# Post-hoc Explainable AI for Black Box Models on Tabular Data

Nedeljko Radulovic

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2023IPPAT028

Thèse de doctorat

# Post-hoc Explainable AI for Black Box Models on Tabular Data

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat: Informatique

Thèse présentée et soutenue à Palaiseau, le 29.09.2023., par

**NEDELJKO RADULOVIĆ**

Composition du Jury :

| | |
|---|---|
| Marcilio De Souto<br>Full Professor, Université d'Orléans<br>(Laboratoire d'Informatique d'Orléans - LIFO) | Président/Rapporteur |
| Dino Pedreschi<br>Full Professor,University of Pisa<br>(Department of Computer Science - DI-UNIPI) | Rapporteur |
| Elisa Fromont<br>Full Professor, Université de Rennes | Examinatrice |
| Armen Aghasaryan<br>Head of AI lab, Nokia Bell Labs | Examinateur |
| Albert Bifet<br>Full Professor, Télécom Paris, Institut Polytechnique de Paris / Artificial Intelligence Institute, University of Waikato, New Zealand | Directeur de thèse |
| Fabian Suchanek<br>Full Professor<br>Télécom Paris, Institut Polytechnique de Paris | Co-directeur de thèse |

PHD THESIS

# Post-hoc Explainable AI for Black Box Models on Tabular Data

*Author:*
Nedeljko RADULOVIĆ



TÉLÉCOM PARIS
Institut Polytechnique de Paris

December 20, 2023

THESIS TITLE:
*Post-hoc Explainable AI for Black Box Models on Tabular Data*

PHD CANDIDATE:
Nedeljko Radulović

SUPERVISORS:
Albert Bifet, professor at Waikato University, Hamilton, New Zealand
Fabian Suchanek, professor at Télécom Paris, France

ACADEMIC INSTITUTION:
Institut de Polytechnique de Paris
Département d'Informatique, de Données et d'Intelligence Artificielle
Laboratoire Traitement et Communication de l'Information (LTCI)

LOCATION:
Palaiseau, France

DEFENSE DATE:
September 29, 2023

# Contents

# List of Figures

# List of Tables

# Abstract

Current state-of-the-art Artificial Intelligence (AI) models have been proven to be very successful in solving various tasks, such as classification, regression, Natural Language Processing (NLP), and image processing. Coupled with the rise of available data, cloud computing power, and plenty of automatization tools, these AI models became a logical choice for numerous applications with high stakes and high volume decision making. The resources that we have in our hands today allow us to train very complex AI models to solve different tasks in almost any field: medicine, finance, justice, transportation, forecast, etc. With the popularity and widespread use of AI models, the need to ensure trust in them also grew. Complex as they come today, these AI models are impossible to be interpreted and understood by humans. The most common example is the Deep Neural Networks, which cannot be interpreted even by its own developer. These models are often called *"black-box"* models. One could ask: "Why do we even need to interpret these models?". Well, while the need for model interpretation may not seem obvious as long as the model performs the task correctly, the same cannot be said for the situation when the model makes a mistake. Even though they may be very rare, depending on their impact, they can be very significant. At that moment, the possibility to interpret the model and discover a cause for an error is very important. Motivated by these reasons, it became mandatory by law in Europe to ensure an explanation for decisions made by AI agents.

In this thesis, we focus on a specific area of research, namely post-hoc Explainable Artificial Intelligence (xAI) for black-box models on tabular data, which aims to provide the approaches to interpret the complex AI models and explain their decisions. Our work was focused on classification and regression tasks for tabular data and has resulted in two approaches, which represent the two main contributions:

1. STACI is a method to interpret complex classification and regression AI models, using surrogate decision trees to produce confident, general, simple, and accurate interpretations.

2. BELLA is a method to explain the individual predictions of complex regression models, using local linear surrogate models that are optimized to provide accurate, simple, and general explanations.

Both methods are model-agnostic post-hoc approaches, which means that the architecture of the black-box model is not important and that the method does not impact the inner workings of the black-box model. An important novelty of both of these approaches is their deterministic nature. As such, they provide consistent explanations and can be used to explain data and the predictions of the black-box models. Additionally, both approaches account not only for accuracy of explanations but for their simplicity and generality, thus ensuring explanations that are more robust and easier to understand. BELLA also provides *"counterfactual"* explanations, that provide information on how data point should be altered to reach the desired output. We confirmed their high performance through extensive experiments and user study.

Additionally, in this thesis, we present a work that contributes to another field of Machine Learning, Data Stream Mining. The importance of data stream mining lies in its ability to harness insights from continuous, high-velocity data sources, thereby addressing the evolving dynamics of contemporary data ecosystems. Data stream mining offers a solution by providing real-time analytics capabilities, making it a critical component in various domains. Industries such as finance, healthcare, and cybersecurity rely on instantaneous analysis of incoming data to detect anomalies, identify emerging patterns, and make informed decisions promptly. The ability to process data streams in real time ensures that organizations can respond swiftly to changing conditions, capitalize on emerging opportunities, and mitigate potential risks.

Moreover, data stream mining is pivotal in handling concept drift, a phenomenon where the underlying patterns in the data evolve over time. Traditional machine learning models, designed for static datasets, struggle to adapt to these changes. In contrast, data stream mining techniques are specifically tailored to handle concept drift, providing a more accurate and reliable means of extracting meaningful patterns from dynamic data streams. Seeing the importance and the success of, already existing, platforms for Machine Learning competitions on static datasets, we have developed a platform for Machine Learning competitions on data streams. SCALAR is an open-source, first of its kind, platform that supports online learning scenarios. The platform has been used for an online Machine Learning competition at IEEE BigData Conference.

# Résumé

Les modèles d'intelligence artificielle (IA) actuels ont fait leurs preuves dans la résolution de diverses tâches, telles que la classification, la régression, le traitement du langage naturel (NLP) et le traitement d'images. Associés à l'augmentation des données disponibles, à la puissance de l'informatique en nuage et à de nombreux outils d'automatisation, ces modèles d'IA sont devenus un choix logique pour de nombreuses applications à fort enjeu et pour la prise de décision en grande quantité. Les ressources dont nous disposons aujourd'hui nous permettent d'entraîner des modèles d'IA très complexes pour résoudre différentes tâches dans presque tous les domaines : médecine, finance, justice, transport, prévisions, etc. Avec la popularité et l'utilisation généralisée des modèles d'IA, la nécessité de leur faire confiance s'est également accrue. Aussi complexes soient-ils aujourd'hui, ces modèles d'IA sont impossibles à interpréter et à comprendre par les humains. L'exemple le plus courant est celui des réseaux neuronaux profonds, qui ne peuvent souvent pas être interprétés même par leur propre développeur. Ces modèles sont souvent appelés modèles de *"boîte noire"*. On pourrait se demander : "Pourquoi avons-nous besoin d'interpréter ces modèles? ". Si la nécessité d'interpréter un modèle ne semble pas évidente tant que le modèle exécute correctement la tâche, il n'en va pas de même lorsque le modèle commet une erreur. Même si elles sont très rares, elles peuvent être très importantes en fonction de leur impact. À ce moment-là, la possibilité d'interpréter le modèle et de découvrir la cause d'une erreur est très importante. C'est pour ces raisons que la loi européenne a rendu obligatoire l'explication des décisions prises par les agents d'intelligence artificielle.

Dans cette thèse, notre focalisation se porte sur un domaine de recherche particulier, à savoir l'Intelligence Artificielle Explicable (xAI) post-hoc pour les modèles de boîte noire sur les données tabulaires, avec pour objectif de

proposer des approches pour interpréter les modèles d'IA complexes et expliquer leurs décisions. Notre travail s'est concentré sur les tâches de classification et de régression pour les données tabulaires et a abouti à deux approches, qui représentent les deux contributions principales :

1. STACI est une méthode d'interprétation des modèles complexes de classification et de régression de l'IA, utilisant des arbres de décision de substitution pour produire des interprétations confiantes, générales, simples et précises.

2. BELLA est une méthode permettant d'expliquer les prédictions individuelles de modèles de régression complexes, à l'aide de modèles de substitution linéaires locaux optimisés pour fournir des explications précises, simples et générales.

Les deux méthodes sont des approches post-hoc agnostiques du modèle, ce qui signifie que que l'architecture du modèle boîte noire n'est pas importante et que la méthode n'a pas d'impact sur le fonctionnement interne du modèle boîte noire. Une nouveauté de ces deux approches est leur nature déterministe. En tant que telles, elles fournissent des explications cohérentes et peuvent être utilisées pour expliquer les données et les prédictions des modèles de boîte noire. données et les prédictions des modèles de boîte noire. En plus, les deux approches tiennent compte non seulement de la précision des explications, mais aussi de leur simplicité et de leurs avantages. de l'exactitude des explications, mais aussi de leur simplicité et de leur généralité, garantissant ainsi des explications plus robustes et plus faciles à comprendre. garantissant ainsi des explications plus robustes et plus faciles à comprendre. BELLA fournit également des explications "*contrefactuelles*", qui fournissent des informations sur la manière dont les points de données devraient être modifiés pour atteindre les objectifs fixés. point de données devrait être modifié pour obtenir le résultat souhaité. Nous avons confirmé leur Nous avons confirmé leur haute performance par des expériences approfondies et des études d'utilisateurs.

En outre, l'exploration des flux de données est essentielle pour gérer la dérive des concepts, un phénomène dans lequel les modèles sous-jacents dans les données évoluent au fil du temps. nomène où les modèles sous-jacents des données évoluent au fil du temps. Les modèles Les modèles traditionnels d'apprentissage automatique, conçus pour des ensembles de données statiques, peinent à s'adapter à ces changements. changements. En revanche, les techniques d'exploration des flux de données sont spécifiquement conçues pour gérer la dérive des concepts, ce qui permet d'obtenir des résultats plus fiables. pour gérer la dérive des concepts, offrant ainsi un moyen plus précis

et plus fiable d'extraire des schémas significatifs à partir de flux de données dynamiques. des modèles significatifs à partir de flux de données dynamiques. L'importance et le succès des plateformes l'importance et le succès des plateformes déjà existantes pour les concours d'apprentissage automatique sur des ensembles de données statiques. sur des ensembles de données statiques, nous avons développé une plateforme pour les concours d'apprentissage automatique sur les flux de données. d'apprentissage automatique sur des flux de données. SCALAR est une plateforme open-source, première en son genre, qui prend en charge les scénarios d'apprentissage en ligne. qui prend en charge des scénarios d'apprentissage en ligne. La plateforme a été utilisée pour un concours d'apprentissage d'apprentissage automatique en ligne lors de la conférence IEEE BigData.

# 1 Introduction

With the widespread use of Artificial Intelligence (AI) agents in various tasks across every aspect of research and industry, it became inevitable to impose legislative control on their usage. Even though they are very powerful and successful in tasks they have been trained for, these models lack one very important characteristic, transparency. Most of these powerful AI models are very complex models such as Deep Neural Networks and ensemble models such as Random Forests or XGBoost. The need to be able to understand how these opaque models came to a certain decision or to understand the global logic of their decision making process led to increased research interest in the explainable models and to establishing explainable AI (xAI) as an independent research field.

In this first chapter, we define the relevant terms and notations, we discuss the motivation, and possible use-cases and give an overview of the thesis.

## Contents

## 1.1 Preliminaries

Since the beginning of human history, people have sought to find ways to make their daily tasks easier and more efficient. This has led to the development of tools and technologies that have allowed us to automate many of the tasks that we once did by hand. One of the earliest examples of automation can be seen in the development of waterwheels, windmills, and other mechanical devices used to power various types of machinery. However, the widespread adoption of automation did not occur until the industrial revolution, which saw the development of steam power, the assembly line, and other technologies that revolutionized the way we live and work.

Since then, the development of computers and software has led to even greater levels of automation, with machines now able to perform tasks once thought impossible for a machine to do. These developments allowed scientists to pursue the idea of creating machines that can perform tasks that would otherwise require human-level intelligence, such as recognizing patterns, making predictions, and solving complex problems. This idea has been named and formulated by John McCarthy in 1956 when he organized the first academic conference on *Artificial Intelligence* (McCarthy et al., 2006).

**Definition 1.1.1.** *Artificial Intelligence is a branch of computer science that is concerned with the development of computers able to engage in human-like thought processes such as learning, reasoning and self-correction. (Kok et al., 2009)*

The early work on AI focused on developing systems that could reason and make decisions based on rules and logic. However, these systems were limited by the fact that they required human programmers to input the rules and logic, which made them inflexible and unable to learn from experience. During the 1980s, a new approach to AI advanced significantly, known as machine learning.

**Definition 1.1.2.** *Machine Learning is the field of study that gives computers the ability to learn without explicitly being programmed. (Samuel, 1959)*

Machine Learning (ML) is a sub-field of AI and it is mainly concerned with statistical techniques that enable learning from data. AI, on the other hand, includes a broader range of techniques that allow machines to perform tasks that usually need human intelligence such as problem-solving, reasoning, language understanding. AI models can be rule-based models, expert systems, and also ML models that learn from data.

Machine learning (ML) has significantly impacted various domains and transformed the way we approach complex problems. The development of machine learning algorithms, along with advances in computer hardware, has

led to rapid progress in AI in recent years. In particular, deep learning, a type of machine learning that uses neural networks to simulate the structure and function of the human brain, has enabled machines to achieve state-of-the-art performance in tasks such as image recognition, speech recognition, and natural language processing.

In its early stages, simple applications of ML included pattern recognition and character recognition, enabling machines to recognize handwritten letters and digits. These early applications laid the groundwork for more advanced ML techniques.

In modern times, machine learning applications have expanded exponentially, revolutionizing industries and improving efficiency. In natural language processing (NLP), ML algorithms power systems that can understand and respond to human language, enhancing human-computer interactions. Additionally, language translation and sentiment analysis benefit from ML-driven approaches, making communication across languages more accessible and efficient.

In the field of computer vision, ML algorithms enable object detection, image recognition, and facial recognition systems, contributing to advancements in autonomous vehicles, surveillance, and medical imaging. ML has also found extensive applications in healthcare, where it aids in disease diagnosis and prognosis.

Recommendation systems suggest products, movies, or content based on user preferences, enhancing user experiences and increasing user engagement. MLs is used in finance, where it is employed in fraud detection, credit risk assessment, and algorithmic trading, contributing to more secure financial transactions and better risk management. Additionally, ML plays a vital role in optimizing supply chains, predicting demand, and improving inventory management, leading to enhanced operational efficiency in various industries.

ML relies on statistical methods to enable machines to learn from data. Depending on the type of data that is available, there exist three main types of machine learning: supervised learning, unsupervised learning, and reinforcement learning.

In **supervised learning**, the algorithm learns to map the input to the output, and it assumes the existence of a training labeled dataset, i.e., the correct output is provided for each input. **Unsupervised learning**, on the other hand, involves training a model on an unlabeled dataset, i.e., a dataset that is similar to a labeled training dataset except that it doesn't include the label. The algorithm learns intrinsic patterns and groupings in the data without any guidance. This helps in a better understanding of the data and its segmentation. An example of unsupervised learning is clustering, where the algorithm groups similar data points together based on similarities in their features. **Reinforcement learning**

is a type of machine learning where the algorithm learns by interacting with an environment and receiving feedback in the form of rewards or punishments. The goal is to find the optimal set of actions to make in order to maximize the cumulative reward over time.

Throughout this thesis, we are focused on the interpretability of supervised machine learning models. Supervised learning involves training a machine learning algorithm on labeled data, meaning the desired output is already known.

**Definition 1.1.3.** ***A labeled training dataset*** *is a set of n d-dimensional data points $X_i$, where $X_i = (x_{i,1}, x_{i,2}, ...x_{i,d})$, and $x_{i,1}, x_{i,2}, ...x_{i,d}$ are features. Each of the data points $X_i$ has one or more corresponding labels $Y_i$ attached to it. These labels are used to train supervised machine learning models, where the model learns to map input data to output labels based on the patterns found in the dataset.*

Table 1.1: Labeled dataset

| Features | | | | Label |
|---|---|---|---|---|
| $x_{1,1}$ | $x_{1,2}$ | $\cdots$ | $x_{1,d}$ | $Y_1$ |
| $x_{2,1}$ | $x_{2,2}$ | $\cdots$ | $x_{2,d}$ | $Y_2$ |
| $x_{3,1}$ | $x_{3,2}$ | $\cdots$ | $x_{3,d}$ | $Y_3$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $x_{n,1}$ | $x_{n,2}$ | $\cdots$ | $x_{n,d}$ | $Y_n$ |

For example, a supervised learning algorithm could be trained to predict the risk of heart disease in patients. In this particular case, the labeled dataset would be data from electronic health records, such as patient age, sex, blood pressure, cholesterol levels. These data are labeled i.e. we already know which of these patients have a heart disease. The algorithm, through the training process,

Table 1.2: Example of a labeled dataset

| Features | | | | Label |
|---|---|---|---|---|
| Age | Gender | Blood Pressure | Cholesterol | Heart Disease |
| 45 | Male | 130/80 | 200 | No |
| 60 | Female | 140/90 | 240 | Yes |
| 55 | Male | 150/95 | 180 | Yes |
| 50 | Female | 120/70 | 160 | No |
| 65 | Male | 135/85 | 220 | Yes |

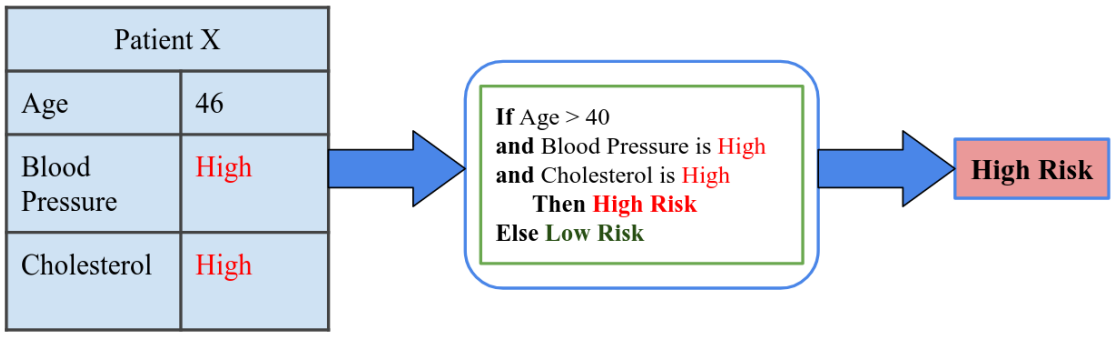



Figure 1.1: Example of a white-box Machine Learning Model

learns from the data such that it can make predictions if the new patient has heart disease (Table 1.2).

**Definition 1.1.4.** *A prediction is the output of a machine learning model when it processes new input data, based on learned patterns from previously seen data. (Alpaydin, 2020)*

We categorize two types of supervised machine learning tasks based on the nature of the label: these are known as classification and regression. In classification tasks, the goal is to predict a categorical label or *class* for each input. On the other hand, in regression tasks, the goal is to predict a continuous numerical value as output. Once the algorithm has been trained on the training dataset to make decisions on new data, we refer to it as a *machine learning model*. In other words, a *machine learning model* is a way to represent patterns in data that can be used to make predictions or decisions about new data. For example, the aforementioned ML model that predicts the risk of heart disease in patients can be a set of rules as shown in Figure 1.1.

The process of creating a machine learning model typically involves several steps. First, a dataset is collected that contains examples of input data and the corresponding output data. This dataset is then used to train the model by adjusting its parameters to minimize the difference between the predicted output and the actual output.

According to the level of understanding of the decision-making process of a machine learning model, we distinguish two groups: interpretable or *"white-box"* (transparent) models and non–interpretable or *"black-box"* (opaque) models.

**Definition 1.1.5.** *White-box model refers to all recognized interpretable machine learning models, e.g. the models that are understandable for humans. The white-box models are, for example: decision trees, linear models, rule based models, etc. (Guidotti et al., 2018b)*

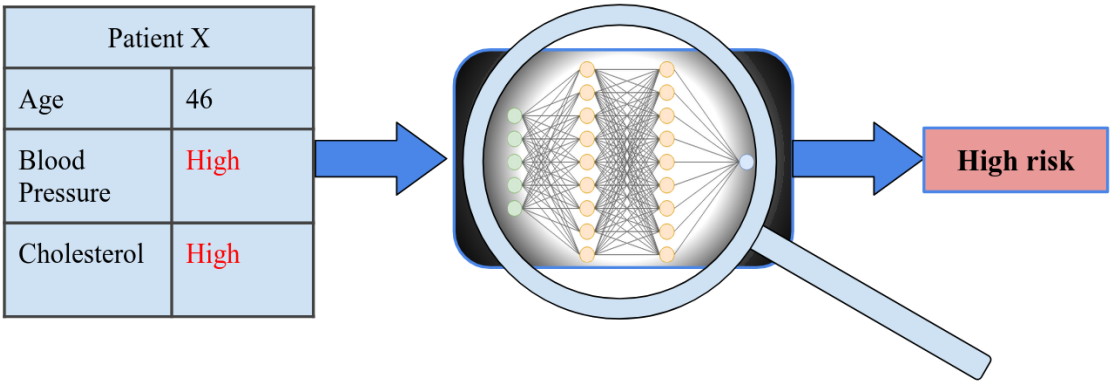



Figure 1.2: Example of a black-box Machine Learning model

One example of a white-box model is presented in Figure 1.1. The decision-making process of *"white-box"* can be described with a set of rules or simple (linear) equations, both of which are easy to understand and interpret.

**Definition 1.1.6.** *In the context of explainable AI (xAI), "black-box" model refers to a machine-learning obscure model, whose internals are either unknown to the observer or they are known but uninterpretable by human. (Guidotti et al., 2018b)*

The "black-box" models are: Deep Neural networks, ensemble models (Random forests) etc. As shown in Figure 1.2, even when we know the architecture of the black-box model itself and the mathematics behind it, it is impossible to deduce the reasons for a certain prediction. On the other hand, we have *"white-box"* models.

Despite the huge progress in ML, there are still many challenges that need to be addressed. One of the biggest challenges is developing ML models and, more generally, all AI systems that are trustworthy and reliable. As AI systems become more complex, it becomes harder to understand how they work and to ensure that they are making decisions that are fair and unbiased. The techniques that help us understand how these complex systems make decisions are the main focus of this thesis.

From the very title of this thesis, we are using the terms *"explainability"* and *"interpretability"*. Interpretability and explainability are both important concepts in the field of artificial intelligence and are often used interchangeably in the literature, but at the same time, there exist slightly different definitions.

**Definition 1.1.7.** *Interpretability is the ability to present the model in terms understandable by humans. (Doshi-Velez and Kim, 2017)*

Interpretability refers to the ability to understand how a machine learning model works and how it arrives at its decisions. It focuses on the internal

workings of the model and the ability to understand its features, variables, and mathematical processes. A model can be interpretable if a human can inspect its internal mechanisms and understand its decision-making process.

On the other hand, explainability refers to the ability to provide an explanation of the model's output in a way that is understandable to a non-technical user.

**Definition 1.1.8.** *Explainability is associated with the notion of explanation as an interface between humans and a decision maker that is, at the same time, both an accurate proxy of the decision maker and comprehensible to humans. (Guidotti et al., 2018b; Arrieta et al., 2020)*

The explanation is the notion that comes from social sciences and it is a complex phenomenon whose definition has been a subject of debate throughout history. Still, there is no consensus about one unique definition of an explanation. Rather there exist many types of explanations depending on what kind of question they answer or who are they meant to. In the context of AI, we are mainly interested in answering the question *" Why? "*. For example, if a patient would ask a doctor: *"Why do I have high risk of getting a heart disease? "*, the doctor could provide an explanation such as: *"Your blood pressure and cholesterol levels are too high"*. If, instead of a doctor, we have a ML model, a corresponding explanation would involve parts of an input that had the biggest impact on the decision.

**Definition 1.1.9.** *An explanation is additional meta information, generated by an external algorithm or by the machine learning model itself, to describe the feature importance or relevance of an input instance towards a particular output classification. (Das and Rad, 2020)*

Even though there is no definitive distinction in the literature, interpretability is often related to the understanding of the whole model while explainability is more related to understanding individual predictions of the model. (Burkart and Huber, 2021) In the example of the heart disease prediction model, interpretability would involve understanding how the machine learning algorithm uses various factors to predict the risk of heart disease. This could be achieved by presenting the decision making logic of the model in understandable terms, for example, list of rules such as in Figure 1.1. Explainability, on the other hand, would involve presenting the patient and doctor with a clear explanation of why the algorithm made a particular prediction, such as highlighting the specific risk factors that contributed to the prediction. In the aforementioned example, it would justify the model's prediction of a high risk of heart disease with the given patient because they are 46 years old and because they have high blood pressure and cholesterol levels.

In this thesis, we will adopt the term interpretability and interpretable for all methods that aim to demystify the decision-making process of black-box models. We will refer to explainability and explanations for methods that explain particular predictions of black-box models.

## 1.2 Motivation

The ability to quickly and successfully learn patterns and relations in huge amounts of data, made ML models are being used in various applications. Today, almost every industry strongly relies on data. Almost everything is being recorded and measured and those records are later used either to help and predict some unwanted events like natural catastrophes (earthquakes, floods) or health issues (heart attacks), they are used in justice to improve policing, in finance for forecasting and decision making. It is obvious that these models handle very sensitive and important tasks, thus we need not just to minimize the number of errors but we also need to understand how these models make their decisions. What follows is that the accuracy is not enough. Even though it is a significant indication of the model's performance, accuracy is just not sufficient to validate and deploy the model into a real-world scenario. Wrong decisions can have a huge impact on people's lives and it is important to ensure that models' decisions are justified and that they are the result of learned patterns in the data rather than a random lucky guess.

As mentioned before, very powerful AI models are used in many different applications and we have many examples of them being successful. On the other hand, there are also examples where AI models didn't behave as expected, once they were deployed in the real world. The first example comes from justice, where AI models are used to predict crime recidivism (the probability that a sentenced criminal will commit crimes again). A statistical study on risk assessment tool [1], shows that offenders of African-American descent were attributed a higher risk score than white offenders. The study found that they were almost twice as likely to be marked higher risk and not re-offend than white people. A higher risk of recidivism can then lead to longer imprisonment, i.e., a material disadvantage for the defendant. Even though ethnicity and recidivism may correlate, it is illicit to predict recidivism based on ethnicity. This is because such a prediction would penalize a person merely for belonging to a certain ethnicity (i.e., what the person *is*), rather than on the past behavior of the person (what the person *does*). It is thus important not just which decision was taken, but why that decision was taken. (Russell and Blackburn, 2020)

[1] https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing

There is one more very well known case of failure of an AI system due to racial bias: Google Photos app labeling black people as gorillas [2]. Even though it is impossible to have a model that is correct 100% of the time, not all errors are the same. If the model labels a photo of a dog as a horse, this mistake would have barely some impact. On the other hand, labeling a photo of black people as gorillas could have higher consequences. Since the model that has been used for this particular application was a complex, opaque model it was not possible to determine how this mistake was made and the fix that was implemented at the moment was simply removing anything related to gorillas.

Another common bias is gender bias, which in the professional world is usually exhibited by employing men in higher paid positions. One example where an AI agent exhibited this kind of bias is Amazon's hiring bot [3]. The AI agent was eventually shut down after it was discovered that it ranked female candidates lower for technical jobs. Another study showed the gender bias in Google online ads [4]. The study revealed that men were shown ads for high paid jobs much more often than women. But since the algorithm for targeting is proprietary and as such a *black-box*, it wasn't possible to determine where the bias comes from.

The "Wolf vs. Husky" experiment presented in Ribeiro et al. (2016), showed how we can have a false perception of the quality of the AI model. The experiment consisted of training a model to recognize if there is a wolf or a husky in the image. Bias has been added intentionally in the data, where all images with wolfs had snow in the background. After the training, the model had high accuracy and without further inspection one could assume that the model is valid. In the second part of the experiment, to verify how the model is making decisions, the important parts (features/pixels) of the images were extracted. These features showed, that the model actually learned that when there is snow in the image – it's an image of a wolf. It is clear that this model will not recognize a husky in snow. The idea of the experiment was to emphasize the importance of being able to interpret and understand the decision making logic of complex AI models.

The aforementioned cases are some examples of the impact that wrong AI decisions can have on the world, people, health, business etc. – even though the models performed the task they were trained on accurately. We thus have to accept that accuracy on testing data alone cannot be the sole criterion of performance for AI models. We also have to *understand* how the model arrives

[2]https://archive.nytimes.com/bits.blogs.nytimes.com/2015/07/01/google-photos-mistakenly-labels-black-people-gorillas/

[3]https://theconversation.com/amazons-sexist-hiring-algorithm-could-still-be-better-than-a-human-105270

[4]https://www.cmu.edu/news/stories/archives/2015/july/online-ads-research.html

at its decision so that we can validate whether the model took its decision *for good reasons*. Therefore, the importance of interpretability in AI models cannot be overstated. By gaining insights into how the model arrives at its predictions, we can identify potential biases, uncover hidden patterns, and detect any weaknesses in the model's reasoning. This interpretability not only benefits the developers but also end-users, stakeholders, and regulators. It enables better debugging and fine-tuning of the models, ultimately leading to improved performance and robustness. Additionally, it aids in complying with regulatory requirements in sensitive domains, ensuring that models do not make decisions based on discriminatory or biased factors.

The EU has adopted the General Data Protection Regulation i.e. GDPR (Goodman and Flaxman, 2017) to legally regulate data protection and privacy. In addition, the GDPR regulates the use of AI models in the decision making process in the way that the right to explanation is guaranteed to anyone being affected by the decision of an AI model. Due to the AI's impact and infiltration in every aspect of modern lives, the EU proposed an updated regulation, popularly known as the AI Act (Commission, 2021). AI Act is the first law on AI proposed by a major regulatory body. This framework aims to regulate the use of AI and to ensure the protection of health, environment, safety and human rights through trustworthy AI. The AI act proposes a framework for categorizing AI systems into 4 tiers, according to the level of risk, they pose to the health and safety of a person: minimal, limited, high and unacceptable (Veale and Zuiderveen Borgesius, 2021). Transparency requirement is aligned with the risk level the system poses – the higher the risk, the higher level of transparency is required. For example, real-time biometric identification systems present unacceptable risk systems and therefore are prohibited. There exist certain exceptions e.g. when peoples' lives are endangered. Systems used in law enforcement, management of critical infrastructure, employment etc. present high risk systems. In such cases, the provider is obliged to ensure the data quality, accuracy and robustness of the system, the technical documentation, logging for traceability, human oversight, etc. The systems with limited risks, such as chatbots, don't have to comply with strict requirements, but the provider needs to inform users that they interact with an AI system rather than with a real person.

The EU countries are among the first ones to implement legal regulations on the use of AI. In that sense, the EU is also encouraging research in the fields of xAI and trustworthy AI. Several big projects are funded by the EU, uniting numerous prestigious researchers and institutions in order to compute meaningful explanations of AI models' decisions and to prevent their undesired

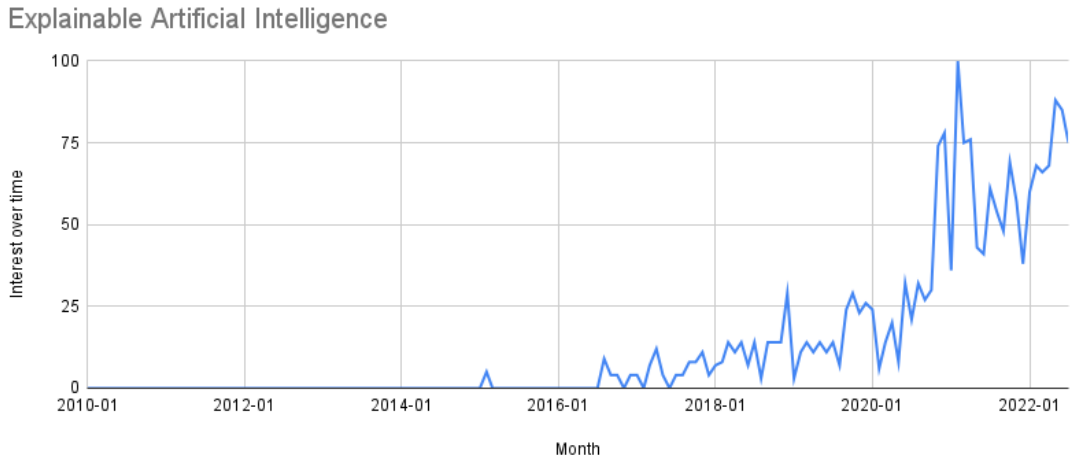



Figure 1.3: Google trends for the term: Explainable Artificial Intelligence[5]

effects [5] and to lay down the foundations of trustworthy AI [6].

Ultimately, interpretability is a necessary aspect of the responsible and ethical use of machine learning. All of this led to the expansion of xAI as a separate research topic, which, in recent years, became very popular. The figures 1.3 and 1.4 show the popularity of the terms *"Explainable Artificial Intelligence"* and *"Interpretable Machine Learning"* respectively, in the Google search engine from 2010 to 2022. The $y$ axis shows the interest in the term, which has been $min-max$ normalized, such that it ranges from $0$ to $100$, and the time (in months) is on the $x$ axis. From these figures, we can see that in the last three years, these terms hit the peak of their popularity.

In this section, we discussed some real-world examples to point out the motivation for xAI. In the rest of this chapter, we will introduce the relevant definitions and notations. Later we will give an overview of the current state-of-the-art.

## 1.3 The taxonomy of interpretable ML methods

The interpretability of ML models can be perceived and categorized according to different criteria. Interpretability can be categorized by the time when it is implemented in the ML model (before or after training), by its scope (local or global) and its applicability (model-specific or model-agnostic). In the following,

[5]https://xai-project.eu/index.html
[6]https://cordis.europa.eu/project/id/952215
[5]Data source: Google Trends (https://www.google.com/trends).

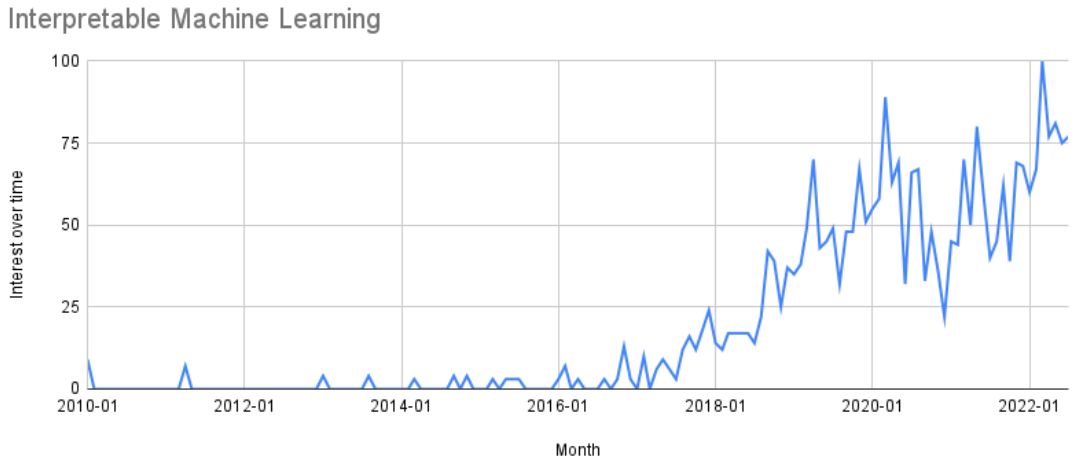



Figure 1.4: Google trends for the term: Interpretable Machine Learning[5]

we explain these notions as an introduction to the taxonomy of techniques for interpretable ML.

From the definition of white-box models, it is clear that their transparency comes built-in with their design. On the other hand, black-box models don't have this characteristic and the usual way to improve their transparency is by applying appropriate techniques after the model has been trained and without interfering with its internal architecture. From this stance, we distinguish two types of interpretability of ML models: *ante-hoc* and *post-hoc*.

**Definition 1.3.1.** *Ante-hoc interpretability is considered to be built-in from the beginning of the model creation. It is fulfilled by using the white-box models. (Burkart and Huber, 2021)*

In our heart disease example, *ante-hoc* interpretability would imply using an interpretable machine learning model such as a decision tree or a rule-based model (as in Figure 1.1) to predict the patient's risk of getting a heart disease. On the other hand, *post-hoc* interpretability would involve a complex model making a prediction and an interpretable model explaining it, as shown in Figure 1.5.

**Definition 1.3.2.** *Post-hoc interpretability is added to the model after its creation. (Burkart and Huber, 2021)*

This is done by applying some of many techniques that aim to provide insights into models' decision making process such as feature importance, interpretable surrogate models, rule extraction etc.

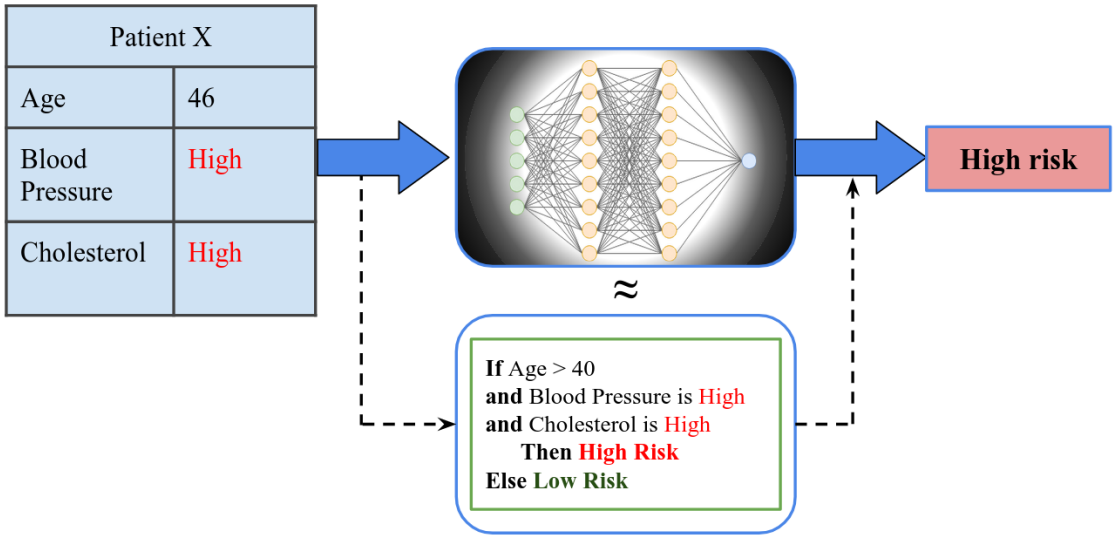



Figure 1.5: Interpreting a black-box model

The second aspect to categorize interpretability approaches is their applicability. Some approaches benefit from the black-box model architecture and thus are tailored to be used only with certain black-box models. These approaches are known as *"model-specific"*. The other group of approaches, *"model-agnostic"* approaches, is more general and they don't rely on the internal architecture of the black-box model and can be used to interpret any model.

In this thesis, we focus on *post-hoc, model-agnostic* techniques of improving the interpretability of complex ML models. More specifically, we will rely on white-box models as the proxy, so called surrogate models, for enabling the interpretability of black-box models without interfering with their inner architecture. That means that interpretability is an added feature of the black-box model. The training of these surrogate models is done only based on the input and the output for the black-box model. With *post-hoc* interpretability we get insights of how the black-box model arrives at its decision and we get to keep the black-box model and exploit its high performance. Even though, *post-hoc* interpretability provides insights, interpreting complex models comprehensively, remains a challenge.

The third criterion for the categorization of interpretability is its scope. We differentiate between the explanations of individual predictions and the interpretation of the whole model.

**Definition 1.3.3.** *Local interpretability holds for an individual prediction and its close vicinity. (Burkart and Huber, 2021)*

**Definition 1.3.4.** *Global interpretability holds for the entire black-box model. (Burkart*

*and Huber, 2021)*

To provide explanations for individual predictions, local interpretable models are usually trained for each prediction separately. Global interpretable models can also be used to explain individual predictions, but in contrast to the local interpretable models, they are trained only once on the whole domain.

Each group of approaches has some limitations: global interpretable models tend to be too complex and local interpretable models apply only to small regions thus not providing general insights about the black-box model. We will discuss these limitations in more detail in Chapters 3 and 4. We will present our two approaches, STACI and BELLA, that address the aforementioned limitations of global and local interpretable models, respectively.

### 1.3.1 Evaluation

How do we evaluate an interpretation model? How do we compare two different interpretation models? These are the questions that we discuss in this section. We define the metrics that are used for the evaluation of the interpretation models.

Similarly to any other ML/AI model, the interpretation models are trained in a way to optimize a specific loss function and are evaluated according to some predefined metric. Evaluation allows us to determine how the model is performing in a given task and to compare different models against each other. Thus, the evaluation is an indispensable step in the process of building any AI/ML model.

Usually, the quality of ML models is measured by how close the predictions it makes are to the real value. This can be measured by different evaluation metrics such as: accuracy, F1 measure, precision, kappa characteristic, mean squared error (MSE), etc. In a general classification/regression scenario, calculating some of the aforementioned metrics would give enough information about the quality of the model. On the other hand, the xAI setting is different. Except for the quality of the predictions we also want to measure how understandable is our model.

The post-hoc interpretation models aim to replicate the behaviour of the black-box models in a simple, understandable fashion. With this in mind, we define the most common characteristics that "good" interpretation models should have: fidelity (in literature also referred to as accuracy or correctness), simplicity (i.e. complexity, compactness), and generality (representativeness).

**Definition 1.3.5.** *Fidelity is the measure of how good is the interpretation model in mimicking the behavior of the black-box model. (Guidotti et al., 2018b)*

Fidelity can be defined through accuracy, F1 measure, MSE etc., depending on the task at hand. The main difference is that in this setting we don't evaluate against the true values but against the values predicted by the black-box model.

The other important desiderata of the interpretation models is how understandable they are to humans. It is not always easy to have humans evaluate and verify manually if the interpretation model is in fact understandable, so it is very common to define a metric that will measure how complex/simple is the interpretation model or the explanations that it provides.

**Definition 1.3.6.** *Simplicity (complexity (Guidotti et al., 2018b), compactness((Nauta et al., 2022))) measures how understandable is the model, through its size.*

It is defined depending on the type of the interpretation model (Guidotti et al., 2018b):

- Linear models: number of features included in the model
- Decision trees: maximum depth of the tree or number of internal nodes
- Rule-based models: number of rules

Fidelity and simplicity (complexity) represent the two most often used evaluation metrics and seem to be widely adopted in the literature. Another important metric is the *generality* (representativeness (Molnar, 2018)) of the interpretations. General interpretations

**Definition 1.3.7.** *Generality represents the percentage of instances (data points) that the interpretation covers. (Molnar, 2018)*

Global interpretation models reflect the global behavior of the black-box model, but as has been mentioned before, they provide interpretations for individual predictions as well. Thus we argue that these interpretations should cover not just a single instance but its neighbourhood also to ensure interpretations that are more stable and more representative of the black-box model's decision-making process.

These metrics constitute a basis for desiderata that interpretable models. At the same time, they define the main difficulties that xAI poses.

## 1.4 The challenges of *post-hoc* interpretability

Our main goal is to learn the same patterns as the complex black-box model with a simple interpretable one. Usually, opaque architectures like deep Neural Networks and Random Forests are used for solving more difficult tasks and thus

they usually capture very complex patterns in the data. If we want to perfectly mimic the behaviour of such a complex model, we might end up with a very complex interpretable model. The dilemma arises is that if this, very complex interpretable, model is actually interpretable in practice. For example, if we consider a rule-based model, like the one in Figure 1.1. This model is very simple and can be summarized by one *if...then* rule. In contrast, if this model could only be summarized by, let's say, hundreds of *if..then* rules, it would make the model very complex and difficult to interpret. At the same time, this model would be able to capture more intricate patterns in the data than the simple one. This would, most often, lead to better performance. Through that, we arrive at the main challenge in xAI approaches. The balance between the complexity of the interpretable model and its performance, is commonly referred to as the *"complexity vs accuracy"* trade-off. In this work, instead of the *complexity* we will use the term *simplicity*, such that the trade-off can be seen as balancing between maximizing both simplicity and accuracy.

We discuss this trade-off only related to *post-hoc* xAI approaches. In general, the simplicity of the model depends on the complexity of the task at stake and the requirements for the model performance. Sometimes it may happen that the simple (interpretable) model will satisfy the requirements and sometimes even perform better than the complex black-box model. As it has been discussed in Rudin (2018), when facing a new machine learning problem, as a first step, it is advisable to try solving it using some of the interpretable models (decision trees, linear models). If the precision requirements have not been met, only then one should consider complex, non-interpretable models such as deep NNs and RFs.

Here we consider a common xAI scenario, where we have a trained black-box model and interpretability needs to be implemented afterwards. We argue that in this case, the *"simplicity vs accuracy"* trade-off is relevant and solving it represents the main goal of xAI. The task that xAI approaches are trying to solve is to accurately explain the decision-making process of the complex model in simple terms.

The aforementioned *"simplicity vs accuracy"* trade-off is present with approaches that assess the interpretability on a global level, e.g. methods that interpret the black-box model as a whole. On the other hand, with approaches that assess the interpretability on a local level, interpreting each individual prediction, this two-way trade-off adopts a third dimension which is *generality*, as shown in Figure 1.6. While being focused on individual predictions, we argue that, in order to ensure that the local interpretable models provide relevant interpretations, these interpretations have to be applicable not just to a single data point but also on its neighbourhood.

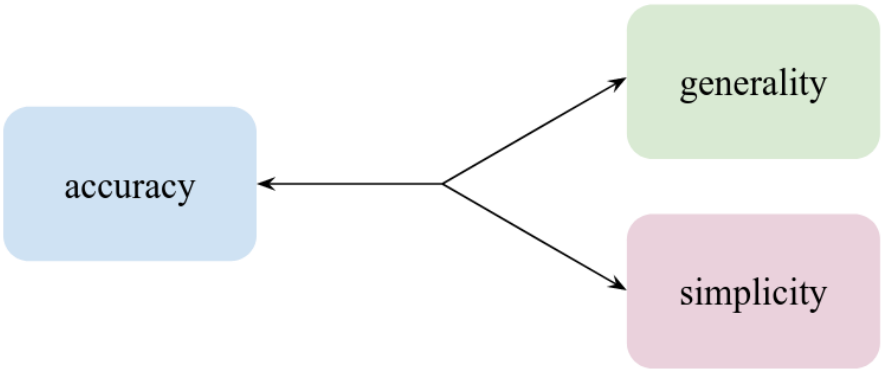



Figure 1.6: 3-way trade-off for local interpretability approaches

The works presented in this thesis aim to tackle and overcome the aforementioned challenges, common for xAI approaches. We present two post-hoc model-agnostic approaches, for global interpretation of classification black-box models - STACI, and for local explanations of predictions of regression black-box models - BELLA.

**STACI** is a global interpretation method for black-box classifiers that tackles the *"simplicity vs accuracy"* trade-off. STACI uses a set of decision trees, trained separately to represent each of the classes. The decision trees are trained once on the whole dataset and thus represent a global interpretation model. We propose a specific training algorithm that results in simple yet accurate interpretation models. At the same time, the optimization function accounts for generality thus providing the interpretations that apply to a larger portion of the dataset. Being accurate, simple and applicable to a wider range of data points, these interpretations have been preferred by the users in the user study that we have conducted for the purpose of validating our approach and the choice of the metrics that it optimizes.

- Nedeljko Radulovic, Albert Bifet, and Fabian Suchanek. Confident interpretations of black box classifiers. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.
  DOI: 10.1109/IJCNN52387.2021.9534234. [*code*]

**BELLA** is focused on explaining individual predictions of regression black-box models. The main idea behind this approach stems from the intention to overcome the three-way *"simplicity vs accuracy vs generality"* trade-off. BELLA provides explanations from local linear models that are trained to account for accuracy and generality, while regularization controls the simplicity of the explanation. Additionally, the explanation is provided in a way that allows users to

compute the explained (approximation of predicted) value by themselves. This, along with the improved generality (the explanation can be applied to multiple data points) proved to be of importance for the users, which we again verified through a user study.

- Nedeljko Radulovic, Albert Bifet, and Fabian Suchanek. BELLA: Black Box Model Explanations by Local Linear Approximations. **UNDER REVIEW** *AAAI - Association for the Advancement of Artificial Intelligence* [Preprint]

Both STACI and BELLA, use only already existing, available, real data points, which makes them deterministic, in contrast to most of the state-of-the-art approaches. The deterministic nature of these approaches guarantees that these interpretation models for one data point always provide the same explanation, which is not the case with the approaches that rely on synthetic data points.

## 1.5 Challenges of data stream mining

The previous sections have focused on explaining the decisions of ML models. This assumes that ML models are already available and working well. However, there is one domain of ML where the scientific community is still struggling with developing these models in the first place: data stream mining. In this section, we discuss the motivations for online learning and define its main challenges.

Artificial intelligence, and especially machine learning has attracted a huge audience of researchers, industry practitioners and freelance enthusiasts. Due to its versatility, people are continuously applying machine learning to solve new problems. Finding the best model for the task at hand is not a one-way street and thus can be a long and complicated process. We will often have several different models that perform similarly and only by tweaking hyper-parameters or relying on specific ideas that stem from our previous experience, we can make a significant difference in performance. It follows that the best model doesn't always come only from the book and theoretical knowledge. This is why companies often when privacy constraints allow it, ask the "public" to solve some classification or regression problems for a reward. To allow for this several platforms have been established, the most popular being, Kaggle. Most of the existing platforms for data science competitions are tailored to offline learning where a static dataset is made available to the participants before the competition starts. This dataset is divided into training and test sets. The training set is used to build and train the model, which is then tested on the test set.

But, today, data is produced in real-time. We are surrounded by Internet of Things (IoT) devices monitoring various activities, every mouse clicks in our browser, each message, and each call on our phone represents a data point in

some huge dataset. Each second billions of these data points are created around the world. Obviously, at some point, it became crucial to have models that can learn and make decisions in real time without storing the data. This is defined as an online learning scenario. In online learning, the data arrive at high speed in real-time and the model has to make predictions in a short time. Also, the model has to be able to learn from these new data.

We recognize that current, state-of-the-art, platforms for machine learning competitions do not conform with the requirements of online learning and thus are not suitable for this kind of competition. We propose a novel platform tailored for machine learning competitions on data streams – `SCALAR`. `SCALAR` supports this data stream machine learning scenario where data is continuously released, in batches every time interval. Predictions for each current batch, that are sent before a defined deadline, are evaluated in real-time, and the results are shown on the live leaderboard. In Chapter 5, we present an implementation of a `SCALAR` and we discuss the machine learning competition that was organized using `SCALAR` on the IEEE Big Data Challenge Cup 2019.

- Nedeljko Radulovic, Dihia Boulegane, and Albert Bifet. Scalar- a platform for real-time machine learning competitions on data streams. *Journal of Open Source Software*, 5(56):2676, 2020. DOI: 10.21105/joss.02676. [*code*]

- Dihia Boulegane, Nedeljko Radulovic, Albert Bifet, Ghislain Fievet, Jimin Sohn, Yeonwoo Nam, Seojeong Yu, and Dong- Wan Choi. Real-time machine learning competition on data streams at the IEEE big data 2019. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3493–3497. IEEE, 2019. DOI:10.1109/BigData47090.2019.9006357

# 2

# eXplainable Artificial Intelligence

In this chapter, we provide a comprehensive literature review on Explainable Artificial Intelligence (xAI). We begin by presenting a taxonomy of xAI approaches, categorizing them based on their scope and applicability. Subsequently, we delve into an in-depth analysis of the most prominent and state-of-the-art xAI approaches, highlighting both their strengths and limitations.

## Contents

## 2.1 Literature review

There have been many works to enable interpretability in complex ML models. In this section, we give a review of some most prominent, state-of-the-art approaches and some most common principles. Explainable AI has received a lot of attention lately in the research community, and several survey papers discuss recent approaches (Beaudouin et al., 2020; Guidotti et al., 2018b; Adadi and Berrada, 2018; Murdoch et al., 2019; Burkart and Huber, 2021). In Figure 2.1 we categorize the most prominent approaches in the field. The two main groups of approaches are *ante-hoc* and *post-hoc*. The ante-hoc group uses models that are interpretable by design: decision trees CART (Breiman et al., 1984), rule-based models, linear models, Bayesian models, kNN etc. One school of thought argues that these methods should be preferred for high stake tasks (Rudin, 2018). These already interpretable, models are often used to interpret the complex black-box models. In this scenario, we refer to them as surrogate models and they add the interpretability *post-hoc*. Thus, contrary to *ante-hoc*, *post-hoc* approaches add interpretability without interfering with the inner architecture of the black-box model. *Post-hoc* approaches are further categorized into the approaches that can be applied to any machine learning model, i.e. *model-agnostic*, and the ones that are tailored to explain specific types of machine learning models, i.e. *model-specific*. We then divide *model-agnostic* approaches based on the scope of the explanations that they provide: global or local. On the other hand, *model-specific* approaches we categorize based on the type of the model that they apply to. Throughout this chapter, we will address each category and give an overview of these approaches.

## 2.2 White-box models

Not all machine learning models are opaque. There exist several classes of machine learning models that can be interpreted and replicated by humans. These models are transparent and are usually referred to as *"white-box"* models. In essence, just by looking at the parameters of the model, once it has been trained, one can compute the prediction for each new data point. The models that allow this are: tree-based (Breiman et al., 1984), rule-based (Rivest, 1987; Letham et al., 2015; Yang et al., 2017), linear (Ustun and Rudin, 2016), Bayesian networks (Friedman et al., 1997) and *k-Nearest Neighbors*. All these approaches propose readily interpretable models. In what follows, we describe these *white-box* models and analyze their transparency using a simple 2-D binary classification example shown in Figure 2.2.

The decision tree model is a widely adopted ML model. It has a hierarchical tree structure with nodes and branches. At each node, the data is split into two separate parts based on the value of one feature, such that the loss function is optimized. The most popular decision tree algorithm is CART (Classification and Regression Trees) (Breiman et al., 1984). The split at each node is binary and is computed to optimize the Gini index.

Now let's consider that we train a CART decision tree to solve the running example classification problem (Figure 2.2), and that supposed decision tree is shown in Figure 2.3. The decision tree algorithm partitions the feature space and for the 2-D binary classification example, assuming that the classifier is perfectly accurate, can be represented as in Figure 2.4. Knowing the values $\alpha, \beta, \gamma$ we can reproduce the output of this decision tree for each data point in feature space and we can also divide the space in the same way the decision tree does it.

The next group of white-box models are rule-based models. Let's consider decision lists proposed by authors in Rivest (1987). These models consist of a list of *if . . . then . . . elseif . . . else* rules. Each item e.g. node in the decision list, is of size $k$, which means that each rule in the list represents a formula in conjunctive normal form of up to $k$ clauses. Contrary to a binary decision tree, where *True* branch can lead to another node, in the decision list, each *True* branch leads to a leaf node. Assuming that the decision list is trained such that it fits the data perfectly (same as we assumed for the decision tree above), it will partition the feature space the same as the decision tree (Figure 2.4) and the model is given as a list of rules in Figure 2.5.

Linear models are another set of simple, interpretable by design models. The output is computed as a linear combination of features. In the case of a binary classification task, the output of the model function is compared to a threshold in order to decide which class the given data-point belongs to. In a given 2-d classification scenario, a linear model function can be written as:

$$y = \alpha x + \beta, \tag{2.1}$$

where $\alpha$ is the weight attached to the feature $x$ and $\beta$ is the intercept. To make a classification for a certain data-point the output $y$ is computed and then compared with a threshold $r$, such that:

$$y = \begin{cases} 1, \textit{if } y \geq r \\ 0, \textit{if } y < r \end{cases} \tag{2.2}$$

Once the model has been trained and the parameters $\alpha, \beta \; and \; r$ are known, one can compute the output for each given data point. Thus, this model can also be replicated and depending on its parameters one can understand how changing the input value (in this case $x$) affects the outcome.

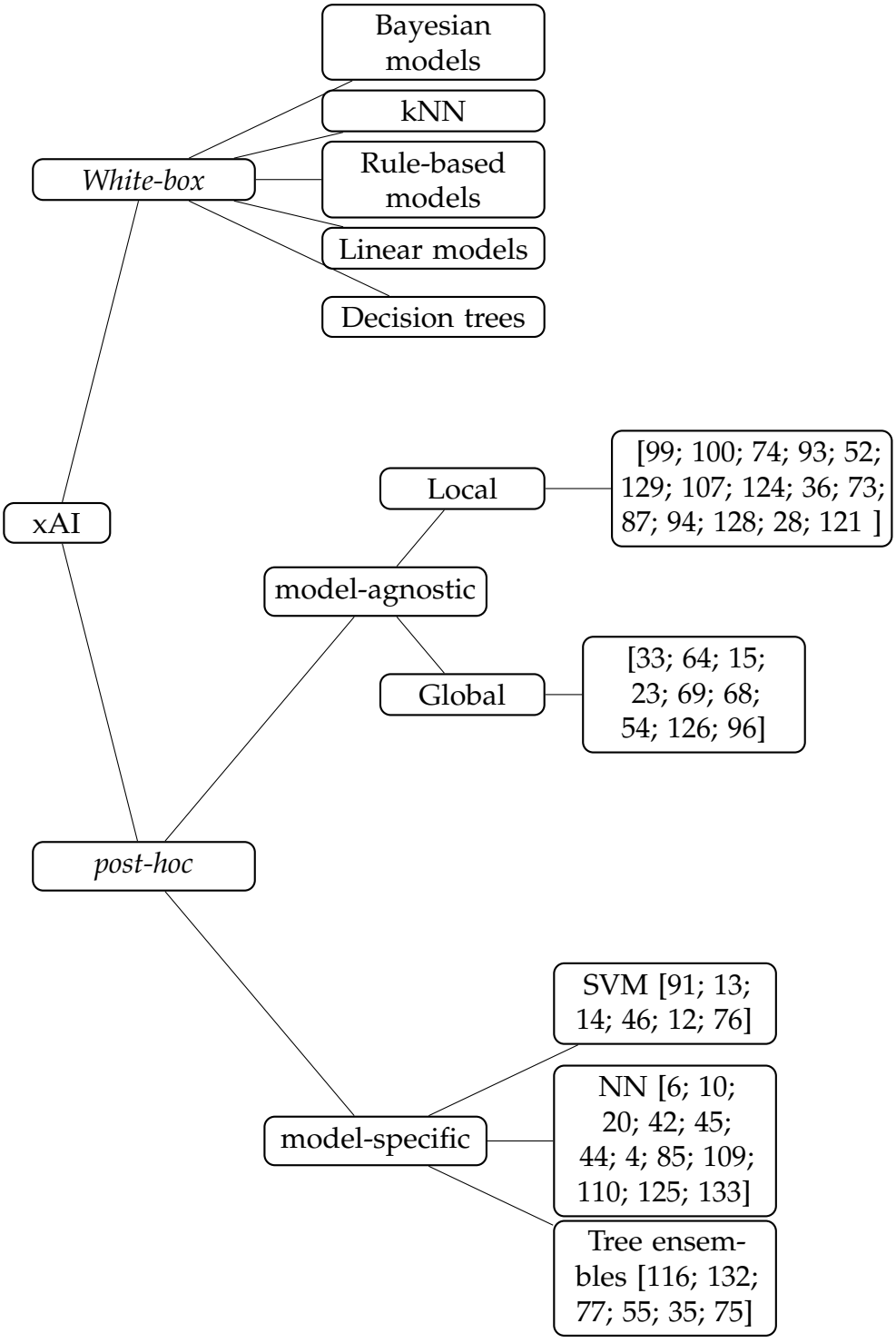



Figure 2.1: Taxonomy of xAI approaches

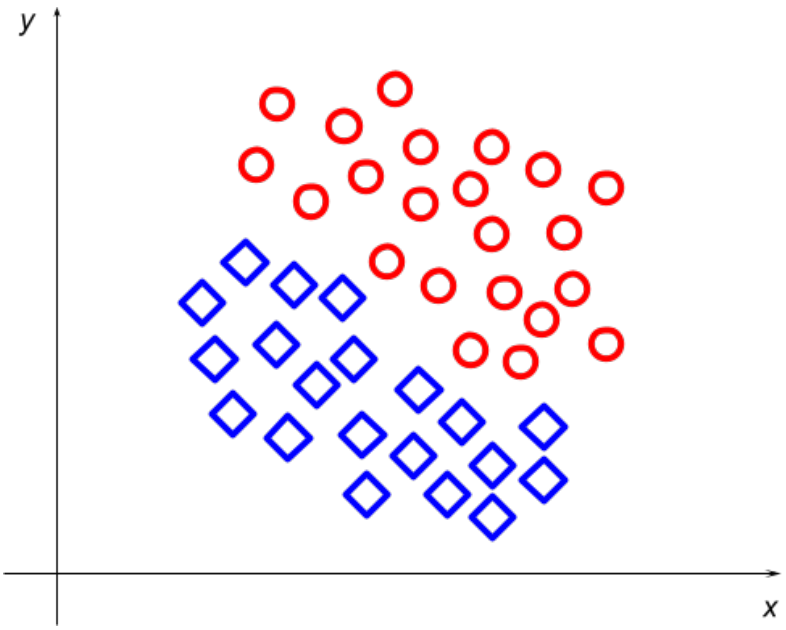



Figure 2.2: Running example: Binary classification

Bayesian models are probabilistic models, usually represented by directed acyclic graphs (DAG). The edges in Bayesian networks depict conditional dependencies between the variables. Based on the visual representation, one can easily reason about how each variable affects the outcome. In Figure 2.7 we show a prototype of a Bayesian network for our running example. For simplicity, we consider that this model has also perfectly learned the training dataset and that it partitions the space as shown in Figure 2.4, e.g. parameters $\alpha, \beta, \gamma$ are the same as for the decision tree. We then define two categorical variables $A, B$ such that:

$$A = \begin{cases} high, \textit{if } y > \alpha \\ medium, \textit{if } \gamma < y \leq \alpha \\ low, \textit{if } y \leq \gamma \end{cases} \qquad B = \begin{cases} 1, \textit{if } x < \beta \\ 0, \textit{otherwise} \end{cases}$$

According to the values of the features $x$ and $y$ the model assigns the probabilities for each of the classes. Having in mind that we assumed a perfect model, the probabilities for classes are either $0$ or $1$.

As the last group of *white-box* models, we mention *k-Nearest Neighbours* (*k*NN) classifier. This classifier outputs the most common class among the data points in some neighbourhoods. In the case of a regression problem, the output is computed by aggregation, e.g. average value. The size of the neighbourhood is defined through the parameter *k* and the neighbouring data points are chosen according to some similarity/distance metric. Given this similarity/distance

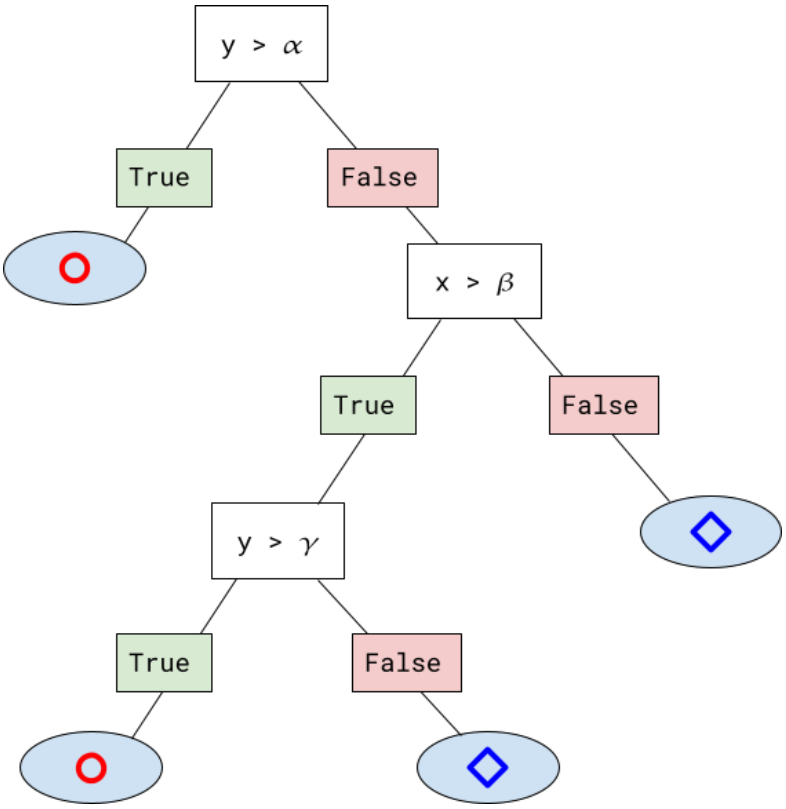



Figure 2.3: White-box model: Decision Tree

metric and the parameter $k$ one can compute the neighbourhood and replicate the output of $k$NN classifier. Let us consider a $k$NN classifier with $k = 5$ and the Euclidean distance as a distance metric. An example of how $k$NN model makes a prediction is shown in Figure 2.8. The neighbourhood of size $5$ is marked by the circle (Euclidean distance). The output of the model for a given data point (green cross) would be a red circle because $\frac{4}{5}$ neighbours (marked with numbers $1 - 5$) are red circles.

## 2.3 Post-hoc interpretability

White-box models come with transparency that is already integrated because of their intuitive architecture. On the other hand, for certain problems, the predictive performance of Deep Learning (DL) and ensemble models has been shown to be higher than with white-box models. Thus, the white-box models are preferred when the interpretability of the model is of crucial importance. In other cases, when predictive performance is the most important, one usually

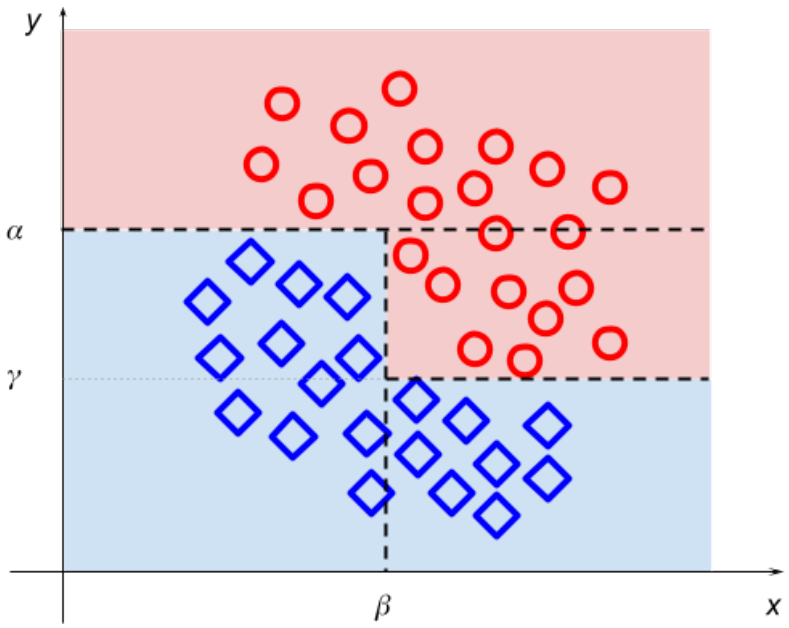



Figure 2.4: Tree/Rule -based classification boundary

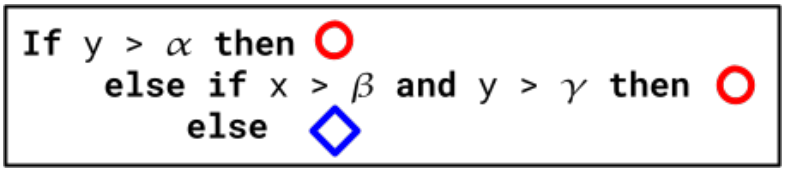



Figure 2.5: White-box model: Decision list

relies on DL and ensemble models. To implement the interpretability while not disrupting the predictive performance of the model we use *post-hoc* interpretation approaches. These approaches do not interfere with the inner workings of the black-box model, thus keeping its performance intact. Some approaches benefit from the architecture of the model to provide interpretations and those are model-specific approaches. These approaches are designed to interpret NNs, Tree ensembles or SVMs. The other group of approaches are more general and they don't utilize the architecture and can be applied to any black-box model. These methods are *model-agnostic*. We review the most prominent approaches from each of the two groups.

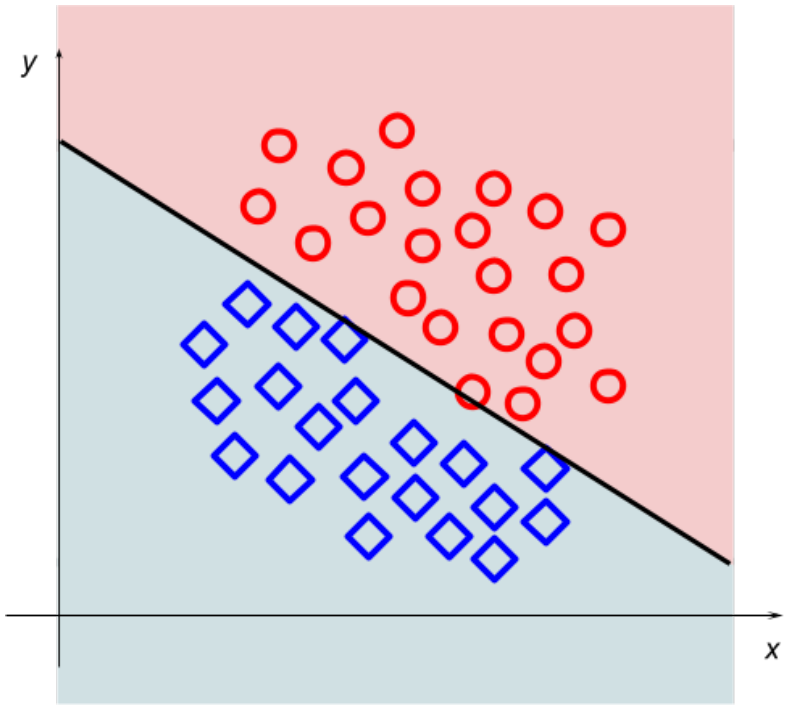



Figure 2.6: White-box model: Linear classifier

### 2.3.1 Model-agnostic methods

Model-agnostic interpretable methods can be applied to any *black-box* model. They do not rely on or interact with the inner workings of the *black-box* model, yet they mainly provide interpretations by learning the input-output relation through probing the model. There exist two main groups of model-agnostic approaches: local and global. Local model-agnostic methods provide interpretations for individual input data points by building a new interpretable model for each data point. Global model-agnostic methods can also provide explanations for individual data points but the main difference is that they train one interpretable model, on the whole space, and probe it with individual data points to provide interpretation. Global methods allow for easier interpretation of a black-box model's behaviour in general but they are less accurate than local interpretable models.

#### 2.3.1.1 Local model-agnostic methods

One of the most important works from this group of approaches for sure is LIME (Ribeiro et al., 2016). It stands for Local Interpretable Model-agnostic

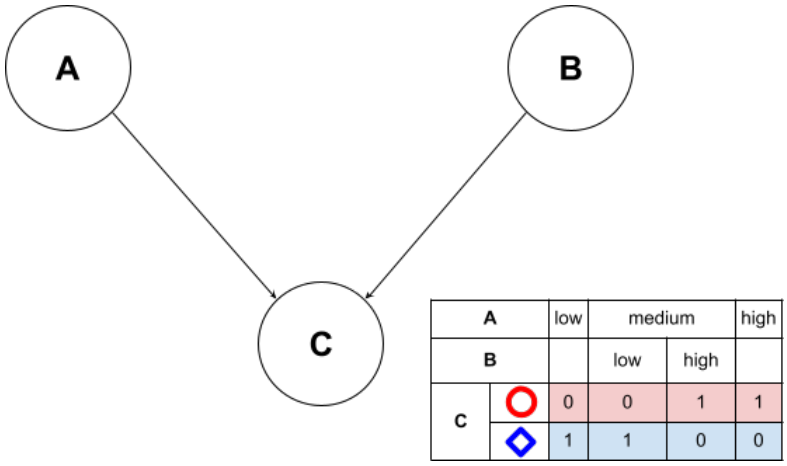



Figure 2.7: White-box model: Bayesian network

explanations and it trains interpretable simple local linear models around individual data points. It then provides an explanation in the form of a list of the most relevant features with their weights. The weights are proportional to the contribution of the features to the outcome probability. The main idea of LIME is to approximate the complex black-box model with a linear model locally, as shown in Figure 2.9. The algorithm synthetically generates new training data points, weighing them according to their distance to the original data point (the one in which the prediction is being explained). The further the synthetic data point is, its weight is smaller. LIME also provides a sampling algorithm, *sub-modular pick*, that combines multiple explanations for individual data points to provide a global explanation.

Another method, MAPLE (Plumb et al., 2018), combines the use of *SILO* (Bloniarz et al., 2016), the Random Forests approach to assign weights to the training examples and *DStump* (Kazemitabar et al., 2017), the feature selection method to assign weights to the features. It then goes on to train a local linear model to compute the explanation.

The same authors proposed another approach (Ribeiro et al., 2018), called Anchors. Anchors are also local approximations of the complex model that provide explanations in the form of sufficient *if . . . then* rules. These rules include only the features that influence the outcome, i.e. changing the value of other features will not have an impact on the outcome.

SHAP (SHapley Additive exPlanations) (Lundberg and Lee, 2017) introduces a game theory approach to compute the contribution of each feature. In coalition game theory, Shapley values are used to make a fair distribution of payout between the players. In the context of xAI, the features are considered as the

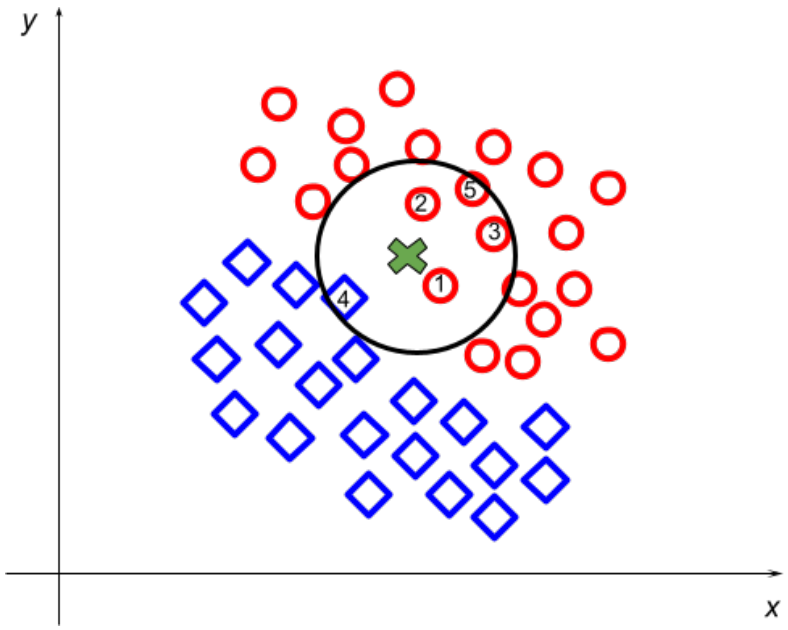



Figure 2.8: White-box model: *k*NN classifier

players, and the Shapley value for a feature corresponds to the average change in prediction when this feature is removed from a coalition. To compute the average change one needs to assess all possible coalitions. SHAP computes the attribution of all features, without removing any of them, which yields explanations that score high on fidelity but low on simplicity. In this work, the authors define an entire group of similar approaches, namely *"Additive feature attribution methods"*. This group of approaches provides an explanation that is a linear function of simplified input features projected to a binary space. This explanation is given with the following equation:

$$g(z) = \phi_0 + \sum_{i=0}^{M} \phi_i z_i, \tag{2.3}$$

where $z \in \{0,1\}^M$ and $M$ is the number of simplified input features. The authors go on to show that in this group fall methods such as LIME and DeepLIFT.

While previous approaches focused on training local models to mimic the behavior of the black-box model, and compute features' contributions, there is another group of approaches that aims to select important features for each prediction. Chen et al. (2018) propose a method *L2X* - Learning to explain, to select the important features. The feature selector maximizes the mutual information between the selected features and the target variable. While the number of features can be fine-tuned as a hyper-parameter, it is usually left

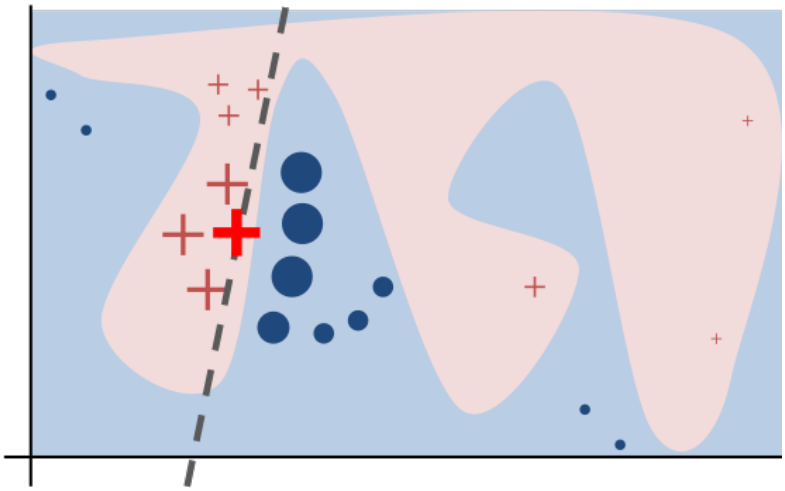


Figure 2.9: Intuition behind LIME

to the user to choose it. A related work by Yoon et al. (2018), uses actor-critic framework to train three neural networks to allow instance-wise feature selection. A recent work by Vo et al. (2022), combines additive feature attribution methods with instance-wise feature selection to compute explanations for multi-class classification problems.

LORE (Local Rule-Based Explanations) (Guidotti et al., 2018a) proposes using local interpretable models, specifically decision trees, from which local rule-based explanations are extracted. Each local interpretable model is trained on a dense set of synthetic data-points generated by a genetic algorithm. The interpretable model provides one rule to explain the decision (the branch of the tree with a given data-point) and several counterfactual rules for reversed decisions (the other branches of the tree).

All of these approaches train the interpretable models using synthetically generated data points. These data points are generated usually by random perturbation methods, which introduces uncertainty in explanations. To avoid this, a deterministic version of LIME has been proposed in Zafar and Khan (2019). DLIME uses hierarchical clustering (HC) to cluster the training data and then k Nearest Neighbours (kNN) to determine which cluster the data point, that's to be explained, belongs to. DLIME provides stable explanations but it requires additional input parameters: the number of clusters for HC and the number of neighbours for kNN.

There exists a specific subgroup of the local interpretation methods that provide counterfactual explanations. These methods usually compute the minimal change needed to be applied to a given data-point such that the black-box model would assign a different class label to it.

CLEAR (White and Garcez, 2019) is the first work that addresses the quality of the explanations, providing counterfactual explanations for each data point and measuring the fidelity of each explanation. It uses the idea of *b-counterfactuals*, which represents the minimal change in the feature in order to gauge the prediction of the complex model. Then, a regression model is fitted in the neighborhood of the data point to find the best explanation.

A more recent work by Dandl et al. (2020), proposes to use Multi-Objective Optimization to compute counterfactual explanations, both for classification and regression. The search for counterfactuals is formalized as a multi-objective optimization problem and the four criteria that constitute a *good* counterfactual are: data-point and its counterfactual are near in feature space, they differ only in a few features, the outcome for the counterfactual is close to the desired set of outcomes and counterfactual is plausible data-point.

Looveren and Klaise (2021) proposes using class prototypes to compute counterfactual explanations. These prototypes are obtained either using encoders or *k-d* trees. This helps to speed up the search process for counterfactual instances. Then, the prototype loss term is included in the objective function to produce more interpretable explanations.

Since counterfactual explanations determine a minimal change needed to revert the outcome, several works (Mothilal et al., 2020; Poyiadzi et al., 2020) argue that *feasibility* is important criteria for counterfactuals to satisfy. Mothilal et al. (2020) propose DiCE (Diverse counterfactual explanations) algorithm that computes the set of diverse and actionable counterfactuals, which are computed using determinantal point processes (DPP), which has been used for subset selection with diversity constraints. By proposing multiple diverse counterfactuals, they increase the probability that some of them will be actionable. On the other hand, Poyiadzi et al. (2020) propose FACE (Feasible and actionable counterfactual explanations) that first creates a graph over data-points and then the counterfactuals are selected based on the shortest path distances that are computed such that they account for the path length and the density over the path. Therefore, the counterfactual explanation doesn't necessarily represent the minimal change but the emphasis is on its feasibility.

#### 2.3.1.2 Global model-agnostic methods

Global interpretation methods try to summarize and interpret the behavior of the black-box model on a whole space. They still can be used to provide individual explanations but they are trained only once on a whole data space.

One of the first global methods was TREPAN (Craven and Shavlik, 1996), which queries the complex model in order to train a decision tree that mimics its behavior. The tree grows in the best-first manner, where the best node is the one

that increases the fidelity of the tree. Another specificity of this approach is that the splits can be based not only on a single feature but on multiple, which can increase the comprehensibility of the tree by reducing its depth.

Dectext (Boz, 2002) refined this idea by using C4.5 decision trees with different types of splits and a specific tree pruning strategy to improve its fidelity. Since the number of instances that are used to compute the best split decreases with the depth of the tree, they propose using a fixed number of instances to decide a split. If there are not enough training instances left, one can use unlabeled instances or generate synthetic ones, label them using the black-box model and use them in the training process of a decision tree. Another method based on decision trees by Johansson and Niklasson (2009) uses a genetic programming algorithm to evolve decision trees. They also propose using different combinations of training and test data to increase the fidelity of the decision tree to a black-box model.

A more recent global approach by Bastani et al. (2017) proposes DTExtract, a method that first fits a mixture of axis-aligned Gaussians to estimate the input distribution over features of the training dataset. Then the method builds binary decision trees iteratively, using an active sampling strategy. At each step of the training of the decision tree, new training data is generated under the constraints of previous nodes in that branch of the tree. Generating additional training data at each step has as its goal to ensure fidelity to the original black-box model.

GIRP (Yang et al., 2018) use a compact binary decision tree to extract the rules that black-box model has learned. The global interpretations are learned from local explanations for individual data points. This approach relies on the contribution matrix which consists of the contributions of input variables to predicted scores for each single prediction and which can be very difficult to obtain.

SP-LIME (Ribeiro et al., 2016) is an extension of the original LIME approach that uses a specific sub-modular pick algorithm to choose a set of local explanations to summarize the global behavior of the black-box model. The main idea of the sub-modular pick algorithm is to cover as many cases as possible with as few as possible local explanations.

K-LIME (Hall et al., 2017) is a variant of LIME (Ribeiro et al., 2016), that instead of using synthetically generated samples to explain local regions, uses clustering or user-defined regions. It provides local explanations of each of K clusters and it trains one global surrogate model. These models are trained to minimize $R^2$.

Apart from linear models and decision trees, there exist methods that use rule-based models for interpreting the black-box models. In Lakkaraju et al. (2016), authors propose interpretable decision sets as surrogate models for interpreting the behavior of the black-box model on the whole feature space. These

decision sets are sets of unordered classification rules and they are the result of optimizing the objective function that accounts for accuracy and interpretability. The same authors proposed MUSE (Model Understanding through Subspace Explanations) (Lakkaraju et al., 2019), which interprets the black-box model by explaining its behavior in subspaces defined by certain features. MUSE also produces explanations in the form of decision sets which consist of rules that are unambiguous and optimized for fidelity and interpretability.

As mentioned before, the main challenge when building global surrogate models is the trade-off between fidelity and complexity. Among the aforementioned methods, we saw different strategies to tackle this trade-off: limiting the number of nodes in the surrogate tree (Bastani et al., 2017; Craven and Shavlik, 1996), applying specific pruning algorithms (Boz, 2002), or stopping the growth of the tree when a node covers the instances of only one class with high probability (Craven and Shavlik, 1996).

### 2.3.2 Model-specific methods

The *model-specific* post-hoc approaches are built to interpret only a certain class of black-box models, contrary to *model-agnostic* methods that can interpret any model. They usually exploit the structure of the black-box model to compute explanations or to be more efficient. We review some of the important works in this area and we categorize them based on the class of black-box model that they interpret: Support Vector Machine (SVM), Neural Networks (NN) and Tree ensembles.

#### 2.3.2.1 Interpreting SVMs

Support Vector Machines (SVM) is a supervised machine learning algorithm that can be used for classification, regression and outlier detection. It uses support vectors to optimize the decision border between the classes. Support vectors are the data points that are the closest to the decision line or hyperplane. The main idea of SVM is to find an optimal line or hyperplane such that the distance to the support vectors is the largest. These hyperplanes can have very complex shapes and thus can be difficult to interpret. In the literature, there exist works that are built specifically to interpret the SVM models. Most of them try to extract rules that conform with the decision boundary of the SVM model.

SVM+Prototypes method has been proposed by Núñez et al. (2002) to extract *if-then* rules from SVM. This method uses a clustering approach to determine prototypes for each class. Then it combines these prototypes with support vectors to define regions (hyper-rectangles or ellipsoids) that are later translated into *if-then* rules.

Another approach (Barakat and Diederich, 2005) suggests using decision trees as a comprehensible model to interpret SVMs. They suggest creating a synthetic dataset where support vectors are labelled with already trained SVMs. This synthetic dataset is then used to train a decision tree which, in that way, represents the comprehensible interpretation of concepts learned by SVMs.

Barakat and Bradley (2007) propose SQRex-SVM method to extract rules from SVMs using sequential covering, which aims to cover as many as possible positive examples and as few as possible negative examples with each learned rule. The method learns the rules directly from correctly classified support vectors, minimizing the effects of noisy and misclassified data.

A recommender system approach was presented by Barbella et al. (2009). This approach defines a kind of similarity measure between a given data point and each support vector and then uses the support vector with the highest similarity to provide an explanation. Also, they propose a technique to compute a minimal change that moves the data point to the decision boundary. They refer to it as inverse classification, which in fact produces counterfactual explanations.

Focusing solely on linear SVMs, an efficient approach to extract rules was proposed by Fung et al. (2005). The algorithm is based on constrained optimization which makes it very efficient. Two optimization criteria have been proposed such that each rule covers the largest volume in feature space or the largest number of data points from the training set.

Active learning-based approach (ALBA) is used in Martens et al. (2008) to extract rules from SVMs. Since the highest noise is found in the regions near the decision boundary, this approach focuses more on these specific regions. It uses support vectors as proxies to generate additional data points in their vicinity using active learning. After this step, one can apply multiple rule induction techniques (C4.5, RIPPER (Cohen, 1995)) can be applied.

#### 2.3.2.2 Interpreting NNs

Neural networks (NN) are a machine learning model that represents the core of deep learning techniques. The structure of NN's is inspired by the structure of the human brain. They consist of, often, a large number of highly interconnected layers of nodes. The connections between the nodes have weights, which hold the knowledge. They show the ability to learn, memorise and generalize based on data that they have been trained on. They are used in many different applications and often reach state-of-the-art results. On the other hand, because of their complexity, they are opaque models and understanding how they reach the decision became of crucial importance. There have been many works on interpreting NNs and here we outline the most prominent ones.

One possible way of introducing interpretability to NNs is to alter their archi-

tecture such that each layer has a clear semantic meaning. One such approach is Self-Explaining Neural Networks (SENN) (Alvarez Melis and Jaakkola, 2018). They propose an architecture that first maps the input into a set of interpretable concepts, then a parametrizer that computes relevance scores for each concept. The last step is an aggregation function that combines the concepts and their scores to produce a prediction. Another approach that offers a specific architecture to ensure the interpretability in NNs is proposed in Angelov and Soares (2020), xDNN (explainable Deep Neural Networks). This approach consists in a specific architecture that extracts class prototypes, which are then used to form a list of interpretable *if-then* rules.

Several works try to extract interpretable rules from already trained NNs. One of the earlier works is KT method (Fu, 1994). The KT method performs a tree search of combinations of relevant attributes (the ones that make neurons fire) to generate a rule for each hidden and output node in the NN. OSRE (orthogonal Search-based Rule Extraction) (Etchells and Lisboa, 2006) fits Boolean rules to the smooth decision surface of NN. It assumes the discrete attributes which are 1-from-N encoded and performs the rule search by successive examination of orthogonal neighbours. In Augasta and Kathirvalavakumar (2012), authors propose Rule Extraction by Reverse Engineering the Neural network (RxREN). This approach uses reverse engineering to prune the neurons that aren't significant for classification and then constructs *if-then* rules for each class using only the significant neurons and data ranges that lead to correct classification. An extension of RxREN was proposed in Biswas et al. (2017), RxNCM (Rule extraction from NN using Classified and Misclassified data), that, as its name suggests, uses both correctly and incorrectly classified data to determine the ranges and significant neurons. In that way, they ensure that not only patterns that can be classified according to one attribute can be discovered, but also the common patterns, the ones that can be classified based on multiple attributes. DeepRED (Zilke et al., 2016) algorithm extracts the rules from each layer of a NN. Then all the rules for one class are merged into a rule set that describes the given class. In the merging process, unsatisfiable rules and redundant terms are deleted and for simplification purposes, the rules are pruned similarly as in RxREN.

Similarly, as with decision trees, we can combine multiple NNs to achieve better results in certain tasks, so called NN ensembles. These models are also opaque and REFNE (Zhou et al., 2003) method has been proposed to interpret them by extracting rules. It looks for values of categorical attributes that ensure that all instances that possess that value belong to the same class, to generate a rule. The process continues by examining the combinations of attributes. To ensure comprehensibility the length of the rules is limited to 3.

The method proposed in Montavon et al. (2017) uses back-propagation to decompose the complex decision function of NN on its inputs. It uses Taylor decomposition to factorize the function of each neuron to its inputs. As the last step, this decomposition results in the relevance score of each pixel that is then visualized in the form of a heatmap, that represents the explanation. Even though by nature this approach applies to image data it can be applied to other data types. Another approach to producing visual explanations of CNNs (convolutional neural networks) is proposed in Selvaraju et al. (2016). Grad-CAM (Gradient-weighted Class Activation Mapping) discovers and visualizes important regions for each class. Deep-LIFT (Shrikumar et al., 2017) introduces concepts of reference input and output. It uses the differences between current and reference input to explain the differences between current and reference output. The reference input is the default one, chosen according to the problem at hand.

A set of methods tries to represent the knowledge contained in trained NN with an interpretable model, namely a decision tree. In Frosst and Hinton (2017) a method for distilling a NN into a soft decision tree has been proposed. The decision tree is trained to mimic the input-output behavior of NN using stochastic gradient descent. To train a decision tree one can use predictions of NN on unlabelled data or, use various data generation methods to generate synthetic data. A method proposed in Wu et al. (2018) includes an additional regularization parameter in the objective function for NN training. In each iteration a decision tree is trained to mimic the behavior of NN and the average path length is computed on a selected part of the dataset. In that way, the optimization algorithm will prefer the models that can be approximated and interpreted with simpler decision trees.

#### 2.3.2.3 Interpreting tree ensembles

Another powerful, yet opaque, group of models are tree ensembles. Tree ensembles, as the name suggests, are the models that combine multiple decision trees into one single model for better predictive performance. Some examples of tree ensemble models are Random Forests (RF), AdaBoost trees, and Gradient Boosted trees. These models are used both for classification and regression. There exist several approaches that are tailored to exploit the specific tree structure of these models, to interpret their predictions.

In Cui et al. (2015), authors address the problem that tree ensembles don't provide an interpretable actionable plan to alter their outcome. Thus they propose a method to extract actionable knowledge from a tree ensemble. This approach computes which feature values should be changed to alter the outcome of the black-box tree ensemble method while minimizing some cost function, e.g. the

number of features changed. A similar idea of tweaking feature values has been proposed in Tolomei et al. (2017). The algorithm focuses only on decision trees in the ensemble that have predicted a negative class (binary classification scenario). Similarly to the previous approach, it tries to find the set of feature values that need to be changed to achieve the positive output of the ensemble model, while minimizing a cost function.

On the other hand, Hara and Hayashi (2016) argues that the main source of the lack of transparency of tree ensemble models is the number of regions that it creates in feature space. They propose to learn a simple model, with a limited number of regions, while minimizing the model error.

Since the tree-based models can be easily rewritten as sets of rules, three rule extraction algorithms from tree ensemble models have been proposed in Mashayekhi and Gras (2017). All three approaches start from the same set of rules, which represent nodes, under certain constraints, from decision trees in the ensemble. Then in the first algorithm, a heuristic search is employed to find the set of rules optimizing the accuracy and coverage. The second approach converts the rule extraction problem to a regression problem and uses the Sparse Group Lasso method, which finds a small group of features that optimizes for sparsity and prediction error. Finally, the third approach is the adaptation of the second one for multi-class classification problems.

TreeExplainer (Lundberg et al., 2020) represents an extended version of SHAP (Lundberg and Lee, 2017) specifically tailored to interpret tree ensemble models. utilizing the tree structure it allows for exact computation of SHAP values. Also, they propose a method to combine local explanations with global ones, to interpret the behaviour of the model on the whole space.

Different visualization techniques are often used to improve interpretability of black-box models, e.g. feature importance, partial dependency plots (PDP), and decision paths. All these techniques are contained in iforest (Zhao et al., 2018) framework. The framework aims to reveal the relationships between features and predictions, to discover underlying working mechanisms (e.g. open the black box) and to allow for case base reasoning.

# 3 STACI

We propose a novel method to interpret the classification results of a black box model a posteriori. We emulate the complex classifier by surrogate decision trees. Each tree mimics the behavior of the complex classifier by overestimating one of the classes. This yields a global, interpretable approximation of the black box classifier. Our method provides interpretations that are at the same time general (applying to many data points), confident (generalizing well to other data points), faithful to the original model (making the same predictions), and simple (easy to understand). This chapter is based on our paper: *Confident interpretations of black box classifiers* (Radulovic et al., 2021) and its extended version (Radulovic et al., 2023a), available as a preprint.

## Contents

## 3.1 Introduction

STACI is a new deterministic approach to interpreting the behavior of complex black-box models. Since it uses decision trees as surrogate models it also allows extraction of interpretations of individual predictions. STACI provides concise and accurate interpretations, and as we have discussed in section 1.4, our approach takes into account the generality of an interpretation. In fact, in this chapter, we argue that aside from accuracy and simplicity, a third important metric is *generality*: the number of data points that the interpretation applies to. If an interpretation applies to more data points, it appears less ad-hoc to a user. The interpretation should also have a high *confidence*, i.e., all data points concerned by the interpretation should be classified in the same way.

It is intuitively clear that these desiderata are pitted against each other: Higher fidelity means higher complexity. This problem is known as the *comprehensibility-complexity trade-off*. In the same spirit, higher generality means lower confidence (akin to the precision-recall trade-off). Finally, higher generality at low complexity also means lower fidelity.

In this chapter, we propose a new methodology that addresses this impasse: We propose to mimic a given black box classifier not by a single surrogate model, but by several – one for each class. In this way, each of the models can be simple while their combination still has high fidelity. Our method has not just a high fidelity and a low complexity, it also provides very general interpretations with high confidence.

Our main contributions are as follows:

- We develop an abstraction of surrogate models, and formalise the quality metrics of surrogate models.
- We present our method STACI[1], which learns surrogate models that are at the same time simple, general, confident, and faithful to the original.
- We perform an extensive empirical evaluation on several popular datasets from the UCI Machine Learning Repository, showing that STACI outperforms other state-of-the-art methodologies in these desiderata.
- We perform a user study that shows that users prefer the interpretations of our method over others.

The rest of the chapter is organised as follows. Section 3.6.1 discusses related work. Section 3.3 presents our new method, STACI. Section 3.6 evaluates our method on several datasets and compares it to a baseline and the state of the art.

[1]Surrogate Trees for A posteriori Confident Interpretations

## 3.2 Desiderata

**Post-hoc Interpretation.** We are given a black box multi-class classification model. We wish to make it interpretable *post-hoc*, i.e., after the model has been trained. There is considerable debate about the meanings of the terms "explainable" and "interpretable" (Beaudouin et al., 2020; Guidotti et al., 2018b; Došilović et al., 2018; Adadi and Berrada, 2018; Doshi-Velez and Kim, 2017). In this chapter, we aim at interpretability in the following sense: *We want to provide a meaning for the results of a model in terms that are understandable to humans* (Doshi-Velez and Kim, 2017).

**Local vs. global interpretations.** Local interpretations help us understand the classification of one given input data point ("Why does the model predict that this particular patient should undergo chemotherapy?"). While these interpretations can be very tailored, they are less well adapted for scenarios where the model is used repeatedly: Local interpretations can be unstable and can provide very different interpretations even in a very close neighborhood (Alvarez-Melis and Jaakkola, 2018). Individual interpretations may also be contradictory to each other (Tan et al., 2018). Global interpretations, in turn, consider the model as a whole ("What are the criteria that make patients more likely to be recommended chemotherapy in general?"). Such interpretations can also help understand an individual classification, but they are more geared towards an understanding of the model as a whole. In this chapter, we study global interpretations.

**Interpretation Models.** Formally, we aim at global post-hoc interpretations of the following form:

**Definition 3.2.1.** *Given a set $\mathcal{S}$ of labeled data points and a labeled input point $i \in \mathcal{S}$, an* interpretation *of point $i$ is a set of conditions that $i$ satisfies so that the majority of the data points in $\mathcal{S}$ that satisfy these conditions carry the same label as $i$.*

An *interpretation model* is then a model that can provide such interpretations. This definition applies to a wide variety of models, be it decision trees, Bayesian rule lists, linear models, or our own method. Let us now discuss some quality metrics of such models.

**Fidelity.** All post-hoc approaches have the problem that the interpretation model usually deviates from the black box model because it has to be simpler than the black box model. If the model deviates for a given point, the approach cannot deliver an interpretation for this point.

Thus, fidelity is just the ratio of points on which the surrogate model agrees with the complex model. In the case of a decision tree trained on $\mathcal{S}$, the fidelity is just the weighted average confidence of the leaf nodes.

**Confidence.** An interpretation will identify some characteristics of the input point, and say that the majority of points with these characteristics are classified in a certain way. Naturally, such an interpretation is more convincing when that majority is larger. To quantify this intuition, we define the notion of confidence:

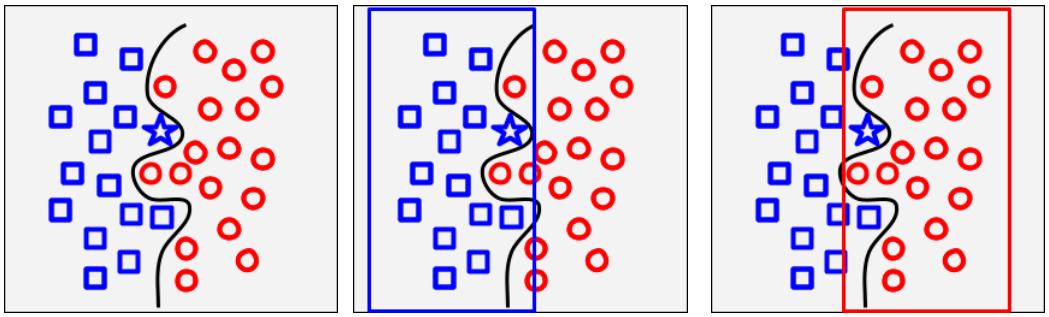
**Definition 3.2.2.** *Given an interpretation model $\mathcal{M}$, a labeled dataset $\mathcal{S}$, and a labeled input data point $i \in \mathcal{S}$ that the model can interpret, the* confidence *at point $i$ is the ratio of data points of $\mathcal{S}$ that satisfy the interpretation of $i$ and share the label of $i$ over the data points of $\mathcal{S}$ that satisfy the interpretation of $i$.*

This definition can be generalized to the *average confidence of the model $\mathcal{M}$* on the set $\mathcal{S}$, which is simply the average confidence for all points of $\mathcal{S}$. In the case of a decision tree trained on $\mathcal{S}$, the average confidence is just the weighted average confidence of the leaf nodes (and thus identical to fidelity). In general, however, the fidelity gauges the percentage of data points where the model applies (no matter their class). The confidence, in contrast, measures whether the data points concerned by the interpretation are of the same class. In this way, confidence corresponds to the precision of the interpretation of the set of cases to which it applies. Optimizing only confidence for a given class may reduce the fidelity.

**Generality.** An interpretation will be more convincing if it applies to more input data points. For example, imagine an interpretation model in the medical domain that provides interpretations based on the social security number of the patient. This model will have high fidelity and high confidence because it just "explains" the illness of the patient by her social security number. To avoid such interpretations, we need the notion of generality:

**Definition 3.2.3.** *Given an interpretation model $\mathcal{M}$, a labeled dataset $\mathcal{S}$, and a labeled input data point $i \in \mathcal{S}$ that the model can interpret, the* generality *at point $i$ is the ratio of data points in $\mathcal{S}$ that satisfy the interpretation of $i$ and share the label of $i$ over all the data points of $\mathcal{S}$ that share the label of $i$.*

The larger that percentage, the more satisfactory the interpretation will be. We define generality as a measure relative to the class size in order to be scale-invariant, and in order to guard against skewed class distributions. Generality thus corresponds to the recall of the interpretation on the set of all data points with the same label.



(a) Complex model (b) Interpretable model for left class (c) Interpretable model for right class

Figure 3.1: In order to approximate the complex decision boundary of the black box model (Figure 3.1a), we train two specialised surrogate models: One, shown in Figure 3.1b, overgeneralizes the "square" class. The other, shown in Figure 3.1c, overgeneralizes the "circle" class.

**Complexity.** The goal of an interpretation model is to provide interpretations that are as simple as possible. Intuitively, the complexity of an interpretation corresponds to the number of conditions it contains: Simpler interpretations have fewer conditions. Formally, the *complexity* of an interpretation depends on the type of the interpretation model. For decision trees, the complexity of an interpretation is usually the length of the path from the root to the leaf node (Guidotti et al., 2018b). The worst-case complexity of a tree is the maximal depth.

## 3.3 Method's Approach

**Goal.** Our approach receives as input a black box classification model. It produces as output a surrogate model, which makes (by and large) the same predictions as the black box model but is simpler and thus easier to understand. When this surrogate model is presented with a new data point of a class $C$, it will produce an interpretation such as: *"This data point has the characteristics $X, Y, Z$ and was classified as $C$. There are 500 other data points with these characteristics and 80% of them are also classified as $C$."*

**Approach.** The key idea of our method is to generate not a single surrogate model but one specialized surrogate model for each class. Each specialized model is trained to overgeneralize its class. Figure 3.1 exemplifies this for a binary classification model: model (a) is the black box model. Model (b) overgeneralizes the "square" class, while model (c) overgeneralizes the "circle" class. We then interpret an incoming data point in two steps:

1. We first classify the data point by the black box model. This may appear to be cheating, but the goal is not to replicate the black box model in its absence. Rather, the goal is to interpret a prediction of the black box model. Thus, this prediction is necessarily available. In Figure 3.1, if we receive the point marked by a star, we classify it as "square".

2. We then use the specialized surrogate model for that class to provide an interpretation. In the example, we use the model of Figure 3.1 (b). The result is an interpretation such as: This data point has the characteristics of Model (b), and was classified as "square"; there are 30 other data points with these characteristics, and 80% of them are also classified as "square".

**Fulfilment of the Desiderata.** For the specialized models, we use decision trees with limited depth. This entails that our interpretations have a *low complexity*. Since we have one specialized tree per class, their combination is still sufficiently complex to approximate the black box model. We thus achieve a *high fidelity*. Finally, our trees are constructed in such a way that they maximize both the percentage of correctly classified points (the *confidence*) and the percentage of points of the target class (the *generality*).

In general, training more trees leads to higher confidence, less complexity per tree, and higher fidelity. Thus, one could think that one should simply train many more trees. However, more trees lead to a decrease in generality. If we train only one tree per class, in contrast, this does not lead to a decrease in generality. This is because generality is computed per class. Furthermore, one tree per class satisfies our goal of providing global interpretations: A single tree provides a human-understandable interpretation of a single, entire class.

## 3.4 Training algorithm

We are given a black box $N$-class classification model, and a set $\mathcal{S}$ of data points. We want to construct a surrogate model that can interpret these points. As is customary, we use the black box model to label the data points $\mathcal{S}$. On this labelled dataset, we train one decision tree per class in a *one-vs-all* fashion.

Our goal is to train our class-specific trees in such a way that they maximize confidence and generality. For this purpose, we do not use standard metrics such as the Gini Impurity Index or the Information Gain when deciding a split on a node in the decision tree. Rather, we employ the *F1 score*.

The F1 score is a measure of the model's accuracy on the dataset and is defined as a harmonic mean of precision and recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.1}$$

Precision and recall are defined for a binary classification scenario. Precision is the ratio of true positives ($TP$), i.e. of positive data points that the model has classified as such, and overall data points that the model has classified as positive. The recall is the ratio of true positives, i.e., the ratio of positive data points that the model has classified as such, overall positive data points in the dataset:

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

where $FP$ and $FN$ represent false positives and false negatives, respectively.

If we recall the definitions of *confidence* (Def. 3.2.2) and *generality* (Def. 3.2.3), we can see that they directly correspond to the definitions of precision and recall, respectively, in our scenario of one tree per class. It is worth mentioning that optimizing only one of these two metrics would lead to pathological solutions. Optimizing only precision would result in splits that are highly confident but that would not cover a lot of data points (i.e., the generality would be low). In the extreme case, the split would single out only one data point, resulting in a confidence of $100\%$. On the other hand, optimizing only for recall, the split would tend to encapsulate as many points of one class as possible, including many points of the other classes, thus significantly degrading the confidence. The F1 score is the harmonic mean of precision and recall and is thus a natural metric to optimize both desiderata together. It complies with our approach in multiple aspects:

- It is an asymmetric metric (false positives and false negatives count differently), and thus it fits our strategy of training one surrogate model per class in a *one-vs-all* manner.
- It is a widely used metric for the evaluation of classifiers, and optimizing it will thus, by definition, not harm *fidelity*.
- It optimizes both *confidence* and *generality*.
- It does not require any user-defined parameters.

In fact, our training algorithm has just one user-defined input: the desired complexity of the interpretation, i.e. the maximal depth of the surrogate decision trees. Our method trains one decision tree per class on the labelled $\mathcal{S}$, using the F1 score to decide node splits, and limiting the depth of the trees as specified by the user. The tree growth stops when the gain in the split metric is less or

equal to zero or when the maximal depth has been reached. The output of our algorithm is a set of decision trees, one for each class.

Let us now discuss how we can provide an interpretation for a given input data point. Algorithm 1 receives as input the black box model $M$, a data point $x$, and the surrogate trees $\{T_c\}_{c=1}^{N}$ for each of the $N$ classes. We first use $M$ to classify $x$. Then we check if the corresponding surrogate tree $T_c$ agrees with $M$. If so, we produce an interpretation: The characteristics of the data point can be read off the path from the root of $T_c$ to the leaf for $x$. The generality and confidence can be found directly at the leaf node. If $T_c$ does not agree with $M$ on $x$, we are in an area that fell victim to the constraint of limited complexity, and we cannot provide an interpretation. (If many data points fall in this area, fidelity suffers.)

**Algorithm 1** Interpret with STACI

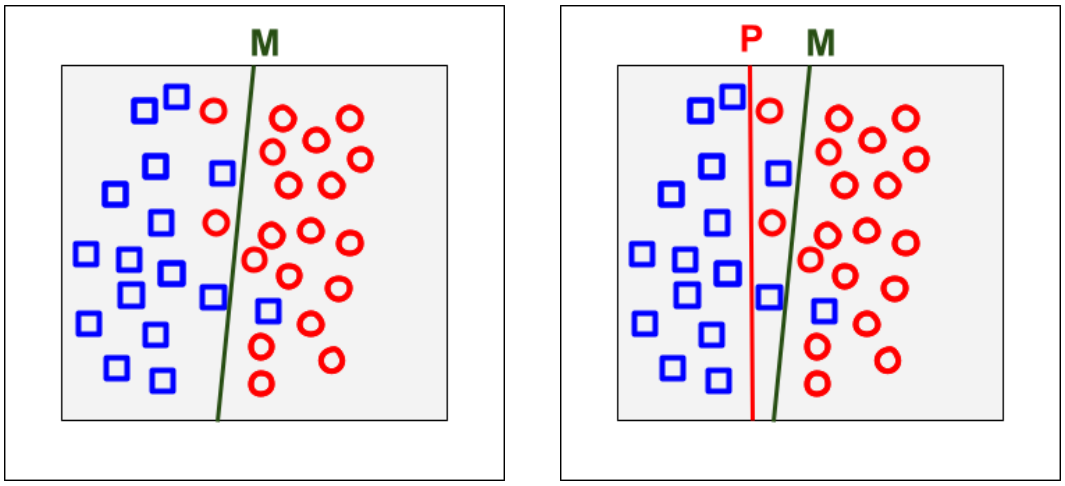
**Input**: black box Model: $M$
Data point $x$
Surrogate trees $\{T_c\}_{c=1}^{N}$

1: $c = M(x)$
2: **if** $T_c(x) = c$ **then**
3: **return** "This data point has the characteristics $T_c$.*pathFor*$(x)$, and is classified as $c$. There are $T_c$.*numSamples*$(x) - 1$ other data points with these characteristics, and $T_c$.*leafConf*$(x)$% of them are also classified as $c$."
4: **end if**
5: **return** "STACI cannot provide an interpretation."

**Theorem 3.4.1.** *Let $M$ (Figure 3.2a) be an interpretable model trained on a labeled dataset $D$ with 2 classes – $C_0$ and $C_1$. Let's denote correctly classified instances as $TC0_M$ and $TC1_M$ and wrongly classified ones as $FC0_M$ and $FC1_M$. Then let $P$ (Figure 3.2b) be an interpretable model, trained on the same dataset $D$, such that $M \neq P$. Similarly, let $TC0_P$ and $TC1_P$ be the sets of correctly classified and $FC0_P$ and $FC1_P$ of wrongly classified instances. Now, consider a case where model $M$ is not perfect, i.e.*

$$F1_M^{C1} = \frac{2|TC1_M|}{2|TC1_M| + |FC1_M| + |FC0_M|} < 1 \wedge F1_P^{C1} \geq F1_M^{C1}, \tag{3.4}$$

*where $F1_M^{C1}$ is $F1$ score for model $M$ when $C_1$ is considered as positive class and analogously $F1_P^{C1}$ is the same for model $P$. We build a* STACI *model as $M + P$, such that model $M$ interprets the instances of class $C_0$ and model $P$ interprets instances of class $C_1$. Then the* STACI *model has a higher fidelity and higher average $F1$ measure than model $M$.*




(a) Single model (b) Two models

Figure 3.2: Binary classification scenario

*Proof.* We will first provide the proof for fidelity and then for the $F_1$ measure.

1° When only one model is available, $M$ (Figure 3.2a), fidelity is actually the ratio of correctly classified instances:

$$fidelity_M = \frac{|TC0_M| + |TC1_M|}{|D|} \tag{3.5}$$

2° After adding the second model $P$(Figure 3.2b), the fidelity can be computed as:

$$fidelity_{M+P} = \frac{|TC0_M|}{|D|} + \frac{|TC1_M|}{|D|} + \frac{|TC0_P \setminus TC0_M|}{|D|} + \frac{|TC1_P \setminus TC1_M|}{|D|} \tag{3.6}$$

We want to prove that $fidelity_{M+P} > fidelity_M$. From Equations 3.5 and 3.6, we have:

$$\frac{|TC0_M|}{|D|} + \frac{|TC1_M|}{|D|} + \frac{|TC0_P \setminus TC0_M|}{|D|} + \frac{|TC1_P \setminus TC1_M|}{|D|} > \frac{|TC0_M| + |TC1_M|}{|D|} \tag{3.7}$$

Removing common factors we get:

$$|TC0_P \setminus TC0_M| + |TC1_P \setminus TC1_M| > 0 \tag{3.8}$$

Which, taking into account the assumption that model $M$ is not perfect, is always true since we can create model $P$ by just changing one of the wrong predictions of the model $M$.

Now we provide the proof for the average $F1$ measure:

1° We compute the average $F1$ metric for the scenario in Figure 3.2a, as:

$$AverageF1_M = \frac{|TC0_M|F1_M^{C0}}{|D|} + \frac{|TC1_M|F1_M^{C1}}{|D|} \tag{3.9}$$

2° For the scenario in Figure 3.2b, the average $F_1$ measure is given by:

$$\frac{|TC0_M|F1_M^{C0}}{|D|} + \frac{|TC1_P|F1_P^{C1}}{|D|} + \frac{|TC0_P \setminus TC0_M|F1_P^{C0}}{|D|} + \frac{|TC1_M \setminus TC0_P|F1_M^{C1}}{|D|} \tag{3.10}$$

Comparing $1°$ and $2°$ we get:

$$|TC1_P|F1_P^{C1} + |TC0_P \setminus TC0_M|F1_P^{C0} + |TC1_M \setminus TC1_P|F1_M^{C1} > |TC1_M|F1_M^{C1} \tag{3.11}$$

According to our assumption: $F1_P^{C1} \geq F1_M^{C1}$, we can exchange:

$$F1_M^{C1}|TC1_P| + |TC1_M \setminus TC1_P| - |TC1_M| + |TC0_P \setminus TC0_M|F1_P^{C0} > 0 \tag{3.12}$$

This gives us two conditions:

$$|TC1_P| + |TC1_M \setminus TC1_P| - |TC1_M| > 0 \vee |TC0_P \setminus TC0_M| > 0 \tag{3.13}$$

Which results in the same conditions as in the proof for fidelity:

$$|TC1_P \setminus TC1_M| > 0 \vee |TC0_P \setminus TC0_M| > 0 \tag{3.14}$$

This comes down to the same conclusion as in the case of fidelity, proving that adding additional model $P$ improves both fidelity and $F1$ measure. □

## 3.5 Regression

The STACI method that has been presented in 3.3 naturally supports only classification problems. The main idea of the approach is to train one interpretable model per class and optimize the $F1$ measure, which accords directly and only to the classification task. In this subsection, we describe the extension of our approach to enable the support for regression tasks.

Contrary to the classification, where the task is to assign a discrete class label to the input data, in the regression task we assign a real value to each input data point. For example, the regression tasks are: predicting the price of real estate, the number of people visiting a certain event, electricity power consumption,

etc. To enable our approach to support regression tasks, we adopt the idea of solving the regression by classification proposed by Torgo and Gama (1996). The main idea is to map the regression task to a classification one, by dividing the original continuous class values into a series of intervals, and then each of these intervals will represent one discrete class. This step can be considered as a data pre-processing step. After mapping the continuous class value into discrete class labels, we train the interpretable models in the same way as for the classification task. The prediction is also made similarly to in the classification task, on the leaf of the decision tree. Instead of the discrete class label, the median value is computed on the leaf. In this way, the accuracy of the predictions doesn't depend only on the number of discrete intervals, but it can also be improved by increasing the tree depth.

The mapping of continuous values to discrete class labels can be seen as a clustering task. One of the main caveats here is the fact that usually, we don't know the true number of clusters. To overcome this challenge we propose several scenarios. Depending on the level of the domain knowledge, the user can choose to provide:

- The width of the intervals i.e. the range, in which case we apply *Equal width intervals* (*EW*) division strategy. The whole range of the target variable is divided in $n$ intervals of the same width, regardless of the number of points inside each interval.

- The size of intervals i.e. the number of points that each interval contains, in which case we apply the *Equal frequency intervals*($EF$) division strategy. Contrary to the previous case, the width of the intervals can vary, but the number of data points in each interval is approximately the same.

- The number of intervals i.e. the number of discrete classes. In this case, the user provides the number of intervals $n$, based on which we compute intervals using both, previously mentioned approaches (*EW* and *EF*) and *K-means* (Hartigan and Wong, 1979). For the first two approaches, we compute the width of the interval (*EW*) and the number of data points pre-interval (*EF*), based on the given number of intervals $n$. For each approach, we compute the *Silhouette score* (Rousseeuw, 1987), and select the division strategy that achieves the highest value. The *Silhouette score* represents the cluster validation metric and it is computed using the following equation:

$$Silhouette\ score = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{3.15}$$

  where $a(i)$ is the average distance between the data point $i$ and the rest of the data points of the same cluster, and $b(i)$ is the minimum average

distance between data point $i$ and all data points in each cluster that $i$ doesn't belong to. The *Silhouette score* takes values in range $[-1, 1]$. The *Silhouette score* of $1$ means that the clusters are well apart.

- Maximal percentage error of discretization, which is computed as weighted mean absolute percentage error (*WMAPE*), which represents a variation of mean absolute percentage error (*MAPE*), adjusted to avoid infinite values of error when the true value is equal to $0$:

$$\text{WMAPE} = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} |y_i|} \tag{3.16}$$

  In this case, we increase the number of intervals until the *WMAPE* is smaller or equal to the given maximal allowed error. Similar to the previous case, we employ all three methods for mapping and choose the one that first reaches the requested maximal allowed error rate. If multiple approaches reach the requested error rate at the same time, we choose the approach that achieves the highest *Silhouette score*.

- No input at all. There are many situations when the user is not familiar with the data and lacks domain knowledge, which would help them to propose how the continuous values should be mapped into discrete classes. If this is the case, there is no need for the user to provide any of the previously mentioned parameters. We employ the non-parametric clustering method *UNIC* proposed by Leopold and Rose (2020). The *UNIC* method computes the cluster borders and the optimal number of clusters, based on the distances between several arbitrarily chosen points and the rest of the dataset. We then use the number of clusters acquired using the *UNIC* algorithm to compute the clusters using previously mentioned methods: *EW*, *EF*, *K-means*. Additionally, we explore the clustering results for several different numbers of clusters, in the neighbourhood of the one proposed by the *UNIC* algorithm. We compare the methods, again, using *Silhouette score* and select the best one.

## 3.6 Experiments

We perform an extensive experimental evaluation of our approach on the datasets of the UCI Machine Learning repository. We compare the performance of our approach with the state of the art method DTExtract (Bastani et al., 2017), the interpretable-by-design method SBRL (Yang et al., 2017), LIME (Ribeiro et al., 2016), and CART (Breiman et al., 1984) as a baseline. We have also conducted a

user study to validate the desiderata defined in Section 3.2 and to determine the users' preferences.

### 3.6.1 Competitors

One of the first global methods was TREPAN (Craven and Shavlik, 1996), which queries the complex model in order to train a decision tree that mimics its behavior. Dectext (Boz, 2002) refined this idea by using different types of splits in the decision tree and a specific tree pruning strategy to improve its fidelity. Another method based on decision trees (Johansson and Niklasson, 2009) uses a genetic programming algorithm to sample new data points, which are then used to learn the behavior of the black box model. A recent global approach (Bastani et al., 2017) proposes DTExtract, a method that first fits a mixture of axis-aligned Gaussians to estimate the input distribution over features of the training dataset. Then the method builds binary decision trees iteratively, using an active sampling strategy.
As mentioned before, the main challenge when building surrogate models is the trade-off between fidelity and complexity. There are different approaches to tackle this trade-off: limiting the number of nodes in the surrogate tree (Bastani et al., 2017; Craven and Shavlik, 1996), applying specific pruning algorithms (Boz, 2002), or stopping the growth of the tree when a node covers the instances of only one class with high probability (Craven and Shavlik, 1996).
In contrast to all of these works, our approach builds not a single surrogate model, but one per class. This allows us to achieve high fidelity and low complexity without corrupting the resulting models through pruning. In our experiments, we compare our approach to the state of the art global post-hoc method, DTExtract (Bastani et al., 2017). We also compare to a model that is interpretable by design (Scalable Bayesian Rule Lists (Yang et al., 2017)), to LIME (Ribeiro et al., 2016), and to CART (Breiman et al., 1984) as a baseline.

### 3.6.2 Settings

**Datasets.** All the datasets used in our experiments are publicly available at the UCI Machine Learning Repository (Dua and Graff, 2017). Table 3.1 shows their key characteristics.
**Metrics.** We report the four metrics from Section 3.2: The complexity is the average depth of the path for the decision trees, the number of rules for SBRL, and the number of features in the explanation for LIME. The confidence is the

[2]We used a subset of Adult dataset, removing less relevant features (race and native-country)

Table 3.1: Datasets

| Dataset | Features | Num. | Cat. | Classes | Instances |
|---|---|---|---|---|---|
| Heart | 13 | 6 | 7 | 2 | 303 |
| Breast | 31 | 31 | 0 | 2 | 569 |
| Diabetes | 8 | 8 | 0 | 2 | 768 |
| Voting | 16 | 0 | 16 | 2 | 435 |
| Sick | 29 | 7 | 22 | 2 | 2800 |
| Hypothyroid | 25 | 7 | 18 | 2 | 3163 |
| Adult[2] | 11 | 5 | 6 | 2 | 30162 |
| Wine | 13 | 13 | 0 | 3 | 179 |
| Dermatology | 34 | 33 | 1 | 6 | 358 |
| Vehicle | 14 | 18 | 0 | 4 | 846 |

average confidence on the decision path of the decision tree, the confidence of the firing rule in the case of SBRL, or the confidence of the set of conditions in the case of LIME. The fidelity is the percentage of data points where the model agrees with the black box model. The generality is the percentage of the data points of one class that is covered by an interpretation.

**Black-box Models.** For every dataset, we train two different black box models, a *Neural Network* and a *Random Forest*. We use the implementations of the scikit-learn Python package (Pedregosa et al., 2011). We train the *Multi Layer Perceptron* classifier with 500 nodes in the hidden layer and the *Random Forest* classifier with 1000 trees. We use 10% of the data as the test set.

**Systems.** For DTExtract[3] and SBRL[4], we use the code published by the authors. For CART, we train the decision tree with limited depth, as we do for our method. We train SBRL using the default parameters and setting the length of the rule list to 10. Since this method supports only categorical features, we discretize the numerical features. Also, SBRL doesn't support the multi-class scenario, and we thus cannot provide results for the Wine, Dermatology and Vehicle datasets. For LIME[5], we use its global version, where `Submodular pick` algorithm is used to provide multiple explanations that represent the black box model as a whole. We fix the number of features to the maximal number of conditions. We allow an arbitrary number of explanations for each dataset, except for Adult, Sick and Hypothyroid datasets, where we limit the number to 1000. To validate our method, we also evaluate the fidelity of a modified STACI method, called

[3]https://github.com/obastani/dtextract

[4]https://github.com/Hongyuy/sbrl-python-wrapper

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

Table 3.2: Fidelity (%) with NN as black box model

| Dataset | DTE | SBRL | LIME | CART | STACI' | STACI |
|---|---|---|---|---|---|---|
| Heart | **87.34** | 85.88 | 84.84 | 80.97 | 79.68 | 84.84 |
| Breast | **94.93** | 91.57 | 87.28 | 89.65 | 91.05 | 93.16 |
| Diabetes | 80.58 | 83.38 | 71.49 | 75.19 | 76.23 | **84.55** |
| Voting | **95.91** | 94.55 | 95.34 | 95.34 | 94.55 | 95.00 |
| Sick | 97.88 | 97.25 | 75.36 | 96.66 | 97.79 | **98.46** |
| Hypo. | 96.39 | 97.88 | 94.32 | 98.99 | **98.45** | **99.31** |
| Adult | 92.35 | 93.88 | 87.56 | 73.53 | **98.23** | **99.58** |
| Wine | 91.11 | **N/A** | 52.78 | 66.67 | 86.67 | **97.78** |
| Derma. | 94.86 | **N/A** | 82.70 | 80.28 | **95.28** | **96.11** |
| Vehicle | 74.47 | **N/A** | 54.71 | 69.06 | 68.24 | **86.35** |

STACI'. This method does not have access to the black box model at testing time. For each prediction, it computes the average confidence along the decision path of each surrogate tree. The confidence of a node in the tree is computed as the ratio of correctly classified data points by the split on that node over the total number of data points on that node. Finally, the method uses the tree with the highest average confidence to make the prediction. Thus, STACI' is a kind of disadvantaged variant of STACI, which has to make do without access to the black box model at testing time.

We train our models on 90% of the dataset, run all experiments on 20 random train/test splits, and report the averaged results. The code for our approach, as well as all experimental data, is available at https://github.com/nedRad88/STACI.

**Fidelity.** The results of the fidelity comparison are shown in Table 3.2 for the NN model, and in Table 3.3 for the RF model. We can see that our method, STACI, outperforms all competitors in most cases. This is not surprising, because STACI has one tree per class to ensure fidelity. Even our disadvantaged method STACI' outperforms competitors in several cases, while in others it has comparable performance. This means that our training algorithm successfully ensured a high fidelity of the specialized surrogate trees.

**Complexity.** Table 3.4 shows the average complexity of the surrogate models. CART and LIME always have the same complexity – simply because we set the maximal depth of the trees (in CART) and the maximal number of conditions (in LIME) to the same value as the maximal depth of the tree for our approach. We show that interpretations provided by our approach are usually shorter than the ones of DTExtract and SBRL.

Table 3.3: Fidelity (%) with RF as black box model

| Dataset | DTE | SBRL | LIME | CART | STACI' | STACI |
|---|---|---|---|---|---|---|
| Heart | 87.10 | 88.06 | 91.13 | 86.94 | 83.87 | **92.90** |
| Breast | 96.32 | 92.21 | 89.82 | 96.49 | 92.63 | **97.89** |
| Diabetes | 87.86 | 87.01 | 71.56 | 85.00 | 81.82 | **94.16** |
| Voting | 97.96 | 96.59 | 98.07 | 97.05 | **98.86** | **98.86** |
| Sick | **99.43** | 94.61 | 73.93 | 99.20 | 97.86 | 98.46 |
| Hypo. | 98.97 | 95.93 | 93.85 | 99.45 | 99.31 | **99.96** |
| Adult | 83.89 | 87.56 | 80.36 | 89.69 | 85.73 | **96.74** |
| Wine | 91.67 | **N/A** | 62.78 | 93.89 | 91.11 | **96.67** |
| Derma. | **96.75** | **N/A** | 76.98 | 90.00 | 93.06 | 96.39 |
| Vehicle | 74.35 | **N/A** | 58.33 | 72.94 | 70.82 | **86.11** |

At the same time, the complexity of STACI remains always limited, even in the worst case. That is not the case for our competitors: Table 3.5 shows the maximal complexity of an interpretation. In the worst case, DTExtract and SBRL will deliver interpretations of more than 10 conditions. STACI, in contrast, always delivers simple interpretations, as the maximum tree depth is fixed. The method can still keep a high fidelity because it uses multiple surrogate trees.

**Confidence.** Tables 3.6 and 3.7 show the average confidence of the interpretations for the two different black box models. As we can see, STACI gives interpretations with higher confidence in most cases.

**Generality.** We compare the generality of our model and DTExtract in Table 3.8. STACI has higher generality in most cases. Even though there is a clear trade-off between confidence and generality (or precision and recall), the results show that our specific training strategy and choice of $F1$ measure successfully solves this challenge.

In summary, our method outperforms the other methods in terms of all four aforementioned criteria: fidelity, complexity, confidence and generality. Thus, STACI overcomes the trade-off between confidence and generality, achieves higher fidelity, and still never delivers long interpretations. Figure 3.3 shows an example interpretation given by our system.

### 3.6.3 Counterfactuality

In this section, we discuss another property of interpretations: counterfactuality. An interpretation for a data point is *counterfactual* if the following is true: If we modify the data point in such a way that the conditions of the interpretation

Table 3.4: Average Complexity

| Dataset | Black | DTE | SBRL | LIME | CART | STACI |
|---|---|---|---|---|---|---|
| Heart | NN | 3.15 | 3.90 | 3 | 3 | **2.89** |
| | RF | 3.11 | **2.29** | 4 | 4 | 3.28 |
| Breast | NN | 2.88 | 4.20 | 3 | 3 | **1.9** |
| | RF | 3.18 | 6.16 | 4 | 4 | **2.88** |
| Diabetes | NN | 2.89 | 5.78 | 3 | 3 | **1.49** |
| | RF | 2.75 | 7.21 | 4 | 4 | **1.85** |
| Voting | NN | 3.11 | **1.57** | 3 | 3 | 1.58 |
| | RF | 3.00 | **1.63** | 3 | 3 | 1.69 |
| Sick | NN | 2.40 | 3.64 | 3 | 3 | **1.40** |
| | RF | 2.25 | 3.77 | 3 | 3 | **2.07** |
| Hypo. | NN | 2.58 | 4.50 | 3 | 3 | **1.20** |
| | RF | 2.16 | 4.78 | 3 | 3 | **1.09** |
| Adult | NN | 3.25 | 8.49 | 4 | 4 | **1.87** |
| | RF | 2.75 | 7.22 | 4 | 4 | **1.83** |
| Wine | NN | 3.95 | **N/A** | 3 | 3 | **2.42** |
| | RF | 4.29 | **N/A** | 4 | 4 | **2.93** |
| Derma. | NN | 4.91 | **N/A** | 3 | 3 | **2.24** |
| | RF | 4.85 | **N/A** | 4 | 4 | **2.36** |
| Vehicle | NN | 3.99 | **N/A** | 3 | 3 | **2.68** |
| | RF | 4.50 | **N/A** | 4 | 4 | **2.91** |

no longer hold, then the black box model classifies the data point differently. Counterfactuality is a very attractive property, and counterfactual interpretations have been considered tantamount to explanations (Miller, 2018; Wachter et al., 2017b). That said, counterfactuality alone is not sufficient: A counterfactual interpretation could just pose a condition that is so extreme that it is guaranteed to catapult the data point out of its current class – as in "You suffer from senility because you are not a baby. If you were a baby, you would not be senile." Such explanations are obviously absurd. Therefore, counterfactuality is always accompanied by the requirement to find the smallest modification that changes the class of the data point (Wachter et al., 2017b) – as in "You suffer from senility because you are older than 100 years." Counterfactuality in this sense is not possible in the global setting, because it inherently depends on the individual data point. Therefore, there are no guarantees that our approach will provide such explanations. However, we can assess counterfactuality a posteriori.

We report the counterfactuality as the percentage of data points for which

Table 3.5: Maximal Complexity

| Dataset | Black | DTE | SBRL | LIME | CART | STACI |
|---|---|---|---|---|---|---|
| Heart | NN | 9.30 | 7.00 | 3 | 3 | **3** |
| | RF | 10.80 | 5.4 | 4 | 4 | **4** |
| Breast | NN | 10.30 | 6.55 | 3 | 3 | **3** |
| | RF | 9.4 | 11.00 | 4 | 4 | **3** |
| Diabetes | NN | 9.50 | 10.40 | 3 | 3 | **3** |
| | RF | 10.2 | 11.90 | 4 | 4 | **4** |
| Voting | NN | 10.85 | 3.80 | 3 | 3 | **3** |
| | RF | 11.30 | 4.10 | 3 | 3 | **3** |
| Sick | NN | 10.50 | 5.60 | 3 | 3 | **3** |
| | RF | 10.60 | 7.60 | 3 | 3 | **3** |
| Hypo. | NN | 9.7 | 6.40 | 3 | 3 | **3** |
| | RF | 10.00 | 6.10 | 3 | 3 | **3** |
| Adult | NN | 10.6 | 12.20 | 4 | 4 | **4** |
| | RF | 10.8 | 18.85 | 4 | 4 | **4** |
| Wine | NN | 7.8 | **N/A** | 3 | 3 | **3** |
| | RF | 6.7 | **N/A** | 4 | 4 | **4** |
| Derma. | NN | 8 | **N/A** | 3 | 3 | **3** |
| | RF | 8.5 | **N/A** | 4 | 4 | **4** |
| Vehicle | NN | 6.4 | **N/A** | 3 | 3 | **3** |
| | RF | 6.4 | **N/A** | 3 | 3 | **3** |

Table 3.6: Confidence (%) of the interpretations (NN)

| Dataset | DTE | SBRL | LIME | STACI |
|---|---|---|---|---|
| Heart | 85.04 | 87.83 | 78.80 | **88.57** |
| Breast | 94.18 | 93.47 | 88.95 | **95.57** |
| Diabetes | 81.71 | **84.50** | 60.11 | 83.47 |
| Voting | 95.17 | 95.56 | 94.84 | **95.91** |
| Sick | 97.17 | 96.55 | 77.96 | **97.83** |
| Hypo. | 97.38 | 97.63 | 86.92 | **98.98** |
| Adult | 92.46 | 93.85 | 75.34 | **98.20** |
| Wine | 88.58 | **N/A** | 90.36 | **92.23** |
| Derma. | 78.63 | **N/A** | 53.12 | **89.21** |
| Vehicle | 66.02 | **N/A** | 40.41 | **74.98** |

Table 3.7: Confidence (%) of the interpretations (RF)

| Dataset | DTE | SBRL | LIME | STACI |
|---|---|---|---|---|
| Heart | 82.28 | 81.75 | **85.74** | 80.38 |
| Breast | 93.49 | 91.86 | 95.33 | **95.52** |
| Diabetes | 76.23 | 77.33 | 68.00 | **80.22** |
| Voting | **96.50** | 95.30 | 94.98 | 95.89 |
| Sick | 97.61 | 93.90 | 74.55 | **97.79** |
| Hypo. | 98.08 | 95.81 | 95.82 | **99.07** |
| Adult | 80.12 | 82.21 | 82.09 | **87.53** |
| Wine | 88.84 | **N/A** | 59.82 | **93.48** |
| Derma. | 78.83 | **N/A** | 64.48 | **90.00** |
| Vehicle | 59.45 | **N/A** | 52.99 | **61.38** |

Table 3.8: Generality comparison and counterfactuality

| Dataset | Black | DTE | STACI | Counterfactuality |
|---|---|---|---|---|
| Heart | NN | 59.21 | **76.63** | 66.81 |
| | RF | 58.83 | **68.35** | 64.63 |
| Breast | NN | 80.31 | **92.59** | 75.20 |
| | RF | 84.82 | **88.67** | 7.88 |
| Diabetes | NN | 66.92 | **74.47** | 72.33 |
| | RF | 64.23 | **71.51** | 90.99 |
| Voting | NN | 73.37 | **95.01** | 64.39 |
| | RF | 82.14 | **95.15** | 69.89 |
| Sick | NN | **94.70** | 94.18 | 17.14 |
| | RF | 93.39 | **94.65** | 30.44 |
| Hypo. | NN | 89.62 | **97.08** | 10.99 |
| | RF | 96.79 | **96.94** | 15.32 |
| Adult | NN | 92.06 | **95.53** | 52.35 |
| | RF | **92.25** | 73.84 | 42.12 |
| Wine | NN | 77.03 | **86.67** | 69.38 |
| | RF | 79.51 | **85.12** | 22.27 |
| Derma. | NN | **91.74** | 91.33 | 12.99 |
| | RF | **91.54** | **91.54** | 9.11 |
| Vehicle | NN | 53.98 | **68.70** | 48.21 |
| | RF | 46.16 | **55.54** | 27.39 |

The datapoint

| | |
|---|---|
| Pregnancies | 5 |
| Glucose | 166 |
| Blood pressure | 72 |
| Skin thickness | 19 |
| Insulin | 175 |
| BMI | 25.8 |
| Diabetes pedigree | 0.59 |
| Age | 51 |

is classified as diabetic. It has these characteristics:

Glucose>154, Insulin>145, Age>30

There are 37 other data points with these characteristics,
and 94.59% of them are also classified as diabetic.

Figure 3.3: Example of a STACI interpretation

the prediction of the black box model changes when we modify the data point so that it no longer satisfies the conditions of the interpretation. The results are shown in Table 3.8. As we can see, our approach is able to achieve a respectable ratio of counterfactuality, despite not being designed for it.

### 3.6.4 Regression experiments

As we have shown in Section 3.5, STACI can also interpret regression black-box models. We run similar experiments as for the classification task. We run experiments for both Neural Network and Random Forest as black-box models. We compare it to TREPAN and LIME but not SBRL, since they do not support regression problems. As a baseline, we again use the CART regression decision tree, with the same depth as the surrogate trees of STACI. We evaluate these methods on several regression datasets from the UCI Machine Learning Repository (Dua and Graff, 2017) and Penn Machine Learning Benchmarks (Romano et al., 2021). The datasets and their characteristics are listed in Table 3.9.

Tables 3.10 and 3.11 show the fidelity results for all the systems. The fidelity is measured through RMSE (smaller is better). We can see that STACI is achieving better results than its competitors in the majority of the cases and in the rest of the cases has comparable performance.

As in the previous section, we measure the average and the maximal com-

Table 3.9: Regression Datasets

| Dataset Features | | Num. | Cat. | Instances |
|---|---|---|---|---|
| Auto MPG | 7 | 6 | 1 | 393 |
| Bike | 12 | 9 | 3 | 8761 |
| Concrete | 8 | 8 | 0 | 1031 |
| Servo | 4 | 0 | 4 | 168 |
| Electrical | 12 | 12 | 0 | 10000 |
| Superconductivity | 81 | 81 | 0 | 21263 |
| White Wine Quality | 11 | 11 | 0 | 4898 |
| Real estate valuation | 5 | 5 | 0 | 414 |
| Wind | 14 | 14 | 0 | 6574 |
| CPU activity | 12 | 12 | 0 | 8192 |
| Echocardiogram | 9 | 6 | 3 | 17496 |
| Iranian Churn | 11 | 8 | 3 | 3150 |

plexity. We again show that STACI provides the shortest interpretations in both scenarios. CART and LIME always have the same complexity, because we set the maximal depth of the trees (in CART) and the maximal number of conditions (in LIME) to the same value as the maximal depth of the tree for STACI. But, taking into account fidelity results, we can see that for the same level of complexity STACI achieves better fidelity.

Finally, we compare the generality of interpretations provided by STACI to those of DTExtract. We do not compare the confidence of the interpretations since it is not defined for regression problems. Only in the case of STACI, it is possible to compute it because the regression problem is solved using first clustering and then classification. Having in mind the regression setup, we compute generality with regards to the total number of data points in the training set. Recall that for the classification task, we reported generality as percentage of data points of that class. In this scenario, the generality of interpretations of STACI is impacted by the clustering step. A bigger the number of clusters produces a lower generality. At the same time, it allows for better fidelity. Nevertheless, STACI is able to achieve both better fidelity and generality than its competitors. On top of that, STACI provides much shorter interpretations. Additionally, no matter the number of clusters, interpretations of STACI cover on average $65\%$ of the data points from the same cluster (class).

Table 3.10: Fidelity ($RMSE$) with RF as black box model

| Dataset | DTE | LIME | CART | STACI' | STACI |
|---|---|---|---|---|---|
| Auto MPG | **1.59** | 3.66 | 2.31 | 3.77 | 2.61 |
| Bike | 283.31 | 640.47 | 356.98 | 537.41 | **145.93** |
| Concrete | 6.85 | 10.40 | 8.61 | 9.53 | **6.18** |
| Servo | 0.23 | 1.22 | 0.60 | 1.46 | **0.22** |
| Electrical | 0.02 | 0.04 | 0.02 | 0.03 | **0.01** |
| Superconductivity | 14.72 | 33.10 | 12.58 | 35.44 | **10.69** |
| White Wine Quality | 0.39 | 0.51 | 0.41 | 0.57 | **0.37** |
| Real estate valuation | 4.46 | 8.69 | 4.94 | 6.96 | **3.44** |
| Wind | 1.89 | 3.95 | 2.26 | 3.10 | **1.42** |
| CPU activity | **3.09** | 19.23 | 4.16 | 6.57 | 4.82 |
| Echocardiogram | 3.80 | 6.59 | 3.72 | 8.60 | 3.88 |
| Iranian churn | 88.51 | 238.54 | 117.98 | 172.72 | **68.10** |

## 3.7 User study

To evaluate which characteristics of the interpretations are subjectively most valuable, we conducted a user study. We used the Diabetes dataset and trained a Random Forest with 1000 trees as a black box model. Then, we trained three interpretable models: DTExtract, LIME and STACI. The length of the interpretation was set to 3 for both LIME and STACI. For LIME we show only the features that had a positive influence (weight) on the outcome. Each participant in the user study was invited to look at a single patient. We showed the data of this patient and the model prediction, and we proposed the interpretations provided by the three approaches. In this study, we are not interested in the visual representation of the interpretations, so we showed all three interpretations in textual form. Each interpretation consists of the set of conditions identified by the model. Since STACI also provides confidence and generality, we computed the same measures for the other two systems as well and provided them to the participant. For LIME, we also show the importance of each feature. We asked the participants to evaluate how satisfactory each interpretation was, using a scale from 1 (least satisfactory) to 5 (most satisfactory). We also asked them to identify which characteristics of the interpretations were important for their choice: the length of the interpretations, the feature importance, the confidence, or the generality. 55 people participated, and most of them have a background in computer science.

Table 3.14 shows the average characteristics of the interpretations provided by each system and the average rating by the users. Our method achieves the

Table 3.11: Fidelity ($RMSE$) with NN as black box model

| Dataset | DTE | LIME | CART | STACI' | STACI |
|---|---|---|---|---|---|
| Auto MPG | 1.73 | 2.17 | 2.22 | 1.97 | **1.34** |
| Bike | 259.33 | 466.37 | 281.53 | 334.43 | **201.61** |
| Concrete | 7.47 | 8.08 | 8.81 | 8.38 | **5.33** |
| Servo | **0.31** | 1.22 | 0.75 | 0.55 | 0.37 |
| Electrical | 0.03 | 0.02 | 0.03 | 0.04 | **0.01** |
| Superconductivity | 21.46 | 35.53 | 12.44 | 13.91 | **9.03** |
| White Wine Quality | 0.23 | 0.43 | 0.27 | 0.36 | **0.20** |
| Real estate valuation | **1.72** | 4.62 | 2.58 | 3.85 | 2.62 |
| Wind | 2.27 | 4.73 | 2.47 | 2.86 | **1.84** |
| CPU activity | 126.91 | **27.41** | 323.67 | 110.75 | 54.86 |
| Echocardiogram | 1.60 | 4.31 | 1.56 | **1.90** | **1.37** |
| Iranian churn | 90.69 | 231.71 | 122.09 | 156.50 | **75.08** |

highest confidence and ranks second in generality after DTExtract. This is because DTExtract classified many instances on the root node. This entails shorter interpretations, and more generality, but comes at the cost of confidence. LIME achieves very low generality, which is because it provides local interpretations, which are not designed to regroup many data points.

Overall, STACI achieves the highest user rating. The reason for this good performance of STACI is shown in Figure 3.4: The users rated confidence and generality as the two most valuable properties of interpretations. These are exactly the metrics that we introduced in this work, and that STACI optimizes. This preference for more general interpretations also explains why LIME, with its local explanations, performs poorly: the participants did not like interpretations that appear tailored to a few data points. The reason why STACI outperforms DTExtract in the user rating is that users value confidence above everything else. Hence, DTExtract's strategy of providing short, general, but low-confidence interpretations falls behind STACI's more balanced approach of optimizing generality and confidence at the same time.

Overall, our user study emphasizes the importance of generality and confidence as qualities of an interpretation. and shows that the interpretations by our method were considered the most satisfactory.

Table 3.12: Average Complexity

| Dataset | Black | DTE | LIME | CART | STACI |
|---|---|---|---|---|---|
| Auto MPG | NN | 4.98 | 4 | 4 | **3.51** |
| | RF | 5.10 | 3 | 3 | **2.75** |
| Bike | NN | 5.27 | 3 | 3 | **2.66** |
| | RF | 5.35 | 3 | 3 | **2.41** |
| Concrete | NN | 5.52 | 4 | 4 | **3.60** |
| | RF | 5.38 | **3** | **3** | **3** |
| Servo | NN | 5.64 | 4 | 4 | **3.24** |
| | RF | 5.53 | 3 | 3 | **2.29** |
| Electrical | NN | 5.61 | 3 | 3 | **2.64** |
| | RF | 5.32 | 3 | 3 | **2.73** |
| Superconductivity | NN | 5.39 | 3 | 3 | **2.82** |
| | RF | 5.47 | 3 | 3 | **2.08** |
| White Wine Quality | NN | 5.44 | 3 | 3 | **2.43** |
| | RF | 5.69 | 3 | 3 | **2.07** |
| Real estate valuation | NN | 5.24 | 3 | 3 | **2.51** |
| | RF | 5.52 | 3 | 3 | **2.83** |
| Wind | NN | 5.04 | 3 | 3 | **2.94** |
| | RF | 5.09 | 3 | 3 | **2.58** |
| CPU activity | NN | 3.71 | 3 | 3 | **2.37** |
| | RF | 5.99 | 3 | 3 | **2.64** |
| Echocardiogram | NN | 5.71 | 3 | 3 | **2.89** |
| | RF | 4.87 | 4 | 4 | **2.73** |
| Iranian churn | NN | 5.93 | 4 | 4 | **3.55** |
| | RF | 5.73 | 3 | 3 | **2.86** |

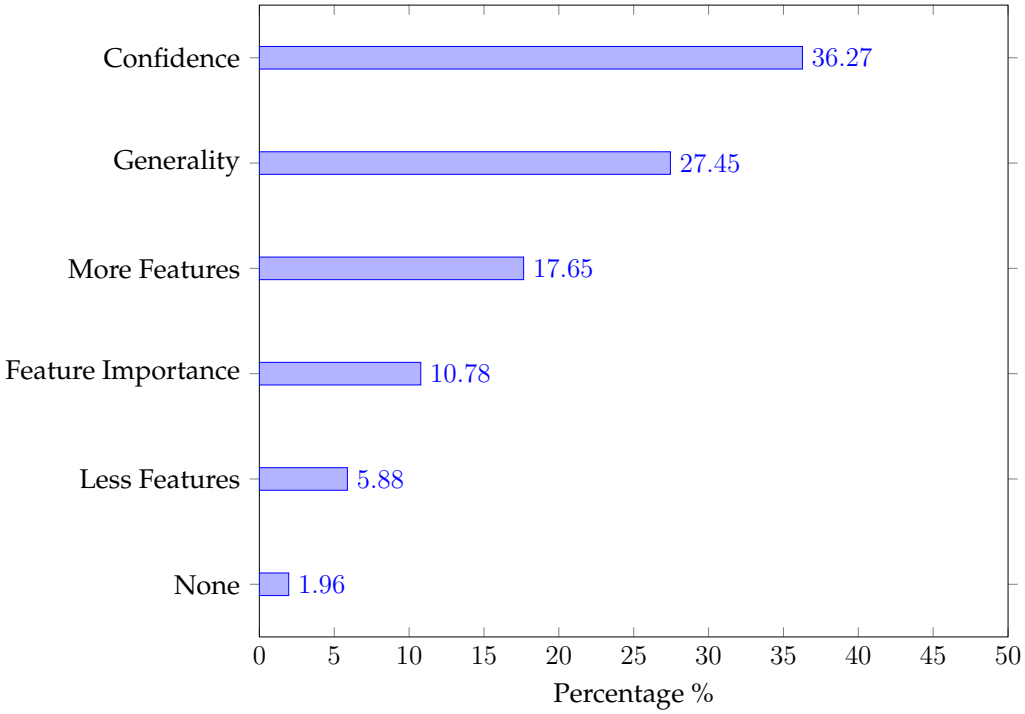
Table 3.13: Maximal Complexity

| Dataset | Black | DTE | LIME | CART | STACI |
|---|---|---|---|---|---|
| Auto MPG | NN | 4.98 | 4 | 4 | **4** |
| | RF | 5.10 | 3 | 3 | **3** |
| Bike | NN | 5.27 | 3 | 3 | **3** |
| | RF | 5.35 | 3 | 3 | **3** |
| Concrete | NN | 5.52 | 4 | 4 | **4** |
| | RF | 5.38 | **3** | **3** | **3** |
| Servo | NN | 5.64 | 4 | 4 | **4** |
| | RF | 5.53 | 3 | 3 | **3** |
| Electrical | NN | 5.61 | 3 | 3 | **3** |
| | RF | 5.32 | 3 | 3 | **3** |
| Superconductivity | NN | 5.39 | 3 | 3 | **3** |
| | RF | 5.47 | 3 | 3 | **3** |
| White Wine Quality | NN | 5.44 | 3 | 3 | **3** |
| | RF | 5.69 | 3 | 3 | **3** |
| Real estate valuation | NN | 5.24 | 3 | 3 | **3** |
| | RF | 5.52 | 3 | 3 | **3** |
| Wind | NN | 5.04 | 3 | 3 | **3** |
| | RF | 5.09 | 3 | 3 | **3** |
| CPU activity | NN | 3.71 | 3 | 3 | **3** |
| | RF | 5.99 | 3 | 3 | **3** |
| Echocardiogram | NN | 5.71 | 3 | 3 | **3** |
| | RF | 4.87 | 4 | 4 | **4** |
| Iranian churn | NN | 5.93 | 4 | 4 | **4** |
| | RF | 5.73 | 3 | 3 | **3** |

Table 3.14: User study

| System | Confidence(%) | Generality(%) | Length | Average Rating |
|---|---|---|---|---|
| DTExtract | 76.61 | 74.68 | 1.16 | 3.12 |
| LIME | 59.18 | 0.41 | 1.86 | 1.93 |
| STACI | 85.23 | 42.42 | 2.52 | **3.91** |

Figure 3.4: User preferences regarding characteristics of interpretations



# 4 BELLA

Understanding the decision-making process of black-box models has become a legal requirement and an additional way to assess their performance. However, the state of the art post-hoc explanation approaches rely on synthetic data generation, which introduces uncertainty and can hurt the reliability of the explanations. Furthermore, they tend to produce explanations that apply to only very few data points. We present BELLA, a deterministic model-agnostic post-hoc approach for explaining the individual predictions of regression black-box models. BELLA provides explanations in the form of a linear model trained in the feature space. BELLA maximizes the size of the neighborhood to which the linear model applies so that the explanations are accurate, simple, general, and robust. BELLA can produce both factual and counterfactual explanations. This chapter is based on our paper Radulovic et al. (2023b), available as a pre-print.

## Contents

## 4.1 Introduction

Several approaches exist to provide post-hoc explanations. However, most of them have been naturally tailored for classification tasks, and only a few can be applied to regression tasks. According to (Nauta et al., 2022), among the papers that propose methods for explaining black-box models, only $10\%$ consider the regression task. The most prominent approaches of those who can handle regression tasks are LIME (Ribeiro et al., 2016), SHAP (Lundberg and Lee, 2017) and MAPLE (Plumb et al., 2018). Even though they have been widely adopted because of their ease of use, they both rely on random feature perturbations that introduce a level of uncertainty that can affect their trustworthiness (Zhang et al., 2019; Slack et al., 2020). Moreover, while these approaches provide accurate explanations, they do not optimize for their generality. Thus, they tend to provide explanations that apply to very few data points. Finally, the explanations by LIME and SHAP are not *verifiable*: they provide intuitive scores for the importance of the different features, but they do not allow the user actually to compute the predicted value from feature values. This means the user cannot see how this explanation applies to neighboring data points.

With our approach, we aim to remedy these shortcomings: We present BELLA (Black-box Explanations by Local Linear Approximations), which is, to the best of our knowledge, the first deterministic model-agnostic post-hoc approach for explaining the individual predictions of a regression. It has the following properties:

- BELLA learns local linear surrogates that are not only *simple*, but also accurately mimic the black-box regression model with high *fidelity* and high *generality*.
- BELLA produces explanations that are *verifiable*, i.e., that humans can use to compute the predicted value from the feature values.
- BELLA can provide both *factual* and *counterfactual* explanations (given a reference value).
- BELLA is *deterministic*, and relies only on actually existing data points. Therefore, it can be used without access to the predictive model, and can interpret any real-valued variable of a tabular dataset.

We perform an extensive evaluation on several popular datasets from the UCI Machine Learning Repository and Penn Machine Learning Benchmarks and show that BELLA outperforms other state-of-the-art methodologies. The rest of the chapter is organised as follows: Section 3.3 presents our new method, BELLA. Section 4.3 evaluates our method on several datasets and compares it to several state-of-the-art methods.

## 4.2 Methodology

### 4.2.1 Goal

We are given a tabular dataset $T \subset F_1 \times ... \times F_n$, where each $F_i$ is a set of *feature values*. We are also given a function $Y : T \to \mathbb{R}$ that yields, for each $x \in T$, a *target value* $Y(x) \in \mathbb{R}$. These target values may, e.g., have been produced by a black-box model, in which case the target value is a *prediction*. Consider now one data point $x \in T$ with its target value $Y(x)$. We aim to compute an *explanation* in the following sense Das and Rad (2020):

**Definition 4.2.1.** *An explanation is additional meta information, generated by an external algorithm or by the machine learning model itself, to describe the feature importance or relevance of an input instance towards a particular output classification.*

If the target value was produced by a black-box model, we cannot be sure post-hoc that the features we identify really contributed to the computation of the target value (the model may just as well have thrown a dice, independently of any feature values). However, if several data points with these or similar feature values produce a similar prediction, we can use abductive reasoning to infer that these features may have contributed to the prediction, and that, hence, the features constitute an explanation. This is in fact common in the literature Ribeiro et al. (2016); Lundberg and Lee (2017); Radulovic et al. (2021); Ignatiev et al. (2019).

**Quality measures..** Several properties of "good" explanations have been proposed. Some of them, such as plausibility and accordance with prior beliefs, require human evaluation. Among the criteria that do not, we commonly find Miller (2019); Guidotti et al. (2018b); Burkart and Huber (2021); Molnar (2018):

1. **Accuracy**: we want the value that the surrogate model explains to be close to value that the black-box model predicts.
2. **Simplicity**: we want the explanation to contain few features.
3. **Robustness**: we want similar data points to have similar explanations.

In addition, users tend to favor explanations that apply to many data points Radulovic et al. (2021). This appears counter-intuitive, as we aim to explain only a single data point, no matter the others. And yet, it is easy to see that an explanation such as "You have a high risk of diabetes because your body mass index is 27, your A1C level is 7%, and your blood sugar level is 210mg/dL" is little satisfactory, as it allows no generalization. More helpful is to know that, *generally*,

people with a body mass index larger than 25, an AIC level above 6.5%, and a blood sugar level of 200 mg/dL have a high risk of diabetes Mayo-Clinic (2023). We would thus like to have:

4. **Generality**: we want the number of data points to which an explanation applies to be large.

### 4.2.2 Desiderata

In addition to the above quality measures, there are also several criteria in the literature that either apply or don't apply to a given method of explanation. One of them is Miller (2018); Wachter et al. (2017a):

5. **Counterfactuality**: the ability to provide a set of modifications to the data point at hand that would entail a change in the decision of the black-box model.

Counterfactuality is obviously of interest to a user who wishes not just to understand the prediction, but also to actively influence it (e.g., after having been attributed a high risk of diabetes based on the results of a blood test).

Several methods for post-hoc explainability use randomization to probe the black-box model. However, this entails that the same data can lead to different explanations, which introduces uncertainty for the user Zhang et al. (2019); Slack et al. (2020). We thus have:

6. **Determinism**: the avoidance of randomization steps

Finally, some methods Plumb et al. (2018) propose explanations that take the form of a linear equation, which allows computing the predicted value from the feature values. This is a very attractive property, as the user can toy with the explanation and apply it also to neighboring data points. We thus have a desideratum that we call

7. **Verifiability**: the possibility to compute the predicted value from the feature values

Given this plethora of quality measures and desiderata, it is not surprising that no existing method (including our own) can satisfy all of them perfectly. However, we can at least show that our method ticks all desiderata, and outperforms existing methods across nearly all quality measures.

### 4.2.3 Method

We are given a tabular dataset $T$, a data point $x \in T$, and a real-valued target value $y$, and we aim to compute an explanation for this target value. Our idea is to find a linear equation $y \approx w_1 \cdot f_1 + w_2 \cdot f_2 + \cdots + w_n \cdot f_n + w_0$, where $w_i$ are real-valued regression coefficients and $f_i$ are feature values of $x$ in $T$. Such an equation tells the user (1) what the important features are and (2) how their can be used to compute the predicted value. To find this equation, BELLA proceeds in three steps (Algorithm 2):

1. Compute the distance of $x$ to the other points in $T$.
2. Conduct a linear search to find the best neighborhood of $x$, according to a defined metric.
3. Train a sparse linear model on that neighborhood, and propose this model as an explanation.

**Algorithm 2** BELLA

**Input**: Dataset $T$ with labels $Y$
Labeled data point $x \in T$
1: $d \leftarrow$ ComputeDistances$(x, T)$
2: $L, N \leftarrow$ OptimalNeighborhoodSearch$(x, T, d)$
3: **return** $L, N$

**Step 1: Computing the distances..** To compute the neighborhood of the input data point, we need a distance measure. A good starting point is to have all numerical features on the same scale so that each feature contributes to the distance measure in the same range. Therefore, we first standardize all numerical features to have a mean of 0 and a standard-deviation of 1.

To compute the distances, we employ the generalized distance function Harikumar and Surya (2015), which consists of three separate distance measures to account for numerical, categorical, and binary data types, as follows:

$$d(x_1, x_2) = \sum_{i=1}^{m_n} d_n(x_{1i}, x_{2i}) + \sum_{j=m_n+1}^{m_c+m_n} d_c(x_{1j}, x_{2j}) + \sum_{k=m_c+m_n+1}^{m_n+m_c+m_b} d_b(x_{1k}, x_{2k}) \tag{4.1}$$

Here, $m_n$, $m_c$ and $m_b$ are the number of numerical, categorical, and binary features, respectively. The distance measure for the numerical attributes $d_n$ is the $L1$ norm $d_n(x_1, x_2) = |x_1 - x_2|$, which is preferred over $L2$, as it is more robust to outliers Hopcroft and Kannan (2014). For categorical features, $d_c$ is the distance measure Ahmad and Dey (2007), which takes into account the distribution of values and their co-occurrence with values of other attributes. The distance between two values $x$ and $y$ of an attribute $A_i$ with respect to attribute $A_j$ is given by:

$$\delta^{ij}(x, y) = P(A_j \in \omega | A_i = x) + P(A_j \notin \omega | A_i = y) - 1$$

Here, $P(A_j \in \omega | A_i = x)$ is the conditional probability that attribute $A_j$ will take a value from the set $\omega$ given that the attribute $A_i$ takes the value $x$. $\omega$ is a subset of all possible values of attribute $A_j$ that maximizes the sum of the probabilities. Since both probabilities can take values from $[0, 1]$, we subtract 1 in order to arrive at $\delta^{ij}(x, y) \in [0, 1]$. Lastly, for binary features, we use the Hamming distance: $d_h(a, b) = 1$ if $a = b$, and zero otherwise. In Line 1 of Algorithm 2, the function *ComputeDistances* returns the distances by Eq. 4.1.

**Step 2: Optimal Neighborhood Search..** After computing the distances, we proceed with the exploration of the neighborhood of the input data point $x$. The goal is to find a set of points, closest to $x$ according to the distance measure, that will serve as a training set for a local surrogate model. Several common techniques could be considered to that end, including kNN, K-Means, and other distance-based clustering methods. In our case, however, we aim to find a neighborhood such that a linear regression model trained on that neighborhood represents an accurate local approximation of the black-box model. Hence, the quality of the neighborhood is proportional to the quality of the performance of the linear model fitted on it. Common drawbacks of regression evaluation metrics metrics are missing interpretability, sensitivity to outliers and near-zero values, divisions by zero, missing bounds, and missing symmetry. We find that the *Berry-Mielke universal R value* $\Re$ Berry and Mielke Jr (1988) avoids most of these pitfalls. $\Re$ represents the measure of agreement between raters and it is a generalization of Cohen's kappa Cohen (1960). $\Re$ measures how much better the model is compared to a naive one (e.g., to a random predictor). $\Re$ takes values from the range $[0, 1]$, and it can be interpreted easily: If $\Re$ is equal to $0$, the model performance is equal to the one of the random model and if it is $1$, then the model has perfect performance. $\Re$ is defined as $\Re = 1 - \frac{\delta}{\mu}$, where $\delta$ and $\mu$ are defined as:

$$\delta = \frac{1}{n} \sum_{i=1}^{n} \Delta(\hat{y}_i, y_i), \quad \mu = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \Delta(\hat{y}_j, y_i). \tag{4.2}$$

Here, $n$ is the number of samples, $y_i$ is the actual target value, $\hat{y}_i$ is the predicted

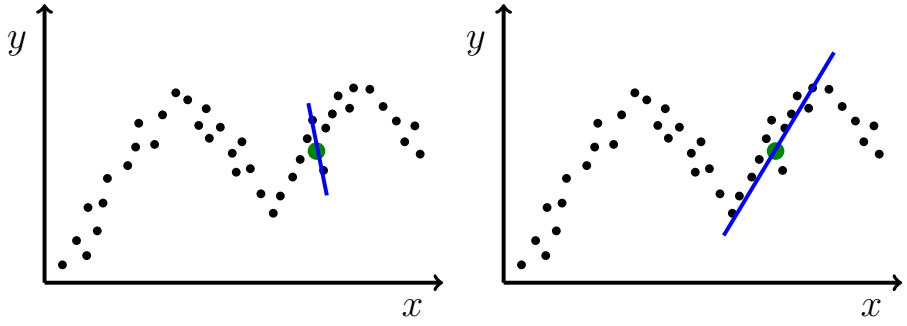



Figure 4.1: Left: an explanation for a data point $x$ that is too specific, applying only to a very small neighborhood. Right: An explanation that applies to a larger neighborhood, which is what we aim at.

value, and $\Delta(\cdot)$ represents the distance function between the true and the predicted value. The original work by Berry and Mielke Jr (1988) uses the Euclidean distance, but later works Janson and Olsson (2001, 2004) propose to use the squared Euclidean distance instead, because this distance is equivalent to the variance of the variable, which further improves the interpretability of $\Re$. We follow this argumentation, and use $\Delta(x, y) = (x - y)^2$. This definition implies that $\Delta$ is in fact equal to the Mean Squared Error (MSE). Thus, by optimizing $\Re$, we are actually optimizing the accuracy of the local model.

Our next goal is to avoid explanations that are too specific, i.e., explanations that apply to very small neighborhoods, as in Figure 4.1 (left). We rather aim for explanations that are at the same time accurate and general (Figure 4.1 (right)). Therefore we include the size of the neighborhood in the optimization function. One way to do this is to maximize the lower bound of the confidence interval of $\Re$. The lower and upper bounds for the confidence interval of $\Re$ are given by Berry and Mielke Jr (1988):

$$CI_{\Re} = \overline{\Re} \pm MOE_{\Re} = 1 - \frac{\overline{\delta} \mp MOE_{\delta}}{\mu} \tag{4.3}$$

Here, *MOE* stands for the Margin of Error. From Equation 4.3, it follows that computing the lower bound of $\Re$ is analogous to computing the upper bound of $\delta$. Therefore, we can compute the margin of error for $\delta$ as $MOE_{\delta} = t\frac{\sigma}{\sqrt{n}}$, where $\sigma$ is the standard deviation of the sample, $n$ is the sample size and $t$ represents the critical value from the t-distribution. We use the t-distribution because it is adapted for small sample sizes, which is what we encounter when we grow the neighborhood. The distribution converges to the normal distribution as the sample size increases.

Due to the non-monotonic nature of the $\Re$ value, we have to explore the whole space to maximize its lower bound. We employ a linear search algorithm (Algorithm 3) to this end.

**Algorithm 3** Optimal Neighborhood Search

**Input**: Labeled data point $x \in T$
Dataset $T$ with labels $Y$
Distances $d : T \rightarrow \mathbb{R}$ of the data points to $x$

Sort $T$ by ascending $d$
$n \leftarrow$ number of features in $T$
$max\Re_{lb} \leftarrow 0, bestN \leftarrow 0, bestL \leftarrow \emptyset$
**for** $i = min(2n, |T|)$ to $|T|$ **do**
    $L \leftarrow TrainLocalSurrogateModel(T[0:i])$
    **if** $\Re_{lb}(L) > max\Re_{lb}$ **then**
        $max\Re_{lb} \leftarrow \Re_{lb}(L)$, $bestN \leftarrow i$, $bestL \leftarrow L$
    **end if**
**end for**
**return** *bestL, T[0:bestN]*

The algorithm receives as input a labeled data point $x$ that is to be explained, a labeled training set $T$, and a vector of distances between $x$ and each point in the training set $T$. We sort the training set by increasing distance to $x$, train a linear model on the first $i$ data points for increasing $i$, and return the set of neighbors for which the lower bound of $\Re$ is maximal. As the neighbourhood is very small in the beginning, the training easily lead to overfitting. Therefore, we consider at least $2n$ data points for our neighborhood, where $n$ is the number of features. This ensures that the estimation of regression coefficients exhibits less than $10\%$ relative bias Austin and Steyerberg (2015).

**Step 3: Building a local surrogate model..** We build our local surrogate model on the neighborhood we have found. To obtain a model with few parameters (i.e., a *simple* model), we use regularization. In terms of feature selection, $L1$ regularization (e.g. Lasso Hastie et al. (2009)) is able to select a nearly perfect subset of variables in a wide range of situations. The only condition for this to work is that there are no highly collinear variables Candès and Plan (2009), which can significantly reduce the precision of estimated regression coefficients. To remove highly collinear features, we compute the *variance inflation factor* (VIF), and, following a rule of thumb Stine (1995), adopt $10$ as the cut-off value for the VIF.

After removing highly collinear features, the next step is to train a linear model with Lasso regularization. Lasso regularization adds a penalty term in

the form of the sum of absolute values of regression coefficients. The objective function is $\min_{\beta \in \mathbb{R}^p}(||y - \beta X||_2^2 + \lambda||\beta||_1)$, where $\lambda$ is the shrinkage parameter. This provides a sparse model, by forcing some coefficients to be zero. Removing some features ensures a better generalization, and results in simpler, and thus more comprehensible explanations. On the other hand, coefficients obtained by minimizing the Lasso objective function are biased towards zero. Therefore, Lasso is preferred for model selection rather than for prediction. The common strategy is to train an Ordinary Least Squares (OLS) linear model on the subset of variables selected by Lasso. This corresponds to a special variation of the relaxed Lasso Meinshausen (2007), with $\phi = 0$.

To determine the value of the shrinkage coefficient $\lambda$, we use 5-fold cross-validation (CV). To preserve the deterministic nature we perform CV on adjacent slices of the dataset, without random shuffles. CV selects the best model in terms of a prediction error. Since the goal of this step is model selection, we want to avoid choosing $\lambda$ too small, and hence we apply the common one-standard-error rule. According to this rule, the most parsimonious model is the one whose error is no more than one standard error above the error of the best model Hastie et al. (2009).

Once we have obtained the most parsimonious model, i.e., the best set of features, we train the final local surrogate model as an OLS model using the features selected by Lasso. This procedure is described in Algorithm 4, and it returns a local linear model.

**Algorithm 4** Train Local Surrogate Model

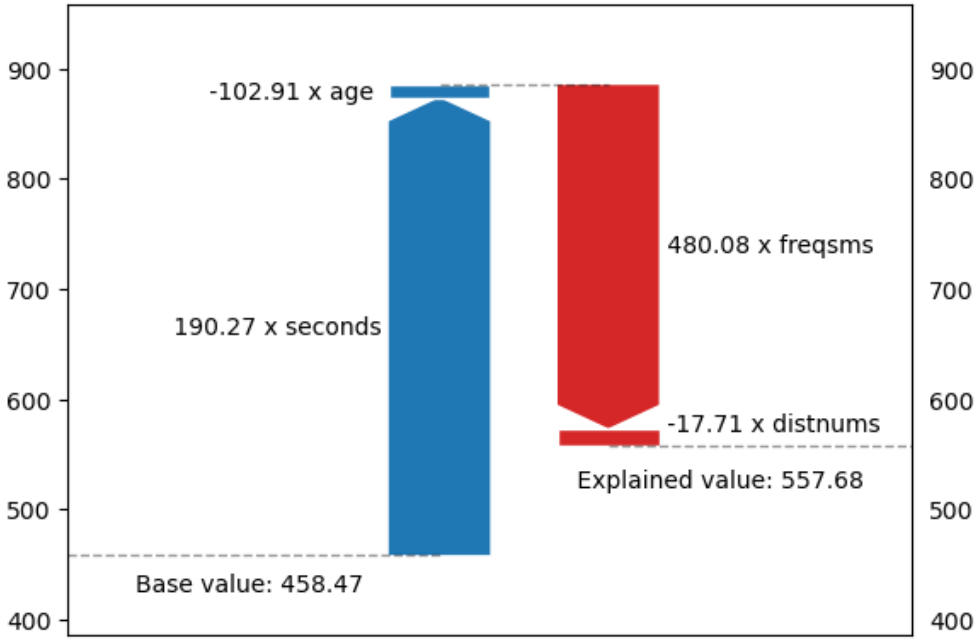
**Input**: Neighborhood of data points $N$
1: $F \leftarrow$ the set of all features in $N$
2: $F' \leftarrow \{f | f \in F \wedge \mathit{VIF}(f) < 10.0\}$
3: $\mathit{Features}_{Lasso} \leftarrow Lasso(cv = 5, \mathit{features} = F')$
4: **return** $OLS(\mathit{Features}_{Lasso})$

**Providing an explanation..** As the final result, BELLA outputs the OLS model computed by Algorithm 4, together with the size of the neighborhood. As an example, consider the Iranian Churn dataset Jafari-Marandi et al. (2020). It contains the (anonymized) customers of a telecommunication company, with their age, subscription length, the satisfaction with the service, etc. The goal is to predict the commercial value of the customer to the company (in dollars).

Let us now consider a given customer, for which a black-box model predicted a commercial value of \$551. The explanation that BELLA can provide for this prediction is shown in Figure 4.2.

All numerical features have been standardized to have mean value equal to $0$ and standard deviation equal to $1$. (Thus, a customer has a "negative age" if they

Figure 4.2: Explanation example.




The value predicted by the model is **551.08** and the explained value is **557.68**. This explanation applies to **476** other instances.

are younger than the average customer.) In the explanation, the base value is the output of the model when all inputs are set to zero (i.e., to their mean value). Each bar shows the total contribution of each feature to the predicted value: The more the customer phones (variable *seconds*), the more revenue the company generates. The age (which is below average for this particular customer), likewise, has a small positive impact. The number of SMS, in contrast, (variable *freqSMS*) impacts the revenue negatively. Finally, the number of distinct phone numbers called (variable *distnum*) has a small negative impact. These sizes of the bars are easy to interpret: The size of each bar is equal to the value of the feature multiplied by the weight computed by our method. Their sum is then directly equivalent to the explained value:

$$y \approx 458.47 + 190.27 \times \mathit{seconds} - 102.91 \times \mathit{age} + \\ 480.08 \times \mathit{freqSMS} - 17.71 \times \mathit{distNums}$$

This computation applies to all data points in the neighborhood of the input data point (to the current instance and 476 others in our example). We thus see that BELLA's explanations are *verifiable* (because they take the form of a linear

equation), *deterministic* (because BELLA does not use any randomized steps), *simple* (because we applied regularization), *general* (because we maximized the neighborhood), and *accurate* (because we optimized the linear model on the local neighborhood). In addition, BELLA does not probe the black-box model. This means that, unlike many of its competitors, BELLA can explain not just the decisions of a black-box model, but any numerical variable in a tabular dataset – even if that variable was not generated by a model at all but merely observed in reality (such as, e.g., housing prices). Let us now turn to the missing desideratum, *counterfactuality*.

**Counterfactual explanations** provide information about a (minimal) change needed to alter the prediction of the black-box model. In a classification scenario, the goal is to make the model predict a different class. In a regression scenario, the goal is not to make the model predict *any other value*, but the value that the user would like to see. In our example of the Iranian churn dataset, an analyst may ask why the model predicted \$551 instead of, say, \$1000. A counterfactual explanation should suggest a set of changes that should be applied to reach this reference value. To provide such an explanation with BELLA, we select candidates, i.e. data points whose target value is in an $\epsilon$-vicinity to the reference value (with $\epsilon = 5\%$). There can be multiple candidates. To find the best one, we optimize two criteria: the distance between the given data point and the candidate, and the amount of change needed. The first criterion will favor candidates that are in the vicinity of the given data point. The second criterion controls the amount of change applied. To alter the outcome, one can usually either apply a small change to several features, or a big change to few features. The second approach is risky: without human intervention, we can end up with a set of features that are difficult or impossible to change (e.g., the age of a customer). Therefore, we rather aim to minimize the average amount of change and suggest smaller adjustments to multiple features (such as frequency of use, or the number of SMS messages). This yields the following objective function:

$$\min_{x_i \in S}(d(x, x_i) + \frac{d(x, x')}{|\Delta|}) \tag{4.4}$$

Here, $x_i$ is a counterfactual candidate data point, $d$ is a distance measure defined in Equation 4.1, $x'$ represents the modified data point $x$ according to the counterfactual explanation and $|\Delta|$ is the number of features that have been modified. The modified data point, $x'$, represents the counterfactual explanation. This process is described in Algorithm 5.

**Algorithm 5** Computing a counterfactual explanation

**Input**: Dataset $T$ of data points $x_i$ with labels $y_i$
Labeled data point $x \in T$
A reference value $y_{ref} \in \mathbb{R}$
Deviation from reference value $\epsilon = 0.05$

1: $X_c \leftarrow \{x_i \in T : y_i \in [y_{ref} - \epsilon, y_{ref} + \epsilon]\}$
2: **for** $x_i \in X_c$ **do**
3: $L_i, N_i \leftarrow \text{BELLA}(T, x_i)$ ▷ Algorithm 2
4: $x_i' \leftarrow x_i$ *modified according to* $L_i$
5: **end for**
6: $x_{ref} \leftarrow argmin_{x_i'}(d(x, x_i) + \frac{d(x,x_i')}{|\Delta|})$
7: **return** $x_{ref}$

The algorithm takes as input the labeled dataset $T$, a labeled data-point $x \in T$, a reference value $y_{ref} \in \mathbb{R}$, and a permitted deviation $\epsilon$ from the reference value. We first choose the set of counterfactual candidates $X_c$ that have the target value in the $\epsilon$ neighbourhood of the reference value $y_{ref}$. For each $x_i$ among these candidates, we compute the explanation using BELLA. This explanation gives us a set of features, and the proposed modification of $x$ is to set all these features of $x$ to the values given by $x_i$.

Among these proposed modifications, we choose the one that minimizes the objective function in Equation 4.4. Figure 4.3 shows how BELLA computes a counterfactual explanation for the data point $x_1$ and the reference value $y_{ref}$. BELLA first identifies a candidate data point $x_{ref}$, whose target value is equal to the reference value $y_{ref}$. Then, it computes the local surrogate model for this candidate (straight blue line). The local surrogate model selects the set of important features whose values should be altered. In this simple example, there is just one feature, $x$, and the data point is just a one-dimensional real value $x_1$. Consequently, the counterfactual explanation suggests changing the value of that feature $x$ from $x_1$ to $x_{ref}$ to achieve the reference value. In more complex scenarios, the surrogate model identifies the most important features to alter.

## 4.3 Experiments

### 4.3.1 Experimental setup

**Datasets.** We performed experiments on datasets from two standard repositories: the UCI Machine Learning Repository Dua and Graff (2017) and the Penn Machine Learning Benchmarks Romano et al. (2021) (shown in Table 4.1).

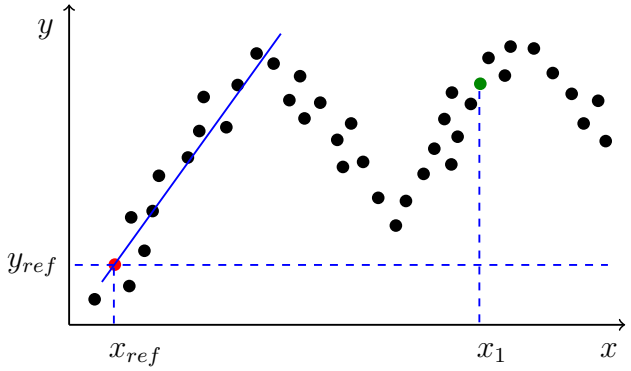



Figure 4.3: Counterfactual explanation using the reference point

Among them is also a high-dimensional dataset, Superconductivity, with 81 features. All categorical features have been one-hot encoded and all numerical features have been standardized. To show that BELLA works with different families of models, we trained a random forest (with 1000 trees), and a neural network (with one hidden layer with 500 nodes) as black-box models.

**BELLA.** Our method is implemented in Python. For the black-box models, we use the implementations of `scikit-learn` Pedregosa et al. (2011). All experiments are run on a Fedora Linux (release 38) computer with an Intel(R) Xeon(R) v4 @ 2.20GHz CPU, a memory of 64 GB, and Python 3.9. All code and the data for BELLA and the experiments will be made publicly available on Github and are part of the submission material.

**Competitors.** We compare BELLA to LIME Ribeiro et al. (2016), SHAP Lundberg and Lee (2017) and MAPLE Plumb et al. (2018). We use the implementations by the authors[1][2][3]. We do not compare to methods that are designed for classification tasks, or that can provide only counter-factual explanations and not factual ones.

### 4.3.2 Experimental results

We compare BELLA's performance against the competitors on the quality measures from Section 4.2.1. All tables show the average performance on the test set of each method with confidence intervals at $\alpha = 95\%$.

**Accuracy** is measured by the Root Mean Squared Error (RMSE) of the local surrogate models wrt. the predictions of the black-box models (Table 4.2, with a

[1]https://github.com/marcotcr/lime
[2]https://github.com/slundberg/shap
[3]https://github.com/GDPlumb/MAPLE/tree/master

Table 4.1: Regression Datasets

| Dataset | Features | Numerical | Categorical | Instances |
|---|---|---|---|---|
| Auto MPG | 7 | 6 | 1 | 392 |
| Bike | 12 | 9 | 3 | 8760 |
| Concrete | 8 | 8 | 0 | 1030 |
| Servo | 4 | 0 | 4 | 167 |
| Electrical | 12 | 12 | 0 | 10000 |
| Superconductivity | 81 | 81 | 0 | 21262 |
| White Wine Quality | 11 | 11 | 0 | 4898 |
| Real Estate Valuation | 5 | 5 | 0 | 414 |
| Wind | 14 | 14 | 0 | 6574 |
| CPU activity | 12 | 12 | 0 | 8192 |
| Echocardiogram | 9 | 6 | 3 | 17496 |
| Iranian Churn | 11 | 8 | 3 | 3150 |

min-max normalized average). SHAP always has an error of 0. This is because it provides exact explanations that apply only to a single data point. Among the methods that apply to a neighborhood of points, MAPLE is generally the best, followed closely by BELLA. LIME comes last.

**Generality** is measured by the number of data points to which the explanation applies (as a percentage of all data points in the training set). For BELLA, we simply return the size of the neighborhood. For MAPLE we return the number of data points that have weights larger than 0. For LIME, an explanation comes with the range of values for each feature. We count the number of data points that fall into this range. The results are shown in Table 4.3. For SHAP, the size of the neighborhood is always 0. This is because SHAP provides feature contributions that are specific for the given data point, and there is no way to apply these explanations to other data points. LIME's explanations are more general, and MAPLE's explanations even more. Still, they are vastly less general than the explanations of BELLA.

**Simplicity** is most commonly measured by the number of features that an explanation contains (Table 4.4). LIME has the same size of explanations as BELLA. This is because LIME takes this parameter as input and we set it to the size of the explanation provided by BELLA. SHAP and MAPLE constantly provide longer explanations than BELLA. MAPLE has higher complexity than SHAP, even though it comes with lower accuracy.

**Robustness** judges how similar the explanations for close data points are. We

Table 4.2: Accuracy comparison (RMSE – smaller is better)

| Dataset | LIME | MAPLE | BELLA | SHAP |
|---|---|---|---|---|
| Auto MPG | 2.16±0.487 | 1.04±0.239 | **0.75**±0.245 | **0.00** |
| Bike | 392.00±23.70 | **67.60**±5.110 | 164.00±11.00 | **0.00** |
| Concrete | 11.70±1.540 | **2.30**±0.382 | 3.71±0.554 | **0.00** |
| Servo | 0.87±0.289 | **0.26**±0.092 | 0.88±0.197 | **0.00** |
| Electrical | 0.03±0.002 | **0.01**±0.001 | 0.02±0.001 | **0.00** |
| Supercond. | 25.20±1.271 | **2.56**±0.357 | 6.84±0.978 | **0.00** |
| White Wine | 0.35±0.041 | **0.19**±0.026 | **0.19**±0.023 | **0.00** |
| Real Estate | 5.22±1.350 | 1.83±0.641 | **0.84**±0.511 | **0.00** |
| Wind | 3.85±0.539 | **1.28**±0.233 | 1.45±0.231 | **0.00** |
| CPU Activity | 18.70±1.700 | **0.75**±0.058 | 0.92±0.114 | **0.00** |
| Echocard. | 9.56±0.250 | **2.09**±0.087 | 2.92±0.119 | **0.00** |
| Iranian Churn | 147.00±20.70 | **2.82**±0.468 | 14.30±2.530 | **0.00** |
| **Norm. avg.** | 0.11±0.008 | 0.03±0.003 | 0.04±0.006 | 0.00 |

Table 4.3: Generality comparison (% - larger is better)

| Dataset | LIME | SHAP | MAPLE | BELLA |
|---|---|---|---|---|
| Auto MPG | 10.08±2.520 | 0.00 | **38.33**±3.100 | 37.27±8.710 |
| Bike | 4.14±0.458 | 0.00 | 5.42±0.088 | **32.11**±2.160 |
| Concrete | 1.10±0.400 | 0.00 | **31.75**±1.000 | 21.55±4.142 |
| Servo | 16.82±4.130 | 0.00 | 74.35±3.730 | **76.31**±9.410 |
| Electrical | 0.02±0.011 | 0.00 | 20.67±0.321 | **21.61**±3.180 |
| Supercond. | 0.01±0.709 | 0.00 | 15.98±0.498 | **30.25**±3.480 |
| White Wine | 0.83±0.243 | 0.00 | **17.89**±0.369 | 15.34±1.950 |
| Real Estate | 3.75±1.710 | 0.00 | **46.34**±3.990 | 16.21±6.410 |
| Wind | 0.24±0.044 | 0.00 | 12.69±0.186 | **94.00**±1.520 |
| CPU Activity | 0.71±0.250 | 0.00 | 9.30±0.227 | **17.50**±1.610 |
| Echocard. | 0.00±0.001 | 0.00 | 5.13±0.060 | **83.76**±1.180 |
| Iranian Churn | 0.65±0.202 | 0.00 | **12.10**±0.417 | **6.41**±2.140 |
| **Average** | 3.21±1.190 | 0.00 | 24.16±1.166 | **37.95**±3.813 |

Table 4.4: Simplicity comparison (smaller values are better). LIME requires the explanation size as input, and we give it the size of the explanation computed by BELLA.

| Dataset | SHAP | MAPLE | BELLA/LIME |
|---|---|---|---|
| Auto MPG | 9.00±0.000 | 6.75±0.208 | **4.90**±0.369 |
| Bike | 11.63±0.038 | 12.44±0.080 | **7.30**±0.115 |
| Concrete | 8.00±0.000 | 7.00±0.000 | **5.42**±0.297 |
| Servo | 13.06±0.218 | 15.71±0.155 | **5.06**±0.645 |
| Electrical | 12.00±0.000 | 12.00±0.000 | **8.63**±0.087 |
| Supercond. | 77.97±0.246 | 79.99±0.008 | **13.10**±0.555 |
| White Wine | 11.00±0.000 | 10.00±0.000 | **7.54**±0.198 |
| Real Estate | 5.00±0.000 | **4.00**±0.000 | 4.05±0.327 |
| Wind | 13.65±0.132 | 13.00±0.000 | **9.47**±0.102 |
| CPU Activity | 12.00±0.000 | 12.00±0.000 | **10.80**±0.204 |
| Echocardiogram | 9.00±0.000 | 8.47±0.026 | **8.23**±0.037 |
| Iranian Churn | 9.15±0.054 | 9.53±0.059 | **5.65**±0.166 |
| **Norm. Avg.** | 0.92±0.002 | 0.91±0.004 | **0.59**±0.022 |

measure robustness as

$$\textit{robustness} = 1 - \frac{1}{n}\sum_{i=1}^{n}\frac{|\beta_{1i} - \beta_{2i}|}{|\beta_{1i}| + |\beta_{2i}|}. \tag{4.5}$$

Here, $n$ is the number of features, and $\beta_{1i}$ and $\beta_{2i}$ are the weights of feature $i$ in the first and second explanation, respectively. Robustness is in the range of $[0, 1]$, with 1 indicating that two explanations are identical. We compute explanations for each data point in the test set, and compute robustness wrt. the $10$ closest data points (Table 4.5). LIME samples $5000$ data points to create a synthetic neighborhood. Thus, LIME can perform slightly better than our approach on datasets that have fewer observations. Still, in the majority of cases, and on average, BELLA outperforms LIME. BELLA also outperforms SHAP by a wide margin. This is because SHAP's explanations are tailored for a single data point. BELLA also outperforms MAPLE. This is because the crisp neighbourhood of BELLA provides much more robust explanations than MAPLE's weighted neighbourhood.

From Tables 4.2, 4.3, 4.4, and 4.5, we can see that at the same level of simplicity, BELLA provides more general, more robust, and more accurate explanations

Table 4.5: Robustness comparison (0 to 1 – larger is better)

| Dataset | LIME | SHAP | MAPLE | BELLA |
|---|---|---|---|---|
| Auto MPG | 0.78±0.023 | 0.68±0.083 | **0.84**±0.037 | 0.80±0.038 |
| Bike | **0.79**±0.026 | 0.70±0.037 | 0.66±0.033 | 0.76±0.050 |
| Concrete | **0.78**±0.047 | 0.72±0.028 | 0.68±0.041 | 0.65±0.081 |
| Servo | **0.77**±0.017 | 0.59±0.021 | 0.56±0.097 | 0.64±0.029 |
| Electrical | 0.63±0.015 | 0.59±0.022 | **0.76**±0.022 | **0.76**±0.041 |
| Supercond. | 0.89±0.014 | 0.87±0.031 | 0.58±0.076 | **0.93**±0.059 |
| White Wine | **0.67**±0.035 | 0.56±0.051 | 0.66±0.030 | 0.65±0.064 |
| Real Estate | 0.70±0.057 | 0.74±0.041 | **0.77**±0.058 | 0.65±0.085 |
| Wind | 0.64±0.037 | 0.62±0.035 | 0.67±0.023 | **0.99**±0.109 |
| CPU Activity | 0.46±0.034 | 0.73±0.035 | 0.76±0.031 | **0.83**±0.039 |
| Echocardiogram | 0.75±0.039 | 0.66±0.034 | 0.55±0.039 | **0.96**±0.017 |
| Iranian Churn | 0.62±0.035 | 0.79±0.045 | **0.84**±0.039 | 0.76±0.049 |
| **Average** | 0.71±0.032 | 0.69±0.039 | 0.68±0.044 | **0.78**±0.055 |

than LIME. BELLA provides less accurate explanations than SHAP and MAPLE, but at the same time, BELLA's explanations are more general, more robust, and vastly simpler.

What follows are the experimental results on the same desiderata with Random Forest as a black-box model (Tables 4.7, 4.8, 4.9, 4.10). Key takeaways are inline with the previous experiments where black-box model was a Neural Network.

**Other desiderata** outlined in Section 4.2.1 were counterfactuality, determinism, and verifiability (the possibility to compute the explained value from the feature values). SHAP offers none of these. Neither does LIME. While both SHAP and LIME compute linear models with feature weights, these models are not verifiable in our sense: There is no way that the user can insert the feature values of a neighboring point into these models and obtain an explained value. This is because the linear models do not operate in the original input feature space. Only MAPLE offers this verifiability. However, it relies on randomization and provides no counterfactuality.

Let us now evaluate the quality of BELLA's counterfactual explanations. We measure the accuracy of an explanation wrt. a reference value. For each data point $x$ with its target value $y$, we set the reference values to $y_{ref} = y \pm 0.3 \times |y_{max} - y_{min}|$. Table 4.6 shows the RMSE of the counterfactual explanations (as well as of the factual explanations by BELLA and LIME for comparison). We

Table 4.6: RMSE of BELLA's counterfactual explanations (smaller is better). Factual explanations for comparison.

| Dataset | Black-box | Factual LIME | Factual BELLA | Counterf. BELLA |
|---|---|---|---|---|
| Auto MPG | 2.30 | 2.16±0.487 | **0.75**±0.245 | 3.74±1.100 |
| Bike | 235.21 | 392.00±23.70 | 164.00±11.00 | 474±20.72 |
| Concrete | 4.75 | 11.70±1.540 | 3.71±0.554 | 1.23±0.410 |
| Servo | 0.34 | 0.87±0.289 | 0.88±0.197 | 0.89±0.203 |
| Electrical | 0.02 | 0.03±0.002 | 0.02±0.001 | 0.03±0.001 |
| Supercond. | 11.44 | 25.20±1.270 | 6.84±0.978 | 58±1.860 |
| White Wine | 0.70 | 0.35±0.041 | **0.19**±0.023 | 0.86±0.058 |
| Real Estate | 7.91 | 5.22±1.350 | **0.84**±0.511 | 8.76±2.231 |
| Wind | 3.07 | 3.85±0.539 | 1.45±0.231 | 4.62±0.204 |
| CPU Activity | 2.97 | 18.70±1.700 | 0.92±0.114 | 2.76±0.370 |
| Echocard. | 12.18 | 9.56±0.250 | 2.92±0.119 | 13.1±0.422 |
| Iranian Churn | 3.27 | 147.00±20.70 | 14.30±2.530 | 28±5.970 |
| **Norm. avg.** | 0.07 | 0.11±0.008 | 0.04±0.006 | 0.12±0.047 |

see that the counterfactual explanations of BELLA are often of similar accuracy as its factual explanations. Even in the cases where the error of counterfactual explanations is an order of magnitude larger, it is still lower or comparable to the error of the factual-only explanations provided by LIME.

### 4.3.3 Verification on an interpretable model

To confirm that the explanations provided by BELLA represent what the black-box model has learned, we evaluate them with regard to an already interpretable model. Instead of a black-box model, we train an Ordinary Least Square linear regression model and consider the $5$ most important features. We then compute the explanations for each data point in the test set with our method. BELLA was able to recover on average $85.12\%$ of the original top-5 features across all datasets from the ones in Table 3.1. This shows that our method provides explanations that generally agree with prior beliefs, as encoded in an interpretable model.

Table 4.7: Accuracy comparison (RMSE – smaller is better)

| Dataset | LIME | MAPLE | BELLA | SHAP |
|---|---|---|---|---|
| Auto MPG | 1.63±0.430 | **0.74**±0.218 | 1.36±0.421 | **0.00** |
| Bike | 322.72±16.83 | **66.14**±5.420 | 176.42±10.58 | **0.00** |
| Concrete | 6.09±0.954 | **2.36**±0.381 | 3.75±0.601 | **0.00** |
| Servo | 0.49±0.129 | **0.16**±0.088 | 0.52±0.146 | **0.00** |
| Electrical | **0.01**±0.001 | **0.01**±0.000 | **0.01**±0.00 | **0.00** |
| Supercond. | 28.40±1.470 | **2.94**±0.409 | 5.41±0.499 | **0.00** |
| White Wine | 0.31±0.027 | **0.16**±0.013 | 0.28±0.023 | **0.00** |
| Real Estate | 4.08±1.202 | **3.16**±1.050 | 3.39±0.756 | **0.00** |
| Wind | 1.49±0.084 | **0.55**±0.036 | 1.01±0.060 | **0.00** |
| CPU Activity | 11.66±1.030 | **0.71**±0.101 | 1.44±0.238 | **0.00** |
| Echocard. | 3.51±0.113 | **1.74**±0.069 | 3.17±0.121 | **0.00** |
| Iranian Churn | 146.68±20.73 | **10.92**±2.335 | 14.28±2.543 | **0.00** |
| **Norm. avg.** | 0.07±0.008 | 0.02±0.004 | 0.04±0.005 | 0.00 |

Table 4.8: Generality comparison (% - larger is better)

| Dataset | LIME | SHAP | MAPLE | BELLA |
|---|---|---|---|---|
| Auto MPG | 4.25±2.653 | 0.00 | 43.91±3.493 | **77.05**±9.229 |
| Bike | 1.30±0.225 | 0.00 | 5.48±0.089 | **39.99**±2.392 |
| Concrete | 0.33±0.168 | 0.00 | **31.18**±1.444 | 23.26±4.079 |
| Servo | 4.63±1.900 | 0.00 | 79.57±5.023 | **81.69**±12.721 |
| Electrical | 0.01±0.001 | 0.00 | **17.12**±0.305 | 16.89±1.882 |
| Supercond. | 0.01±0.159 | 0.00 | 14.93±0.439 | **53.55**±2.386 |
| White Wine | 0.68±0.245 | 0.00 | 18.44±0.306 | **65.31**±2.879 |
| Real Estate | 2.52±0.968 | 0.00 | 49.42±3.936 | **55.38**±12.020 |
| Wind | 0.37±0.057 | 0.00 | 12.41±0.176 | **99.97**±0.014 |
| CPU Activity | 0.75±0.146 | 0.00 | 9.54±0.215 | **51.63**±1.570 |
| Echocard. | 0.31±0.040 | 0.00 | 5.90±0.081 | **86.40**±1.080 |
| Iranian Churn | 1.5±0.216 | 0.00 | 11.91±0.429 | **12.49**±1.843 |
| **Average** | 1.18±0.565 | 0.00 | 24.98±1.328 | **55.21**±4.341 |

Table 4.9: Simplicity comparison (smaller values are better). LIME requires the explanation size as input, and we give it the size of the explanation computed by BELLA.

| Dataset | SHAP | MAPLE | BELLA/LIME |
|---|---|---|---|
| Auto MPG | 8.00±0.000 | 7.90±0.097 | **3.90**±0.382 |
| Bike | 12.00±0.032 | 12.57±0.080 | **7.43**±0.303 |
| Concrete | 8.00±0.000 | 7.00±0.000 | **5.40**±0.301 |
| Servo | 10.12±0.845 | 15.94±0.125 | **6.76**±1.068 |
| Electrical | 12.00±0.000 | 12.00±0.000 | **8.06**±0.201 |
| Supercond. | 48.90±1.159 | 80.00±0.007 | **15.50**±0.344 |
| White Wine | 11.00±0.000 | 10.00±0.000 | **6.70**±0.291 |
| Real Estate | 5.00±0.000 | 4.00±0.000 | **3.76**±0.256 |
| Wind | 13.63±0.048 | 13.00±0.000 | **7.82**±0.155 |
| CPU Activity | 12.00±0.000 | 11.00±0.000 | **10.35**±0.523 |
| Echocardiogram | 9.00±0.000 | 7.48±0.025 | **5.65**±0.145 |
| Iranian Churn | 9.19±0.043 | 9.44±0.063 | **5.56**±0.176 |
| **Norm. Avg.** | 0.90±0.006 | 0.89±0.003 | **0.57**±0.022 |

Table 4.10: Robustness comparison (0 to 1 – larger is better)

| Dataset | LIME | SHAP | MAPLE | BELLA |
|---|---|---|---|---|
| Auto MPG | **0.87**±0.017 | 0.67±0.044 | 0.65±0.060 | 0.70±0.028 |
| Bike | 0.77±0.004 | 0.66±0.005 | 0.51±0.063 | **0.79**±0.009 |
| Concrete | **0.76**±0.020 | 0.65±0.019 | 0.70±0.050 | 0.69±0.035 |
| Servo | **0.90**±0.019 | 0.79±0.038 | 0.52±0.108 | 0.75±0.030 |
| Electrical | **0.82**±0.002 | 0.50±0.004 | 0.59±0.028 | 0.81±0.006 |
| Supercond. | 0.76±0.006 | 0.85±0.005 | 0.49±0.033 | **0.80**±0.011 |
| White Wine | 0.62±0.007 | 0.51±0.007 | 0.59±0.042 | **0.77**±0.013 |
| Real Estate | 0.63±0.040 | 0.69±0.038 | 0.66±0.057 | **0.85**±0.060 |
| Wind | 0.71±0.006 | 0.63±0.006 | 0.61±0.027 | **0.98**±0.002 |
| CPU Activity | 0.52±0.005 | 0.69±0.008 | 0.65±0.040 | **0.84**±0.006 |
| Echocardiogram | 0.81±0.004 | 0.52±0.002 | 0.46±0.037 | **0.99**±0.011 |
| Iranian Churn | 0.75±0.017 | **0.79**±0.011 | 0.65±0.055 | 0.78±0.012 |
| **Average** | 0.74±0.012 | 0.66±0.016 | 0.59±0.050 | **0.81**±0.019 |

# 5 SCALAR

SCALAR is a new platform for real-time machine learning competitions on data streams. Following the intent of Kaggle, which serves as a platform for organizing machine learning competitions adapted for batch learning, we propose SCALAR as a novel platform designed specifically for stream learning in real-time. SCALAR supports both classification and regression problems in data streaming setting. SCALAR is open source and it has been published in Journal of Open Source Software (JOSS) (Radulovic et al., 2020). Also, it has been used to organize, first of its kind, a competition on data streams on IEEE Big Data Cup (Boulegane et al., 2019). In this section, we describe its architecture and discuss the results and the experiences of the competition.

## Contents

## 5.1 Introduction

While xAI focuses on providing insights into the "black-box" of AI decision-making, data stream mining addresses the dynamic and continuous nature of modern data streams. In this chapter, we will uncover the unique challenges posed by the continuous and dynamic nature of modern data streams. Data stream mining plays a crucial role in extracting knowledge and patterns from rapidly evolving data streams in real-time.

Here, we are focused on one specific segment of the machine learning community: publicly available platforms for machine learning competitions, such as `Kaggle`[1] and `Alibaba Tianchi`[2]. These platforms, especially Kaggle, are widely recognized for data science competitions and collaborative machine learning projects, and have played a crucial role in advancing the field of machine learning. Designed predominantly for batch learning, the participants access large datasets, apply pre-existing models, and submit predictions to compete for the best-performing solutions. This framework has been instrumental in fostering innovation, attracting talent, and promoting the development of cutting-edge algorithms that excel in batch learning scenarios.


However, as the landscape of data continues to evolve rapidly with the emergence of real-time data streams, the limitations of Kaggle in facilitating online learning become apparent. Online learning, also known as incremental or streaming learning, poses unique challenges as models must continuously adapt to new incoming data. The lack of a similar platform tailored for online learning hinders the collaborative efforts and experimentation necessary to tackle real-time, dynamic data analysis. With all this in mind, following the principles of Free and Open Source Software (FOSS), we developed and proposed `SCALAR`, a platform for real-time machine learning competitions. `SCALAR` supports this data stream machine learning scenario where data is continuously released, in batches every time interval. Predictions for each current batch, that are sent

[1] https://www.kaggle.com/
[2] https://tianchi.aliyun.com/

before a defined deadline, are evaluated in real-time, and the results are shown on the live leaderboard.

SCALAR has been used for organizing, a first Real-time Machine Learning Competition on Data Streams (Boulegane et al., 2019) as part of the IEEE Big Data 2019 Cup Challenges[3].

### 5.1.1 Streaming learning setting

Most of the existing platforms for data science competitions are tailored to offline learning where a static dataset is made available to the participants before the competition starts. This dataset is divided into training and test sets. The training set is used to build and train the model, which is then tested on the test set.

In online learning, data arrive in a stream of records (instances) and the model needs to be trained incrementally as new instances are released. Since the data arrive at high speed, predictions have to be issued within a short time. Having in mind this specific setup of the data stream mining scenario (Figure 5.1), every model should use a limited amount of time and memory, process one instance at a time and inspect it only once and the model must be able to predict at any time (Bifet et al., 2010).

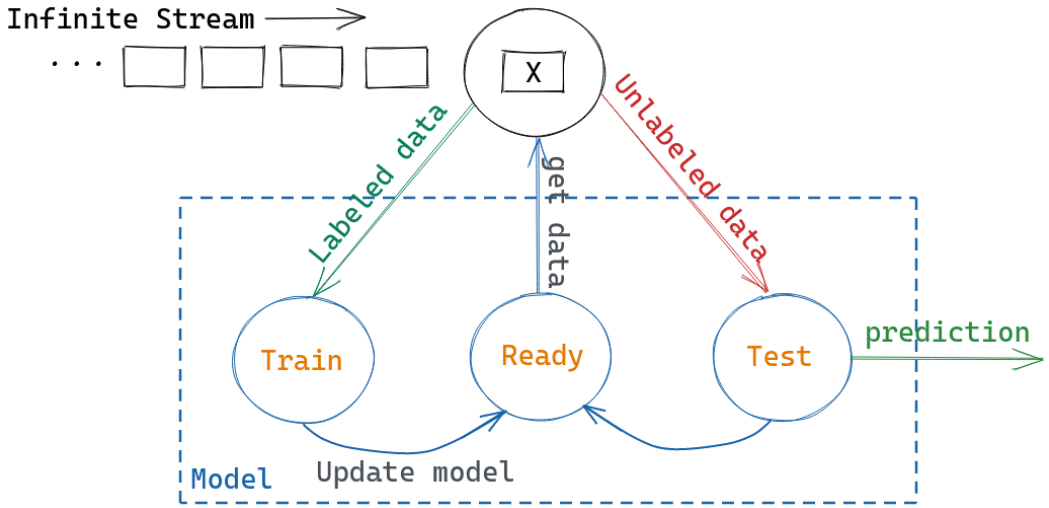



Figure 5.1: Stream data mining scenario

The model is updated using the labelled instances and then evaluated on new unlabeled instances. This scenario represents the prequential evaluation scenario

[3]http://bigdataieee.org/BigData2019/BigDataCupChallenges.html

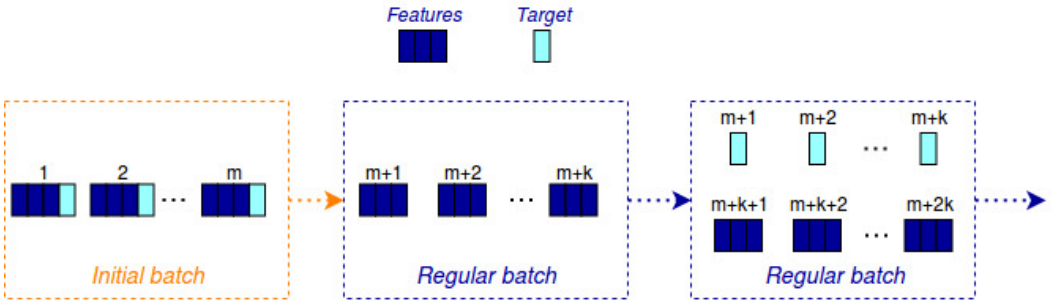



Figure 5.2: Initial and regular batches in the data stream

or *"test-then-train"* setting. To make this scenario possible in SCALAR, we first assume that the records in the data stream arrive in batches and that each batch can be of size 1 or more. Then, we define two different kinds of batches:

- **Initial batch:** This batch is used to build and train the initial model. It is aimed to avoid the cold start problem and as such contains both features and targets. The initial batch usually has a large number of instances.
- **Regular batch:** The regular batch contains new test instances while providing training instances of previous batches that are used to evaluate the quality of the model up to the current time.

## 5.2 Platform for real-time machine learning competitions

In this section, we describe the architecture and the development details of SCALAR. We will give an overall overview of all parts of the system and then we will focus specially on the online evaluation engine and the user interaction with the platform.

### 5.2.1 Architecture

To support all the aforementioned requirements for stream data mining, and to be able to organize competitions in such a scenario, a specific dedicated platform was needed. To the best of our knowledge, there doesn't exist such a platform, thus we propose SCALAR. Figure 5.3 highlights the architecture of the platform that includes a web application and a streaming engine following the fundamentals of Information Flow Processing (IFP) (Cugola and Margara, 2012). We explain layer by layer:

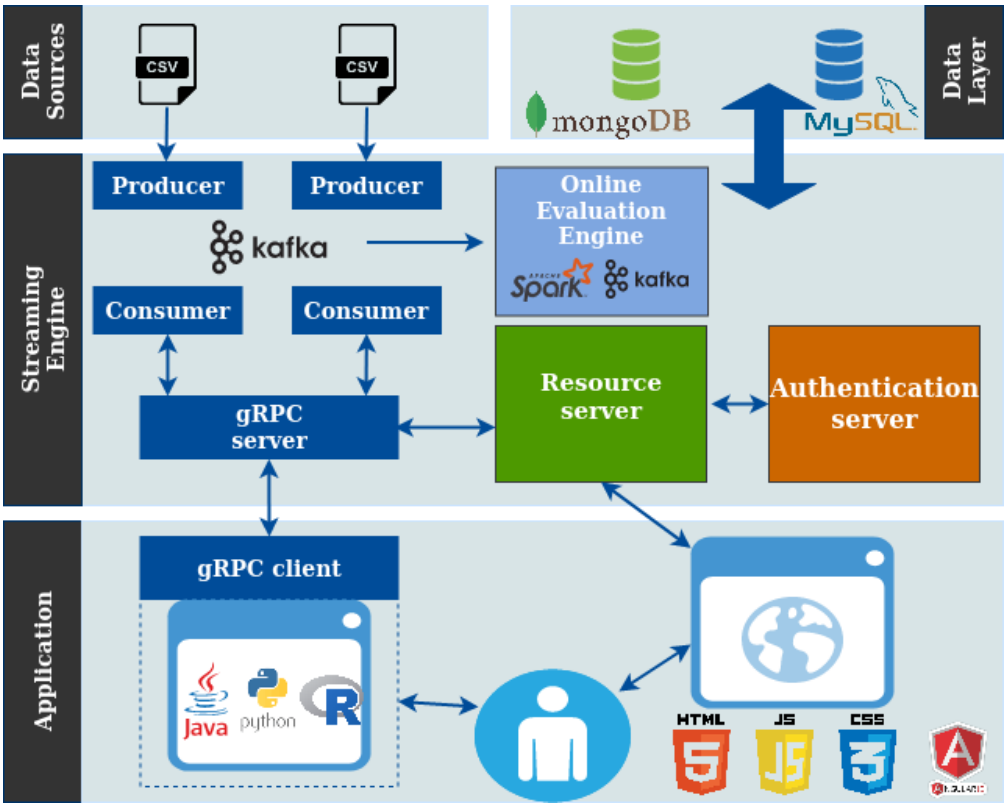



Figure 5.3: Architecture of the platform

- **Data sources:** SCALAR allows creating competitions by providing data in the form of a CSV file. That file is used to recreate a continuous stream following predefined competition settings such as the time interval between two releases and the number of instances at a time.

- **Data layer:** represents the persistence node in the system where different kinds of information are stored. MongoDB is used for unstructured data (e.g. user's predictions, records from the stream, evaluation metrics) and MySQL for structured data (competition information, user information).

- **Streaming Engine:** is responsible for handling the streams. From ***CSV*** files, Kafka recreates multiple streams to be sent to multiple users. SCALAR provides a secure and independent bi-directional streaming communication between the user and the streaming engine. This allows each user to consume training and test instances and submit the respective predictions according to the competition's predefined data format. The resource server is the ***API*** that handles all authenticated requests from the client applica-

tion whereas the authorization server is in charge of users' identification. The Online evaluation engine handles both the stream of instances and the streams of participants' predictions in order to compute and update the evaluation metrics online before storing them in the dedicated database.

- **Application:** The application layer consists of two parts: a web application and a client application. Web application represents a user-friendly interface that allows participants to register, subscribe to competitions, and follow leaderboard and models' performance online. The client application is provided in order to accommodate participants' code to solve the machine learning problem at hand. It delivers a secure communication channel to receive the stream of training and test instances and send their respective predictions to the server.

## 5.3 Development

`SCALAR` was designed using state of the art big data processing tools where streams are handled by `Kafka` ((Kreps et al., 2011), (Garg, 2013)) and the online evaluation is performed using `Spark Structured Streaming`[4] ((Zaharia et al., 2010), (Armbrust et al., 2018)). In order to ensure safe and fast bi-directional streaming communication between users and the server, we use a combination of `gRPC`[5] and `Protobuf`[6]. We use `Docker` ((Merkel, 2014)) for the deployment to provide portability and easy integration across `Linux`, `Windows` and `macOS` operating systems.

The code, the links to `Docker` containers and the documentation are available at: https://github.com/nedRad88/SCALAR.

### 5.3.1 Implementation details

**Wrappers for multiple language support.** The platform has been built in a way that it should provide complete independence for each user. Every user should be able to use different programming languages, software stacks, libraries, and additional data for building and improving his model. For communication between the server and the client, we used gRPC framework with Protobuf. This combination provides secure communication, full-duplex bidirectional streaming and an easy way to describe services.(gRPC, 2018) (Developers, 2018) The services and data formats are defined once in a ***.proto*** file. Also, this framework

[4]https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html
[5]https://grpc.io/
[6]https://developers.google.com/protocol-buffers

is language- and platform-neutral and supports several programming languages (Java, Python, Go, C#, Ruby). As mentioned earlier, gRPC and Protobuf support both Java and Python so in this case, we can use those languages directly.
**R programming language.** Except for Java and Python, there is another programming language that is used more and more for data science and data analysis tasks, and that is R. R is a programming language and software environment for statistical computing and graphical analysis(CRAN, 2018). R is interpreted language and usually is used from command-line but also has several front-ends of which the most important is RStudio[7]. R has a variety of libraries and packages that provide statistical and graphical techniques. Those libraries provide classification, clustering, regression techniques, linear and nonlinear modelling, time series analysis and many others. Its source code is written in C, Fortran and R. It runs and compiles under many UNIX platforms (including Linux), Windows and MacOS. It is available free of charge and is a competitor to some commercial statistical tools such as SAS[8], SPSS[9], and Stata[10]. It is widely used by statisticians and data scientists so it was natural to include R as one of the programming languages supported on our platform.

There is one important difference when it comes to using R for our platform. As explained earlier, users need to connect to the server through gRPC channel and communicate over Protobuf protocol. Both of these are not supported by the R programming language. In order to enable R for our platform, we developed a package/wrapper that offers extended capabilities for R and allows users to connect to gRPC server.

A specific package for R, Reticulate offers an appropriate solution. The Reticulate package offers a set of tools for interoperability between R and Python(Ushey et al., 2022). It provides several solutions:

- Calling Python from R ( sourcing Python scripts, importing Python modules, R markdown)
- Translation between Python and R objects (R and Pandas data frames)
- Binding to different versions of Python

For our use case, the most interesting feature is importing Python modules. Since we already have enabled communication with the server via Python script,

[7]https://www.rstudio.com/products/rstudio/
[8]https://www.sas.com/en_us/solutions/analytics.html
[9]https://www.ibm.com/analytics/spss-statistics-software
[10]https://www.stata.com/

it would be convenient to use that script and its classes in order to enable communication from R. The main idea is to import classes for communication from Python and call them in R. In that way, we establish communication and use rich libraries in R for processing the data.

**Online evaluation engine.** Our platform is specially designed to work with data streams. The stream of records (instances) is provided from the server side and streams of predictions are provided by users. Due to various use cases and also to be able to verify if our model is working correctly, we need to have an online evaluation. The main idea is to be able to compute evaluation metrics according to competition settings online, as the stream of predictions arrives and provide the live graphics of the quality of the model to users. We have several requirements for a stream processing tool for our system. The stream processing tool should provide:

- low latency
- exactly once fault tolerance
- integration with Kafka
- Python ***API***
- parallelism

Apache Spark is an open-source framework for cluster-computing. Originally, it supports batch computations on large data. With its API - Spark Streaming, Spark is able to do stream processing in a micro-batch setting. This means that Spark is not able to do true stream processing but it considers records in the stream as micro-batches and applies the same logic as in batch processing. The base of its architecture is RDD - resilient distributed dataset. It is a read-only multiset of data distributed over a cluster of machines, which is maintained in a fault-tolerant way. Another Spark API is a DataFrame. The DataFrame is similar to RDD as it also represents an immutable collection of data but it is organized in named columns, similar to a table in a relational database. It is designed in a way to make big data processing even easier and more understandable to a wider audience. (Bill Chambers, 2018)

Spark has four main components on top of the foundation of the framework Spark Core. Spark Core provides basic I/O functionalities and scheduling. It has a programming interface for Java, Scala, Python, and R and it is based on RDD abstraction. On top of the Spark Core, there are four components: Spark SQL, Spark MLlib, GraphX and Spark Streaming.

For the purpose of our system, we are interested in two components: Spark Streaming and Spark SQL. Spark SQL is a component of Spark that introduced

the concept of DataFrame. It provides support for structured data and semi-structured data. Spark Streaming is a component of Spark that uses fast scheduling from Spark Core to provide streaming analytics. It ingests data in micro-batches and performs RDD transformations on the micro-batches. It has the capability to use as data sources: Twitter, Kafka, Flume, TCP I/O sockets. The pipeline of the online evaluation engine is shown in figure 5.4.

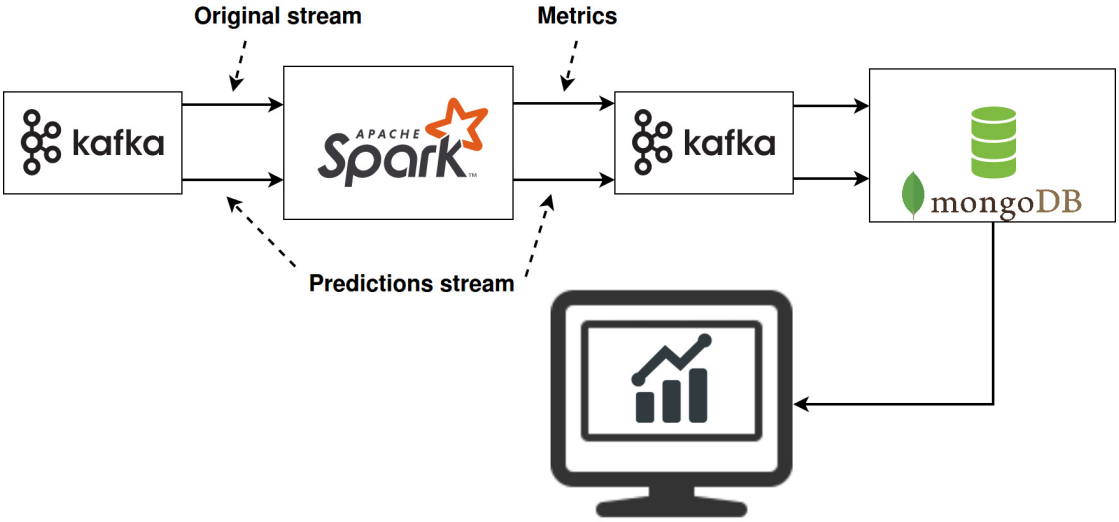



Figure 5.4: Online evaluation engine pipeline

The central point in this pipeline is the Spark module. On input, it is connected to the Kafka cluster and subscribes to Kafka topics for certain competitions. For every competition, there will be two data streams. One is the stream of records sent by the server and the other is the stream of predictions from users. These two streams are then processed in Spark to compute evaluation measures according to competition settings. On the output side, the stream of predictions and evaluation measures is then written again to Kafka and after to MongoDB.(Neha Narkhede, 2017) After storing the results in MongoDB, on a request from web application the results will be streamed to the live chart and leaderboard.

As already mentioned earlier, the latest versions of Spark introduced a new concept of Structured streaming. Structured streaming is based on Spark SQL component. The basic idea behind this concept is to think of the data stream in the same way as the SQL table. Every new record in the data stream is considered as a new row in the table. Since we consider stream as infinite we refer to the table as unbounded.(Apache, 2018) It is obvious that keeping this whole table in memory is not feasible and because of that after computations data are written

either on disk or to some of the available sinks.

Data are read from specified input at every trigger interval. Trigger interval can be set manually (for example 1s), which means that the memory table will be updated with new rows every interval. Since we are actually working with micro-batches, during every interval we consider the memory table to be static and with that in mind we apply operations and transformations as if it was an ordinary SQL table.

**Baseline competitor.** Whenever we need to evaluate our model, as explained earlier, we compute appropriate evaluation metrics. Metrics alone, usually, will not give the perfect idea of how good our model is. We know, depending on the evaluation metric, which values are considered good and which are not, but to have some more precise clue about the quality of our model we need to compare it with some baselines. For this purpose, we implemented an artificial competitor that will represent the baseline model used for comparison. Having in mind the nature of the platform that we are building, one way to evaluate your model will also be the comparison to other competitors, but their models remain unknown to other users. The baseline model is important as it is the basic model and if your model is any good, it should be better than the baseline.

Since we support both classification and regression we had to provide different types of baselines. Depending on the competition type appropriate baseline will be activated. Some of the most simple classifiers out there are the ones that predict: random value, constant value, average value etc. As mentioned earlier, to provide support for various types of competition we implemented:

1. **Classification baseline** - We chose the majority class classifier. This model is used for classification and it predicts the most common class for the target. During the training period it counts class appearances and later based on those numbers predicts a class. This classifier is really simple and usually will not provide high accuracy, but it is good to compare with to realize if the model that you are using is any good.

2. **Regression baseline** - As regression baseline, we chose average value predictor. This model keeps the statistic of the average value of the target and predicts the average value at a given moment. It is also very simple as we only keep track of the sum of values and the number of instances. It will not provide high accuracy but again will give a clue if our model is better than the simple one.

## 5.4 Real-time Machine Learning competition on SCALAR

We have conducted a real-case competition on data streams using SCALAR for IEEE Big Data Cup Challenge (Boulegane et al., 2019). We have involved several competitors from all over the world to adhere to this novel competition. Our competition scenario falls within the network activity monitoring application. This includes time series forecasting where the goal is to predict at time $t$ the $n$ upcoming values of the series based on values observed in the past. We describe below the data used to run the competition in addition to the settings related to the release of the stream and the evaluation.

### 5.4.1 Data

We have long-term, continuous data from the network activity recorded every second summarising the density of communications in terms of the number of messages conveyed. The dataset under study is synthetic but strongly inspired by real time-series following this use-case. The time series has been specifically designed to meet the challenges of stream mining, comprising:

- **Concept drift**: stands for a change in the statistical properties of the target variable that we are trying to predict. The change can be *Sudden* or *Gradual*. *Sudden* change occurs when the distribution has remained unchanged for a long time, then changes in a few steps to a significantly different one. *Gradual* change occurs when, for a long time the distribution experiences at each step a tiny, barely noticeable change, but these accumulated changes become significant over time (Gama et al., 2014).
- **Anomalies**: are patterns in data that do not conform to a well defined notion of normal behavior (Chandola et al., 2009). We address two types of anomalies: *Point* and *Group* anomalies. If an individual data instance can be considered as anomalous to the rest of the data, then the instance is referred to as a *point anomaly*. On the other hand, *group anomaly* is a set of consecutive points where the points are relatively normal, but as a whole they are unusual.

### 5.4.2 Workflow

We describe below the setting used to run the competition following the streaming setting. The stream settings are as follows:

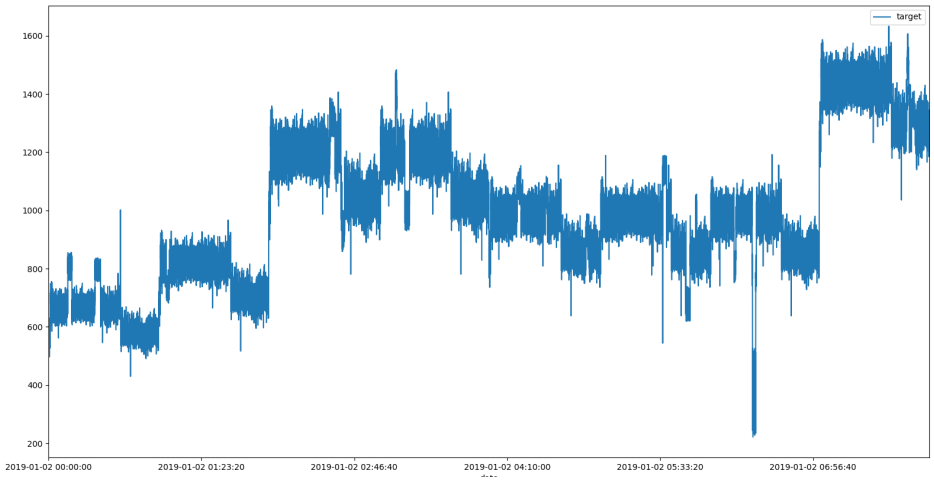



Figure 5.5: Data stream used in competition

- **Initial batch**: we have initially provided a time series lasting for $2$ days with a record every second summarizing the number of messages conveyed in the network. A period of time was granted to competitors to tune their models at first before the stream of test instances is released.
- **Regular batch**: a batch of fixed size is sent at regular time intervals ($5$ seconds). The latter contains $5$ test instances to be predicted before the deadline assigned (arrival of the next batch). The batch contains also true values corresponding to the test instances released in the batch before. If predictions are submitted after the due date, the competitor is penalized.

**Evaluation.** We conduct a prequential evaluation scenario where each instance is first used to test and then to train. We provide a range of evaluation metrics to support time-series forecasting tasks and keep track of the time delay while submitting the prediction. We compute time latency for each competitor to show the average delay for submitting. One can distinguish between models that are less time-consuming and greedy. We provide the following evaluation metrics where $n$ is the number of samples, $Y$ is the true value and $\hat{Y}$ is the predicted value:

1. **Mean absolute percentage error (MAPE)**: By definition, this measure cannot be used if the target takes zero value. It gives the value of error as a percentage. The definition is given by equation 5.1.

$$MAPE = 100\% \frac{1}{n} \sum_{t=1}^{n} |\frac{Y_t - \hat{Y}_t}{Y_t}| \tag{5.1}$$

2. **Mean absolute error (MAE)**: Represents the absolute difference between continuous values. The definition is given by equation 5.2.

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|Y_t - \hat{Y}_t| \tag{5.2}$$

3. **Mean squared error (MSE)**: Measures the average of squares of error. The definition is given by equation 5.3.

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(Y_t - \hat{Y}_t)^2 \tag{5.3}$$

## 5.5 Winning Solutions

In this section, we present the solutions which achieved the best scores. Three solutions that yielded the best results are:

**1. Adaptive moving average on n previous equally distant windows method (AMA).**

This model takes advantage of the periodicity of the data stream. It takes into account the current window and previous *k* windows where the distance between all the windows is the period of the stream. The period of the stream is computed using Partial Auto-correlation Function (PACF). The size of the window is updated during the stream. For every one of these windows, the average value is computed.

$$window\_mean_k = \frac{1}{win\_size}\sum_{i=t-k*period}^{t-k*period+win\_size} Y_i \tag{5.4}$$

The model keeps only $n$ windows whose mean is the closest to the mean value of the current window. This avoids considering the mean of the windows during anomalies periods. The prediction for the next $m$ values is made by computing the distance between the mean of each window and every one of the next $m$ points following the window.

$$prediction\_mean_l = \frac{1}{n}\sum_{i=1}^{n}|Y_l - window\_mean_i| \tag{5.5}$$

where $l$ in $[1, m]$. The prediction is then computed as the sum of the mean value of the current window and the average distance for every of $m$ points.

$$prediction_l = (current\_window\_mean + prediction\_mean_l) \tag{5.6}$$

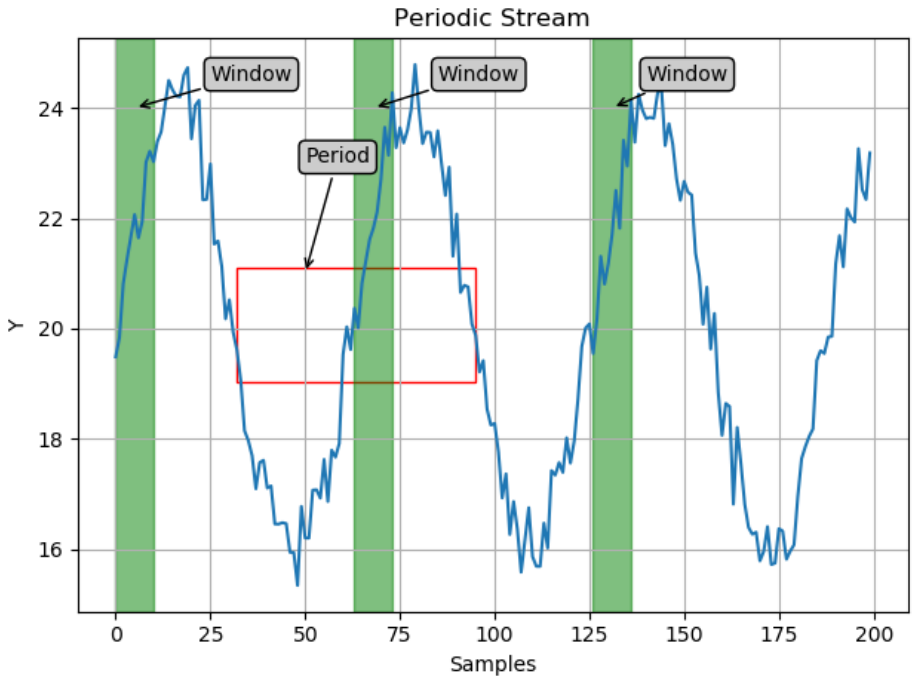



Figure 5.6: Windows with a distance equal to stream period

where $l$ in $[1, m]$.

Additionally, the model adapts its prediction methods to deal with anomalies. Anomalies are assumed to be the values whose distance to the window mean value is greater than the threshold $t$. If the prediction is to be made inside, what is detected as, an anomaly period, the prediction is made by a simple moving average algorithm. The method requires a vector of 9 parameters such as the windows size, the threshold to detect anomalies, the number of windows to consider, etc... During the competition a custom genetic algorithm makes these parameters evolve to fit as best as possible the past 30 minutes data.

**2. Moving average method with Robust Random Cut Forest for anomaly detection (MA-RRCF).**

The second solution uses the moving average method to predict the next values and Robust Random Cut Forest (RRCF)(Guha et al., 2016). The prediction model, in this case, relies on the fact that the data has an increasing, monotonous pattern and thus using the moving average method for prediction seems adequate. The training phase consists of keeping the last $m$ values, where $m$ is the

size of the batch in the stream, and computing the average value:

$$prediction_t = \frac{1}{m} \sum_{i=t-m}^{t-1} Y_i \tag{5.7}$$

The average value of the last $m$ points is then used for the whole next batch and then updated in the same way. To deal with anomalies this solution proposes using $\mathrm{RRCF}$. $\mathrm{RRCF}$ provides anomaly detection on streaming data. It uses the ensemble of independent Robust Random Cut Trees. It constructs a tree of 10-1000 vertices from a random sampling of the pool and creates more trees of the same size 1-1000 times which creates the forest. It decides whether the new data point is an anomaly or not by injecting it into the trees and analyzing how much that changes the complexity of the forest($e.g.$ depth). Previous values in the stream are used to build the trees and new values are inserted and the average displacement degree is computed to decide if the point is an anomaly or not. If the degree is larger than the threshold, that point is assumed to be an anomaly and then it is replaced by the average computed on the whole stream.

**3. Mondrian Forest with Robust Random Cut Forest for anomaly detection (MF-RRCF).**

The third solution proposes using regression Mondrian Forest (Lakshminarayanan et al., 2014) for prediction and $\mathrm{RRCF}$ for detecting anomalies. Mondrian Forests are fast incremental random forests that use Mondrian Processes(Roy et al., 2008) to build the ensembles of random decision trees. For this specific use case, since the data stream represents the univariate time series, to use Mondrian Forest to make predictions the data had to be transformed. It is necessary to provide the features and the target to Mondrian Forest, so the stream is transformed in a way that previous $n$ values are considered as features for the current value. The example of the time series is given in Table 5.1 and its transformation to the data stream is shown in Table 5.2.

The second part of the solution deals with detecting the anomalies. Similar to the previous solution, here also $\mathrm{RRCF}$ are used for anomaly detection. The difference is that anomaly values are being replaced not by the average on the whole stream but by the average of the last received batch.

### 5.5.1 Results

In this part, we present the results achieved by the mentioned solutions. We present the results achieved during the competition following all the rules of the competition which include penalties for late predictions or not sending predictions at all. After we present the results that have been achieved during

| $date$ | $target$ |
|---|---|
| $00:00:01$ | $397$ |
| $00:00:02$ | $388$ |
| $00:00:03$ | $341$ |
| $00:00:04$ | $361$ |
| $00:00:05$ | $359$ |
| $00:00:06$ | $369$ |
| $00:00:07$ | $387$ |
| $00:00:08$ | $415$ |

Table 5.1: Time series

| $date$ | $target$ | $previous1$ | $previous2$ | $previous3$ |
|---|---|---|---|---|
| $00:00:04$ | $361$ | $341$ | $388$ | $397$ |
| $00:00:05$ | $359$ | $361$ | $341$ | $388$ |
| $00:00:06$ | $369$ | $359$ | $361$ | $341$ |
| $00:00:07$ | $387$ | $369$ | $359$ | $361$ |
| $00:00:08$ | $415$ | $387$ | $369$ | $359$ |

Table 5.2: Time series transformed into the data stream with a number of features $n = 3$

independent testing of provided solutions, on a wider test set and without any disturbance where prediction has been sent for all records in the stream on time. We evaluate models using Mean Absolute Percentage Error($\mathrm{MAPE}$), described in section 5.4.2. Also, we note the time latency, to measure how fast models were in sending predictions.

We compare the proposed methods with the baseline model, which in the case of regression was the model predicting the average value of whole data. In the competition results, we ignore the time latency of the baseline model since it doesn't include the delays caused by the network. The results show that in both scenarios the Adaptive Moving Average method had the best performance. Also with the lowest time latency. The better performance comes from the adaptive nature of the algorithm, which chooses the best algorithms online while the stream is running and specifically the adaptive way of treating the anomalies. The differences in the results in Table 5.3 and Table 5.4 are coming from delays in network communication. Due to those delays, some predictions have been received late and thus penalized, which caused higher values of $\mathrm{MAPE}$ and

| Team | Method | MAPE | Latency$[s]$ |
|---|---|---|---|
| Ghislain Fievet | $AMA$ | **12.71261** | **0.68632** |
| Sohn Jimin | $MA - RRCF$ | 13.10381 | 0.91592 |
| Yeonwoo Nam | $MF - RRCF$ | 13.73831 | 1.04440 |
| Baseline | $MEAN$ | 16.50072 | 0.0055 |

Table 5.3: Competition results

| Team | Method | MAPE | Latency$[s]$ |
|---|---|---|---|
| Ghislain Fievet | $AMA$ | **3.64622** | 0.09418 |
| Yeonwoo Nam | $MF - RRCF$ | 3.78128 | 0.27986 |
| Sohn Jimin | $MA - RRCF$ | 4.16643 | 0.11134 |
| Baseline | $MEAN$ | 16.03232 | **0.00347** |

Table 5.4: Test results

latency too. Also, we have participants who started the competition after the official start causing them to be penalized for the records that they have missed.

# 6 Conclusion

## 6.1 Summary

**Post-hoc interpretations of black-box models.** The first part of this manuscript focused extensively on post-hoc interpretations of black box models for tabular data. Importantly, the use of post-hoc interpretations does not negate the inherent advantages of leveraging powerful black-box models. Instead, it empowers us to harness the full potential of these sophisticated algorithms while ensuring that their decision-making processes remain interpretable and comprehensible.

We recognize that interpreting complex black-box models in a comprehensible way is the main challenge of post-hoc interpretations. Additionally, we acknowledge that most of the state-of-the-art approaches rely on randomization techniques and synthetic data which introduces uncertainty and hinders the trust in AI models overall. Finally, the interpretations of individual predictions are computed to reflect the black-box model's behavior only for a single instance or a small artificial neighbourhood. Thus, we proposed two post-hoc approaches: STACI and BELLA. While they address the interpretability from different scopes, both methods address the aforementioned limitations of current state-of-the-art approaches and provide deterministic, accurate, comprehensible, and general interpretations.

**STACI** is a deterministic *post-hoc* method for providing interpretations of black box classification models. It interprets the black-box model on the global level but thanks to the tree structure of surrogate models it provides interpretations for individual predictions as well. Our method uses one surrogate decision tree per class, each trained using the *F1 score* as a metric to decide a split. The resulting models provide simple, confident, but general interpretations. Originally it

was designed only for classification, but then it was extended to support also regression problems.

Since it relies only on the existing data points from the training set, it is a deterministic approach and the generality of its interpretations is grounded in real data points. We have shown that our new method outperforms state of the art methods in terms of fidelity, complexity, confidence and generality. The experiments suggest that with our method we obtain higher accuracy, coupled with higher generality and lower simplicity. Not only we balance this trade-off better than state-of-the-art approaches, but we achieve better results on each metric. Finally, our user study confirms that the metrics we proposed, confidence and generality, are important features of an interpretation and that users prefer our interpretations over others.

**BELLA** is a deterministic approach to provide post-hoc local explanations for any regression black-box model, or indeed any static tabular dataset with a numeric variable to be explained. It can provide both factual and counterfactual explanations. The local linear approximations are computed by optimizing an objective function that ensures accurate, general, and simple explanations.

Translating the idea of generality from our earlier work to regression problems, resulted in an algorithm to construct an optimal neighbourhood, which depicts similar data points that can be explained with one simple local linear model. We showed through detailed experiments that BELLA outperforms state-of-the-art approaches in terms of generality and simplicity of explanations, while achieving high accuracy. Again, our user study showed, that even in the case of individual explanations, generality plays an important role. Additionally, being able to point out real data points that are affected by the same decision, adds to users' trust in the model.

**Data Stream Mining.** Data stream mining is of paramount importance in today's fast-paced world, where data is generated at an unprecedented rate from various sources such as sensors, social media, and IoT devices. Unlike traditional batch data processing, data stream mining deals with continuous and rapidly changing data streams in real-time or near-real-time. While there are platforms like Kaggle that excel in hosting data science competitions on static datasets, there is indeed a lack of similar platforms specifically tailored for data stream mining competitions. This gap represents a missed opportunity for the data science community to explore and tackle real-world challenges related to streaming data. Therefore, we developed SCALAR to address these drawbacks of the popular existing platforms, tailored only for batch learning.

**SCALAR** is the first of its kind, a platform for data science competitions on data streams. Its architecture was designed to support the distribution of the data to the users and the processing of incoming data from users in real-time.

In contrast to the existing data science competition platforms, SCALAR stands alone as the one that supports online learning scenarios.

SCALAR has been developed in Python and it relied on popular open source tools for processing big data and data streams. It was designed to let users choose their preferred working environment. Users have to conform to the communication protocol, but other than that, they are free to use different programming languages, and different tools to train their machine learning model. The platform provides real-time evaluation and a leaderboard. It integrates also the baseline models, so that users can track how their model performs in comparison to others to the baseline model as well. We had the opportunity to try the platform in a real world setting, as we organized a machine learning competition on data streams. It proved to be a very exciting experience both for us, the organizers, as well as for the participants.

## 6.2 Future Works

STACI and BELLA showed that accuracy is not the only important parameter to ensure users' trust in the model. Both methods allow for further improvements. In many applications, human intervention could lead to more plausible explanations. For example, our counterfactual explanations could be improved if users specified which features can be modified. STACI can be further improved such that it automatically computes the model complexity and stops growing the tree once the criteria have been met.

To extend BELLA, one can investigate if density-based clustering algorithms can be used for computing the neighborhoods. Finally, a common challenge for all distance-based machine learning algorithms is the search for more efficient ways to compute neighborhoods. For example, one way can be to investigate the Approximate Nearest Neighbours methods.

For SCALAR, an important step would be to focus on growing the community around it. SCALAR is a prototype solution for the stream data mining platform and there still remains a lot of possibilities to improve the platform. This would allow us to test the platform for scalability and also to show the benefits that our platform offers.

Some of the possible ways to improve the platform is to explore ways to enable it to consume data from various sources. This can be extended through options offered by Kafka Connect. Also, the system should be able to send push notifications to users in order to inform them if there is some problem in communication or if the predictions are arriving late.

### 6.2.1 Perspectives

We hope that the results that our methods achieve, as well as the users' preferences demonstrated through our user studies, will lead towards a definite set of desiderata that will define what *"a good explanation"* is. While it is crucial that explanations depict the reasoning of the black-box models, we aimed to investigate what other characteristics the explanations should have to satisfy the expectations of their recipients. Especially with recent advancements, AI is becoming omnipresent, and it affects almost every person on this planet. Therefore, we tried to determine which parameters are important to end users.

We believe that getting to the transparent and responsible AI is a battle in multiple fields. On one side, it needs to ensure faithful interpretation of black-box models and on the other, it needs to ensure that those can be presented and justified in an understandable way. As xAI research continues to advance, it will remain a vital component in the ethical and sustainable integration of AI across various domains, supporting the realization of a more transparent, responsible, and socially beneficial AI-powered future.

# Bibliography

A. Adadi and M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *Access*, 6, 2018.

A. Ahmad and L. Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering*, 63(2):503–527, 2007.

E. Alpaydin. *Introduction to machine learning*. MIT press, 2020.

D. Alvarez Melis and T. Jaakkola. Towards robust interpretability with self-explaining neural networks. *Advances in neural information processing systems*, 31, 2018.

D. Alvarez-Melis and T. S. Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

P. Angelov and E. Soares. Towards explainable deep neural networks (xdnn). *Neural Networks*, 130:185–194, 2020.

Apache. Spark structured streaming programming guide. https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#structured-streaming-programming-guide, 2018.

M. Armbrust, T. Das, J. Torres, B. Yavuz, S. Zhu, R. Xin, A. Ghodsi, I. Stoica, and M. Zaharia. Structured streaming: A declarative api for real-time applications in apache spark. In *Proceedings of the 2018 International Conference on Management of Data*, pages 601–613, 2018.

A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al. Explainable artificial

intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

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

P. C. Austin and E. W. Steyerberg. The number of subjects per variable required in linear regression analyses. *Journal of clinical epidemiology*, 68(6):627–636, 2015.

N. Barakat and J. Diederich. Eclectic rule-extraction from support vector machines. *International Journal of Computational Intelligence*, 2(1):59–62, 2005.

N. H. Barakat and A. P. Bradley. Rule extraction from support vector machines: A sequential covering approach. *IEEE Transactions on Knowledge and Data Engineering*, 19(6):729–741, 2007.

D. Barbella, S. Benzaid, J. M. Christensen, B. Jackson, X. V. Qin, and D. R. Musicant. Understanding support vector machine classifications via a recommender system-like approach. *DMIN*, 1:305–311, 2009.

O. Bastani, C. Kim, and H. Bastani. Interpreting blackbox models via model extraction. *arXiv preprint arXiv:1705.08504*, 2017.

V. Beaudouin, I. Bloch, D. Bounie, S. Clémençon, F. d'Alché Buc, J. Eagan, W. Maxwell, P. Mozharovskyi, and J. Parekh. Flexible and context-specific ai explainability: a multidisciplinary approach. *SSRN 3559477*, 2020.

K. J. Berry and P. W. Mielke Jr. A generalization of cohen's kappa agreement measure to interval measurement and multiple raters. *Educational and Psychological Measurement*, 48(4):921–933, 1988.

A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.

M. Z. Bill Chambers. *Spark: The Definitive Guide, Big Data Processing Made Simple*. O'Reilly Media, 2018.

S. K. Biswas, M. Chakraborty, B. Purkayastha, P. Roy, and D. M. Thounaojam. Rule extraction from training data using neural network. *International Journal on Artificial Intelligence Tools*, 26(03):1750006, 2017.

A. Bloniarz, A. Talwalkar, B. Yu, and C. Wu. Supervised neighborhoods for distributed nonparametric regression. In *Artificial Intelligence and Statistics*, pages 1450–1459. PMLR, 2016.

D. Boulegane, N. Radulovic, A. Bifet, G. Fievet, J. Sohn, Y. Nam, S. Yu, and D.-W. Choi. Real-time machine learning competition on data streams at the ieee big data 2019. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3493–3497. IEEE, 2019.

O. Boz. Extracting decision trees from trained neural networks. In *SIGKDD*, 2002.

L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

N. Burkart and M. F. Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

E. J. Candès and Y. Plan. Near-ideal model selection by $l1$ minimization. *The Annals of Statistics*, 37(5A):2145–2177, 2009.

V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

J. Chen, L. Song, M. Wainwright, and M. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *International conference on machine learning*, pages 883–892. PMLR, 2018.

J. Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

W. W. Cohen. Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier, 1995.

E. Commission. *Proposal for a Regulation Laying down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act)*. European Commission: Brussels, Belgium, 2021, 2021.

CRAN. An introduction to r programming language. https://cran.r-project.org/, 2018.

M. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, 1996.

G. Cugola and A. Margara. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv.*, 44(3):15:1–15:62, June 2012. ISSN 0360-0300. doi: 10.1145/2187671.2187677. URL http://doi.acm.org/10.1145/2187671.2187677.

Z. Cui, W. Chen, Y. He, and Y. Chen. Optimal action extraction for random forests and boosted trees. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 179–188, 2015.

S. Dandl, C. Molnar, M. Binder, and B. Bischl. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*, pages 448–469. Springer, 2020.

A. Das and P. Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

G. Developers. Protcol buffers developer guide. https://developers.google.com/protocol-buffers/docs/overview, 2018.

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

F. K. Došilović, M. Brčić, and N. Hlupić. Explainable artificial intelligence: A survey. In *MIPRO*, 2018.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

T. A. Etchells and P. J. Lisboa. Orthogonal search-based rule extraction (osre) for trained neural networks: a practical and efficient approach. *IEEE transactions on neural networks*, 17(2):374–384, 2006.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine learning*, 29(2):131–163, 1997.

N. Frosst and G. Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

L. Fu. Rule generation from neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(8):1114–1124, 1994.

G. Fung, S. Sandilya, and R. B. Rao. Rule extraction from linear support vector machines. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 32–40, 2005.

J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):44, 2014.

N. Garg. *Apache Kafka*. Packt Publishing Ltd, 2013.

B. Goodman and S. Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3), 2017.

gRPC. grpc documentation. https://grpc.io/docs/, 2018.

S. Guha, N. Mishra, G. Roy, and O. Schrijvers. Robust random cut forest based anomaly detection on streams. In *International conference on machine learning*, pages 2712–2721, 2016.

R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, 2018a.

R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys*, 51(5), 2018b.

P. Hall, N. Gill, M. Kurka, and W. Phan. Machine learning interpretability with h2o driverless ai. *H2O. ai*, 2017.

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

S. Harikumar and P. Surya. K-medoid clustering for heterogeneous datasets. *Procedia Computer Science*, 70:226–237, 2015.

J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1): 100–108, 1979.

T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

J. Hopcroft and R. Kannan. *Foundations of data science*. Cambridge University Press;, 2014.

A. Ignatiev, N. Narodytska, and J. Marques-Silva. Abduction-based explanations for machine learning models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33(01), pages 1511–1519, 2019.

R. Jafari-Marandi, J. Denton, A. Idris, B. K. Smith, and A. Keramati. Optimum profit-driven churn decision making: innovative artificial neural networks in telecom industry. *Neural Computing and Applications*, 32:14929–14962, 2020.

H. Janson and U. Olsson. A measure of agreement for interval or nominal multivariate observations. *Educational and Psychological Measurement*, 61(2): 277–289, 2001.

H. Janson and U. Olsson. A measure of agreement for interval or nominal multivariate observations by different sets of judges. *Educational and Psychological Measurement*, 64(1):62–70, 2004.

U. Johansson and L. Niklasson. Evolving decision trees using oracle guides. In *Symp. on Computational Intelligence and Data Mining*, 2009.

J. Kazemitabar, A. Amini, A. Bloniarz, and A. S. Talwalkar. Variable importance using decision trees. *Advances in neural information processing systems*, 30, 2017.

J. N. Kok, E. J. Boers, W. A. Kosters, P. Van der Putten, and M. Poel. Artificial intelligence: definition, trends, techniques, and cases. *Artificial intelligence*, 1: 270–299, 2009.

J. Kreps, N. Narkhede, J. Rao, et al. Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, volume 11, pages 1–7, 2011.

H. Lakkaraju, S. H. Bach, and J. Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016.

H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec. Faithful and customizable explanations of black box models. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 131–138, 2019.

B. Lakshminarayanan, D. M. Roy, and Y. W. Teh. Mondrian forests: Efficient online random forests. In *Advances in neural information processing systems*, pages 3140–3148, 2014.

N. Leopold and O. Rose. Unic: A fast nonparametric clustering. *Pattern Recognition*, 100:107117, 2020.

B. Letham, C. Rudin, T. H. McCormick, D. Madigan, et al. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3), 2015.

A. V. Looveren and J. Klaise. Interpretable counterfactual explanations guided by prototypes. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 650–665. Springer, 2021.

S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *NEURIPS*, 2017.

S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1): 56–67, 2020.

D. Martens, B. Baesens, and T. Van Gestel. Decompositional rule extraction from support vector machines by active learning. *IEEE Transactions on Knowledge and Data Engineering*, 21(2):178–191, 2008.

M. Mashayekhi and R. Gras. Rule extraction from decision trees ensembles: new algorithms based on heuristic search and sparse group lasso methods. *International Journal of Information Technology & Decision Making*, 16(06):1707–1727, 2017.

Mayo-Clinic. Diabetes, 2023. URL https://www.mayoclinic.org/diseases-conditions/diabetes/diagnosis-treatment/drc-20371451.

J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12–12, 2006.

N. Meinshausen. Relaxed lasso. *Computational Statistics & Data Analysis*, 52(1): 374–393, 2007.

D. Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2, 2014.

T. Miller. Contrastive explanation: A structural-model approach. *arXiv preprint arXiv:1811.03163*, 2018.

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

C. Molnar. A guide for making black box models explainable. *URL: https://christophm. github. io/interpretable-ml-book*, 2:3, 2018.

G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern recognition*, 65:211–222, 2017.

J. Montiel, J. Read, A. Bifet, and T. Abdessalem. Scikit-multiflow: A multi-output streaming framework. *The Journal of Machine Learning Research*, 19(1):2915–2914, 2018.

R. K. Mothilal, A. Sharma, and C. Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 607–617, 2020.

W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

M. Nauta, J. Trienes, S. Pathak, E. Nguyen, M. Peters, Y. Schmitt, J. Schlötterer, M. van Keulen, and C. Seifert. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *arXiv preprint arXiv:2201.08164*, 2022.

T. P. Neha Narkhede, Gwen Shapira. *Kafka: The Definitive Guide, Real-Time Data and Stream Processing at Scale*. O'Reilly Media, 2017.

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

F. Pedregosa, G. Varoquaux, Gramfort, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2011.

G. Plumb, D. Molitor, and A. S. Talwalkar. Model agnostic supervised local explanations. *Advances in neural information processing systems*, 31, 2018.

R. Poyiadzi, K. Sokol, R. Santos-Rodriguez, T. De Bie, and P. Flach. Face: feasible and actionable counterfactual explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 344–350, 2020.

N. Radulovic, D. Boulegane, and A. Bifet. Scalar-a platform for real-time machine learning competitions on data streams. *Journal of Open Source Software*, 5(56): 2676, 2020.

N. Radulovic, A. Bifet, and F. Suchanek. Confident interpretations of black box classifiers. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

N. Radulovic, A. Bifet, W. Maxwell, and F. Suchanek. Confident interpretations of black box classifiers. In *preprint*, pages 1–8. arxiv, 2023a.

N. Radulovic, A. Bifet, and F. Suchanek. Bella: Black box model explanations by local linear approximations. In *PREPRINT*. **READY FOR SUBMISSION TO A CONFERENCE**, 2023b. URL https://arxiv.org/abs/2305.11311.

M. T. Ribeiro, S. Singh, and C. Guestrin. Why should i trust you? – explaining the predictions of any classifier. In *SIGKDD*, 2016.

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

R. L. Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

J. D. Romano, T. T. Le, W. La Cava, J. T. Gregg, D. J. Goldberg, P. Chakraborty, N. L. Ray, D. Himmelstein, W. Fu, and J. H. Moore. Pmlb v1.0: an open source dataset collection for benchmarking machine learning methods. *arXiv preprint arXiv:2012.00058v2*, 2021.

P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

D. M. Roy, Y. W. Teh, et al. The mondrian process. In *NIPS*, pages 1377–1384, 2008.

C. Rudin. Please stop explaining black box models for high stakes decisions. *Stat*, 1050:26, 2018.

B. Russell and S. Blackburn. *Why I am not a Christian: and other essays on religion and related subjects*. Routledge, 2020.

C. Russell. Efficient search for diverse coherent explanations. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 20–28, 2019.

A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

M. K. Sarker, N. Xie, D. Doran, M. Raymer, and P. Hitzler. Explaining trained neural networks with semantic web technologies: First steps. *arXiv preprint arXiv:1710.04324*, 2017.

R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.

R. A. Stine. Graphical interpretation of variance inflation factors. *The American Statistician*, 49(1):53–56, 1995.

S. Tan, R. Caruana, G. Hooker, P. Koch, and A. Gordo. Learning global additive explanations for neural nets using model distillation. *arXiv preprint arXiv:1801.08640*, 2018.

R. C. Team et al. R: A language and environment for statistical computing, 2013.

G. Tolomei, F. Silvestri, A. Haines, and M. Lalmas. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 465–474, 2017.

L. Torgo and J. Gama. Regression by classification. In *Brazilian symposium on artificial intelligence*, pages 51–60. Springer, 1996.

K. Ushey, J. Allaire, and Y. Tang. *reticulate: Interface to 'Python'*, 2022. https://rstudio.github.io/reticulate/, https://github.com/rstudio/reticulate.

B. Ustun and C. Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3), 2016.

M. Veale and F. Zuiderveen Borgesius. Demystifying the draft eu artificial intelligence act—analysing the good, the bad, and the unclear elements of the proposed approach. *Computer Law Review International*, 22(4):97–112, 2021.

V. Vo, V. Nguyen, T. Le, Q. H. Tran, G. Haffari, S. Camtepe, and D. Phung. An additive instance-wise approach to multi-class model interpretation. *arXiv preprint arXiv:2207.03113*, 2022.

S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31: 841, 2017a.

S. Wachter, B. D. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *CoRR*, abs/1711.00399, 2017b. URL http://arxiv.org/abs/1711.00399.

A. White and A. d. Garcez. Measurable counterfactual local explanations for any classifier. *arXiv preprint arXiv:1908.03020*, 2019.

M. Wu, M. Hughes, S. Parbhoo, M. Zazzi, V. Roth, and F. Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32-1, 2018.

C. Yang, A. Rangarajan, and S. Ranka. Global model interpretation via recursive partitioning. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1563–1570. IEEE, 2018.

H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. In *ICML*, 2017.

J. Yoon, J. Jordon, and M. van der Schaar. Invase: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*, 2018.

M. R. Zafar and N. M. Khan. Dlime: A deterministic local interpretable model-agnostic explanations approach for computer-aided diagnosis systems. *arXiv preprint arXiv:1906.10263*, 2019.

M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, et al. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.

Y. Zhang, K. Song, Y. Sun, S. Tan, and M. Udell. " why should you trust my explanation?" understanding uncertainty in lime explanations. *arXiv preprint arXiv:1904.12991*, 2019.

X. Zhao, Y. Wu, D. L. Lee, and W. Cui. iforest: Interpreting random forests via visual analytics. *IEEE transactions on visualization and computer graphics*, 25(1): 407–416, 2018.

Z.-H. Zhou, Y. Jiang, and S.-F. Chen. Extracting symbolic rules from trained neural network ensembles. *Ai Communications*, 16(1):3–15, 2003.

J. R. Zilke, E. Loza Mencía, and F. Janssen. Deepred–rule extraction from deep neural networks. In *International conference on discovery science*, pages 457–473. Springer, 2016.

ECOLE DOCTORALE

**Titre:** L'IA Explicable a posteriori pour les modèles de boîte noire sur les données tabulaires

**Mots clés:** Apprentissage automatique, Interprétabilité, Explicabilité, Classification, Régression, Données tabulaires

**Résumé:** Les modèles d'intelligence artificielle (IA) actuels ont fait leurs preuves dans la résolution de diverses tâches, telles que la classification, la régression, le traitement du langage naturel (NLP) et le traitement d'images. Les ressources dont nous disposons aujourd'hui nous permettent d'entraîner des modèles d'IA très complexes pour résoudre différents problèmes dans presque tous les domaines : médecine, finance, justice, transport, prévisions, etc. Avec la popularité et l'utilisation généralisée des modèles d'IA, la nécessité d'assurer la confiance dans ces modèles s'est également accrue. Aussi complexes soient-ils aujourd'hui, ces modèles d'IA sont impossibles à interpréter et à comprendre par les humains. Dans cette thèse, nous nous concentrons sur un domaine de recherche spécifique, à savoir l'intelligence artificielle explicable (xAI), qui vise à fournir des approches permettant d'interpréter les modèles d'IA complexes et d'expliquer leurs décisions. Nous présentons deux approches, STACI et BELLA, qui se concentrent sur les tâches de classification et de régression, respectivement, pour les données tabulaires. Les deux méthodes sont des approches post-hoc agnostiques au modèle déterministe, ce qui signifie qu'elles peuvent être appliquées à n'importe quel modèle boîte noire après sa création. De cette manière, l'interprétabilité présente une valeur ajoutée sans qu'il soit nécessaire de faire des compromis sur les performances du modèle de boîte noire. Nos méthodes fournissent des interprétations précises, simples et générales à la fois de l'ensemble du modèle boîte noire et de ses prédictions individuelles. Nous avons confirmé leur haute performance par des expériences approfondies et étude d'utilisateurs.

**Title:** Post-hoc Explainable AI for Black Box Models on Tabular Data

**Keywords:** Machine Learning, Interpretability, Explainability, Classification, Regression, Tabular Data

**Abstract:** Current state-of-the-art Artificial Intelligence (AI) models have been proven to be very successful in solving various tasks, such as classification, regression, Natural Language Processing (NLP), and image processing. The resources that we have at our hands today allow us to train very complex AI models to solve different problems in almost any field: medicine, finance, justice, transportation, forecast, etc. With the popularity and widespread use of the AI models, the need to ensure the trust in them also grew. Complex as they come today, these AI models are impossible to be interpreted and understood by humans. In this thesis, we focus on the specific area of research, namely Explainable Artificial Intelligence (xAI), that aims to provide the approaches to interpret the complex AI models and explain their decisions. We present two approaches STACI and BELLA which focus on classification and regression tasks, respectively, for tabular data. Both methods are deterministic model-agnostic post-hoc approaches, which means that they can be applied to any black-box model after its creation. In this way, interpretability presents an added value without the need to compromise on black-box model's performance. Our methods provide accurate, simple and general interpretations of both the whole black-box model and its individual predictions. We confirmed their high performance through extensive experiments and a user study.

**Institut Polytechnique de Paris**
91120 Palaiseau, France