get around. Efforts should be concentrated on improving its libraries to allow all possibilities suggested by the theory. This especially concerns example-based methods, which despite producing rewarding human-friendly explanations, are not even implemented. On the same note, with more time, this paper would have liked to compare the implementation in Python and in R. This could offer to future projects both more insights and possibilities regarding the choice of the programming language. Furthermore, future researches should target different machine learning algorithms and different data types, in particular deep learning architectures and images or texts. Being able to interpret those constitute the next step towards building more advanced AI, which should be able to explain its decisions as we humans do. This is a big step compared to the current state of things that involves the ability to produce human-friendly explanations from a wide variety of models. Such elaborated task would require progress in a multitude of fields, including interpretability.

A Appendix: Accuracy vs Explainability

Below is displayed a sort of intuitive graph of the different machine learning algorithms, whose position is determined by their accuracy and their explainability. This is not accurate and scientific, it should solely serve the reader as an intuition.

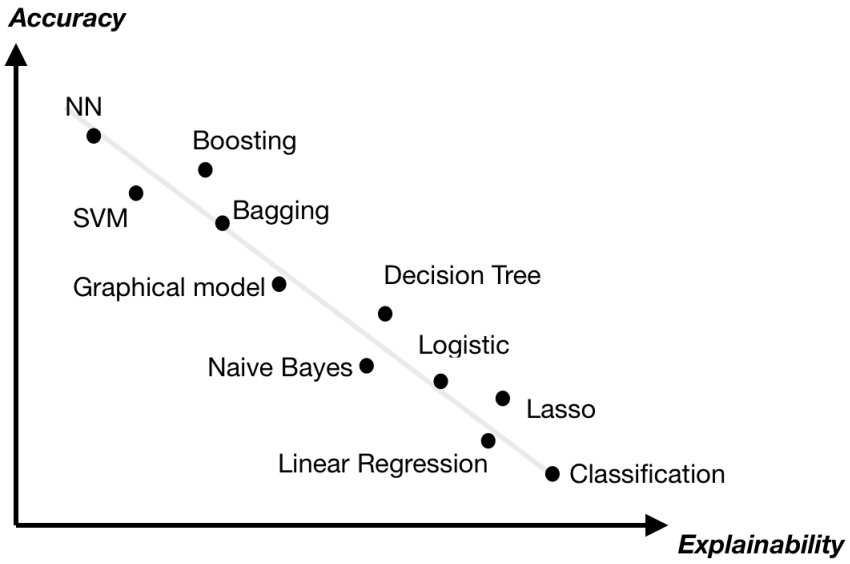

Figure 1: Accuracy vs Explainability of the main machine learning algorithms

Note that 'NN' denotes Neural Networks, 'SVM' stands for Support Vector Machine and 'classification' denotes classification rules.

B Appendix: XGBoost

To gain a better understanding of XGBoost, it is necessary to understand the GBM and gradient tree boosting algorithms, on which it is built [49].

Gradient Boosting Machine (GBM) algorithm

- Input: training data $\{(x_i, y_i)\}_{i=1}^n$, differentiable loss function $L(y, f(x))$, n observations and M iterations (number of predictors applied).
- Initialise with a constant value: $f_0(x) = arg\min_\gamma \sum_{i=1}^n L(y_i, \gamma)$
- For all $m = 1$ to M:
  1. Compute pseudo residuals by calculating the negative gradient:
     $r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$ for $i = 1, ..., n$
  2. Fit a base learner $h_m(x)$ to pseudo residuals using training data $\{(x_i, r_{im})\}_{i=1}^n$

3. Choose a gradient descent step size as
   $\gamma_m = arg\min_\gamma \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + \gamma h_m(x_i))$.
4. New model: $f_m(x) = f_{m-1}(x) + \gamma_m h_m(x)$

- Output $f_M(x)$

Gradient Tree Boosting algorithm (GBT)

- Input: training data $\{(x_i, y_i)\}_{i=1}^{n}$, differentiable loss function $L(y, f(x))$, n observations and M iterations (number of predictors applied).
- Initialise with the constant value: $f_0(x) = arg\min_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$
- For all $m = 1$ to M:
  1. Compute pseudo residuals by calculating the negative gradient:
     $r_{im} = -\left[\dfrac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$ for $i = 1, ..., m$
  2. Fit a decision tree $h_m(x)$ to pseudo residuals using training data $\{(x_i, r_{im})\}_{i=1}^{n}$
  3. Let T be the number of leaves of h_m. Each region (leaf node) is defined by R_{jm} for $j = 1, ..., T$. Determine the structure $\{R_{jm}\}$ by selecting the splits that maximise *gain*, where $gain = \dfrac{1}{2}\left[\dfrac{G_L^2}{n_L} + \dfrac{G_R^2}{n_R} - \dfrac{G_{jm}^2}{n_{jm}}\right]$ (see greedy algorithm for split finding).
  4. Compute predictions $h_m(x) = \sum_{j=1}^{T} b_{jm} 1_{R_{jm}}(x)$ where b_{jm} denotes the value predicted in each region R_{jm}
  5. Compute leaf weights $\{w_{jm}\}_{j=1}^{T}$ for the learnt structure (for each leaf). We choose separate optimal value for each region:
     $\forall j \in \{1, ..., T\}, \quad w_{jm} = arg\min_w \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + w)$.
  6. New model: $f_m(x) = f_{m-1}(x) + \eta \sum_{j=1}^{T} w_{jm} 1_{R_{jm}}(x)$. b_{jm} can be discarded and $\eta \in [0, 1]$ is the learning rate, it is used to apply shrinkage in XGBoost.
- **Output:** $f(x) = f_M(x)$

In the 'Exact greedy algorithm for split finding', g_i denotes the first order gradient and h_i denotes the second order gradient (for an instance index $i \in I$).
This algorithm enumerates over all possible splits on all features to find the best one. In other words, it enumerates over all features and uses unique values of each feature as candidate split points. It then chooses the split that maximizes the splitting criterion (based on gain in this case), which is computationally demanding. In order to so efficiently, it must first sort the data according to feature values and visit instances in sorted order to accumulate the gradient statistics greedily.

Algorithm .4 Exact greedy algorithm for split finding

Input: I, set of instances at a node
Input: d, dimension of the feature space
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ to m **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  for j in sorted (I, by $x_j^{(k)}$) **do**
    $G_L \leftarrow G_L + g_j, \quad G_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L, \quad H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L + \lambda} + \frac{G_L^2}{H_R + \lambda} - \frac{G^2}{H + \lambda})$
  end for
end for
Output: Split with max score

There exist two variants to the 'Exact greedy algorithm for split finding', namely 'Approximate algorithm for split finding' and 'Sparsity aware split finding'. They are all described in the XGBoost paper [36].

C Appendix: Model construction

This dataset contains 3 continuous features, 7 dummy variables and 11 categorical variables (not binary). Here is a quick description of them.

customerID - Customer ID
gender - Whether the customer is a male or a female
SeniorCitizen - Whether the customer is a senior citizen or not (1, 0)
Partner - Whether the customer has a partner or not (Yes, No)
Dependents - Whether the customer has dependents or not (Yes, No)
tenure - Number of months the customer has stayed with the company
PhoneService - Whether the customer has a phone service or not (Yes, No)
MultipleLines - Whether the customer has multiple lines or not (Yes, No, No phone service)
InternetService - Customers internet service provider (DSL, Fiber optic, No)
OnlineSecurity - Whether the customer has online security or not (Yes, No, No internet service)
OnlineBackup - Whether the customer has online backup or not (Yes, No, No internet service)
DeviceProtection - Whether the customer has device protection or not (Yes, No, No internet service)

TechSupport - Whether the customer has tech support or not (Yes, No, No internet service)
StreamingTV - Whether the customer has streaming TV or not (Yes, No, No internet service)
StreamingMovies - Whether the customer has streaming movies or not (Yes, No, No internet service)
Contract - The contract term of the customer (Month-to-month, One year, Two year)
PaperlessBilling - Whether the customer has paperless billing or not (Yes, No)
PaymentMethod - The customer's payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))
MonthlyCharges - The amount charged to the customer monthly
TotalCharges - The total amount charged to the customer
Churn - Whether the customer churned or not (Yes or No)

Data manipulation

Note that the original data was slightly modified to facilitate its comprehension while visualising it; without real consequences. In particular, I replaced the event 'No internet service' by 'No' for the following columns: *OnlineSecurity*, *OnlineBackup*, *DeviceProtection*, *TechSupport*, *StreamingTV* and *StreamingMovies*. It follows because they all are conditioned on the categorical feature *InternetService*, where the category 'No' represents a small proportion of the customer base. Similarly, we change the value 'No phone service' to 'No' in the *MultipleLines* column. Lastly, all three multiple categorical variables were encoded as dummy variables.

There are only 11 missing values, all of them for the *TotalCharges* feature. They affect exclusively customers with zero tenure, one-year or two-year contract and monthly charges strictly greater than 0. It is possible that these consumers have never paid what they owe to the company. These observations are considered to be involuntary churn and therefore, as mentioned earlier, will be excluded from the analysis. No outliers are found (for continuous variables), meaning that the quasi-totality of the dataset is considered.

Unbalanced repartition of the dataset

The main difficulty faced in this phase is the unbalanced repartition of the dataset. From the data visualisation part, there are indeed approximately 73 % of non-churners. Hence, if the model was predicting that every customer would stay, which is wrong, it would still achieve a 0.73 accuracy. Although it is not that accentuated, and given that the positive class defines churners, there exists many more Type II errors[15] than Type I errors[16] (see confusion matrix). This implies that the telecom company fails to identify many customers that are about to churn, which is highly undesirable because its purpose

[15] you predict that a customer will stay while he actually churns, called False Negative
[16] you predict the customer will churn while he actually does not, also called False Positive

is to retain existing customers. To limit this type of error, one solution would be to set a different discrimination threshold[17]. From graph X, a threshold of 0.36 would be optimal and would yield better results. This means that if a customer is likely to churn with probability higher or equal to 0.36 according to our model, it is classified in the churn category. Another solution, which we prefer, is to use SMOTE to oversample the minority class of our training dataset and train the model on this new data. This approach is detailed below and is applied in this paper. SMOTE, by increasing the percentage of churn instances to 50%, augments the number of Type I errors and reduces the amount of Type II errors while maintaining similar performance. The new confusion matrix reveals the trade-off operated between false negative (FN) and false positive (FP) errors, which is exactly what we want. In this case, the optimal threshold is approximately back to 0.5. We are not interested in further reducing the number of Type II errors via a smaller threshold not only because it will hurt performance but also because we cannot tolerate too many Type I errors. Although we do not have enough information on that, the firm might try to retain churners via some promotional offers. Making a large number of those to customers that were not about to leave the company would cause losses to the company. Modifying the proportion of Type I/II errors further therefore depends on the firm's strategy regarding churn customers.

	Predictive class	
True class	churn	non-churn
churn	211 (TP)	170 (FN)
non-churn	98 (FP)	928 (TN)

Table 2: Confusion matrix of initial model

Comparing this confusion matrix to Table 1 (core project) is very interesting as it illustrates the trade-off between false negative and false positive errors that we just mentioned. While performance metrics are very similar for both models, our predictions for the final model are a bit less accurate for non-churners but more accurate for churners, compared to the initial model.

SMOTE

I first looked at the class unbalanced problem in general. In particular, I found [50] useful. I then focused on SMOTE (Synthetic Minority Oversampling), whose pseudo-code [51] (originial paper) is explained below:

1. Split between training and test set; focus solely on the training set.
2. Randomly pick a point from the minority class (churn).
3. Compute the k-nearest neighbours for this point (for some pre-specified k).
4. Introduce synthetic examples along the line segments joining each neighbour and

[17] minimal probability at which the positive class (churn) is chosen over the negative class. By default, it is equal to 0.5

the point of interest.

5. Depending upon the amount of over-sampling required, the synthetic examples obtained from the k-nearest neighbours are randomly chosen.
6. The test set is left unchanged

In fact, we use a slightly different version of SMOTE, called SMOTENC (NC for Numerical Continuous), which simply enables us to deal with both categorical and numerical variables. Otherwise, the newly created samples would attribute new numeric values to our categorical features. Note that we cannot cross-validate after using SMOTE. Indeed, the results would not be relevant as some newly created instances sent to the validation set would have extremely similar instances in the training set, which induces the presence of overfitting.

Feature selection

Finally, no new attribute was added and no feature was deleted by RFECV. RFECV is similar to RFE but suppresses features automatically in a recursive fashion through a cross-validated selection of the best number of features, with respect to the Area Under the Curve metric.
Recursive feature elimination (RFE) is a feature selection method that fits a model and removes recursively the weakest feature, which is defined by its importance in the model.

Formulas of the performance metrics used

$$Accuracy = \frac{TP + FP}{TP + FP + TN + FN}, \quad f1 = \frac{Precision + Recall}{2}$$

$$Precision = \frac{TP}{TP + FP}, \quad Recall = \frac{TP}{TP + FN}, \quad Specificity = \frac{TN}{TN + FP}$$

The ROC curve (Receiver Operating Characteristic) plots *Recall* against $Specificity$. The area under this curve, called AUC, is equal to the probability that a classifier ranks a randomly chosen positive instance (churn class) higher than a randomly chosen negative one assuming 'positive' ranks higher than 'negative', meaning the customer truly churns.

D Appendix: Interpretation methods

Feature interaction

It measures how much variation of the predicted outcome depends on the existing interaction between features. It gives an indication about how strongly one feature interacts with another variable (or all other variables) in the model. As a reminder, feature interaction

and feature correlation are not the same. Two features x_j and x_k interact if the effect of x_j on model predictions depends on the value of x_k [52]. They are correlated if the values of x_j relate in some way with the values of x_k.

A good interaction indicator is provided with Friedman's H-statistic:

$$H_{jk}^2 = \frac{\sum_{i=1}^{n}[f_{\{j,k\}}(x_j^{(i)}, x_k^{(i)}) - f_j(x_j^{(i)}) - f_k(x_k^{(i)})]^2}{\sum_{i=1}^{n}(f_{\{j,k\}}(x_j^{(i)}, x_k^{(i)}))^2}$$

where $f_s(x_s)$ represents the partial function of x_s with $S \subseteq \{j, k\}$, defined for PDPs.

The result is the proportion of variance explained by the interaction between two features, given as a figure comprised in $[0, 1]$. A similar step holds for the interaction between one and all other features. The only modification required is to replace $x_k^{(i)}$ by $x_{\setminus j}^{(i)}$, which includes all variables except x_j.

This method presents the advantages of being built on an underlying theory through partial dependence plots and of being comparable across features, since the result belongs to $[0, 1]$. However, despite presenting a meaningful interpretation, it does not tell us how the interaction is shaped and does not characterise strong interaction, which is model specific.

Feature importance

There exist many ways to measure feature importance. They are usually specific to a model. *XGBoost*, for instance, offers three possibilities:

- 'gain': the average gain of the feature when it is used for building trees, where the gain implies the relative contribution of the corresponding feature to the model calculated by taking each feature's contribution for each tree in the model. Refer to the exact algorithm for split finding in Appendix B. This is the most widely used.
- 'weight': the number of times a feature is used to split the data across all trees. It is expressed as a frequency with respect to the total number of splits, all trees included. However, this might be misleading. Continuous features are used more times than categorical variables because they contain more split possibilities. They will consequently be attributed very high importance, which is often overvalued.
- 'cover': the average coverage of the feature when used for building trees, where coverage is defined as the number of samples affected by the split. In different terms, the cover metric means the relative number of observations related to this feature, expressed as a percentage for all features' cover metrics.

Two other popular tree based feature importance methods that will be used in this paper

(for the method described in the next section) are briefly overviewed below.

Entropy impurity. Entropy is a measure of homogeneity for a set S. We would like to use the reduction in entropy (or highest information gain) to evaluate importance. The feature presenting the highest information gain is the most influential and should be used for the split [53].

$$Entropy(S) = -\sum_i p_i log_2(p_i)$$

$$\Delta i(s) = I.Gain(S, x_j) = Entropy(S) - \sum_i \frac{\mid S_{z_i} \mid}{\mid S \mid} Entropy(S_{z_i})$$

where p_i denotes the frequency of items of type i in the set (for classification problems), x_j is the attribute considered, z_i the possible values of x_j, S_{z_i} a subset of S where $x_j = z_i$. Careful, this method has bias towards the highest branching factor.

Gini Importance or *Mean Decrease in Impurity (MDI)* is just the expected error rate at node N if the category label is selected randomly from the class distribution present at time N [53]. $i(N) = \sum_{ij} p_i p_j$. MDI is biased towards preferring variables with more categories and does not work well with correlated variables.

Lastly, there is a model-agnostic feature importance method that we will mention in the paper, called *permutation feature importance.* Importance is given as a metric that accounts for the increase in model prediction error after we permuted the values of the feature of interest. A feature is important if permuting its values increase the model error.[54, 55]

Algorithm .5 Permutation feature importance

Input: Trained model $\hat{f}$, dataset X of dimension $N \times p$, target vector y, error measure $L(y, \hat{f})$.
$e_{orig} \leftarrow L(y, \hat{f}(X))$
for j=1,...,p **do**
  choose a random permutation $\psi : \{1, ..., N\} \longrightarrow \{1, ..., N\}$ of features values for x_j
  $X_{perm} \leftarrow \psi(X)$
  $e_{perm} \leftarrow L(y, \hat{f}(X_{perm})$
  $FI \leftarrow \frac{e_{perm}}{e_{orig}}$ and $FI' \leftarrow e_{perm} - e_{orig}$
end for
sort features by descending FI or FI'.

In summary, the above code infers to compute the error of the original model, then re-compute the error but having permuted the feature of interest's values. This breaks the association between this predictor and the true outcome y. Dividing or subtracting the two errors gives a measure of importance for the feature of interest. This can be repeated for all features.

Feature Importance is a very powerful interpretation tool and one of the most widely used. Note that its results are tied to the model error and to the random choice of permutations. The negative sides of this method are its bias towards unrealistic instances (due to permutation) and its incapacity to deal with correlated variables. Lastly, one interrogation remains concerning where to apply it: should we use the training or test set?

Shapley Value

Literally, the Shapley value method proceeds as follows:

1. Define a feature of interest x_j and a coalition $S \subseteq \{x_1^{(i)}, ..., x_p^{(i)}\} \setminus x_j^{(i)}$
2. Keep the instance's value for the features in the coalition and pick a random value for the ones that are not.
3. Compute the prediction for the coalition $S \cup x_j^{(i)}$. Subtract from it the prediction of the coalition S, the only difference being that a random value is chosen for the feature x_j from all its grid values, instead of $x_j^{(i)}$
4. Weight this (divide) by the product of the number of coalitions of size $\mid S \mid$ and all possible coalition sizes. Store the result and repeat the process for a different coalition until they have all been considered. Note that the order in which feature appears in S matters. Sum all the results stored to obtain the Shapley value for the feature x_j.
5. Repeat the process for all features.

Algorithm .6 Shapley value method

Input: instance regarded, say $x^{(i)}$
for $j = 1$ to p **do**
  $R \leftarrow 0$
  for all subsets $S \in X \setminus \{x_j^{(i)}\}$ **do**
    $S' \leftarrow (v(S \cup \{x_j^{(i)}\}) - v(S)) \quad \frac{\mid S \mid !(p- \mid S \mid -1)!}{p!}$
    $R \leftarrow R + S'$
  end for
  $\phi_j^{(i)} \leftarrow R$
end for
Ouput: $(\phi_1^{(i)}, ..., \phi_p^{(i)})$

References

[1] Tom M. Mitchell. *The Discipline of Machine learning*. 2006

[2] Andrew Moore. AI is not 'magic dust' for your company, says Googles Cloud AI boss. *MIT Technology Review*, 2018.

[3] Zachary Lipton. From AI to ML to AI: On Swirling Nomenclature Slurried Thought. *Approximately Correct*. 2018

[4] R. Guidotti, A.Monreale, S.Ruggieri, F.Turini, D.Pedreschi, F.Giannotti. A survey of methods for explaining black boxes. *arXiv preprint arxiv: 1802.01933*. 2018.

[5] David Gunning, Explainable Artificial Intelligence (XAI). DARPA, `https://www.darpa.mil/attachments/XAIProgramUpdate.pdf`. 2016

[6] L. Gilpin, D. Bau, B.Yuan, A. Bajwa, M. Specter and L. Kagal. Explaining Explanations: An Approach to Evaluating Interpretability of Machine Learning. 2018

[7] T. Miller. Explanation in Artificial Intelligence: Insights from the Social Sciences. *arXiv preprint arXiv:1706.07269*. 2017.

[8] Kim, Been, Rajiv Khanna, and Oluwasanmi O. Koyejo. Examples are not enough, Learn to Criticize! Criticism for Interpretability. *Advances in Neural Information Processing Systems*. 2016

[9] Kate Crawford. Artificial Intelligence's White guy problem. *The New York Times*, 2016

[10] Caruana, Rich, Lou, Yin, Gehrke, Johannes, Koch, Paul, Sturm, Marc, and Elhadad, Noe mie. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. *KDD*. 2015.

[11] Kim, Been. Interactive and interpretable machine learning models for human machine collaboration. PhD thesis, Massachusetts Institute of Technology, 2015.

[12] Cathy ONeil, *Weapons of Maths Destructions*. 2016

[13] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images.*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.pp 427-436. 2015.

[14] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 86-94. 2017.

[15] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. *IEEE European Symposium on Security and Privacy (EuroSP)*. pp. 372-387. 2016.

[16] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* 2017.

[17] The Future of Life Institute. Asilomar AI Principles. `https://futureoflife.org/ai-principles/`. 2017.

[18] Max Tegmark. *Life 3.0*. 2017.

[19] Ribeiro, Marco Tulio, Singh, Sameer, and Guestrin, Carlos. why should i trust you?: Explaining the predictions of any classifier. In *KDD*, 2016.

[20] Kim, Been. Interactive and interpretable machine learning models for human machine collaboration. PhD thesis, Massachusetts Institute of Technology, 2015.

[21] Huysmans, Johan, Dejaeger, Karel, Mues, Christophe, Vanthienen, Jan, and Baesens, Bart. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 2011.

[22] Goodman, Bryce and Flaxman, Seth. European union regulations on algorithmic decision-making and a right to explanation. *arXiv preprint arXiv:1606.08813*, 2016.

[23] Zachary C. Lipton. The Mythos of Model Interpretability. *arXiv preprint arXiv:1606.03490*. 2017

[24] B. Mittekstadt, C. Russel, S. Wachter. Explaining Explanations in AI. The Alan Turing Institute, 2018.

[25] C. Molnar. *Interpretable Machine Learning*. 2018

[26] Krening, Samantha, Harrison, Brent, Feigh, Karen, Isbell, Charles, Riedl, Mark, and Thomaz, Andrea. Learning from explanations using sentiment and advice in rl. *IEEE Transactions on Cognitive and Developmental Systems*, 2016.

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

[28] David-Hillel Ruben. *Explaining explanation*. 2004

[29] Lipton, Peter. Contrastive Explanation. Royal Institute of Philosophy Supplements 27. pp 247-266. 1990.

[30] Nickerson, Raymond S. Confirmation Bias: A Ubiquitous Phenomenon in Many Guises. *Review of General Psychology 2 (2)*. Educational Publishing Foundation: 175. 1998.

[31] Doshi-Velez, Finale, and Been Kim. Towards A Rigorous Science of Interpretable Machine Learning, no. Ml: 113. *arXiv preprint arXiv: 1702.08608*. 2017.

[32] D. Van den Poel and B. Larivire, Customer attrition analysis for financial services

using proportional hazard models, European Journal of Operational Research, vol. 157, no. 1, pp. 196-217, 2004.

[33] D. G. M. Mozer, R. Wolniewicz and H. Kaushansky. Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry. IEEE Transactions on Neural Networks, 11: pp 690-696, 2000.

[34] L.J.S.M. Alberts, Churn Prediction in the Mobile Telecommunications Industry, 2006.

[35] Leslie Valiant, *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World* , 2013.

[36] T. Chen, C. Guestrin, XGBoost: A Scalable Tree Boosting System, 2016.

[37] *https://github.com/AlexDuvalinho/Explainable-Artificial-Intelligence-XAI-*

[38] Friedman, Jerome H. Greedy function approximation: a gradient boosting machine. *Annals of statistics*: 1189-1232. 2001

[39] Goldstein, Alex, et al. Peeking Inside the Black Box: Visualizing Statistical Learning with Plots of Individual Conditional Expectation. *arXiv preprint arXiv:1309.6392*, 2014.

[40] Apley, Daniel W. Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models. *arXiv preprint arXiv:1612.08468*. 2016.

[41] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. *arXiv preprint arXiv:1602.04938* 2016.

[42] Shapley, Lloyd S. A Value for N-Person Games. Contributions to the Theory of Games 2 (28): pp 307-317. 1953.

[43] Eyal Winter, The Shapley Value, *Handbook of game theory with economic applications*, 2002.

[44] Strumbelj, Erik, Igor Kononenko, Erik trumbelj, and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems 41 (3)*: 647-665. 2014.

[45] G. Rodriguez, Lecture Notes - Princeton University, Chapter 7: Survival analysis, *https://data.princeton.edu/wws509/notes/c7.pdf* 2010

[46] B.S Clarke and J.L Clarke, *Predictive Statistics: Analysis and Inference beyond models*, pp 206-248, 2018

[47] Python documentation for survival analysis, *https://lifelines.readthedocs.io/en/latest/*

[48] D. Collett, *Modelling Survival Data in Medical Research*, 3rd Edition, 2003.

[49] J. Friedman, Greedy Function Approximation: A Gradient Boosting Machine 1999 (paper GBM)

[50] Veronikha Effendy, Adiwijaya and Z.K.A. Baizal, Handling imbalanced data in customer churn prediction using combined sampling and weighted random forest, *ResearchGate*, 2014.

[51] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: Synthetic Minority Over-sampling Technique, *arXiv preprint arXiv:arXiv:1106.1813*, 2002.

[52] Friedman, Jerome H, and Bogdan E Popescu. Predictive Learning via Rule Ensembles. *The Annals of Applied Statistics*. JSTOR, 916-954. 2008.

[53] Duda, Hart and Stork. *Pattern Classification*, Wiley-Interscience, 2000.

[54] Ref: Breiman, Leo. Random Forests. *Machine Learning, vol 45*. Springer: pp 5-32. 2001.

[55] Fisher, Aaron, Cynthia Rudin, and Francesca Dominici. Model Class Reliance: Variable Importance Measures for any Machine Learning Model Class, from the Rashomon Perspective. *ResearchGate*. 2018.