Towards the improvement of decision tree learning

A perspective on search and evaluation

Cecília Nunes

TESI DOCTORAL UPF / 2019

THESIS SUPERVISORS
Oscar Camara, Anders Jonsson

Department Engineering and Information and Communication Technologies Academic Coordination Unit

Universitat Pompeu Fabra Barcelona

Acknowledgements

I can’t begin to express my gratitude towards my supervisors, official and unofficial ones, for their continuous support on so many different levels. To Anders Jonsson, Oscar Camara, Bart Bijnens, Hélène Langet and Mathieu De Craene, it was an honor to have had the opportunity to work under your supervision. Thank you for all that you have taught me.

I would also like to acknowledge the support and companionship of all the incredible people I’ve met along the way, in the UPF, BCN MedTech, Cardiofunxion or in Medisys, including Carlos, Paula, Nerea, Fede, Rubén, Sergio, Marta, Oualid, Eric, Bruno, Guille, Amelia, Laura, Judith, Irem, Christian, Hadrien, Hernan, Constance, Caroline, Jean-Michel, and Simone. To Nathalie, thank you for all the help and dedication, despite the fights with Concur! To Aurelio, thank you for your outstanding dedication and professionalism, and to Lydia for being the secret boss of the department. To the amazing first-floor people of Residence André de Gouveia, without whom it would have been impossible to survive through a gray Parisian winter: Rosi, Bruno, Gi, Lia, Mafalda, Filipa, Filipa 2, Filipa 3, Sofia, Sílvia, Nikos, Danilo, Madre, Louise, Margarida, Dora, Michelle, and Berta. To all the people of the department with whom I’ve shared one or more much needed vermuts on Friday evenings. A very special thanks to Roland for allowing me to use his painting in the cover. A big hug to my friends back home and all over the place: Catarina, Gulce, Joana, Cláudia, the amazing ”Menas", Olga, Maria, Federico, Matteo, Georgi, Carlos, Theresa, Maria, Gabriel, Jeni, and Mariana, my Portuguese partner in crime throughout this PhD.

Finally, I would like to address a thank you to those whom I am lucky to call family (biological or otherwise, alive or already on the first floor), and have mysteriously persevered in putting up with me: avó Gina, avô Armando, avó Cecília, avô Amaro, Luísa, José João, Dinora, my crazy sisters Beatriz and Teresa, as well as my pipocas Maguie and Mafa, for always being there for me despite the distance. A wholehearted thank you to Xavi for the love and support.

Abstract

Data mining and machine learning (ML) are increasingly at the core of many aspects of modern life. With growing concerns about the impact of relying on predictions we cannot understand, there is widespread agreement regarding the need for reliable interpretable models. One of the areas where this is particularly important is clinical decision-making. Specifically, explainable models have the potential to facilitate the elaboration of clinical guidelines and related decision-support tools. The presented research focuses on the improvement of decision tree (DT) learning, one of the most popular interpretable models, motivated by the challenges posed by clinical data.

One of the limitations of interpretable DT algorithms is that they involve decisions based on strict thresholds, which can impair performance in the presence noisy measurements. In this regard, we proposed a probabilistic method that takes into account a model of the noise in the distinct learning phases. When considering this model during training, the method showed moderate improvements in accuracy compared to the standard approach, but significant reductions in number of leaves.

Standard DT algorithms follow a locally-optimal approach which, despite providing good performances at a low computational cost, does not guarantee optimal DTs. The second direction of research therefore concerned the development of a non-greedy DT learning approach that employs Monte Carlo tree search (MCTS) to heuristically explore the space of DTs. Experiments revealed that the algorithm improved the trade-off between performance and model complexity compared to locally-optimal learning. Moreover, dataset size and feature interactions played a role in the behavior of the method.

Despite being used for their explainability, DTs are chiefly evaluated based on prediction performance. The need for comparing the structure of DT models arises frequently in practice, and is usually dealt with by manually assessing a small number of models. We attempted to fill this gap by proposing an similarity measure to compare the structure of DTs. An evaluation of the proposed distance on a hierarchical forest of DTs indicates that it was able to capture structure similarity.

Overall, the reported algorithms take a step in the direction of improving the performance of DT algorithms, in particular in what concerns model complexity and a more useful evaluation of such models. The analyses help improve the understanding of several data properties on DT learning, and illustrate the potential role of DT learning as an asset for clinical research and decision-making.

Resumen

La minería de datos y el aprendizaje de patrones se encuentran cada vez más debajo de muchos aspectos de la vida cotidiana moderna. La preocupación creciente sobre el impacto de basarse en predicciones difíciles de explicar o comprender hace que haya un consenso amplio respecto a la necesidad de modelos interpretables y robustos. Una de las áreas donde esto es particularmente importante es en la toma de decisiones clínicas. Específicamente, los modelos interpretables tienen el potencial para facilitar la elaboración de guías clínicas y herramientas relacionadas de soporte a la decisión. La investigación que se presenta en este manuscrito se centra en la mejora del aprendizaje de los árboles de decisión ("Decision Trees", DT, en inglés), uno de los modelos interpretables más populares, motivada por los retos que ofrecen los datos clínicos.

Una de las limitaciones actuales de los algoritmos de DT interpretables es que implican decisiones basadas estrictamente en umbrales que pueden deteriorar la precisión en presencia de medidas con ruido. Al respecto, hemos propuesto un método probabilístico que considera un modelo de ruido en las distintas fases de aprendizaje. Al considerar este modelo en la fase de entrenamiento, el método demuestra mejoras moderadas en la precisión del algoritmo DT, comparado con el método clásico, aunque produce reducciones significativas en el número de hojas (e.g. niveles) del árbol de decisión.

Los algoritmos clásicos de DT siguen un enfoque óptimo a nivel local que, a pesar de proporcionar buenos resultados a un coste computacional bajo, no garantiza árboles de decisión óptimos. Así, la segunda dirección de la investigación en este doctorado se dirigió al desarrollo de una metodología de aprendizaje de árboles de decisión no voraz ("non-greedy" en inglés) que usa una búsqueda de árboles de Monte Carlo ("Monte Carlo Tree Search", MCDS en inglés) para explorar de manera heurística el espacio de DTs posibles. Los experimentos realizados revelaron que el algoritmo usando MCTS mejoró el balance entre la precisión en los resultados y la complejidad del modelo, comparado con el aprendizaje óptimo a nivel local. Asimismo, el tamaño de los datos y las interacciones entre las características tuvieron un rol importante en el comportamiento del método.

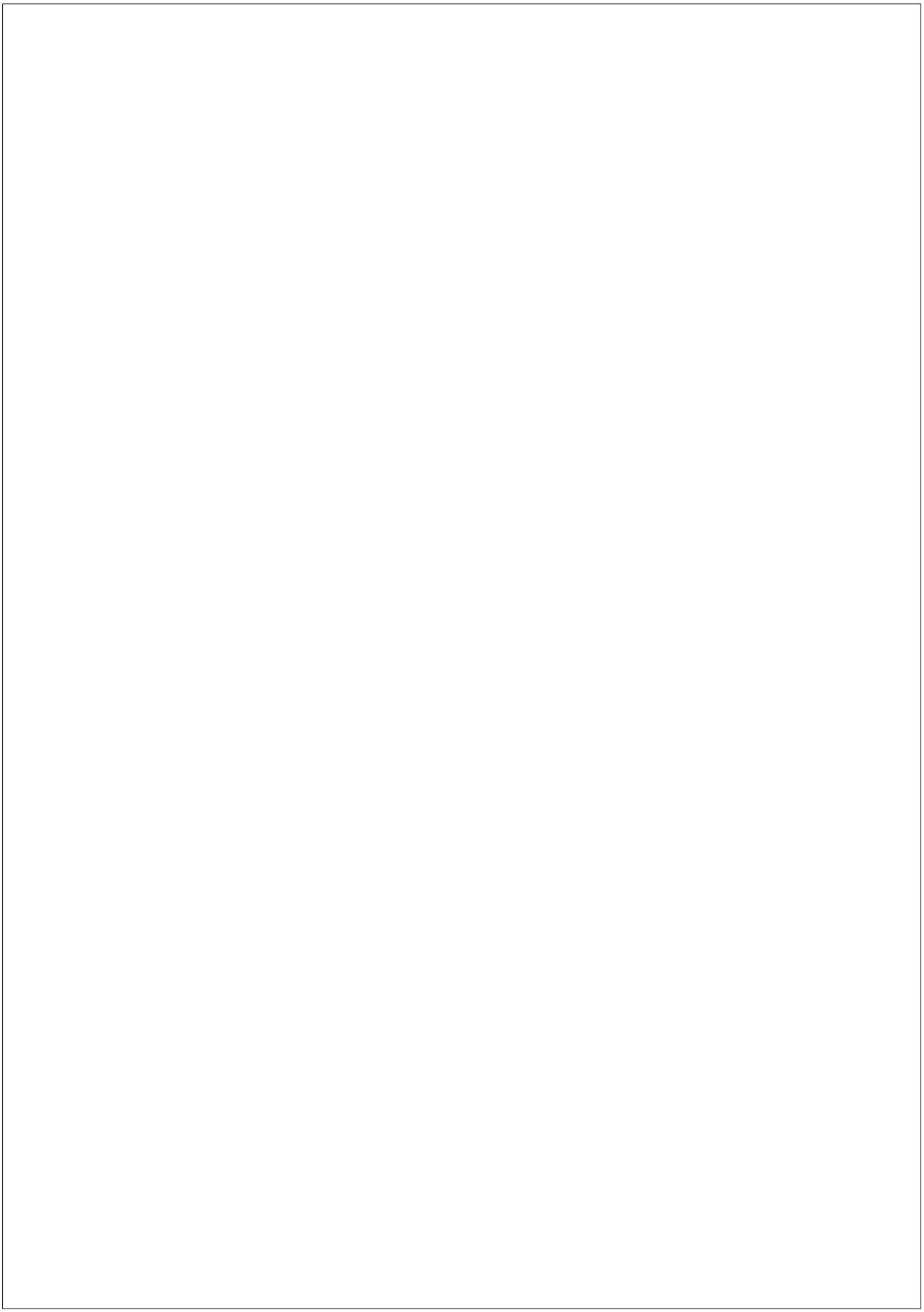
A pesar de emplearse por su explicabilidad, los árboles de decisión son principalmente evaluados con criterios basados en la predicción. La necesidad de poder comparar la estructura de diferentes modelos de DT es frecuente en la práctica y usualmente se trata evaluando manualmente un pequeño número de modelos. Durante esta tesis intentamos cubrir

esta necesidad proponiendo una medida de similitud para comparar la estructura de los árboles de decisión. Una evaluación basada en la medida propuesta aplicada a un bosque jerárquico de DTs indicó que era capaz de capturar la similitud estructural. De manera global, los algoritmos descritos dan un paso hacia la dirección de mejorar la precisión de los algoritmos basados en árboles de decisión, especialmente en lo concerniente a la reducción de la complejidad de los modelos y a una evaluación más práctica de ellos. Los análisis efectuados mejoran la comprensión de varias de las propiedades de los datos en el aprendizaje de DT, demostrando su rol potencial como recurso en la investigación y toma de decisiones clínicas.

Contents

List of Figures

List of Tables



Chapter 1

INTRODUCTION

Data mining is an approach to the extraction of information or predictions from data, using a combination of techniques from statistics, machine learning (ML), database and visualization technologies. It is a multidisciplinary field experiencing growth since the 1990s with the maturity of computing, network and storage technologies. Uncovering patterns in data involves a deep understanding of the problem to be addressed, designing the experiment or analysis to perform, gathering pertinent data, and choosing the learning method to employ. The latter is the object of ML, a field concerned with the development of models and algorithms to make computer systems execute tasks without being explicitly programmed.

Thanks to the improvements in data storage capacity and computing power, data mining and ML are increasingly at the core of many aspects of modern life. Machine-made decisions are now present in areas ranging from financial transactions, medical and legal scoring systems, investment decisions, credit and insurance qualification to scientific research, recommendation systems for marketing, political and cultural content, and even relationships. Since the first nationwide data-collection efforts in the United States Census of 1790, algorithmic treatment has become ubiquitous in many areas of human endeavor. The advent of information technology represents an evolutionary leap for a species whose survival depends on information sharing, and whose very cognition is now believed to have evolved as a result of argumentative communication [121].

1.1. The case for interpretability

ML systems have outperformed humans in several specific tasks, like object recognition in images, website enhancement, voice recognition, or playing Atari games. Despite allowing a more efficient use of resources

in numerous contexts, there are widespread concerns regarding the ubiquity of automated decision in our lives, and the associated exposure of our personal information. In particular, there is apprehension about relying on decisions that we cannot fully understand, made by the so-called *black box* algorithms.

The use of black box systems raises issues concerning the fairness, safety and trust associated with *opaque* decisions [78]. A controversial example related to fairness in machine decision is that of the *Correctional Offender Management Profiling for Alternative Sanctions* (COMPAS) score, a proprietary indicator of criminal recidivism, used by judges in the United States of America (USA). The score, whose inner logic is protected by corporate secret, has been recently found to be strongly biased against black American citizens, unfairly doubling their risk of re-offense compared to white defendants [2].

Another area that has sparked the discussion about the safety and liability of machine decisions is automated-driving systems. Such vehicles have been involved in several fatalities, including Tesla’s Autopilot system – three driver fatalities – and Uber’s Volvo refitted for autonomous driving – one pedestrian death. In the latter case, it was demonstrated that the reaction time of the vehicle was slower than a human’s [175], raising ethical concerns regarding driving liability and the distorted perception of control that is given to the vehicle by the driver.

Being able to understand the model increases users’ trust and ability to identify how the model fails. In 1995, a retrospective study led by the University of Pittsburgh aimed at predicting pneumonia mortality, in order to identify low-risk patients that could be safely treated as outpatients [36]. The team trained several algorithms, including a rule-base classifier and an artificial neural network. The latter was selected as the best-performing model. The rule-based classifier learned a rule that said that asthma translated into lower risk. When confronted with this, the clinicians explained that cases of pneumonia with a history of asthma were considered high risk. These patients therefore had a higher change of being admitted to the intensive care unit, effectively reducing their mortality. This selection bias was identified by the interpretable rule-based model, while the neural network could have meant great danger for asthma patients [26].

1.1.1. The legal incentive

With the growing concerns associated to a “black box society", policy makers have shown an increasingly firmer stance in regulating the use of personal data, and the accountability of data-handling organizations.

The *General Data Protection Regulation* (GDPR) is a 2016 initiative by the European Union to legally regulate the collection, storage and use of personal information [55]. Implemented in May 2018, the regulation was a replacement of the *Data Protection Directive* (DPD) [54]. Unlike regulation, directives are not directly enforceable as law in each member nation. In its turn, the DPD was developed in 1995, sedimenting the *Guidelines on the Protection of Privacy and Transborder Flows of Personal Data* first laid out in 1980 by the Organisation for Economic Cooperation and Development (OECD) [139]. This effort crafted the first international recommendations for the protection of personal data and the proclamation of privacy as a fundamental human right. The GDPR goes a step further, stapling those principles onto European law.

The “right to an explanation"

A novelty introduced by the GDPR is its stance on the automated processing of personal data, also known as *profiling*. The GDPR’s Article 22 (*Automated individual decision-making, including profiling*) introduces provisions for regulating machine-made decision. The main idea is that automatic decision-making is prohibited, unless consent is explicitly given or implied by contract. The article raises important issues for ML, in particular regarding what has elsewhere been called the “*right to explanation*". Concretely, Article 22(3) ensures that users have the right to demand human intervention in an automatic decision, as well as the right to contest that decision. Articles 13, 14 and 15 go one step further, stating that the data processor must provide “meaningful information about the logic involved". Despite the controversy regarding the implications of this statement, with some legal experts deeming the text imprecise [124], there is a widespread agreement on the urgent need for transparency and interpretability in automated decision-support systems. The legal incentive has thus been motivating the ML and data mining communities to prioritize explainability.

The need for transparency has also been addressed overseas. In the USA, the *Equal Credit Opportunity Act* is a law first enacted in 1974 to ensure that credit applicants are not unfairly discriminated against [52]. The Women’s Business Ownership Act of 1988 introduced amendments to enforce the right to a timely notification in the event of adverse credit action, such as a denied loan application, accompanied by a statement of *specific reasons* [9]. The law applies regardless of the method used to reach the decision. This is largely implemented through the long-established *reason code* system. Inspired by the GDPR, multiple American states

have started proposing new bills or amendments to existing law in order to increase consumers’ privacy rights, the *California Consumer Privacy Act* being a notable example [186].

1.2. Defining explainability

The requirement for “meaningful information about the logic involved" or “specific reasons" draws the questions: what defines a sufficient explanation, and what does it mean for a model to be interpretable? In this text, the terms interpretability, comprehensibility and explainability are used as synonyms to denote the extent to which a model is easily understandable by humans. In ML and data mining tasks, this often translates to providing some qualitative understanding of the relationship between the input variables and the property which is the object of study. However, each technique has its way of modeling problems and its own account of interpretability, resulting in a lack of consensus about such definitions [164]. Aspects that complicate the discussion include:

- The diversity of application contexts and requirements: The explanatory power needed for telling cats from dogs in a photo *app* is different from that needed when segmenting a liver in hepatic-surgery planning.
- The time available to evaluate the explanation: E.g. an emergency triage algorithm needs to be understood quickly, while a judge may have more time at her disposal to interpret a legal scoring algorithm.
- The level of expertise of the recipient of an explanation.
- The perspective of those building the model: We may be interested in finding explanations for practical applications – data-mining perspective – or prioritize the understanding of the inner workings of an algorithm – ML perspective.

1.2.1. An interpretable model?

Although different definitions existing, there is general agreement regarding properties that correlate with increased model interpretability:

- Well-defined *“cognitive chunks"* [46] or semantic units, i.e. demarcated concepts that can be used as intuitive units to explain a model. For example, rules in propositional learning, tests at the nodes of a

decision tree, prototype instances [10], the multiplication by a scalar parameter in a linear model, or the set of pixels that display a car in an image.

- Sequentiality: when the model/explanation provides a consecutive series of operations, instead of a complex combination of them.
- Monotonicity: models that express monotonic relations between the input and output variables, and/or whose operations favor monotonicity [148].
- Sparsity: it is generally perceived that a system with a smaller number of components is easier to understand, be it for example the number of input variables, the number of parameters of an optimization problem, or number of layers of a neural network.
- Low-dimensionality: this is the idea of sparsity applied to the number of variables used to model a concept. This is the idea behind dimensionality reduction technologies, like feature selection.
- Modularity: a model is easier to interpret if its elements can be independently understood, and their combination with other components is also understandable. E.g. rule-base learners can be decomposed into individual rules; generalized additive models without interaction terms are more modular than those with interactions, as they are decomposable into functions of the individual input variables.
- Adaptability: researchers have suggested that the ability to impose application-based constraints to the model, such as specifying a maximum model size, can improve model interpretability and usability [169, 179].

Regardless of the properties of a model, there is a recognized trade-off between its representational capacity and degree of explainability [73]. In other words, while more complex models may capture more intricate patterns and therefore have higher predictive performance [27], they are usually harder to understand. As a result, researchers have proposed approaches to optimize models both for predictive performance and interpretability. Studies that elaborate on this trade-off however tend to oversimplify the assessment of explainability by expressing it in terms of model size [61, 78]. While large models are unarguably impenetrable, model size is unable to express semantic information about the model. For example, a study evaluated the degree of comprehensibility of classification rules for detecting early signs of dementia by evaluating them with two trained

neurologists [147]. Although a smaller number of rules was correlated with increased acceptance by users, the coherence between the rules and background knowledge was a more important factor.

1.2.2. Interpretability as an end

The preoccupation with producing intelligible models is not new [157]. But given the recent increase in public interest and legal incentives, the past decade has seen a prolific body of research on the development of explainable systems [78]. As of May 2019, the query *"interpretability 'machine learning' "* in *Google scholar* yields nearly 65,300 results. The growing number of diverse contributions prompted efforts to bring consistency and formalism to otherwise-abstract notions of interpretability [61, 46, 78].

ML models are traditionally evaluated in terms of generalization performance [61], under the assumption that the ability to generalize translates into effective learning [157]. The 1990's, 2000's and 2010's saw the development of accurate algorithms like support-vector machines (SVMs) [15], ensemble learning [43] and deep artificial neural networks [109]. The interpretability wave has agitated this notion, with researchers defending the optimization of systems, not only for task performance, but also for properties like safety, fairness and reliability. Since these criteria are difficult to quantify, model interpretability arises as a common solution, allowing verification by humans using criteria of any nature. It also fosters the idea of humans-as-a-test-set, i.e. the validation of models with domain experts, deployed in the target application. Altogether, the need for different types of evaluation is nowadays widely recognized. While core ML work may be evaluated on in silico benchmarks, applied research ought to be evaluated directly on the end-task.

The adoption of interpretable decision-support systems requires enabling technology. Incidentally, research on interpretable models can be divided in two types of approaches. The first type consists in building an explanatory approximation of an existing model, which offers an accurate representation of its mechanism, while reducing its complexity. This idea arises from the excellent generalization ability of recent learning algorithms, such as deep learning architectures or gradient boosting [64]. These models tend to produce black-box predictions, encouraging research on making them explainable [183, 85, 194]. Algorithms for finding explanations often focus on expressing the contributions of each input feature to a given target property, e.g. through rule extraction based on the prediction model [117].

Alternatively, the second type of approach assumes that the properties

inherent to some families of models already satisfy certain interpretability needs, and focus on the improvement of the associated algorithms [78]. This is the approach of the research reported in this thesis. Examples of models recognized as interpretable include decision trees (DTs), rule-based learning [177, 192], decision tables [105], Bayesian networks, and generalized additive models, in particular linear models. How interpretable these approaches are depends on their sparsity. Naïve Bayes and linear-kernel SVMs are additional examples of models that can be made interpretable when used with suitable visualizations [126, 93].

The focus on interpretable models and on moving away from accuracy-oriented design sets the canvas for this thesis. The following section shifts the discussion to introduce the notions of clinical reasoning and decision-support systems in medicine. Along with interpretability, the potential for improvement of models used in clinical decision-support also serves as motivation for the work reported in this manuscript.

1.3. Towards evidence-based medicine

Clinical reasoning involves integrating scientific evidence, empirical knowledge, and information from multiple uncertain sources. Achieving a diagnosis and deciding a course of action requires weighing a complex set of risks, costs and benefits. This challenging process susceptible to variability, errors and suboptimal care [45]. Evidence-based medicine (EBM) was introduced as a response to these challenges, in an attempt to prevent errors, reduce decision variability and avoid cognitive biases.

EBM is the principled integration of scientific evidence in healthcare practice. Before the advent of EBM, there were general practice policies devised by committees of experts, but clinical decision was largely based on the experience of the physicians. The concept initially denoted an epidemiological approach to medical training, and later developed to describe the disclosure of the strength of the evidence supporting recommendations [50]. To facilitate an evidence-based practice, healthcare practitioners nowadays rely on multiple decision-support tools, some of which are introduced in the following pages.

1.3.1. Clinical guidelines and related decision-support tools

Clinical practice guidelines (CPGs) are official recommendations that help physicians make diagnostic and therapeutic decisions. They aim to facilitate clinical reasoning, promote consistent practices and help manage resources. They also provide an educational tool for doctors in training.

Since their proliferation throughout the 1980’s and the 1990’s, guidelines have been promoting EBM and have demonstrated significant improvements in care [77]. Despite their success and widespread adoption, assessments of guideline effectiveness are too few compared to the number of proposed manuscripts [103].

Guidelines are created by medical societies or government agencies. Their elaboration is a challenging process undertaken by appointed expert panels, sometimes composed by multiple working groups. Guideline development involves [34, 35, 76, 171]:

- A systematic review of the literature to gather evidence;
- Summarizing evidence from numerous (sometimes conflicting) sources;
- An explicit grading of evidence strength;
- Generating consensus through argumentation: the teams may have different views on many clinical issues. Their resolution tends to be a political discussion, dependent on the influence of the experts and interactions between working groups;
- Producing recommendations that generalize well based on results obtained from controlled populations;
- Expressing the recommendations in a clear and consistent format.

Medical societies continuously strive to face these challenges and improve guidelines. Although numerous methodological principles for issuing guidelines have been established, several surveys observed that guideline quality standards were largely unmet as of 2012 [35]. Guidelines sometimes have conflicting recommendations or insufficient quantification of evidence strength [76]. Some reviews have noted the need for systematic strategies for international comparisons [151], and more efficient procedures for monitoring and updating existing CPGs [69].

Regarding their format, CPGs are mainly presented in textual form, structured as sets of clinical situations for which evidence-based procedures are recommended. Each recommendation is accompanied by a classification of the strength of its supporting evidence. Guideline texts often contain graphical representations that summarize information and *decision algorithms* that target a specific diagnostic or therapy [103]. In addition, guidelines often advocate the use of *scoring systems*. Finally, the dissemination into practice of guideline recommendations, decision algorithms or scoring systems depends on their integration in medical information systems [57]. Incidentally, CPGs are increasingly made available

as computer programs and in phone *apps* [53]. Computer-generated reminders, such as critical-value alerts, or advice for drug prescription, stand out as effective aids for practitioner performance [70].

Medical decision algorithms Keffer highlights the distinction between guidelines and medical decision algorithms [103]. While guidelines aim to be comprehensive, medical algorithms are more restricted sets of instructions or recommendations that summarize information about a condition. They are constructed manually, targeting well-defined decision scenarios, like diagnosis, treatment selection or clinical workflow. Unlike guideline textbooks, decision algorithms are intended to be used as quick references. They focus on the central aspects of a problem e.g. they consider the most important predictors, or the most prevalent etiologies, and may therefore not contemplate all corner cases. Medical algorithms are often presented in a graphical formats such as tables, lists, flowcharts, or decision trees (DTs).

DTs are graphical decision tools, where each node displays a test on one of more predictor variables, and each leaf contains a prediction or recommendation. The operator starts evaluating the tree at the top, and follows a path according to the outcomes of each test. In a report about the medical laboratory test cycle, Wians describes the usefulness of tree-like recommendations in diagnoses that rely on laboratory tests [184]. According to the author, DT representations are ideal for clinical contexts, since they:

- are logical and sequential,
- can be learned automatically using existing algorithms,
- are interpretable and can therefore be understood by all medical personnel,
- can be easily updated, and
- can be integrated into clinical software.

Some of the work reported in this manuscript refers to decision-support for cardiology. As an example, Figure 1.1 displays a medical algorithm the management of severe aortic stenosis, expressed as a DT. The algorithm is taken from ESC guidelines for the management of valvular heart disease [6].

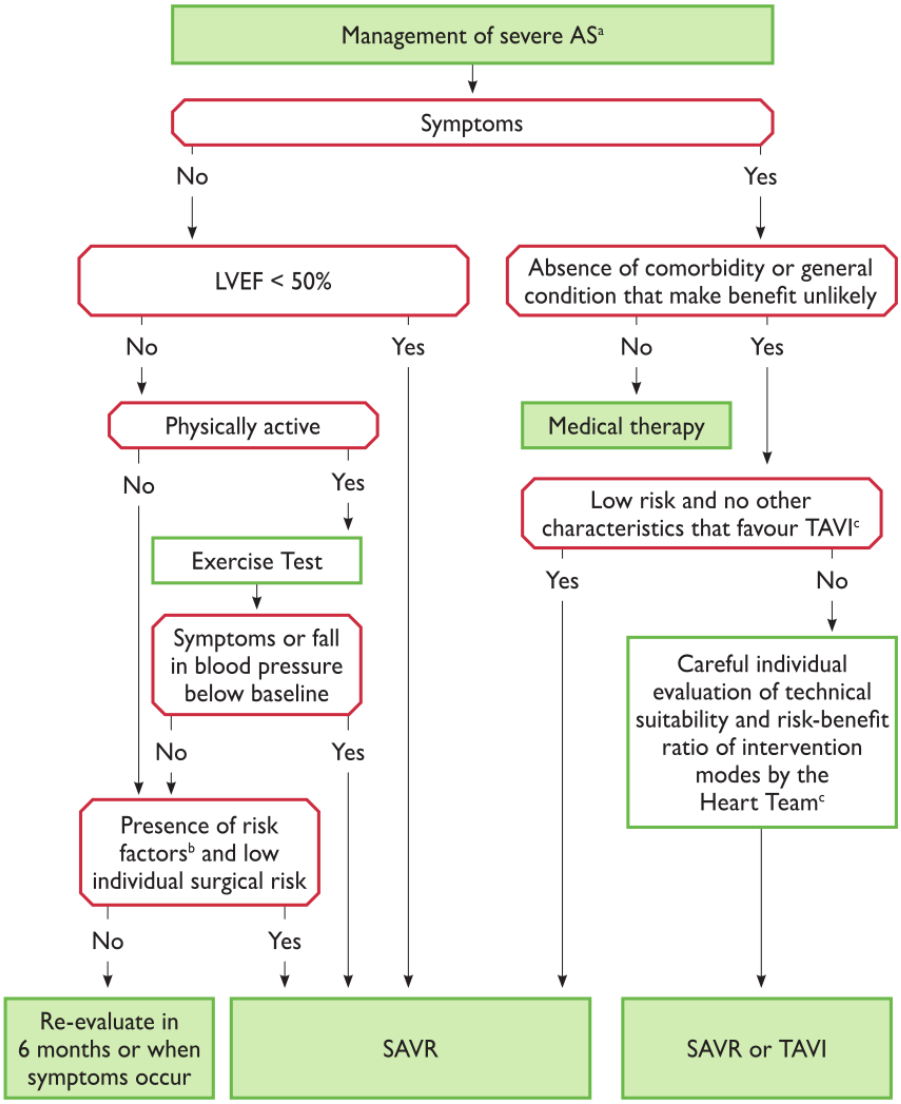


Figure 1.1: Algorithm for the management of severe aortic stenosis from 2017 ESC guidelines for the management of valvular heart disease [6]; AS – aortic stenosis, LVEF – left ventricular ejection fraction; SAVR – surgical aortic valve replacement; TAVI – transcatheter aortic valve implantation.

Medical scoring systems Scoring systems are classification or regression models that express the relationship between multiple input variables and a given aspect of importance for clinical decision. They are meant to provide information in a timely manner, using variables that are objective, generally obtainable and difficult to falsify [130]. Risk scores that estimate

the potential for adverse outcomes are often recommended in CPGs. As an example, the 2013 American College of Cardiology Foundation/American Heart Association (ACCF/AHA) guidelines for the management of heart failure recommend the use of validated scores to assess the risk of mortality in ambulatory or hospitalized heart failure patients. The text describes the usefulness of risk scores and biomarkers in guiding therapeutic decision in routine assessments, e.g. promoting a quicker transition to advanced care [188].

Some scores are based on qualitative observations about the patient, e.g. obtained through anamnesis or physical examination. For example, the New York Heart Association (NYHA) class is an ordinal measure that classifies the severity of heart failure based on symptoms related to physical activity, in particular shortness of breath and angina. In contrast, other scoring systems are often linear models of a set of selected variables. For instance, the EuroSCORE – European System for Cardiac Operative Risk Evaluation – assesses the risk of postoperative mortality of cardiac surgery in European populations. It is a widely-employed logistic regression model, first proposed in 1999 [130] and later updated in 2012 [131].

The elaboration of guidelines is often an argumentative endeavor towards the integration of evidence into practice recommendations with an intuitive format. Indeed, the evidence used to develop guidelines, decision algorithms and scoring systems is obtained through the application of statistics and data mining to clinical data. Health decision-support and guideline elaboration is therefore one of the areas where data mining can potentially bring benefit, in the sense that it possesses the methods to construct and evaluate interpretable models directly from data. Mining of clinical data and the related challenges are briefly introduced in the following section.

1.3.2. Data mining for clinical decision

Clinical data mining is the application of statistics and machine learning techniques to medical data, with the aim of (1) understanding the data, (2) assisting healthcare professionals, and (3) improving the methodologies for the analysis these data [91]. Mining of medical data has been an active research field since the 1970’s, in parallel with the development of ML and EBM. ML models have rivaled human performance in certain image-based diagnostic tasks, e.g. in gynecology, ophthalmology, dermatology, or radiology. Examples include the use of deep artificial neural networks to detect of diabetic retinopathy and macular edema in images of the retinal fundus [79], the identification of precancerous modifications in

images of the female cervix [89], and the detection of malignant melanoma in skin images [83]. Radiology is one of the specialties where computer-aided diagnosis has seen the greatest developments, with ML model being increasingly including in routine imaging software systems [180]. Nonetheless, the translation of automated decision-support tools into practice has been limited, with the technical achievements significantly outnumbering the cases that affect patient management [41, 32]. There are various factors contributing to this reality.

First of all, the development of a model depends on existing collaborations between the institutions providing the data and those performing the analysis, with all the challenges associated with multidisciplinary research. Furthermore, a proposed model needs to ensure sufficient predictive performance, which in the medical domain is usually low when compared to other fields [26]. This aspect is a requirement for acceptance within medical societies, along with model interpretability and coherence with existing knowledge [147]. Finally, the models need to be deployed in vendor software systems [142], and externally validated with the end-users at the point of care.

Challenges of working with clinical data Clinical or medical data are the records generated from clinical practice or controlled trials. Challenges that often arise when mining such data include missing values, measurement uncertainty, unbalance in the number records of each patient group, and limited dataset sizes. Clinical data acquisition, labeling and curation are time-consuming, expensive, and require highly-trained personnel. Therefore, medical datasets are often small when compared to other domains. Small datasets increase the chances of the sample failing to represent the underlying distribution. Data scarcity is also close related to the problem of uncertainty. The uncertainty in clinical measurements can be caused by observer-related variability [174], variations in manufacturer-dependent technologies [60], or individual patient factors [72]. Models constructed on a dataset with few examples are more likely to be affected by noisy measurements. Moreover, guideline recommendations are usually based on fixed thresholds, unable to account for the noise in the data. Another challenge is that medical data usually arise from multiple heterogeneous sources, leading to different types of variable formats [95]. This requires either extensive preprocessing or using models that handle different types of features. All these challenges can affect the performance of the learned models, needed to ensure acceptance by medical experts [166, 108]. Improving data quality is however resource-intensive and often unfeasible [99].

Another important aspect when mining medical data is understanding the assumptions carried by the data. Even if we succeed at training an accurate model, the correct interpretation and extrapolation of the results requires a complete understanding the context that originated the data. For example, retrospective studies are more prone to misguided conclusions compared to prospective studies, as e.g. in the pneumonia mortality study [36], mentioned in Section 1.1. Although we can find associations and correlations in data from retrospective studies, it is not possible to draw causal inferences. In conclusion, we add that the complexity of medical data can sometimes benefit from models that allow manual adaptations [91].

1.4. Decision tree models

Decision trees (DTs) are one of the most widely-employed tools in data mining and decision-support, owing to their interpretability [78]. A DT is a graphical diagram with a hierarchy of nodes, where each path leads to a terminal node containing a prediction. DTs are considered interpretable provided they have a relatively small number of nodes, and the input variables are themselves understandable by the intended user. Most commonly, each inner node contains a test on one of more input variables, which compares their observed value to a threshold. According to the output of the test, the decision is directed to one of the resulting branches. Compared to rule-based models, the hierarchical DT structure provides information about the relative importance of each feature [61]. Clinical guidelines often present decision algorithms in the form of DTs, increasing their potential for acceptance by the medical community and incorporation into clinical software [103]. Considering the growing interest on explainable ML, and that DTs are a recognized interpretable model with potential for clinical integration, the main focus of the research reported in this manuscript is on the improvement of methods to learn DTs.

1.4.1. Decision tree learning and associated challenges

An optimal DT is one that maximizes prediction performance, while minimizing the number of decisions needed to reach a prediction. Learning such a tree is computationally complex, owing to the discrete and sequential nature of the tests at the nodes [90]. Approximate solutions can however be built using the well-established top-down recursive heuristics [167]. As it happens, DTs learned through those procedures do not generalize as well as more recent approaches [27], such as gradient-boosted trees [64]

or deep neural networks [109]. However, the latter models tend to produce black-box predictions [183]. And despite attempts to make boosting and deep learning explainable, DTs remain one of the few general and efficient approaches to produce an intuitive visual output. The following paragraphs motivate the main directions improvement said DTs learning approaches, addressed through research here documented.

Perspective on learning One of the limitations of interpretable DTs and their learning algorithms is that they generally involve decisions based on strict thresholds. This may not suitable in medical contexts, where data are prone to uncertainty, and physiological ranges have variability. Learning DTs from noisy data can reduce prediction performance [160]. Indeed, several algorithms have explored the idea of *softening* split thresholds to make DTs robust to the uncertainty [48, 92, 160, 176, 190]. Existing solutions do not however provide a conclusive account of the benefit of such an approach, nor do they conduct an complete evaluation of the effect of modeling the uncertainty in the distinct stages of model development. As such, this text reports on research efforts towards acknowledgment of uncertainty in input measurement in DT learning and evaluation.

Although standard top-down DT algorithms gain on the side of efficiency, the locally-optimal strategy does not guarantee an optimal DT. Indeed, one of the main limitations of this learning approach is its propensity for overfitting [88], typically handled through pruning strategies. The greedy approach is also known to be sensitive to noise and irrelevant attributes in the training data [159], and to underperform in the presence of complex feature interactions [18]. Greedy is also highly dependent on the particular sample of training data [51], specially when dealing with few examples. Existing global optimization approaches are not practical, as they either impose a fixed DT structure [8], or use multivariate test functions, reducing the interpretability of the output trees [137]. Owing to their efficiency, locally-optimal DT algorithms are often employed as base learners in ensemble methods [18, 168]. Ensembles improve the performance and stability of a base learner, by optimizing the combined predictions of multiple models built with it. Tree ensembles such as random forests [18] or gradient boosting machines [64] are considered the state-of-the-art in many applications. It is however challenging to understand the predictions of an ensemble of multiple trees. The successes of tree-based ensembles and evolutionary DT learning further suggest the potential to improve upon greedy algorithms by performing a more extensive search. Ensuring sufficient predictive performance is an important requirement for the acceptance of by clinical experts. Improving upon locally-optimal learn-

ing therefore constitutes one of the directions of the work report in this manuscript.

Perspective on evaluation Another relevant aspect when learning DTs is how the algorithms cope with datasets arising from noisy environments or complex distributions. Quinlan [161] performed experiments with one dataset that indicated that when subject to small amounts of noise in the training data, the prediction performance of standard top-down induction suffers little. When corrupting the data with higher noise levels, the impact of the noise on the performance was more evident when corrupting the test data. Nonetheless, a conclusive evaluation of the effect of noise on DT performance in more learning problems has yet to be carried out. Owing to their hierarchical structure, DTs are able to express some degree of interaction between the features [51]. Some authors however argue that DT learning algorithms follow a discriminative approach, focusing only on the relation between target and inputs [86]. Another dataset property that can be challenging for DT learning is that of distributions where the target variable cannot be easily approximated by a function of a small number of features, i.e. datasets with many *weak* inputs [17].

Despite being used for their interpretability, the merit of a DT model is almost always evaluated by measures of prediction performance. The assessment of interpretability is often oversimplified to measures of model size, including number of nodes, number of used attributes, or branch depth [167, 78]. These metrics fail to capture the decision structure expressed by the model [61]. Nonetheless, the task of qualitatively comparing DTs arises frequently in practical scenarios, and is usually dealt with by manually inspecting a small number of models. A third direction of work therefore consists in proposing a way of evaluating DTs that takes into consideration their structure.

1.5. Objectives

In light of the above considerations, the main objective of this thesis is to improve algorithms for learning interpretable DTs, with the aim of assessing and increasing their value as a tool for clinical research and decision-support. This work is motivated by some of aforementioned challenges posed when building models with clinical databases. Specifically, it aims to:

- Develop a DT learning approach that takes into account the uncertainty in the data, under the hypothesis that this can improve prediction performance, while safeguarding interpretability.

- Clarify the distinct impact of having noise in the data used for learning DTs or in the data used for generating predictions.
- Assess the merit of alternative search approaches in improving upon locally-optimal DT learning, specifically Monte Carlo tree search (MCTS) algorithms, which have demonstrated success in identical combinatorial search problems.
- Characterize the relationship between the performance of DT learning algorithms and certain dataset properties, including the uncertainty in the input measurements, dataset size and interactions between the features.
- Promote the evaluation of DT models based on their decision structure and coherence with existing knowledge, steering away from accuracy-only model assessments.
- Investigate the value of DT learning as a tool for clinical research and decision analysis in a concrete clinical scenario.

Chapter 2

LEARNING DECISION TREES

In graph theory, a tree is an undirected graph in which every two vertices are connected by exactly one path, as the example of Figure 2.1. Specifically, a decision tree (DT) is a rooted directed tree, where the inner vertex labels express a query/action, and the terminal vertices provide an insight. The hierarchical structure captures the relations between the elements of a problem in a visually intuitive manner. For this reason, DTs are employed in a variety of domains, in research contexts or commercial systems [187]. They are an acknowledged methodological asset in decision analysis and data science. The research described in this manuscript deals with the latter context. The present chapter focuses on how to learn such trees from data. Before doing so, let us revisit some key concepts in machine learning (ML).

2.1. Machine learning concepts

As introduced in Chapter 1, ML is concerned with the development of models and algorithms to make computers perform tasks without being explicitly programmed. More generally, it can be defined as the systematic study of systems that extract knowledge from experience [58]. Knowledge is expressed in a model that makes some assumptions about the problem under study, while experience is gathered in the form of data.

More concretely, a random variable X can be informally defined as a function whose values depend on the outcome of a random event. A phenomenon can be described by a set of random variables $\mathbf{X} = (X_1, ..., X_M)^T$ with joint distribution $P_{\mathbf{X}}$. Sampling this distribution composes a dataset, and we will consider settings where the samples are assumed to be independent and identically distributed. Often, each sample $\mathbf{x}$ is associated with a particular concept modeled by a target variable Y, in which case

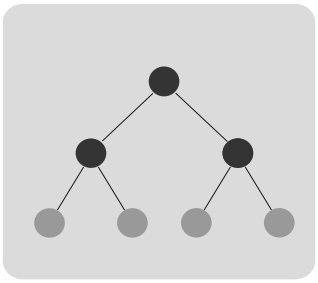


Figure 2.1: Abstract representation of a decision tree.

the joint distribution becomes $P_{\mathbf{X}Y}$. ML models, or the people who employ them, are directly or indirectly interested in learning more about this distribution. Some models have a descriptive purpose and therefore aim to characterize the relations between the variables. On the other hand, predictive models focus on how to make predictions about the target concept, as is the case of DTs. In Bayesian terms, this is related to distinction between generative and discriminative models. While the former attempt to learn the joint distribution $P_{\mathbf{X}Y}$, discriminative models focus on the conditional distribution of the target variable given the input, $P_{Y|\mathbf{X}}$.

Types of models A related distinction is the one made between unsupervised and supervised learning. Unsupervised-learning methods target unlabeled data, i.e. data where there is no target variable, or its value is unknown. They typically focus on finding patterns in the data based the similarity between the samples, for example through dimensionality reduction or clustering. In contrast, when the data is labeled, we talk about supervised learning. In its turn, the target variable Y can be continuous or categorical. Regression models map the input domain to a continuous variable, while classification models aim at identifying a correct category. Reinforcement learning is another setting where an agent aims to learn the best sequence of actions in response to a reward, obtained from interacting with an unknown environment. DT variants have been employed in different tasks, including clustering [113], regression [19], classification [19, 100, 159], probability estimation [153], or reinforcement learning [155]. We focus on DTs for classification.

Peter Flach [58] further describes models in three non-mutually-exclusive categories: geometric, probabilistic and logic. Geometric models construct a mathematical description based on the geometry of the input space, e.g. a separating hyperplane. Probabilistic models make explicit assumptions and inferences about the underlying distributions, $P_{\mathbf{X}Y}$ or $P_{Y|\mathbf{X}}$. Logical

models are those which can be decomposed into logical rules that are understandable by humans. They are algorithmic in nature, as they draw inspiration from computer science and logic [44]. DTs are a type of a logical model. On a different account, Duda et al. [47] characterize DTs as non-metric methods, since they do not rely on a measure of similarity between the samples.

Training and evaluation The chosen model is then trained using an appropriate learning algorithm, which usually amounts to determining the parameters of the model. In addition, model training and selection usually require tuning a number of hyperparameters that can affect the learning algorithm and the output model. Training can be done directly using the variables $\mathbf{X}$ or features/attributes extracted from them.

After the model is trained, it is crucial to know what to expect from it. Generalization is the ability of a model to represent knowledge that carries over to unseen cases, despite having been acquired from a sample. It is assessed by measuring how well the model extrapolates to data which was not used during training. The extent to which a model can express more complex patterns is known as model capacity. The appropriateness of a model and its capacity depend on (1) the underlying data distribution, (2) the representativeness of the sample, and (3) the resources available to handle the data and train the model. When the model fails to capture structure in the data because its capacity is too low, we say that it underfits the data. In contrast, the model may memorize the variability associated to the sampling process rather than the phenomenon of interest, which is known as overfitting.

Naturally, choosing a criterion to evaluate the performance of a model depends on the objective of the problem. Regardless of the type of algorithm, experiment or evaluation criterion, the data used to estimate the performance must be carefully isolated from any model learning and selection tasks. For example, the samples are often divided between a training dataset and a test dataset. If a model performs well on a large-enough number of test samples, and the generalization performance is close to the training performance, we can then conclude that the model has a good chance of generalizing well.

2.2. Decision tree learning

Categorized as logical models, DTs express a hierarchy of logical operations. From a probabilistic perspective, a DT attempts to represent the

distribution $P_{Y|\mathbf{X}}$. Each internal node (Figure 2.1 - black circles) prompts a test on $\mathbf{X}$ which divides the domain among the resulting child nodes, conditioning the probability of Y on the outcome of the test. A sequence of tests along a branch directs the user to a terminal node or leaf (Figure 2.1 - gray circles), which provides a prediction about Y. Usually, the prediction is achieved by selecting the most prevalent class in the leaf node. But given the empirical estimate of $P_{Y|\mathbf{X}}$ provided at each node, any classification threshold can be chosen e.g. through a receiver operating characteristic (ROC) analysis. The hierarchical structure and probabilistic understanding confer a good degree of interpretability, which is the main reason for the popularity of DT models.

An optimal DT is one which maximizes generalization performance while having minimum complexity. The complexity of a DT is hard to measure and is associated with the difficulty in achieving a prediction and interpreting the complete tree. It is usually estimated by the number of nodes or branch depth [167]. DT learning algorithms aim to obtain such a DT for a given dataset. Hyafil and Rivest have however showed that constructing an optimal binary tree is NP-complete [90], where optimality had been defined as minimizing the number of tests required to classify a dataset, and a binary DT is one where each internal node has two child nodes. Even the related problem of finding an equivalent DT of minimal size is NP-complete, where two trees are considered equivalent if they produce the same result for all inputs [191].

Most approaches propose heuristics to tackle the complexity of the DT learning problem. Specifically, most algorithm variants follow a locally-optimal top-down recursive strategy [84, 167]. Although the greedy nature of the approach does not guarantee optimality, it offers a good trade-off between computational demands and prediction performance, and is widely used in recent applications. The strategy is also conceptually simple, conveying transparency to the learning algorithm itself. Additionally, greedy methods have few requirements on the format and distribution of the input variables, which makes for easier data preparation compared to other approaches. For example, the input features can be numerical or categorical and do not need to be normalized. Most DT algorithms also include mechanisms to handle missing data. The following section describes this family of algorithms.

2.2.1. Locally-optimal decision tree learning

Locally-optimal DT algorithms are composed by two main stages: *induction* and *pruning* [84, 167].

- Induction consists in "growing" the tree from top to bottom by choosing the best test for each internal node. We start at the root of the DT and consider the complete training dataset, D. Locally-optimal induction selects the test function $t(\mathbf{x})$ that maximizes an objective, also known as *split criterion*, for those local data [167]. The node is labeled with $t(\mathbf{x})$, which specifies a separating hyperplane in the input space, and divides the training data between the resulting child nodes. Induction proceeds recursively in the newly-generated nodes. Each new node is first evaluated with respect to a *stop criterion* to determine if it should become a terminal node, or if learning should proceed. In the latter case, the algorithm goes on to select the test functions which maximize the split criterion for the local data at the new nodes. This strategy is summarized in Algorithm 1.

- Pruning corresponds to the set of operations that remove unpromising nodes, when such an action is expected to improve generalization performance. Pruning controls the complexity of the DT, playing a crucial role in preventing overfitting.

One of the first DT recursive partitioning algorithms was developed by Quinlan during the early 1980s and was known as ID3 [158]. The approach used binned continuous variables and an entropy-based split criterion. Around the same time, another group independently proposed the classification and regression trees (CART) approach [19]. Both ID3 and CART construct binary trees, but the latter uses an impurity-based split selection criterion and performs pruning. The χ^2-square automatic interaction detector (CHAID) [100] and the quick unbiased efficient statistical tree (QUEST) approaches build DTs using both continuous and categorical variables. While the former employs no pruning, QUEST uses cross validation toward this aim. They belong to a family of DT approaches stemming from the automatic interaction detector (AID) [125], which employ statistical tests to select the test functions. The ID3 algorithm was later superseded by the C4.5 algorithm [159], which improved the former by supporting both continuous and categorical variables, handling missing values, and employing a better pruning procedure. Many variants and extensions have since been proposed based on these foundational algorithms [84, 167, 88].

None of the aforementioned approaches has been found to clearly dominate another [47]. Comparisons [122, 24] usually target different approaches for split selection, early stopping, or pruning. The split criteria employed in C4.5 have been found to perform acceptably in most cases [47]. The research reported in this manuscript mostly builds upon C4.5.

Algorithm 1 General algorithm for top-down recursive induction of decision trees. The method employs a *split criterion* to locally select the test for each node, and a *stop criterion* to determine if a new node should be terminal.

Input: dataset $\mathcal{D}$ with variables $X_j, j = 1, ..., M$ and class Y
Output: a univariate decision tree that predicts Y

function BUILDDECISIONTREE($\mathcal{D}$)
  Create node n with the dataset $\mathcal{D}$
  if $\mathcal{D}$ complies with stop criteria **then**
    return Leaf node n with class probability estimated from $\mathcal{D}$
  end if
  Find best test $t(\mathbf{x})$ according to split criterion
  Label the n with $t(\mathbf{x})$
  Partition $\mathcal{D}$ in subsets $\mathcal{D}_b$, $b = 1, 2, ...$ according to the outcomes of $t(\mathbf{x})$.
  for each $b = 1, 2, ...$ **do**
    Let n_b be a new child node of n
    $n_b \leftarrow$ BUILDDECISIONTREE($\mathcal{D}_b$)
  end for
  return n
end function

2.2.2. Split search and selection

We now explain how top-down induction methods go about selecting a test for each internal node. Again, consider the input $\mathbf{X}$ with dimensions X_j, $j = 1, ..., M$, and the output Y. Drawing samples from $P_{\mathbf{X}Y}$ composes the training dataset D. During induction, suppose that each new node n sees the subset $D_n \subset D$. The question is how to determine the boolean function $t(\mathbf{x})$ that best discriminates the instances in D_n according to Y.

Test functions

Univariate DTs are those in which the test function involves a single input variable X_j. Let $[\tau, \infty)$ denote the real interval of values greater than or equal to τ. If X_j is continuous, the function usually takes the form:

$$t(\mathbf{x}) = \mathbf{1}_{[\tau,\infty)}(x_j), \tag{2.1}$$

where τ is the cutoff value, and $\mathbf{1}_A(x)$ is the indicator function:

$$\mathbf{1}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise.} \end{cases}$$

Algorithmically, if the outcome of the test is 0 the instance $\mathbf{x}$ is passed to the left child node, and to the right child node otherwise. DTs with this type of functions split the input domain according to hyperplanes that are orthogonal to the input dimensions.

On the contrary, multivariate trees are those whose functions may depend on several features. Different types of test functions have been proposed in this context, including linear [19, 25, 115, 129], linear logistic [67, 92], quadratic [67], or as a multilayer perceptron [80]. Yıldız and Alpaydın [189] summarize these approaches and the associated learning algorithms. Multivariate DTs are generally able to represent complex patterns more compactly than univariate ones. As a consequence, they often achieve better performances with fewer nodes. However, the actual nodes may be difficult to interpret as sequences of logical operations [21]. For this reason, we favor the use of univariate tests and make the dependence on a variable X_j explicit by writing $t(\mathbf{x}) = t(x_j)$.

There are several strategies for making univariate splits on categorical variables. Some approaches, like CART, employ binary tests by composing two subsets of the categories, S_L and S_R, for the left and right child nodes of n. For example, if a variable takes values $\{red,\ blue,\ green\}$, examples of split subsets include $S_L = \{red\}$ and $S_R = \{blue,\ green\}$, or $S_L = \{red,\ blue\}$ and $S_R = \{green\}$, and so on. The test function becomes $t_n(\mathbf{x}) = \mathbf{1}_{S_L}(x_{j_n})$. Other methods, like C4.5 or CHAID, consider n-ary splits, where each partition corresponds a distinct value of the categorical variable. Although n-ary splits can provide a more succinct representation for categorical variables, recursive partitioning methods are known to be biased towards splits with many possible splits [47, 88, 159]. We employ the binary-splitting approach of CART for categorical values.

Entropy-based split criteria

The entropy of variable Y is computed as:

$$H_\alpha(Y) = -\sum_y p(y) log_\alpha p(y),$$

and represents the uncertainty about Y, or the number of questions we expect to ask in order to find the value of Y. Taking $\alpha = 2$ means that the entropy is measured in *bit* and can be interpreted as the expected number of binary questions needed to learn the value of Y.

One possible approach is to select the function $t(\mathbf{x})$ that removes the most uncertainty about Y. This corresponds to choosing the variable X_j and cutoff value τ which maximize the mutual information between $t(\mathbf{x})$ and the class Y:

$$j_n, \tau_n = \arg\max_{j,\tau} I\big(t(\mathbf{x}; j, \tau); Y\big), \tag{2.2}$$

where the subscript indices refer to node n. This is the split criterion used in ID3, where $I\big(t(\mathbf{x}); Y\big)$ is known as *information gain*. $I\big(t(\mathbf{x}); Y\big)$ is equal to the difference between the entropy of Y in D_n and the entropy of Y in the resulting nodes:

$$I\big(t(\mathbf{x}); Y\big) = H\big(Y\big) - H\big(Y|t(\mathbf{x})\big). \tag{2.3}$$

The term $H\big(Y|t(\mathbf{x})\big)$ is the conditional entropy of Y given the result of $t(\mathbf{x})$. Equation 2.2 is equivalent to:

$$\begin{aligned} j_n, \tau_n &= \arg\min_{j,\tau} H\big(Y|t(\mathbf{x})\big) \\ &= \arg\min_{j,\tau} \sum_{b \in \{0,1\}} p\big(t(\mathbf{x}) = b\big) H(Y|t(\mathbf{x}) = b), \end{aligned} \tag{2.4}$$

where

$$H(Y|t(\mathbf{x}) = b) = -\sum_{y} p(y|b) log\big(p(y|b)\big). \tag{2.5}$$

with the sum taken over the possible values of Y. The probabilities in Equations 2.4 and 2.5 are estimated empirically from the training data as:

$$\hat{p}(b) = \frac{N_n(b)}{N_n} \text{ and } \hat{p}(y|b) = \frac{N_n(y,b)}{N_n(b)}, \tag{2.6}$$

where N_n is the number of training instances in node n, $N_n = |D_n|$, $N_n(b)$ is the number of instances in n associated to outcome $t(\mathbf{x}) = b$, and $N_n(y, b)$ is the number of such cases that belong to class y.

ID3 uses test functions as defined in Equation 2.1, employing the information gain as a split criterion. C4.5 [159] extends ID3 to categorical variables by allowing n-ary splits, where each resulting branch corresponds to a value of the categorical attribute. Since an entropy measured in *bits* is not suitable for n-ary splits, the information gain criterion is biased towards splits with many child nodes [84, 159]. The C4.5 algorithm attempts to overcome this bias by dividing the conditional entropy $H\big(Y|t(\mathbf{x})\big)$ by the

information associated with the split itself, in a quantity known as the *gain ratio*:

$$\frac{H\big(Y|t(\mathbf{x})\big)}{H\big(t(\mathbf{x})\big)}.$$

Although the gain ratio may compensate for the bias to some extent, performing n-ary splits has the additional drawback of reducing the size of the data subsets in the subsequent nodes very quickly [65]. Moreover, it is also believed to be biased towards splits with very small entropy, i.e. where one of the resulting nodes has most of the samples [38]. In the latest release of C4.5 [163], the best split for each numerical attribute is first selected using information gain. Subsequently, the gain ratio is used to compare the best splits found for all variables, categorical and numerical. The algorithms proposed in the remaining of this manuscript simplify this approach and avoid the bias by employing binary test functions for all variables and using the information gain as a split criterion. Tests on categorical variables are therefore done as in CART, by computing the information gain for all 2-combinations of the values of categorical attributes. In all other regards, the proposed approaches follow the C4.5 algorithm. Top-down recursive induction of binary trees has a worst-case computational complexity of $M|D|\log_2|D|$, where the worst case corresponds to all variables being numerical, and with $|D|-1$ candidate test functions per variable.

Other split criteria

The split criterion employed by the CART approach is identical to the information gain. However, instead of measuring the impurity of the class Y at node n using the entropy, the method employs the *Gini index*:

$$G(n) = 1 - \sum_y p(y)^2.$$

Similar to Equation 2.3, the quantity that is now maximized is the average decrease in Gini index achieved by splitting according to $t(\mathbf{x})$:

$$G(n) - \sum_{b\in\{0,1\}} p\big(t(\mathbf{x})=b\big)G(n_b),$$

where n_b is the node corresponding to outcome $t(\mathbf{x})=b$.

The CHAID and QUEST algorithms employ statistical tests as split criteria, specifically the F-statistic and the χ^2 test of independence for continuous and categorical variables, respectively. CHAID may perform n-ary splits for all types of variables, while QUEST builds binary trees.

Empirical comparisons between various univariate split selection criteria, including the information-based criteria, the Gini index or the statistical measures, have concluded that the choice of split criterion does not significantly affect the accuracy of the output DT [122, 24, 114]. Each criterion is superior in some cases and inferior in others [84, 167]. Several researchers add that the split criterion is not as crucial for the prediction performance as the choice of stopping criteria and pruning procedures [19, 47].

2.2.3. Pruning

Top-down DT induction is prone to overfitting, reflecting noise and sampling deficiencies in the training data. As introduced above, pruning, or tree simplification, refers to the set of procedures that remove nodes and subtrees from a DT, when the generalization error would otherwise be higher. By controlling the size and complexity of the trees, pruning methods contribute to their generalizability and, arguably, their interpretability [47, 88]. Prepruning, also known as early stopping, corresponds to halting the induction of certain DT branch. In contrast, postpruning removes nodes from a finalized tree after its induction is terminated.

Prepruning

Prepruning can be implemented through the specification of stopping criteria. Common stop criteria include imposing a minimum number of training instances required for a split [159], specifying a maximum branch depth [149], or setting a minimum number of instances for the resulting child nodes. Another approach consists in stopping the induction when the improvement of the split selection measure is below a given threshold [161]. Prepruning is typically hard to optimize [84, 19]. It also explores less tree structures than postpruning, specially if subtree raising is used [159]. Consequently, inducer algorithms tend to combine loose stop criteria with a postpruning phase [167].

Postpruning

Since the test functions are computed from the training data, the idea behind most postpruning approaches is to remove nodes or subtrees according to an estimate of the generalization performance and/or tree complexity. Node and subtrees may be converted into leaves, or replaced by a

sub-branch. Specifically, there are two main families of methods to obtain such estimates of generalization performance: pruning based on an independent dataset and pessimistic pruning [159].

As the name suggests, the first type of postpruning involves the use of an independent data sample, unused to learn the split functions, for computing the error estimated or other pruning metric. Examples include *cost-complexity pruning* [19], which generates several possible DTs by cutting the branches with the worse trade-off between the increase in training errors per removed leaves. The best pruned tree is then selected based on the performance on the independent validation set. *Reduced-error pruning* [156] traverses the DT down-top and removes branches which lead to increased error an the independent dataset. As a shortcoming, these techniques require an independent pruning set, decreasing the data available for induction, which can deteriorate the performance of the DTs [20, 159].

The second type of postpruning attempts to obtain realistic generalization errors based only on the training data through the use of pessimistic estimates of predictive performance. The method employed in ID3 adjusts the training-set based error estimates by imposing a continuity correction, assuming the probability of an error at each node follows a binomial distribution. Moreover, nodes whose adjusted error is at least one standard error smaller than the parent node's are not pruned. The following paragraphs explain the even more pessimistic approach delivered by C4.5, sometimes referred to as error-based pruning (EBP), which is superior to the former [20]. Both pessimistic strategies were found to lead to the most accurate DTs, when empirically compared to other methods [20].

Pruning in C4.5

As a type of pessimistic pruning, the method employed in C4.5 estimates the number of errors incurred by the test functions when generalized to unseen data. In other words, this involves comparing the number of expected errors of each node to its child nodes, and deciding on whether to preserve the split, based only on the training data.

After induction has finished, a given leaf ℓ of a fully grown DT provides an empirical estimate of the probability of a Y for a given instance $\mathbf{x}$, $p(y|\mathbf{x}, \ell)$. The instance gets classified with the most prevalent class in the leaf, i.e. the predicted class is the one which maximizes $p(y|\mathbf{x}, \ell)$. For the remaining of this section, let us simplify the notation and refer to the $p(y|\mathbf{x}, \ell)$ as $p(y)$.

Consider that the training data at a node has majority class y and

therefore each instance misclassification follows a Bernoulli distribution with probability $p(\neg y)$. To estimate the number of generalization errors in ℓ, C4.5 takes a pessimistic estimate $\hat{p}$ of the misclassification probability $p(\neg y)$. We now show how C4.5 computes such an estimate based on the training data.

During training, the leaf sees the subset D_ℓ with N instances, $N(y)$ of which belong to class y. Let us assume that the number of training errors $N(\neg y)$ follows a binomial distribution, $N(\neg y) \sim B\big(N, p(\neg y)\big)$. One approach to obtaining a pessimistic error rate is to first compute the confidence interval of the estimate of $p(\neg y)$, (p_L, p_U), under a specified confidence level. Then, $\hat{p}$ is obtained by taking the upper-limit of the confidence interval: $\hat{p} = p_U$. In C4.5, a two-tailed confidence level is set by specifying the *pruning confidence factor* c, such that the probability that $p(\neg y) > \hat{p}$ is smaller than $\frac{c}{2}$. The Clopper-Pearson method is used to compute the binomial confidence limits [33].

The number of expected errors is finally computed by multiplying the number of instances by the pessimistic error estimate, $\hat{N}(\neg y) = N \times \hat{p}$. The number of expected errors is calculated for three scenarios:

1. Maintaining the test function at the node, and therefore keeping the child branches.

2. Removing the test at the node, and replacing it by a leaf.

3. Removing the test at the node and replacing it by the sub-branch with fewest predicted errors.

and the smallest-error scenario is selected. The process starts at the bottom-most left-most node, and traverses the tree upwards from child to parent and from left-to-right. If the factor c is small, the confidence interval will be wider, resulting in a higher upper-limit p_U. As a consequence, the estimate $\hat{p}$ will be more pessimistic for leaves with less instances, and the tree will be more aggressively pruned. On the other contrary, if the confidence factor c is large, $\hat{p}$ will be relatively smaller for nodes with few instances in comparison with to the parent node, therefore resulting in a more conservatively pruned DT.

Postpruning methods can be combined with probability smoothing [135], which is known to improve several pruning methods by favoring the exploration of smaller trees without compromising accuracy [16]. The algorithms proposed in the remaining of this manuscript use the pruning approach of C4.5 enhanced with the Laplace correction. To employ the correction, the number of errors $N(\neg y)$ used in the computation of the confidence interval

(p_L, p_U) is replaced by:

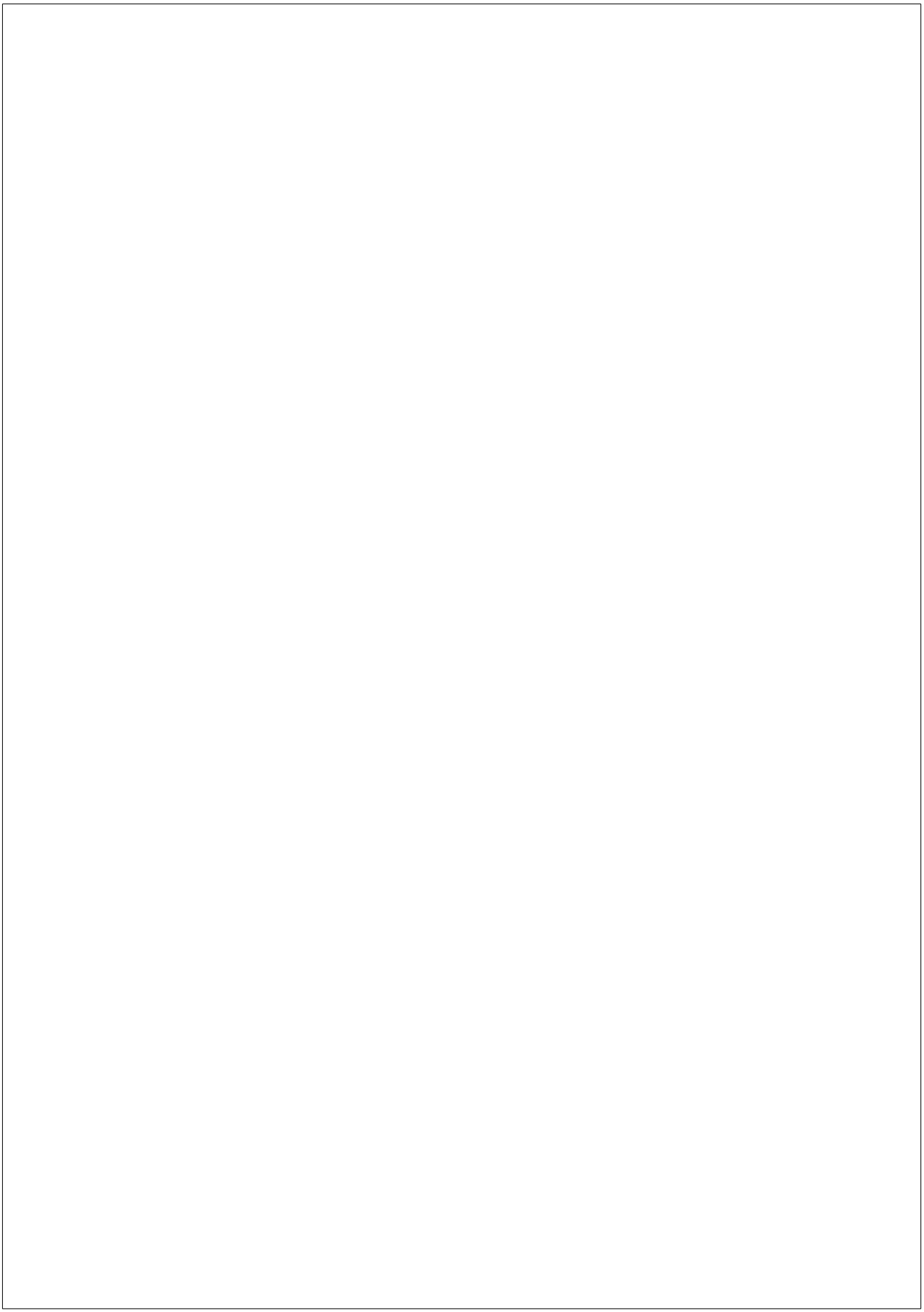
$$\frac{N\big(N(\neg y)+1\big)}{N+C}$$

where C is the number of classes.

2.2.4. Handling missing values

Missing data affects both learning and the application of DT models. Some algorithms ignore the missing cases during induction, which can reduce the significance of the test functions [63]. Other approaches, like CART, propose the use of surrogate splits for variables with unknown values when evaluating an instance through the tree [19]. The idea is to use an alternative test function baring resemblance to the original one, but for which the attribute value is known.

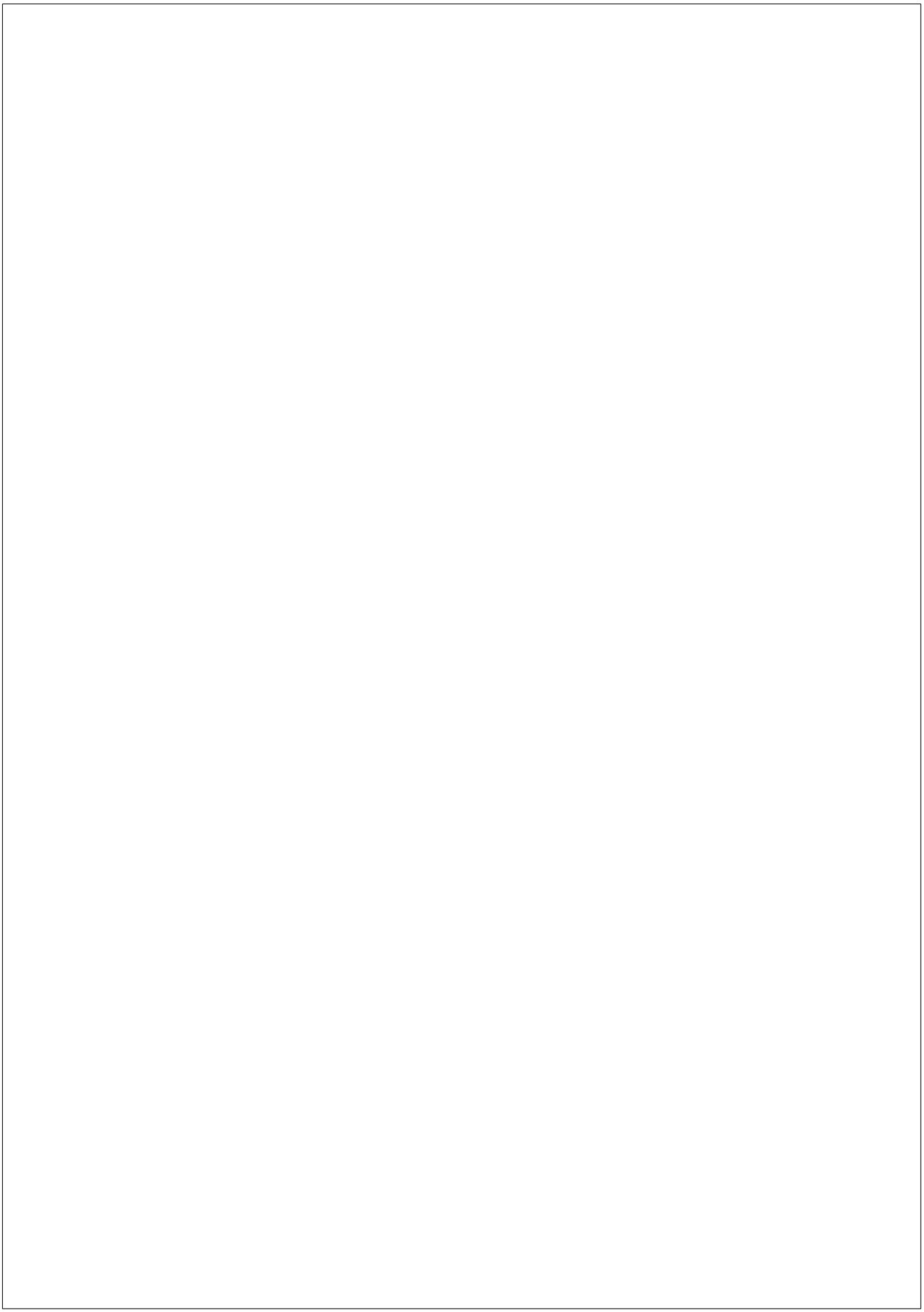
In C4.5, missing values are dealt with in two ways. During induction, the split selection criterion is multiplied by the proportion of known-value instances in the node, making variables with a lot of missing cases proportionally less likely to become a threshold. Secondly, when instances traverse the DT, either during training of evaluation, a soft split on the missing variable is done. In other words, after encountering the test with unknown value, the instances get fractionally split between the resulting nodes, according to the probabilities estimated from the known-value training instances.



Chapter 3

DECISION TREE LEARNING FOR UNCERTAIN MEASUREMENTS

Clinical decision requires reasoning in the presence of imperfect data. Decision trees (DTs) are a well-known decision support tool, owing to their interpretability, fundamental in safety-critical contexts such as medical diagnosis. However, learning DTs from uncertain data leads to poor generalization, and generating predictions for uncertain data hinders prediction accuracy. Several methods have suggested the potential of probabilistic decisions at the internal nodes in making DTs robust to uncertainty. Some approaches only employ probabilistic thresholds during evaluation. Others also consider the uncertainty in the learning phase, at the expense of increased computational complexity or reduced interpretability. The existing methods have not clarified the merit of a probabilistic approach in the distinct phases of DT learning, nor when the uncertainty is present in the training or the test data. We present a probabilistic DT approach that models measurement uncertainty as a noise distribution, independently realized: (1) when searching for the split thresholds, (2) when splitting the training instances, and (3) when generating predictions for unseen data. The soft training approaches (1, 2) achieved a regularizing effect, leading to significant reductions in DT size, while maintaining accuracy, for increased noise. Soft evaluation (3) showed no benefit in handling noise.



> It’s easy to make perfect decisions with perfect information. Medicine asks you to make perfect decisions with imperfect information.
>
> Mukherjee [127]

3.1. Introduction

As introduced in Chapter 1, clinical data mining is the application of machine learning to medical databases as a tool to assist clinical research and decision-making. Its value is dependent on the performance and interpretability of the models, as well as their appropriate integration in clinical software [142]. While many methodological successes exist, examples that affect patient management are seldom found [32]. Incidentally, the uncertainty in clinical measurements limits the usefulness of the learned models, as it reduces their predictive performance [166, 108]. Improving the quality of the data is however resource-intensive and often unfeasible [99]. We hypothesize that the acknowledgment of uncertainty in the models, in particular by making use of existing knowledge regarding the reliability of each measurement, can improve their performance and usefulness as a decision-support tool.

The uncertainty of a measurement denotes the lack of knowledge about its exact value, and is often caused by noise in the acquisition [96]. Clinical data uncertainty can originate from various sources, such as:

- variability in diagnostic practices [30];
- variations associated to the technology manufacturer [60];
- inter- and intra-observer variability [174];
- patient factors such as body habitus or claustrophobia, which can affect the choice and quality of an imaging test [72];
- the use of distinct modalities to achieve a measurement [116]. For example, in the determination of device size for left-atrial appendage closure, consistency between computerized tomography, transesophageal echocardiography and angiography was observed in only 22% of the cases [116].

  On a different example, the estimation of left-ventricular ejection fraction (LVEF), the variability between cardiac magnetic resonance and

echocardiography resulted in 28% of the population having opposing device eligibility [82]. LVEF is one of the most commonly-used descriptors in cardiology.

As discussed in Section 1.3, clinical reasoning involves making guideline-abiding decisions resorting to such data. The experienced physician is able to assess the reliability of each measurement and integrate it with knowledge from his/her training and experience. This endeavor is all the more challenging, when we consider the emphasis on internal validity of clinical research, as opposed to external validity; i.e. the rigor of clinical studies contrasts with the unpredictability and diversity of the target populations. Some authors highlight the contrast between the rigorous design of study protocols and the actual conditions to which the evidence is employed [75].

3.1.1. Uncertain measurements and decision tree learning

Learning decision trees (DTs) from noisy measurements using the standard top-down algorithms can overfit to the noise and fail to generalize. Generating predictions for noisy instances can generate incorrect predictions [160]. As we saw in Chapter 2 Equation 2.1, univariate tests compare the input variables with *hard thresholds*. Small errors in the measurements can therefore lead to the instance following an opposing path, which may lead to a different medical decision. Moreover, the distance between the measurement and the threshold is not taken into account [160]. Several algorithms explore the idea of using *soft thresholds* to make DTs robust to noise [160, 48, 176, 92, 190]. Such approaches weigh the contribution of all child branches to the prediction. Some methods focus on cognitive uncertainty, while others focus on statistical uncertainty or noise.

Fuzzy DT methods use fuzzy logic to handle cognitive uncertainty, which deals with inconsistencies in human reasoning [190]. For example, instead of considering cognitive categories such as *{hot, mild, cold}* as *crisp* or strict sets, fuzzy logic uses a membership function for each category to account for the variability in human classification. Using fuzzy DTs to handle statistical noise involves setting the fuzzy membership functions to express uncertainties around the DT thresholds, which need to be known in advance. Discretization algorithms can be alternatively used, but those methods are computationally costly [181], and the accuracy of the resulting DTs strongly depends on the chosen algorithm [170].

Probabilistic DTs, on the other hand, focus on *stochastic uncertainty* or random noise, arising from the unpredictable behavior of physical systems and measurement limitations:

- Quinlan [162] proposed a method using piecewise-linear thresholdss (PLTs) for evaluating a DT, assuming a two-modal uniform distribution for the noise. The parameters of this distribution are set using a statistical heuristic based on the training data. Dvorák and Savický [48] employed a variation of this method, where the parameters were estimated through simulated annealing. Experiments in one dataset led to 2-3% error rate reductions compared to CART [19], suggesting the potential of the method and the need for an evaluation on more data. No significant differences were found between the two parameter-estimation methods, but the authors highlight the computational cost of simulated annealing. As a limitation, the approach is applied to a finalized DT, so the uncertainty is not accounted for during training.

- The uncertain decision tree (UDT) algorithm [176] extends the probabilistic splits to the training phase, assuming a Gaussian model for the noise, and achieving accuracy gains in 10 datasets. The method takes an oversampling strategy, where each measurement is replaced by s points that represent the distribution. The authors propose optimized searches to control the increased training time, resulting from the oversampling. In UDT, the same noise model is used for training and evaluation. In medical research and practice, however, the noise in the data used to obtain evidence can be very different from the noise of the data to which is evidence is extrapolated [75]. For this reason, a learning algorithm ideally supports independent uncertainty models for the training and the target data.

- Soft approaches have also been proposed for multivariate DTs. One algorithm [92] employs logistic regression, treating the separation of data among the children as a classification problem. The method led to small accuracy gains in 10 classification datasets, with significant reductions in tree size. Hierarchical mixtures of experts are another multivariate tree architecture [97], where each test is a softmax linear combination of all variables, with performance gains compared to CART. Multivariate DTs can compactly model complex phenomena, and generally improve accuracy with fewer nodes. They are however intended to generate predictions that do not need to be understood, lacking the interpretability of univariate DTs.

These approaches hint at the potential benefit of a probabilistic approach in improving DT performance. However, the strategies used to model noise are either limited to the evaluation phase, or they do not consider independent noise distributions for learning and evaluation phases.

Moreover, the behavior of the algorithms upon increasing noise remains unclear, and no distinction was made between the impact of noise in the training data and noise in the test data. Preliminary experiments on one dataset [161] indicate that when subject to small amounts of noise in the training data, the DT performance is not significantly affected. In the presence of higher noise levels, the impact was more evident when corrupting the test data. Overall, a flexible approach for handling measurement uncertainty has not been established for the construction of interpretable DTs, and an account of the its effect on DT performance has yet to be exhausted.

In this chapter, we describe a probabilistic learning approach to handle measurement uncertainty, which is modeled as a distribution of noise added to the actual measurements. Unlike previous methods, the uncertainty distribution is independently considered:

1. During training, when searching for the best threshold at each node, which we call soft search (SS);

2. During training, in the propagation of the training instances through the DT, denoted soft training propagation (STP);

3. During evaluation, in the propagation of the test instances through a finalized tree, which we refer to as soft evaluation (SE).

To promote interpretability, we consider univariate DTs. The intended use of this approach envisions the integration of existing knowledge about the uncertainty, e.g. based on clinical experience or from the meta-analysis of clinical studies. As a proof-of-concept, we opted for a normally-distributed noise model, and evaluate its impact on the distinct learning phases. But any distribution of noise can be used. Furthermore, we study the effect of corrupting the data used for training or testing in the models. Moreover, the proposed SS approach attempts to keep the computational cost under control.

In the following section, we introduce a probabilistic interpretation for the DTs. The manuscript then proceeds with a description of the proposed soft DT algorithm components, SS, STP and SE, followed by the experiments to illustrate and evaluate them.

3.1.2. Probabilistic interpretation of the splits

Consider $p(y|\mathbf{x})$ the probability of y given instance $\mathbf{x}$, based on which we can make a prediction about y. Consider also the binary DT rooted in node n, with left and right child nodes n_L and n_R. Let us associate the

outcome b of the test $t(\mathbf{x})$ at n to the child branches as $b \in \{n_L, n_R\}$. We can estimate $p(y|\mathbf{x})$ from this DT as:

$$\begin{aligned}\hat{p}(y|\mathbf{x}) &= \textstyle\sum_{b\in\{n_L,n_R\}} p(y|b,\mathbf{x})p(b|\mathbf{x}) \\ &= \textstyle\sum_{b\in\{n_L,n_R\}} p(y|b)p(b|\mathbf{x}) \\ &= p(y|n_L)p(n_L|\mathbf{x}) + p(y|n_R)p(n_R|\mathbf{x}),\end{aligned} \tag{3.1}$$

where y and $\mathbf{x}$ are conditionally independent given b.

Equation 3.1 is based on Bayesian model averaging [87], which translates the uncertainty of the distinct models, in this case the two subtrees n_L and n_R, into uncertainty in the class prediction. Instead, probabilistic DTs use this idea to express the uncertainty about the input $\mathbf{x}$. This uncertainty can be modeled through the posterior $p(n_L|\mathbf{x})$, also known as the *gating function*, $g(\mathbf{x})$ [92]. The estimate of $p(y|\mathbf{x})$ becomes:

$$\hat{p}(y|\mathbf{x}) = p(y|n_L)g(\mathbf{x}) + p(y|n_R)\big(1 - g(\mathbf{x})\big). \tag{3.2}$$

The gating function is the probabilistic interpretation of $t(\mathbf{x})$. If we assume that the observed value is accurate, node n performs a *hard split* as in Equation 2.1:

$$g(\mathbf{x}) = \mathbf{1}_{[\tau,\infty)}(x_j), \tag{3.3}$$

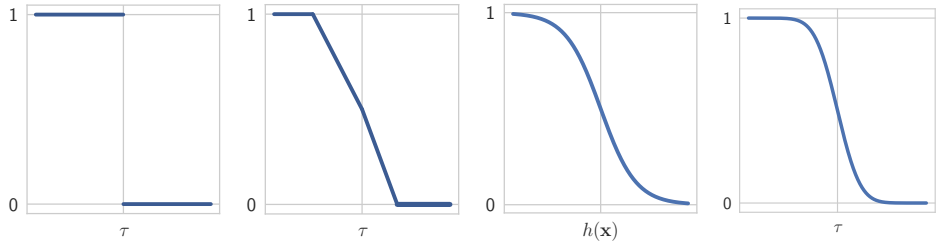
In this case, if x_j is close to τ, small variations in its value can drastically change the estimate $p(y|\mathbf{x})$, and produce incorrect predictions [160]. Probabilistic DTs instead model the uncertainty of each variable X_j as a distribution of noise, and $g(\mathbf{x})$ represents the cumulative density function (CDF) of the chosen distribution. A small variation of x_j around the threshold will smoothly alter $p(y|\mathbf{x})$. The shapes of the gating function employed in the probabilistic approaches introduced in Section 3.1.1 are displayed in Figures 3.1b to 3.1d.

When searching for the best split for n, we use the data subset D_n to obtain the probabilities of Equation 2.6. E.g. we can estimate $p(n_L)$, or equivalently $p(n_R)$, in terms of $g(\mathbf{x})$:

$$\hat{p}(n_L) = \frac{\sum_{\mathbf{x}\in D_n} p(n_L|\mathbf{x})}{\sum_{\mathbf{x}\in D_n} 1} = \frac{\sum_{\mathbf{x}\in D_n} g(\mathbf{x})}{|D_n|} \tag{3.4}$$

3.2. Proposed approach

We propose a probabilistic DT approach to handle uncertainty by modeling it as a distribution of noise around the observed value. The noise model should expresses existing knowledge about the uncertainty. This model is decomposed into three independent components:



(a) Indicator (b) Piecewise linear (c) Sigmoid where $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ (d) Normal CDF

Figure 3.1: Proposed shapes of the gating function $g(x)$.

1. When searching for the split thresholds during training, which we call soft search (SS) (Section 3.2.1),
2. When propagating the training instances through the tree, or soft training propagation (STP) (Section 3.2.2), and
3. When propagating test instances through the constructed DT, or soft evaluation (SE) (Section 3.2.3).

As a proof-of-concept, we consider that the noise of variable X is additive and captured by the variable $\mathscr{E}_X \sim \mathcal{N}(0, \sigma^2)$. The normal distribution can be useful for various types of data, owing to the Central Limit Theorem, and is fully described by its mean and variance. As such, we consider it a good assumption to study our approach.

3.2.1. Soft search

Let us focus on the computation of the information gain for a variable X for node n. Suppose that X takes V *distinct* values sorted as $\{x^{(1)}, ..., x^{(V)}\}$ with $x^{(i)} < x^{(i+1)}$ and $i = 1, ..., V-1$, in the training data D. Note that $|D| \geq V$. Let N be the number of instances in n, let $N(x^{(i)})$ be the number of instances in n taking value $x^{(i)}$, and $N(y, x^{(i)})$ the number of such instances that belong to class y. Let τ denote the candidate threshold for a split based on X.

In C4.5, the search is done by computing the information gain in Equation 2.2 for each candidate value $\tau = x^{(i)}$, $i = 2, ..., V$. Since the method assumes exact measurements, we have $g(\mathbf{x}) = \mathbf{1}_{[\tau,\infty)}(x_j)$, and the probabilities $p(b)$ and $p(y|b)$ are estimated as in Equation 2.6. We now describe how to obtain these probabilities efficiently, when considering a model for the noise in the measurements.

If we consider the model $\mathscr{C}_X \sim \mathcal{N}(0, \sigma^2)$, the gating function will be the normal CDF. Let us denote the numerator of Equation 3.4 as $\rho(\tau)$, representing the density of training examples falling on the left child node:

$$\rho(\tau) = \sum_{\mathbf{x} \in D} p(n_L|\mathbf{x}) = \sum_{\mathbf{x} \in D} g(\mathbf{x}) = \sum_{\mathbf{x} \in D} \int_{-\infty}^{\tau} K(\tau - x, \sigma) d\tau, \tag{3.5}$$

where $K(\tau - x, \sigma)$ is the Gaussian kernel function centered on the measurement x with variance σ^2:

$$K(\tau - x, \sigma) \propto exp\big(-\tfrac{(\tau - x)^2}{2\sigma^2}\big). \tag{3.6}$$

This corresponds to using Gaussian kernel density estimation for the probabilities used to compute the information gain. Since $\rho(\tau)$ is no longer constant between each $x^{(i)}$ and $x^{(i+1)}$, the candidate thresholds need not be the values in the dataset. We take the equidistant values:

$$\tau = \tau_{min},\ \tau_{min} + \delta\sigma,\ \tau_{min} + 2\delta\sigma,\ ...,\ \tau_{max} \tag{3.7}$$

with $\tau_{min} = x^{(1)} - \frac{\omega}{2}\sigma$ and $\tau_{max} \leq x^{(V)} + \frac{\omega}{2}\sigma$. The parameters δ and ω respectively control the resolution and window of the search, $\omega > \delta$. For efficiency, we consider the kernel adjusted to be zero for $|\tau - x| > \omega$. Using the values of Equation 3.7, the number of information gain computations is limited to $\frac{1}{\delta}(\tau_{max} - \tau_{min}) = \frac{1}{\delta}(x^{(D)} - x^{(1)} + \omega\sigma)$.

To compute the information gain efficiently, we keep two running sums of the density $\rho(\tau)$ on the left and on the right of τ: $\rho_L(\tau)$ and $\rho_R(\tau)$. The search is initialized with $\tau = \tau_{min},\ \ \rho_L(\tau) = 0$, and $\rho_R(\tau) = N$. Before each update in τ, the left and right sums are respectively incremented and decremented the quantity $\Delta\rho(\tau)$:

$$\Delta\rho(\tau) = \begin{cases} \sum_{i=1}^{V} N(x^{(i)}) \Phi\big(\frac{\tau - x^{(i)}}{\sigma}\big), & \text{if } \tau = \tau_{min} \\ \sum_{i=1}^{V} N(x^{(i)}) \Big[\Phi\big(\frac{\tau - x^{(i)}}{\sigma}\big) - \Phi\big(\frac{\tau - \delta\sigma - x^{(i)}}{\sigma}\big)\Big], & \text{if } \tau_{min} < \tau < \tau_{max} \\ \sum_{i=1}^{V} N(x^{(i)}) \Big[1 - \Phi\big(\frac{\tau - \delta\sigma - x^{(i)}}{\sigma}\big)\Big], & \text{if } \tau = \tau_{max} \end{cases} \tag{3.8}$$

where $\Phi(\cdot)$ is the CDF of the standard normal distribution. Equation 3.8 corresponds to adding and subtracting the areas under the CDF between

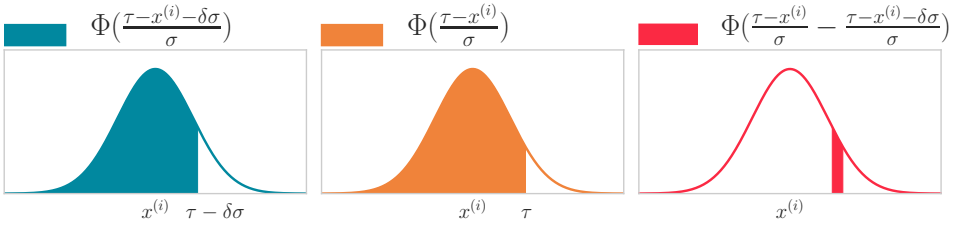


Figure 3.2: Weight of a density increment relative to value $x^{(i)}$ when iterating through the candidate values τ, as in Equation 3.8. The areas of the shaded regions correspond to the quantities in the legends.

each two values of tau, as depicted in Figure 3.2. The quantities $\rho_L(\tau)$ and $\rho_R(\tau)$ are updated as:

$$\rho_L(\tau) \leftarrow \rho_L(\tau) + \Delta\rho(\tau)$$
$$\rho_R(\tau) \leftarrow \rho_R(\tau) - \Delta\rho(\tau),$$

and τ is then updated to $\tau + \delta\sigma$. The contribution of measurement $x^{(i)}$ to $\Delta\rho(\tau)$ is proportional to:

$$\Phi\left(\frac{\tau - x^{(i)}}{\sigma}\right) - \Phi\left(\frac{\tau - x^{(i)} - \delta\sigma}{\sigma}\right)$$

and the last increment $\Delta\rho(\tau_{max})$ ensures that the density contributions of $x^{(i)}$ sum to $N(x^{(i)})$.

Similarly, to estimate $\hat{p}(y|\boldsymbol{n}_L)$, we consider the sum of the densities per class, $\rho(y, \tau)$. The density increments per class $\Delta\rho(y, \tau)$ are computed by replacing the number of instances $N(x^{(i)})$ by $N(y, x^{(i)})$ in Equation 3.8. Finally, the estimated probabilities $\hat{p}(\boldsymbol{n}_L)$ and $\hat{p}(y|\boldsymbol{n}_L)$ are used to minimize the conditional entropy $H(Y|B)$ in Equation 2.4. Searching for the threshold using the set of values in Equation 3.7 and the density increments $\Delta\rho(\tau)$ and $\Delta\rho(y, \tau)$ acts as a Gaussian filter to the information gain. We refer to this approach as soft search (SS).

The standard deviation of $\mathscr{C}_X$ is set to $\sigma = u_s\bar{x}$, where u_s is the *uncertainty factor* of the SS and $\bar{x}$ is the sample mean of X. This choice of σ was made because clinicians often report the uncertainty as a fraction of the absolute values of the attributes [42]. In the following experiments, u_s is the same for all variables, but it could be specified independently.

3.2.2. Soft training propagation

We can also account for the uncertainty when splitting the training data during induction, after the split variable X and threshold τ have been se-

lected for node n. We denote this as soft training propagation (STP). The two soft training approaches, SS and STP can be used in combination or independently.

As before, a soft split is achieved by setting the gating function to the CDF of $\mathscr{E}_X$, centered in the measurement x:

$$g(\mathbf{x}) = F_{\mathscr{E}_X}(\tau - x) = \Phi\big(\tfrac{\tau - x}{\sigma}\big). \tag{3.9}$$

with $F_{\mathscr{E}_X}(.)$ the CDF of $\mathscr{E}_X$. Each training instance is fractionally divided between the child nodes, according to the probability $g(\mathbf{x})$. The standard deviation of $\mathscr{E}_X$ is set to $\sigma = u_p \bar{x}$, with u_p the STP uncertainty factor, and $\bar{x}$ the variable mean.

STP leads to increased learning times. Using hard splits, each instance is sent down a single branch, and the total number of instances remains constant at each level of the tree. The number of information gain computations at any depth is bounded by the dataset size and the number of attributes, $|D|M$. With STP, each instance is sent down all the branches, with weights determined by the gating function. Therefore, a node sees more instances during the search compared to the hard approach. The number of gain computations at a given DT depth d is raised to $2^d|D|M$.

3.2.3. Soft evaluation

The uncertainty may also be accounted for when classifying test cases with a finalized DT. Soft evaluation (SE) is achieved by setting the gating function as in Equation 3.9, but with $\sigma = u_e \bar{x}$. The evaluation uncertainty factor is denoted as u_e and $\bar{x}$ is obtained from the *training* data. From Equation 3.2, the class probability estimates for instance $\mathbf{x}$ become:

$$\hat{p}(y|\mathbf{x}) = p(y|\boldsymbol{n}_L)\Phi(\tfrac{\tau - x}{\sigma}) + p(y|\boldsymbol{n}_R)\big(1 - \Phi(\tfrac{\tau - x}{\sigma})\big). \tag{3.10}$$

As described in Section 3.1.2, this approach adjusts $\hat{p}(y|\mathbf{x})$ to reflect the choice of noise distribution, in this case, Gaussian. The uncertainty factors u_s, u_p, and u_e are maintained independently for the three components (SS, STP and SE).

3.2.4. Motivation on a toy example

To better motivate the use of the soft algorithm components, we consider a toy example with a input variable X and class $Y \in \{0, 1\}$.

Soft search

Suppose we have a training dataset of four examples:

$$\{(x_{(1)}, 0), (x_{(2)}, 0), (x_{(3)}, 1), (x_{(4)}, 1)\}$$

with $x_{(1)} = x_{(2)} = -2$ and $x_{(3)} = x_{(4)} = 2$. Figure 3.3a displays the information gain $I(X;Y)$ computed by C4.5 or the SS for the corresponding candidate splits, denoted by τ. C4.5 would choose $\tau = 2$ as a split, while SS would choose a split close to $\tau = 0$, potentially avoiding the misclassification of test examples in the interval $(0, 2)$, for which there is no training data. If the measurements are noisy such that e.g. $x_{(2)} = 0.2$ and $x_{(3)} = -0.2$, C4.5 would select $\tau = 0.2$ or $\tau = 2$, as seen in Figure 3.3b. The SS would select a value close to zero.

Soft training propagation

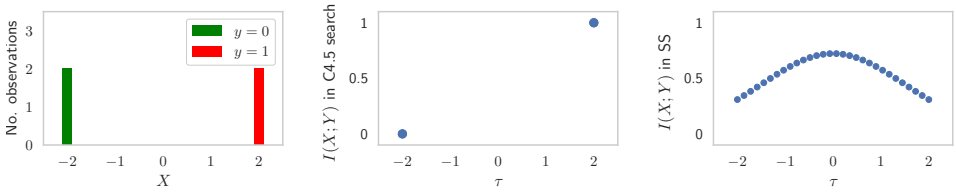
Let us now discard two training examples, and keep $\mathbf{x}_{(1)} = (-2, 0)$ and $\mathbf{x}_{(4)} = (2, 1)$. We model the uncertainty of X, such that $x_{(1)} \sim \mathcal{N}(-2, \sigma^2)$ and $x_{(4)} \sim \mathcal{N}(2, \sigma^2)$ are drawn from normal distributions with mean $\mu_{(1)} = -2$ and $\mu_{(2)} = 2$, and common variance σ^2. Let us also consider the test point $\mathbf{x}_{(t)} = (x_{(t)}, 1)$. We analyze the probability of misclassifying $\mathbf{x}_{(t)}$. We note that the sum (s) and difference (d) of normal distributions are also normally-distributed:

$$\begin{aligned}
s &= x_{(1)} + x_{(4)} \sim \mathcal{N}(-2 + 2, \sigma^2 + \sigma^2) = \mathcal{N}(0, 2\sigma^2), \\
d &= x_{(1)} - x_{(4)} \sim \mathcal{N}(-2 - 2, \sigma^2 + \sigma^2) = \mathcal{N}(-4, 2\sigma^2), \\
m &= \tfrac{s}{2} = \tfrac{x_{(1)} + x_{(4)}}{2} \sim \mathcal{N}(\tfrac{1}{2} \cdot 0, \tfrac{1}{2^2}) = \mathcal{N}(0, \tfrac{\sigma^2}{2})
\end{aligned}$$

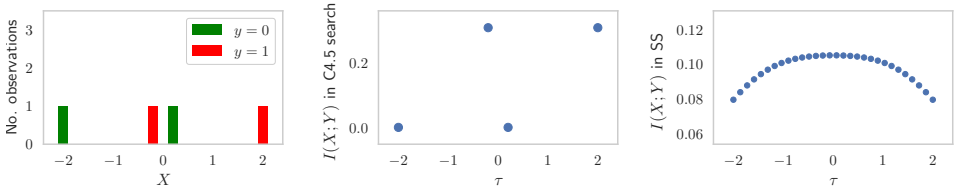
Let us assume that the search selects the midpoint m as a split value. The prediction $\hat{y}(x_{(t)})$ depends on whether the difference $c = x_{(t)} - m$ is greater than or less than 0. Specifically, the probability of misclassifying $\mathbf{x}_{(t)}$ is composed by two terms:

$$P\big(\hat{y}(x_{(t)}) = 0\big) = P(d < 0)P(c < 0) + P(d > 0)P(c > 0).$$

In case $x_{(1)}$ is smaller than $x_{(4)}$, C4.5 assigns class 0 to $x_{(t)}$ iff $x_{(t)} < m$. If $x_{(1)}$ is greater than $x_{(4)}$, C4.5 assigns class 0 to $x_{(t)}$ if $m < x_{(t)}$. Each of the above probabilities is given by the CDF associated with the corresponding distribution. If we shift each distribution to have 0 mean, we can express



(a) Example with $(x_{(1)}, y_{(1)}) = (x_{(2)}, y_{(2)}) = (-2, 0)$ and $(x_{(3)}, y_{(3)}) = (x_{(4)}, y_{(4)}) = (2, 1)$.



(b) Example with $(x_{(1)}, y_{(1)}) = (-2, 0)$ and $(x_{(4)}, y_{(4)}) = (2, 1)$, and noisy measurements $(x_{(2)}, y_{(2)}) = (0.2, 0)$ and $(x_{(3)}, y_{(3)}) = (-0.2, 1)$.

Figure 3.3: Motivating example to show the effect of employing the soft search (SS) on the information gain $I(X;Y)$, with $\sigma_s = 0.3$.

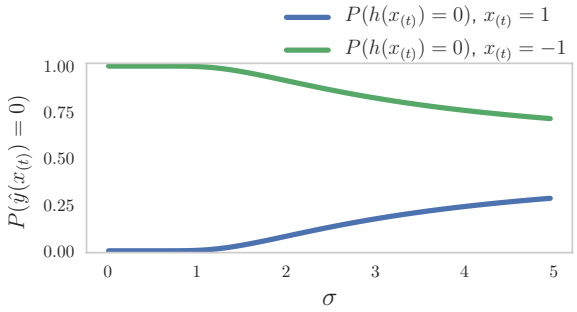
the probabilities as:

$$P(d < 0) = P(d + 4 < 4) = F_{d+4}(4) = \Phi\left(\tfrac{4}{2\sigma}\right),$$
$$P(d > 0) = 1 - P(d < 0) = 1 - \Phi\left(\tfrac{4}{2\sigma}\right).$$

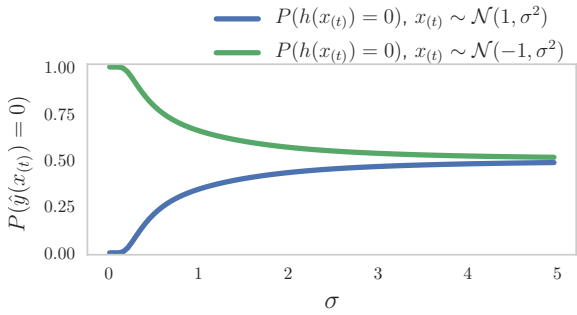
F_{d+4} is the CDF of $d + 4$. Figure 3.4a displays the probability of misclassifying $x_{(t)}$ as a function of σ when $x_{(t)}$ is certain. When $x_{(t)} = 1$, and so $c > 0$, $P\big(\hat{y}(x_{(t)}) = 0\big)$ is nearly zero until the distributions of the training instances start to overlap with increasing σ. The inverse occurs for $c < 0$, where the error probability starts to decrease. Figure 3.4a shows how the prediction probabilities change as the model expresses less confidence on the training data, when using STP.

Soft evaluation

Let us now express the uncertainty about the test example, such that $x_{(t)}$ is also normally distributed with variance σ^2. We can obtain $P(c < 0)$ and



(a) $P\big(\hat{y}(x_{(t)})$ when we assume $x_{(t)}$ to be certain.



(b) $P\big(\hat{y}(x_{(t)})$ when $x_{(t)}$ is assumed to have noise.

Figure 3.4: Probability of misclassifying $(x_{(t)}, 1)$ as function of the standard deviation σ of the normal uncertainty model. In 3.4a, the uncertainty model is considered only for the training instances $x_{(1)}$ and $x_{(4)}$, simulating soft training propagation (STP), while $x_{(t)}$ is exact. In 3.4b $x_{(t)}$ has normally-distributed noise, as when using in soft evaluation (SE).

$P(c > 0)$ as:

$$\begin{aligned} P(c < 0) &= P(c - 1 < -1) \\ &= F_{c-1}(-1) \\ &= 1 - F_{c-1}(1) \\ &= 1 - \Phi\big(\tfrac{\sqrt{2}}{\sqrt{3}\sigma}\big), \end{aligned}$$

$$\begin{aligned} P(c > 0) &= 1 - P(c < 0) \\ &= \Phi\big(\tfrac{\sqrt{2}}{\sqrt{3}\sigma}\big). \end{aligned}$$

Figure 3.4b displays $P\big(\hat{y}(x_{(t)} = 0)$ for $x_{(t)} \sim \mathcal{N}(1, \sigma^2)$ and $x_{(t)} \sim \mathcal{N}(-1, \sigma^2 2)$, which now converges faster to 0.5. In other words, the class probabil-

ity estimates became less extreme as a consequence of expressing less confidence in the measurement $x_{(t)}$.

3.3. Experiment design

The experiments assess the effect of the proposed uncertainty model in the distinct DT learning phases, and the impact of corrupting the training versus the test data with noise. Improving prediction performance consists in maximizing the generalization accuracy, i.e. the fraction of correctly classified test examples, while minimizing model complexity [102]. DT complexity is assessed by measuring its number of leaves.

Each proposed soft component, SS, STP and SE is independently compared to C4.5. The C4.5 pruning and missing-value strategies are equally employed in all experiments, as described in Sections 2.2.3 and 2.2.4. We also extend the evaluation of the PLT [162] and UDT [176] approaches to more data and noise scenarios.

3.3.1. Data description and pruning confidence factor

Table 3.1 displays the employed datasets. We sought clinical data from open resources, including the University California Irvine (UCI) machine learning (ML) [112] and the KEEL [1] repositories. Non-ordinal features were excluded. We synthesized 5 additional datasets using an adaptation of the method by Guyon [81], available in Scikit-learn's implementation *make_classification* [149][1]. The datasets were generated with different numbers of variables and instances, varying correlation between the variables, and varying numbers of hypercube vertices.

Since optimal DT size is problem- and dataset-dependent [94], the pruning confidence factor is not optimized. Instead, for each dataset the confidence factor is fixed such that the average DT size achieved by C4.5 through cross validation (CV) is 15 leaves. This corresponds to a binary tree wit nearly 4 levels, considered a manageable/interpretable number of decisions in a visual clinical guideline. Some of the datasets were too small to reach 15 leaves, so their confidence factor is set to either 10 or 5 leaves. The confidence factor is fixed across all experiments.

[1] The method creates normally-distributed clusters around vertices of a hypercube. Each vertex is assigned to a class, which is followed by introducing some correlation between the attributes.

Table 3.1: Classification datasets used in the experiments.

Dataset	Source	No. variables	No. instances	No. classes
Hepatitis	KEEL	6	155	2
Heart disease	UCI ML	8	303	2
Pima Indians diabetes	UCI ML	8	768	2
South African heart	UCI ML	8	462	2
Breast cancer Wisconsin	UCI ML	39	569	2
Dermatology	UCI ML	34	366	6
Haberman’s breast cancer	UCI ML	3	306	2
Indian liver patient	UCI ML	9	582	2
BUPA liver disorders	UCI ML	6	345	2
Vertebral column (2 classes)	UCI ML	12	310	2
Vertebral column (3 classes)	UCI ML	12	310	3
Thyroid gland	UCI ML	5	215	3
Oxford Parkinson’s disease	UCI ML	22	194	2
SPECTF	UCI ML	44	266	2
Thoracic surgery	UCI ML	3	470	2
Synthetic 1	Guyon	15	30×500	2
Synthetic 2	Guyon	15	30×400	2
Synthetic 3	Guyon	20	30×300	2
Synthetic 4	Guyon	25	30×200	3
Synthetic 5	Guyon	20	30×250	3

3.3.2. Experiments

For each real dataset, 30 random train-test permutations were created, containing respectively 70% and 30% of the data. Stratified sampling was used, so that class proportions are equal in all samples. For each synthetic dataset, distinct instances were generated and divided into 30 different sets, which were then split into 70%-30% train-test samples.

The experiments are performed with varying degrees of noise to the data. The noise added to a variable X in a data subset is sampled according to $\mathcal{N}(0, n\bar{x})$, with n the *noise factor* and $\bar{x}$ the training subset mean of X. The same n is used for all its variables. All randomness was generated with fixed seeds for reproducibility.

Experiment 1: noise added to the training data

Experiment 1 studies the approaches with noise added to the training data, so we denote the training noise factor as n_{train}. The uncertainty factors u_s, u_t, u_e of the SS, STP and SE approaches, as well as the UDT parameter w, control the standard deviation of the uncertainty model. As such, we tune them by cross validation (CV) for each dataset and n_{train}. The SS parameters ω and δ are respectively set to 6.0 and 0.1. Initial experiments showed that they do not significantly impact the results, provided that ω is

large enough to contain most of the density of the uncertainty distribution, and δ is small enough to ensure a large set of candidate splits. The PLT parameters τ^- and τ^+ are derived using the method proposed by Quinlan [162]. We summarize the steps for model tuning and evaluation:

> For each $70\% - 30\%$ train-test permutation, n_{train} and model:
> 1. Hold out the 30% test set
> 2. If model has parameter to set (u_s, u_t, u_e, or w), use the 70% training set to tune it by CV:
>
> (a) Compute 10 stratified CV folds.
>
> (b) Add noise to the CV training folds, distributed as $\mathcal{N}(0, n_{train}\bar{x})$. Do not add noise to the CV validation folds.
>
> (c) Tune the parameter to maximize CV accuracy.
>
> 3. Add noise to the initial 70% training set, distributed as $\mathcal{N}(0, n_{train}\bar{x})$.
> 4. Learn a tree using the noisy training set, and the selected parameter value, if applicable.
> 5. Evaluate the tree on the 30% test set, to which no noise was added.

C4.5 and PLT do not undergo step 2.

Experiment 2: noise added to the test data

Experiment 2 evaluates the merit of the models built on data without added noise applied to noisy test cases. Noise is added to the data used for evaluation, and we denote n as n_{test}. Model evaluation is done as:

> For each $70\% - 30\%$ train-test permutation, n_{test} and model:
> 1. Hold out the 30% test set
> 2. If model has parameter to set (u_s, u_t, u_e, or w), use the 70% training set to tune it by CV:
>
> (a) Compute 10 stratified CV folds.
>
> (b) Do not add noise to the CV training folds. Add noise to the CV validation folds, distributed as $\mathcal{N}(0, n_{test}\bar{x})$.
>
> (c) Tune the parameter to maximize CV accuracy.
>
> 3. Learn a tree using the initial 70% training set, and the selected parameter value, if applicable
> 4. Add noise to the 30% test set, distributed as $\mathcal{N}(0, n_{test}\bar{x})$.
> 5. Evaluate the tree on noisy the 30% test set.

As before, C4.5 and PLT do not undergo step 2. Note that noise is added to the CV validation folds.

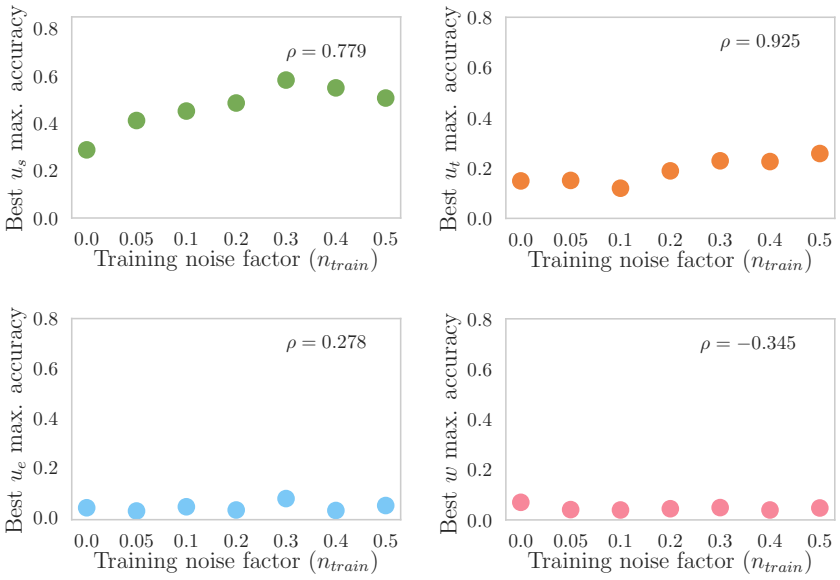


Figure 3.5: Average uncertainty (u_s, u_t, u_e) and window (w) factors selected to maximize cross validation (CV) accuracy in the presence of noise added to the CV training folds.

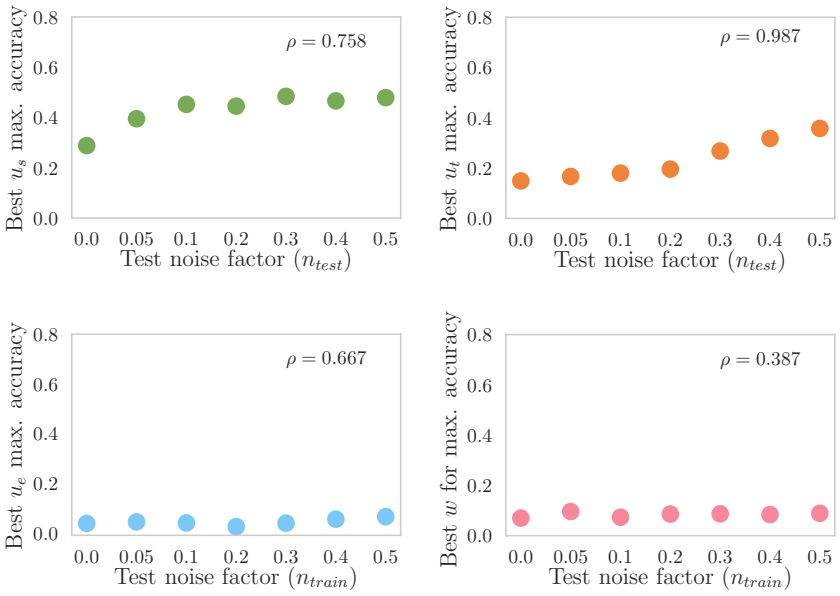


Figure 3.6: Average uncertainty (u_s, u_t, u_e) and window (w) factors selected to maximize cross validation (CV) accuracy in the presence of noise added the CV validation folds.

Parameter tuning

Figures 3.5 and 3.6 display the mean values of the parameters that control the uncertainty distribution for SS, STP, SE and UDT, selected through CV in step 2 of the experimental design. The mean is taken over all datasets.

3.4. Results

Section 3.4.1 illustrates SS, STP and SE on a single variable. In Section 3.4.2, we show the results of Experiments 1 and 2.

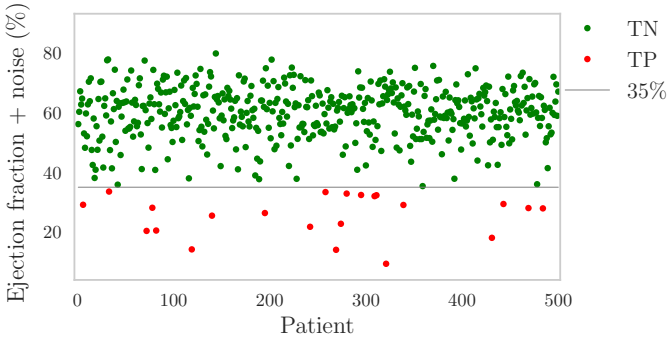
3.4.1. Illustration on a single variable

We take the left-ventricular ejection fraction (LVEF) estimates of the *Data Science Bowl Cardiac Challenge* [132]. LVEF is a variable of critical importance in cardiology. Implantable device therapy is officially recommended for $LVEF < 35\%$ [152]. Therefore, we take cases with $LVEF < 35\%$ to have positive eligibility, i.e. $Y = \mathbf{1}_{(0,35\%)}$, as shown in Figure 3.7a. Adding random noise to these data results in 7 false negatives (FN) and 5 false positives (FP), as seen in Figure 3.7b.

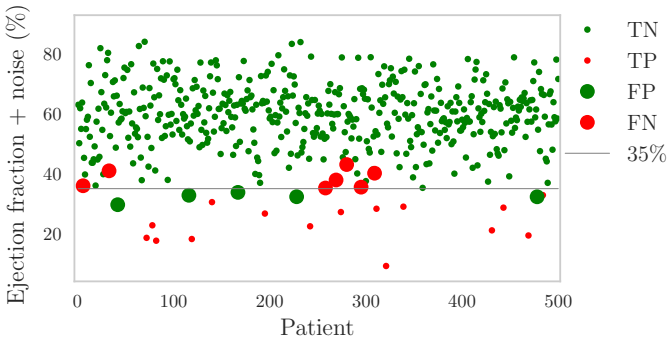
Soft search: In Section 3.2.4, we motivated the SS as way of increasing the set of candidate splits and smoothing the information gain. We now observe this on real measurements. Figure 3.8a displays the number of patients for each class and LVEF value, $N(y, x^{(i)})$, like a histogram. Figure 3.8b shows the same data with noise. Figure 3.8c shows the SS density increments $\Delta\rho(y, \tau)$, and SS information gain.

Soft search methods such as SS or UDT increase the set of candidate splits. The LVEF dataset has 500 instances. Employing UDT with a resampling factor $s = 100$ would raise the number of information gain computations from $5e2$ to $5e4$ [176]. On the other hand, setting the SS parameters e.g. as $w = 8$, $\delta = 0.05$ and $u_s = 0.1$ would limit this number to a maximum of approximately $2.3e3$ computations.

Soft training propagation: To illustrate how STP alters the class probability estimates, we considered the noisy data of Figure 3.7b. We learned two single-node two-leaf trees using the standard training propagation and STP with $u_p = 0.1$. Table 3.2 shows that the class probability estimates are less extreme when STP is employed, as they reflect the choice of noise distribution.



(a) LVEF measurements.



(b) LVEF measurements with added noise.

Figure 3.7: (a) Left-ventricular ejection fraction (LVEF) data of the Data Science Bowl Cardiac Challenge. (b) Same data with noise sampled from $\mathcal{N}(0, 0.1\bar{x})$, with $\bar{x}$ the mean. $LVEF < 35\%$ indicates therapy eligibility leading to true negatives (TN), true positives (TP), false positives (FP), false negatives (FN).

Soft evaluation: Figure 3.7b shows that 7 FN and 5 FP were introduced by the noise added to the LVEF dataset. Table 3.3 shows the probability estimates of those misclassified examples, obtained using hard or soft evaluation, with $u_e = 0.1$. The numbers of FN and FP are different because the algorithm learned a threshold of 35.37% rather than 35%.

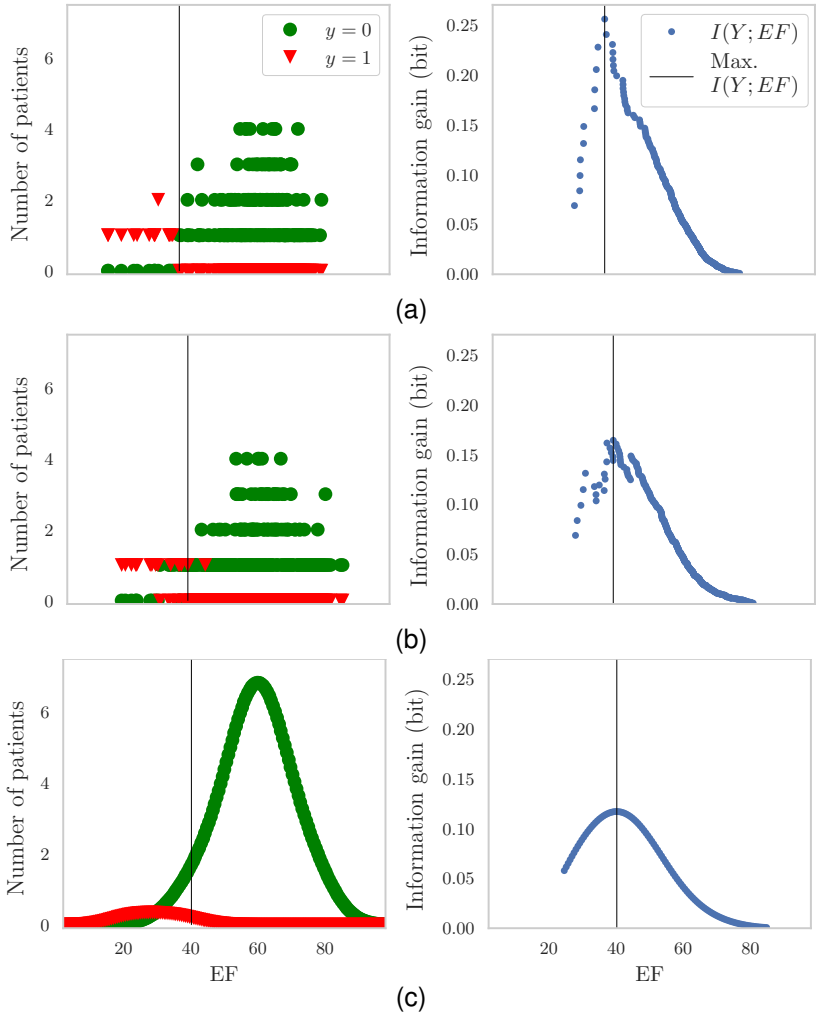


Figure 3.8: Information gain computation the left-ventricular ejection fraction (LVEF) measurements of Figure 3.7, using the standard search (a,b) or soft search (SS) with $u_s = 0.1$ (c). The left (a) and (b) plots show the number of patients for each class and $x^{(i)}$, $N(y = 0, x^{(i)})$ and $N(y = 1, x^{(i)})$. The left (c) plot displays the SS density increments, $\Delta\rho(y, \tau)$. The right plots show the corresponding information gain.

Table 3.2: Class probability estimates at leaves n_L and n_R of the trees learned on the noisy left-ventricular ejection fraction (LVEF) dataset of Figure 3.7b. Decision trees (DTs) learned using standard or soft training propagation (STP).

	Standard training propagation	STP
$\hat{p}(y=0\|n_L)$	0.000	0.310
$\hat{p}(y=1\|n_L)$	1.000	0.790
$\hat{p}(y=0\|n_R)$	0.981	0.979
$\hat{p}(y=1\|n_R)$	0.018	0.021

Table 3.3: Class probability estimates for the misclassified examples of the noisy left-ventricular ejection fraction (LVEF) data in Figure 3.7b, estimated by the tree learned with C4.5 on the non-noisy data in Figure 3.7a.

		Hard evaluation		SE	
	Patient	$\hat{p}(y=0\|\mathbf{x})$	$\hat{p}(y=1\|\mathbf{x})$	$\hat{p}(y=0\|\mathbf{x})$	$\hat{p}(y=1\|\mathbf{x})$
False negative	6	1.00	0.00	0.54	0.46
	33	1.00	0.00	0.83	0.17
	269	1.00	0.00	0.67	0.33
	280	1.00	0.00	0.91	0.09
	295	1.00	0.00	0.51	0.49
	309	1.00	0.00	0.79	0.21
False positive	42	0.00	1.00	0.16	0.84
	116	0.00	1.00	0.33	0.67
	167	0.00	1.00	0.39	0.61
	228	0.00	1.00	0.30	0.70
	359	0.00	1.00	0.49	0.51
	478	0.00	1.00	0.30	0.70

3.4.2. Experimental results

The average number of leaves, test accuracy and running time are computed over 30 train-test permutations, for each experiment and dataset. The absolute difference to C4.5, averaged over all datasets, is displayed in Tables 3.4 and 3.5.

Since absolute results of distinct datasets should not be directly compared [40], we use standardized metrics. The results of each dataset and method were standardized by the dataset's baseline result. The baseline is obtained by C4.5 without added noise, and estimated with the 30 permutations. The standardization consists in subtracting the baseline mean, and dividing by the baseline standard deviation. E.g. the baseline of the

Table 3.4: Results for Experiment 1, where noise was added to the training data. Standardized metrics and absolute difference to C4.5 are presented, averaged over all datasets. The standardized results are tested for statistically significant differences according to a Wilcoxon signed-ranks test [185], where each result is tested against C4.5 with the same n_{train}. Standardized results with $p < .001$ highlighted in bold font.

Metric	Method	Training data noise factor (n_{train}) 0.00	0.05	0.10	0.20	0.30	0.40	0.50
No. leaves standardized by C4.5 with $n_{train} = 0$	C4.5	0.00	0.44	0.72	1.09	1.42	1.75	2.03
	SS	-0.48	**-0.78**	**-0.81**	**-0.41**	**-0.01**	**0.40**	**0.75**
	STP	**-1.11**	**-2.19**	**-2.06**	**-1.95**	**-0.90**	**-1.01**	**-0.16**
	SE	0.00	0.44	0.72	1.09	1.42	1.75	2.03
	PLT	0.00	0.44	0.72	1.09	1.42	1.75	2.03
	UDT	0.46	0.64	0.83	1.25	1.81	1.97	2.17
Absolute diff. in no. leaves to C4. with same n_{train}	SS	-2.9	-5.1	-5.6	-5.4	-5.4	-5.3	-5.0
	STP	-5.4	-8.4	-8.6	-9.3	-8.9	-9.3	-8.7
	SE	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	PLT	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	UDT	1.5	0.4	0.6	0.7	1.1	0.4	0.2
Accuracy standardized by C4.5 with $n_{train} = 0$	C4.5	0.00	-0.52	-0.87	-1.27	-1.59	-1.65	-2.06
	SS	0.26	-0.14	-0.39	-0.79	-1.05	-1.23	-1.40
	STP	0.32	-0.03	-0.30	-0.70	-0.86	-1.06	-1.24
	SE	-0.60	-0.57	-0.87	-1.27	-1.59	-1.81	-1.94
	PLT	-0.51	-0.67	-1.01	-1.38	-1.68	-1.75	-2.14
	UDT	0.09	-0.45	-0.72	-1.10	-1.36	-1.70	-1.81
Absolute diff. in accuracy (%) to C4. with same n_{train}	SS	0.8	1.2	1.7	1.6	1.8	1.3	2.1
	STP	0.9	1.6	2.0	1.9	2.3	1.9	2.6
	SE	-2.0	-0.3	0.0	-0.1	0.0	-0.5	0.3
	PLT	-1.2	-0.5	-0.4	-0.3	-0.2	-0.2	-0.2
	UDT	0.3	0.1	0.5	0.4	0.6	-0.3	0.7
Absolute diff. running time (s) to C4. with same n_{train}	SS	2.7	5.6	4.5	4.9	5.1	4.3	5.2
	STP	1.0	1.2	1.4	1.9	1.9	1.5	1.5
	SE	0.0	0.1	-0.1	0.1	0.0	0.1	0.2
	PLT	0.1	0.2	0.2	0.2	0.2	0.2	0.2
	UDT	40.6	45.0	37.6	49.0	53.9	37.3	44.2

Table 3.5: Results for Experiment 2, where noise was added to the test data. Standardized metrics and absolute difference to C4.5 are presented, averaged over all datasets. The standardized results are tested for statistically significant differences according to a Wilcoxon signed-ranks test [185], where each result is tested against C4.5 with the same n_{test}. Standardized results with $p < .001$ highlighted in bold font.

Metric	Method	Test data noise factor (n_{test}) 0.00	0.05	0.10	0.20	0.30	0.40	0.50
No. leaves standardized by C4.5 with $n_{test} = 0$	C4.5	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SS	-0.48	**-1.10**	**-1.07**	**-0.81**	**-0.93**	**-0.81**	**-0.77**
	STP	**-1.11**	**-1.73**	**-1.89**	**-2.05**	**-1.90**	**-2.16**	**-1.98**
	SE	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	PLT	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	UDT	0.46	0.62	0.49	0.45	0.47	0.33	0.49
Absolute diff. in no. leaves to C4. with same n_{test}	SS	-2.9	-4.3	-4.3	-3.4	-4.0	-3.5	-3.4
	STP	-5.4	-7.0	-7.3	-7.7	-7.5	-7.7	-7.5
	SE	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	PLT	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	UDT	1.5	1.7	1.3	1.2	0.9	0.9	1.5
Accuracy standardized by C4.5 with $n_{test} = 0$	C4.5	0.00	-0.97	-1.38	-2.01	-2.59	-3.06	-3.50
	SS	0.26	-0.50	-0.82	-1.45	-1.85	-2.27	-2.62
	STP	0.32	-0.40	-0.72	-1.18	-1.54	-1.91	-2.15
	SE	-0.60	-0.85	-1.19	-1.79	-2.32	-2.78	-3.19
	PLT	-0.51	-1.16	-1.60	-2.28	-2.90	-3.43	-3.86
	UDT	0.09	-0.64	-1.02	-1.74	-2.39	-2.94	-3.46
Absolute diff. in accuracy (%) to C4. with same n_{test}	SS	0.8	1.5	2.0	1.8	2.2	2.4	2.7
	STP	0.9	1.7	2.1	2.5	3.2	3.4	4.1
	SE	-2.0	-0.2	0.0	-0.1	-0.1	-0.2	-0.1
	PLT	-1.2	-0.5	-0.5	-0.6	-0.6	-0.7	-0.7
	UDT	0.3	0.6	0.8	0.5	0.3	0.1	-0.1
Absolute diff. running time (s) to C4. with same n_{train}	SS	2.05	4.8	6.0	4.8	4.7	3.3	5.29
	STP	1.7	2.8	2.8	3.2	3.7	3.4	3.40
	SE	0.1	0.1	0.1	0.1	0.0	0.0	0.0
	PLT	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	UDT	40.9	37.7	33.9	38.1	39.0	40.3	39.9

Heart disease dataset has mean 15.0 leaves and standard deviation 3.0 leaves. In this case, standardized results 0.0, -1.0 and 1.5 translate into 15.0, 12.0 and 19.5 leaves, respectively. Tables 3.4 and 3.5 display the standardized metrics, averaged over all datasets, and Figures 3.9 and 3.10 show the corresponding boxplots. Computational times are merely indicative, as the experiments were run on a cluster, and the specifications each machine may vary slightly.

All approaches show an increase in DT size and a reduction in test accuracy as the noise factor grows. The decrease in accuracy was sharper when the noise was present in the test data compared to training data, as seen in Table 3.5.

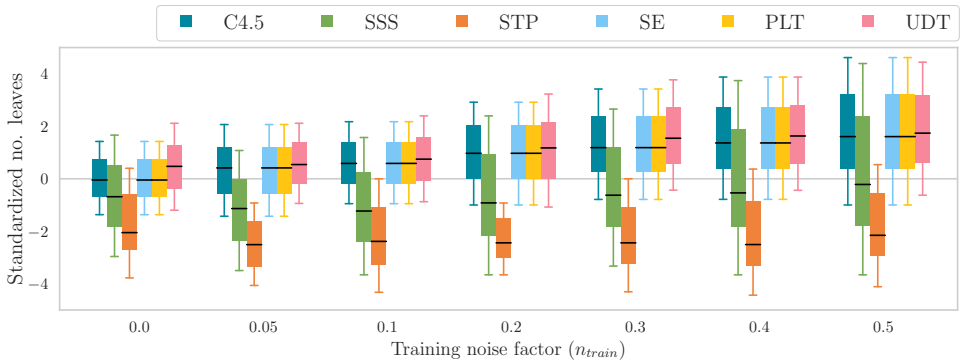
SS, STP and UDT show a maintenance or non-statistically significant improvement in accuracy compared to C4.5, in all noise scenarios. In Table 3.4, we see that STP had higher accuracy compared to the other methods for all n_{train} and n_{test}. For SS, the average absolute increases in accuracy ranged from 0.8 to 2.1% in Experiment 1, and from 0.8 to 2.7% in Experiment 2, with growing noise levels. When using STP, these improvements ranged from 0.9 to 2.6% and from 0.9 to 4.1%, in Experiments 1 and 2 respectively. For the SS and STP approaches, the maintenance of accuracy was accompanied by statistically significant reductions in the number of leaves compared to C4.5 and UDT. The SS tree size reduction was statistically significant for noise factors greater than 0.00. STP had a further reduction in tree size, significant for all n_{train} and n_{test}. The maintenance of accuracy by UDT compared to C4.5 was accompanied by an increase in the number of leaves. The method was considerably slower than SS and STP.

In Experiment 1, the accuracy obtained by SE and PLT were equal or smaller than those obtained through hard evaluation, as seen in Table 3.4. This is particularly evident when $n_{train} = 0$. The number of leaves remains unchanged as these methods do not affect training.

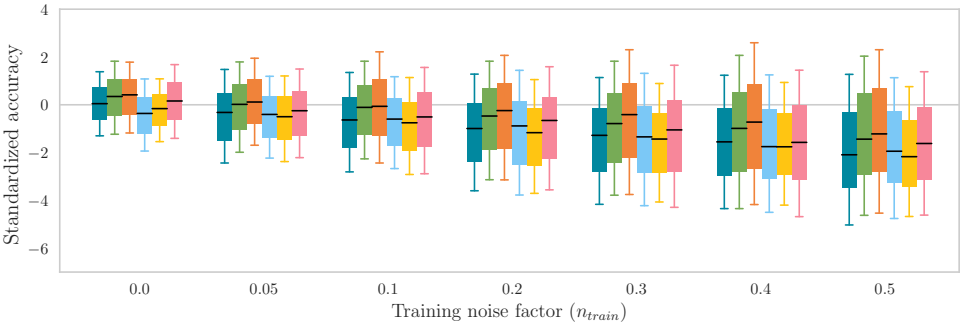
On the contrary, Table 3.5 shows that SE accuracy was greater than that of hard evaluation in for $n_{test} \geq 0.05$. However, this increase was not statistically significant. It suggests that the uncertainty distribution considered in the SE approach does better at capturing the noise added to the data, compared to PLT.

Changing the pruning confidence factor

C4.5 uses the error-based pruning (EBP) method of section 2.2.3, controlled by the confidence factor parameter, c. As a complement to the results, we investigate if the reduction DT size obtained with the soft

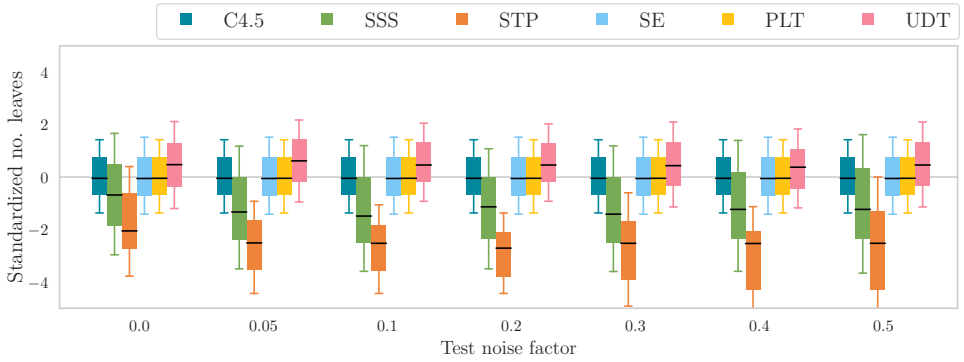


(a) Standardized number of leaves of Experiment 1 - training data noise.

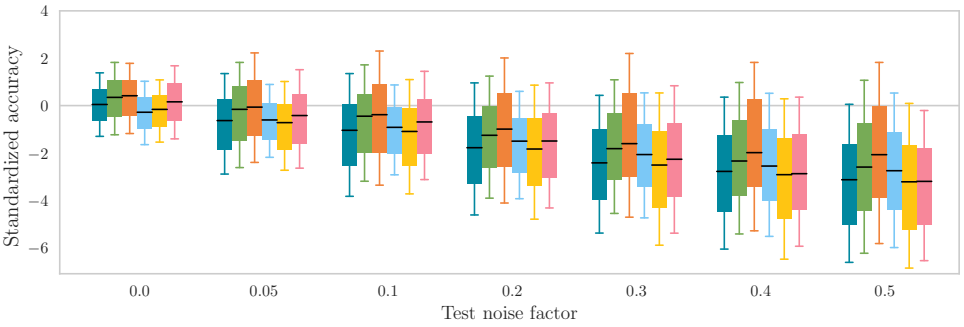


(b) Standardized test accuracy (%) of of Experiment 1 - training data noise.

Figure 3.9: Results of Experiment 1 displayed as boxplots of the standardized metrics for all datasets. Models trained on data with increasing levels of noise $\sim \mathcal{N}(0, n_{train}\bar{x})$, and evaluated on data without added noise. The baseline was obtained with C4.5. Method name abbreviations: SSS - soft split search, STP - soft training propagation, SE - soft evaluation, PLT - PLT - piecewise-linear thresholds, and UDT - uncertain decision tree. The number of leaves obtained with C4.5, SE and PLT is the same.

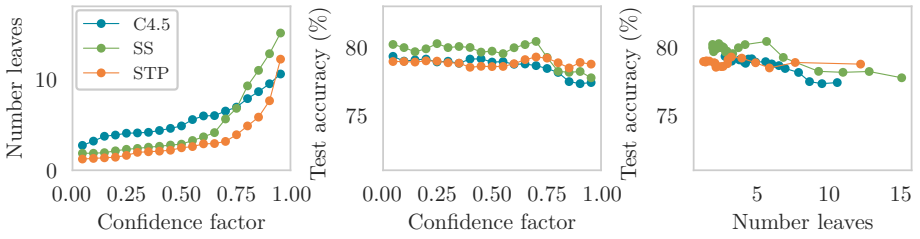


(a) Standardized number of leaves of Experiment 2 - test data noise.

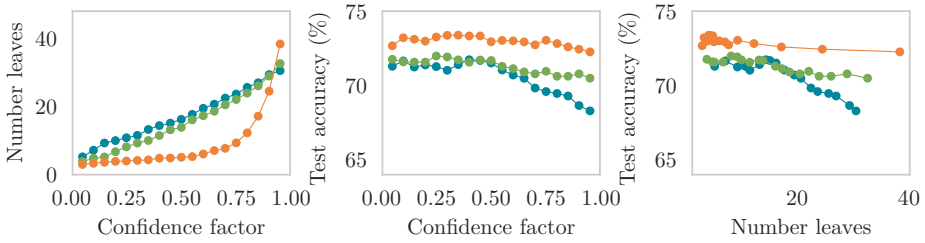


(b) Standardized test accuracy (%) of Experiment 2 - test data noise.

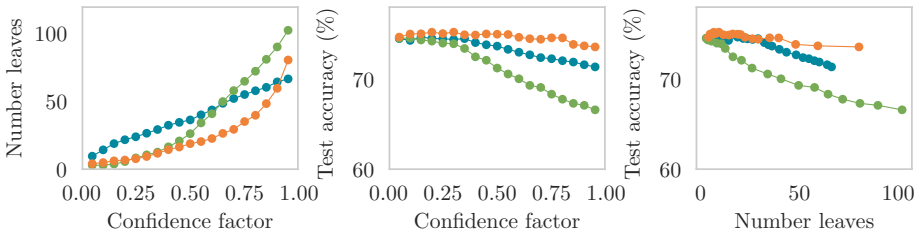
Figure 3.10: Results of Experiment 2 displayed as boxplots of the standardized metrics for all datasets. Models trained on data without added noise, and evaluated on data with noise $\sim \mathcal{N}(0, n_{test}\bar{x})$. The baseline was obtained with C4.5. Method name abbreviations: SSS - soft split search, STP - soft training propagation, SE - soft evaluation, PLT - piecewise-linear thresholds, and UDT - uncertain decision tree. The number of leaves obtained with C4.5, SE and PLT is the same.



(a) Hepatitis dataset

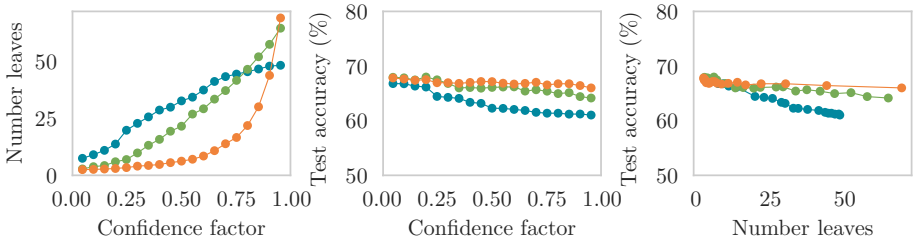


(b) Heart disease dataset

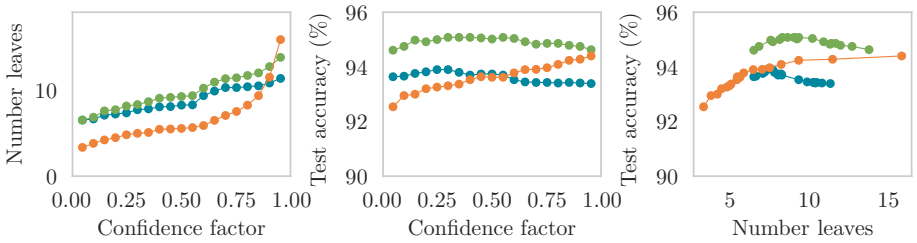


(c) Pima Indians diabetes dataset

Figure 3.11: Illustration on five datasets of the effect of varying the pruning confidence factor using C4.5 and soft search (SS) with $u_s = 0.1$, with training noise factor $n_{train} = 0$.

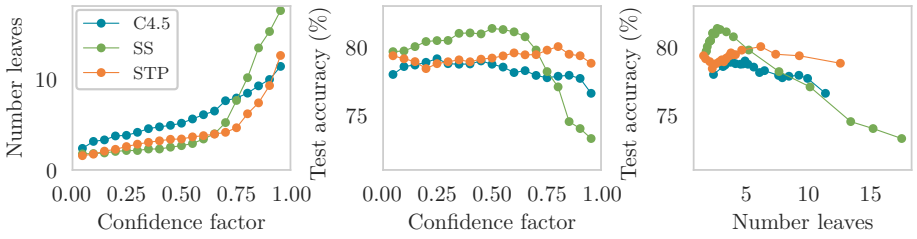


(d) South African heart disease dataset

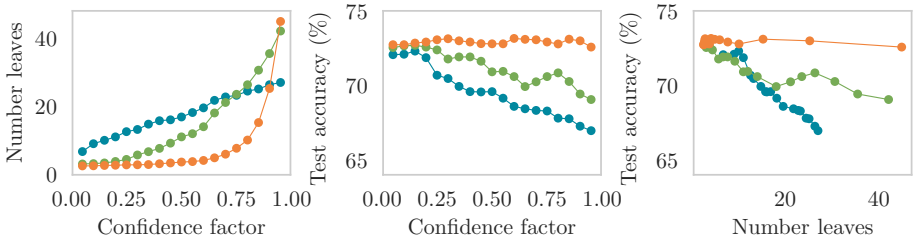


(e) Breast Cancer Wisconsin dataset

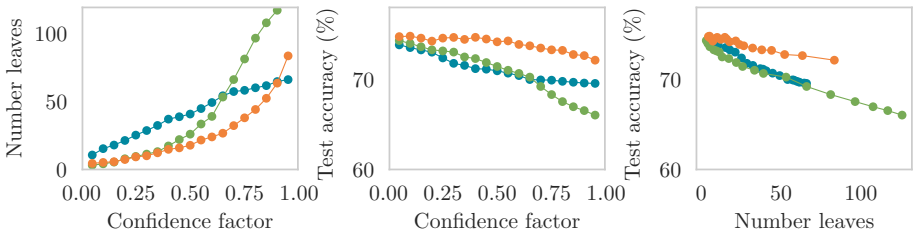
Figure 3.11: (cont.) Illustration on five datasets of the effect of varying the pruning confidence factor using C4.5 and soft search (SS) with $u_s = 0.1$, with training noise factor $n_{train} = 0$.



(a) Hepatitis dataset

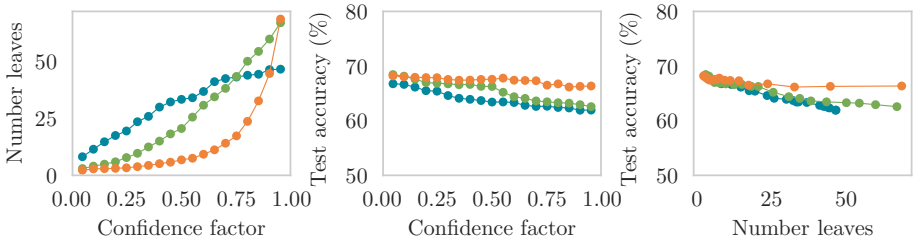


(b) Heart disease dataset

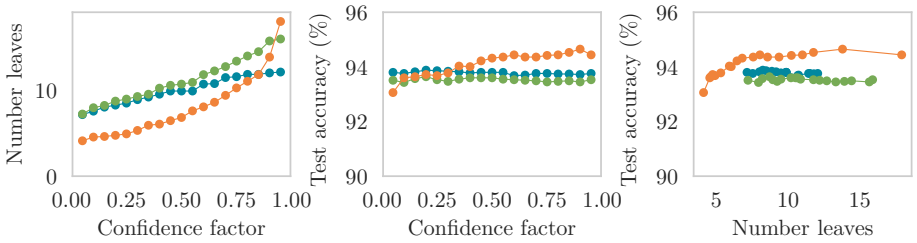


(c) Pima Indians diabetes dataset

Figure 3.12: Illustration on five datasets of the effect of varying the pruning confidence factor using C4.5 and for soft search (SS) with $u_s = 0.1$, with training noise factor $n_{train} = 0.1$.



(d) South African heart disease dataset



(e) Breast Cancer Wisconsin dataset

Figure 3.12: (cont.) Illustration on five datasets of the effect of varying the pruning confidence factor using C4.5 and for soft search (SS) with $u_s = 0.1$, with training noise factor $n_{train} = 0.1$.

training approaches could have been achieved by employing a different value of c. Toward this aim, we evaluated the models learned using C4.5, SS or STP for $c \in [0, 1]$ with two distinct noise levels: $n_{train} = 0$ and $n_{train} = 0.1$. The results can be seen in Figures 3.11 and 3.12 for the first 5 datasets of Table 3.1. Both in the case with no-added noise and with $n = 0.1$, the soft approaches led to smaller trees compared to C4.5, specially for the lower range of c. The differences in accuracy vary between the datasets. In general, the soft methods have comparable accuracy compared to C4.5.

3.5. Discussion

We proposed a probabilistic DT approach to handle uncertain data, which separates the uncertainty model in three independent algorithm components. Our experiments evaluate these components in their ability to handle varying degrees of noise in the training and test data.

The first observation is that corrupting the data decreases the accuracy of the predictions, specially if the noise is in the test data. This is consistent with preliminary observation by Quinlan for one dataset [161]. The probability of learning an accurate model from a noisy dataset is greater than the probability of generating correct predictions for individually-noisy instances, as the noise in the training data is likely to average out. Accordingly, learning on data with increasing noise results in larger DTs, as the models attempt to learn the particularities of the training set.

The results indicate that SS, STP and UDT are at least as robust to noise as C4.5, with non-significant improvements in accuracy. This was observed both for training or test data noise. UDT results are consistent with previous experiments with accuracy gains in the order of 3%, but where DT size had not been reported [176].

All soft training methods had longer running times compared to C4.5, the slowest being UDT. When comparing the search approaches, we observe that, by employing the discretization in Equation 3.7, SS increases the set of possible thresholds compared to C4.5, while preventing the computation of the information gain for values $x^{(i)}$ that are very close. UDT generates s samples for each measurement. The number of entropy computations per attribute is bounded by $s|\mathcal{D}|$, and therefore grows with the size of the dataset. Using SS, this number is bounded by $\frac{1}{\delta}(\tau_{max} - \tau_{min})$, and does not grow with $|\mathcal{D}|$.

While maintaining accuracy, SS and STP led to significantly smaller DTs. All approaches built larger trees for increasing n_{train}, as they start to overfit to the noise. SS and STP were able to cope with this by reducing the num-

ber of splits. This can be interpreted as a consequence of having class probability estimates that reflect the uncertainty, illustrated in Figure 3.4a and Table 3.2, on the pruning algorithm.

The EBP pruning approach used in C4.5 performs a statistical test on the training data to make a pessimistic estimate of the generalization error. Using a soft training approach expresses less confidence in the data, causing the pruning algorithm to remove more nodes. Tree size reductions were also observed for the multivariate sigmoid-split approach [92]. However, they were most likely caused by the use of multivariate split functions, which are able to express complex rules more compactly, at the cost of reduced interpretability.

We also investigated if the DT size reduction could have been obtained by adjusting the extent of pruning. When employing more aggressive pruning, SS and STP resulted in smaller models with similar accuracy to C4.5. This tendency was inverted in the overfitting range, i.e. when the output trees are larger. This suggests that the soft approaches lead to poorer estimates of generalization error for nodes with few training instances, compared to using hard splits with the EBP approach.

Given that the uncertainty models in SS, STP and UDT are all Gaussian, an explanation for the disparity in results obtained by the proposed soft training approaches and UDT could be the mismatch between the uncertainty model considered in the algorithms and the distribution of the noise added to the data. In our experiments, the same distribution was used in SS and STP and in the noise model, i.e. the standard deviation was a factor of the variable mean. In UDT, the standard deviation is instead a factor of the range of the data. This is also suggested by the values of the w parameter selected through CV displayed in Figures 3.5 and 3.6, which are smaller than u_s and u_t. Some degree of correlation between the parameters u_s, u_t and the noise factors n_{train}, n_{test} was observed. But no significant correlation was observed between w and n_{train} or n_{test}.

We hypothesize that the soft learning works more effectively as the uncertainty model approximates the real noise distribution. We therefore recommend the use of uncertainty models derived by domain knowledge. In our experiments, noise was simulated as an experimental proof-of-concept. To validate our approach on concrete clinical decision problems, the uncertainty distribution and its parameters should be estimated beforehand for each variable. Such an estimation may be based, for instance, on the meta-analysis of clinical studies and on empirical knowledge.

SE and PLT did not improve accuracy given noisy training data, compared to the standard hard split approach. When noise was added to

the test data, SE led to non-significant increases in accuracy. This suggests that modeling uncertainty to target training data noise is only effective when this model is incorporated in the training phase, and not during evaluation. As such, we do not recommend the use of soft evaluation to handle training noise.

The disparity between the SE and the PLT results may also be explained by the consistency between the uncertainty model considered by the algorithms, and the model used to corrupt the data. PLT had demonstrated 2-3% error rate reductions on a previous experiment using a single dataset, where the shape of the uncertainty model had been optimized.

3.6. Conclusions

This chapter presents a probabilistic DT learning approach to handle the uncertainty in the data, where the uncertainty model is separated in three independent algorithm components. The background context is to provide interpretable models for clinical decision support, with the motivation that the acknowledgment of uncertainty will facilitate the adoption of automated learning approaches in practice.

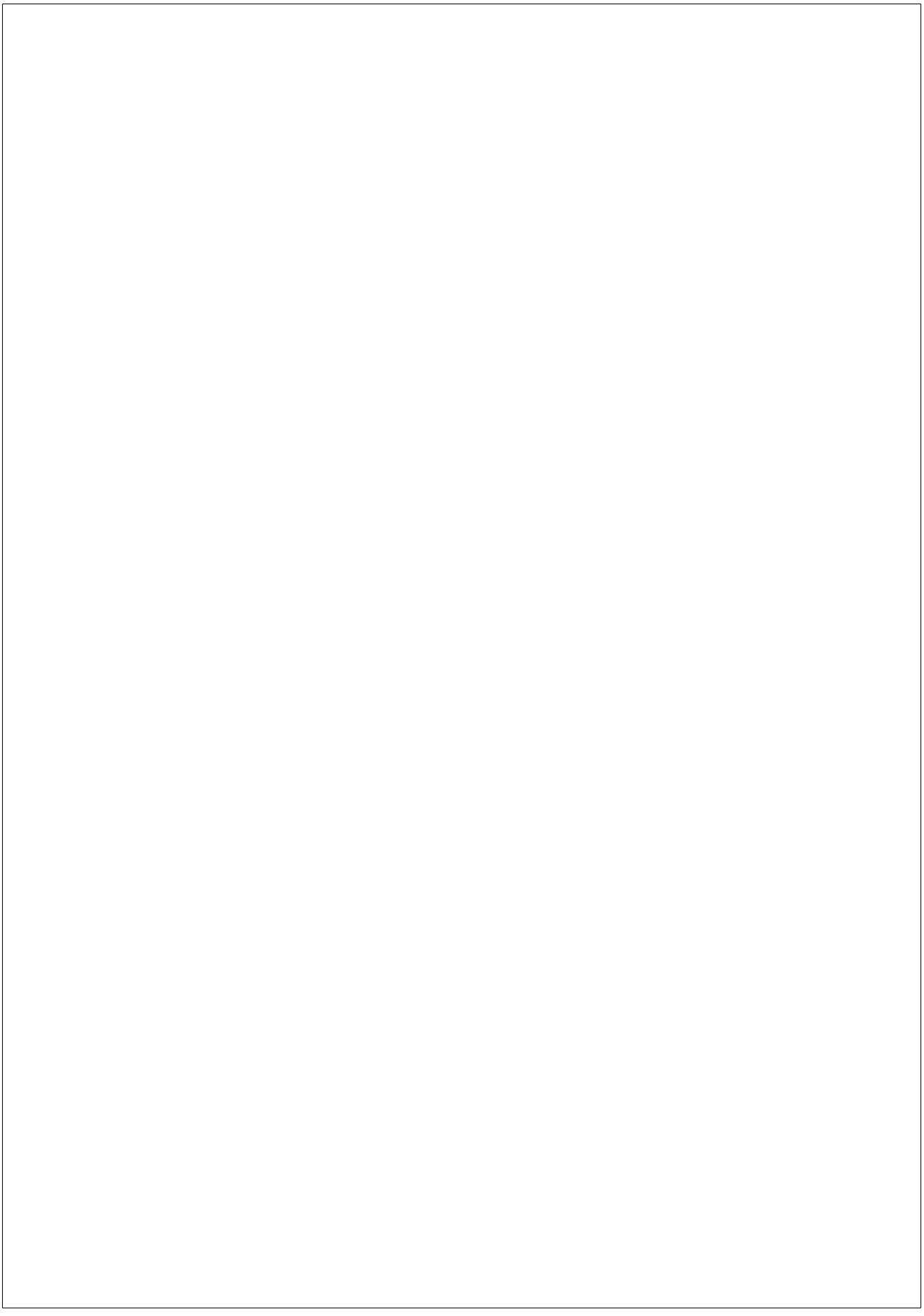
Previous DT algorithms suggest the potential of probabilistic approaches to improve prediction robustness, and the need for an evaluation on more datasets and levels of noise. The impact of considering an uncertainty model in the learning phase or during evaluation had not been evaluated, as well as the effect of having noise is the training examples or the test examples.

In our approach, the uncertainty representation is incorporated: in the learning phase when searching for the optimal thresholds (SS), when propagating the training data through the tree (STP), and in the evaluation phase when obtaining predictions for unseen data (SE). Any model can be chosen to capture the uncertainty. Our purpose is to incorporate clinical knowledge about the reliability of measurements. But as a proof-of-concept, we model the uncertainty as normally-distributed noise.

According to the experiments, corrupting the test data seems to have a more severe impact on accuracy than adding noise to the training data. Upon increased noise, the soft training components, SS, STP and UDT, show maintained or improved accuracy compared to C4.5. STP and SS produced significant reductions in tree size, with STP outperforming the latter. This was not the case of UDT, possibly given the disparity between the noise model in the data and the uncertainty model in the algorithm. The running times of SS and STP were lower than those of UDT. None of the soft evaluation approaches shows significant benefit compared to hard

evaluation. Overall, we recommend using SS and STP with an uncertainty model that approximates as much as possible the real noise in the data. Finally, our study shows the importance of the acknowledgement of data uncertainty when learning decision models. Ideally, when designing clinical studies, an assessment of the reliability of each measurement should be considered part of the database.

Future work directions include testing the combined use of the distinct soft approaches, a limitation in the current analysis. The reported findings are also not complete without an evaluation of the soft approaches with domain-specific noise distributions, and a study of the conditions under which the soft training provides benefit, namely regarding the complexity of the data. For highly separable data, the benefit of any soft approach is expected to be limited.



Chapter 4

MONTE CARLO TREE SEARCH FOR LEARNING DECISION TREES

Standard decision tree (DT) learning algorithms follow a locally-optimal approach that trades off prediction performance for computational efficiency. Such methods can however be far from optimal, and it may pay off to spend more computational resources to increase performance. Monte Carlo tree search (MCTS) is an approach to approximate optimal choices in exponentially large search spaces. We propose a DT learning approach based on the Upper Confidence Bound for Trees (UCT) algorithm, including procedures to expand and explore the space of DTs. To mitigate the computational cost of our method, we employ search pruning strategies that discard some branches of the search tree. The experiments show that proposed approach outperformed the C4.5 algorithm in 20 out of 31 datasets, with statistically significant improvements in the trade-off between prediction performance and DT complexity. The approach improved locally-optimal search for datasets with more than 1000 instances, or for smaller datasets likely arising from complex distributions.

4.1. Introduction

The previous chapter concentrated on how the uncertainty of the input measurements affects decision trees (DTs), and elaborated on an approach to account for it when learning and evaluating interpretable DTs. This chapter tackles the issue of improving DT performance from a different perspective.

As introduced in Chapter 1, an optimal DT maximizes generalization performance, while minimizing the number of decisions needed for a prediction. Learning an optimal DT is NP-complete, owing to the discrete and sequential nature of the splits [90]. Standard DT algorithms follow the greedy strategy explained in Chapter 2, providing good performance at low computational cost. They are not however guaranteed to yield an optimal DT. The greedy approach is also known to be unstable, reducing the confidence in the output models, regardless of their interpretability [49].

Another relevant aspect is how DTs and locally-optimal learning cope with datasets arising from complex distributions. Some authors point out that the greedy approach may be unable to model complex feature interactions [167]. Others coincide, arguing that DT learning algorithms follow a discriminative approach, focusing only on the relation between target and inputs [86]. On the other hand, other authors have hypothesized that thanks to their hierarchical structure, trees are able to express some degree of interaction [51]. Another dataset property that can be challenging for DT learning is that of distributions where the target variable cannot be easily approximated by a function of a small number of features, i.e. datasets with many *weak* inputs [17].

Several methods have been proposed to learn DTs through global optimization. The approaches either use multivariate test functions, which limits interpretability [137], or they impose a DT structure [8]. Evolutionary DT learning showed improvements in performance and model expressiveness [5], suggesting the potential to improve upon locally-optimal learning.

Owing to their efficiency, locally-optimal DT approaches are often employed as base learners in ensemble methods [168, 18]. Ensembles improve the performance and stability of a base learner, by optimizing the combined predictions of several models built with it. Tree ensembles like random forests [18] or gradient boosting machines [64] are considered the state-of-the-art in many applications. These techniques perform respectively a broader and more targeted exploration of the space of DTs, reinforcing the potential to improve upon greedy search. Nonetheless, a large ensemble of DT is not easily interpretable.

Monte Carlo tree search (MCTS) is a family of algorithms that estimate

the next best action for a given domain, by taking heuristic samples of the decision space [104, 29]. The approach has had success in search problems with large branching factors [22]. To the best of our knowledge, MCTS has not been employed for learning prediction models, such as DTs.

The present chapter describes a novel non-greedy DT learning approach that uses MCTS to perform a controlled exploration of the space of DTs. The method is a single-player adaptation of the Upper Confidence Bound for Trees (UCT) algorithm [104], which we call UCT-DT, with specific procedures for generating candidate DTs and estimating their performance. Furthermore, we propose search pruning approaches to discard redundant branches of the search tree. Unlike previous uses of MCTS, UCT-DT outputs the search space, allowing the user to select the DT that best fits the application. We focus on the performance of a single DT, as opposed to an ensemble of trees, targeting applications in which interpretability is crucial.

Before explaining the UCT-DT algorithm, the following paragraphs introduce some concepts related to global DT optimization and MCTS. An empirical evaluation then studies the components of the proposed approach and how it compares to locally-optimal and evolutionary learning. The results are complemented by a discussion about how certain properties of the data impact the DT learning procedure, in its myopic and look-ahead counterparts.

4.2. Alternatives to locally-optimal learning

Several non-greedy approaches exist for learning DTs. Norouzi et al. [137] proposed an algorithm for globally optimizing the test functions at each node, as well as the class distribution parameters at the leaves. The approach however assumes a linear function of all the input variables at each node, which can be less interpretable compared to univariate DTs.

Evolutionary approaches have also been proposed to build DTs non-greedily, as summarized in the survey by Barros et al. [5]. The methods achieved improvements in prediction performance and in capturing attribute interactions, compared to locally-optimal learning [62]. Evolutionary DT methods can be categorized as: (1) methods that consider DTs as individuals and perform evolutionary induction, and (2) methods that apply evolutionary design to DT components. The former is known as tree-encoding, and has been considered the most flexible representation, as it allows variable-sized DTs [141].

One of such tree-encoding approaches [106], which we refer to as evolutionary tree-encoding (ETE), showed median improvements over C4.5 in

the order of 1% in accuracy with a median 30-leaf reduction in DT size. The method first generates an initial population of trees, based on the best local test functions for the dataset. The proposed operators for crossover and mutation then randomly shuffle the functions at different nodes. The chosen evolutionary fitness function sets a trade-off between accuracy and number of leaves for each DT, using the α parameter:

$$fitness(DT) = accuracy(DT) - \alpha \cdot size(DT)$$

ETE attempts to maximize the fitness of each DT population. Larger values of α lead to smaller trees, and the parameter needs to be tuned for each dataset.

4.3. Monte Carlo tree search

Monte Carlo tree search (MCTS) is a heuristic search algorithm that learns sequences of decisions for domains usually modeled as Markov decision processes (MDPs) [104, 22]. The method has had success in problems with high branching factors, and its benefit is best realized when adapted to suit the target domain [29]. Successes include playing Atari games such as *Ms Pac-Man* [68], and *Total War: Rome II* [28]. The most notable accomplishment of MCTS is that of computer *Go*, where the method was able to beat one of the best human players [173]. *Go* is a board game with a large branching factor and long-range spatiotemporal interactions, which makes it unfeasible for other strategies like alpha-beta pruning to succeed at beating a professional player. Before describing the MCTS algorithm, let us introduce Markov decision processs (MDPs).

4.3.1. Markov decision processes

The problem of teaching an agent how to behave in an unknown environment is often modeled as an MDP. An MDP is a Markovian process of *state* variables, where the probabilities of transition between states are conditioned on *actions*. It is composed by:

- A finite set of states, $\mathcal{S}$.
- A finite set of actions, $\mathcal{A}$.
- State-transition probabilities, $P[S_{t+1} = s' | S_t = s, A_t = a]$, with $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

- A reward function associated to each state and action, $R(s, a)$.
- A discount factor, $\gamma \in [0, 1]$.

Given a sequence of states and rewards, the return G_t at timestep t is the sum of the rewards obtained at each state transition after t. The γ factor penalizes transitions that take place later:

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$

For a given MDP and state s, we are interested in estimating how good it is to be in that state with respect to the long-term reward. This is given by the *value function*, $v(s)$, as the expected return of starting in state s:

$$v(s) = \mathbb{E}[G_t | S_t = s] \tag{4.1}$$

Finally, the behavior of the agent is modeled by a *policy* $\pi(a|s)$, which is a distribution of the actions given the states:

$$\pi(a|s) = P[A_t = a | S_t = s],$$

and solving an MDP corresponds to finding an optimal policy that maximizes the expected return.

4.3.2. Algorithm overview

For a given MDP, MCTS iteratively builds a partial search tree, where each node represents a state and each branch denotes an action. The method uses Monte Carlo simulations to estimate the value of each node, which are later employed to guide the search towards the most promising nodes. Each MCTS iteration is composed by four phases, illustrated in Figure 4.2: *selection*, *expansion*, *simulation* and *backpropagation*. The procedures for selection and expansion define how the search tree is explored. During simulation and backpropagation, estimates of the value of the states are obtained and updated throughout the search tree.

Selection and expansion At the start of an iteration, the selection policy is applied starting at the root, in order to choose the most promising node to visit. The policy should balance out the *exploration* of unseen states with the *exploitation* of known states. One approach to achieving this is that of the UCT algorithm [104]. Let us consider the multi-armed bandit problem, which models the sequential choice of actions with unknown reward, in order to maximize the long-term reward of consistently selecting

the best action. In bandit problems, the regret is the loss incurred by the policy not always choosing the best action. For a given policy, we are interested in finding an Upper Confidence Bound (UCB) on the probability that its regret will be lower than a given value. The UCB1 policy solves the bandit exploration-exploitation dilemma by selecting the action that maximizes the aforementioned upper bound [3]. Based on this result, the UCT algorithm [104] handles the MCTS exploitation-exploration dilemma by treating node selection as a bandit problem, and employing UCB1 as a selection policy. The extent of exploration is controlled by the C_p parameter, $C_p > 0$, where larger values favor a more exploratory search.

After selection, the node is returned if it represents a terminal state. Otherwise, the node gets *expanded* by taking an action out of the set of available actions, creating and returning a new node with a new state.

Simulation and backpropagation A *simulation* or *rollout* is then run from the returned node according to a *simulation policy*, returning an estimate of its value. Monte Carlo methods only require a black-box simulator, avoiding the need for an explicit probability representation. The estimate is backpropagated to the ancestor nodes, updating their value statistics. MCTS is an anytime algorithm, and so the number of iterations is set based on the available time and computation resources. After the iterations are completed, the estimated best action for the root state is returned.

4.4. Proposed approach

We propose an approach to learn DTs based on MCTS – specifically, the UCT algorithm. The method takes as input a *training* dataset, divided into an *induction* set and a *validation* set. We start by modeling DT learning as an MDP, such that each node of the search tree contains a DT and each branch corresponds to the addition of a test function (Section 4.4.1). We then define the approaches for expansion and simulation. During expansion, promising candidate actions are generated based on their information gain computed on the induction set, as presented in Section 4.4.2. The simulation phase estimates how good a given state is, based on the DT prediction performance on the validation set, as described in Section 4.4.3. Selection and backpropagation are performed as in the UCT algorithm. Section 4.4.4 presents strategies for tackling the exponential growth of the search. Figure 4.2 displays an overview of the approach, which we call UCT-DT.

4.4.1. Decision tree learning as a Markov decision process

As before, we consider DT with univariate test functions, denoted $t(\mathbf{x})$. For continuous variables, they are defined as in Equation 2.1. DT learning can be approximated by the Markov decision process (MDP) where each state s represents a DT, and each action a represents the addition of a test $t(\mathbf{x})$ to leaf ℓ, $a = \big(\ell, t(\mathbf{x})\big)$. This is illustrated in Figure 4.1. The reward $r(s)$ is 0 for all non-terminal states. If s is terminal, it is a finalized DT and $r(s)$ is a measure of its prediction performance. The definition of a terminal state is further elaborated in this section.

As defined in Equation 4.1, the value of state s is the expected long-term reward of starting in s. In our case, the value represents the estimated performance of the finalized DT that we expect to learn, having the DT in s as a starting point. Unlike the previous chapter, we consider the average of the F1 scores computed for each class as a measure of prediction performance. The average F1 is more robust to class imbalances compared to the accuracy, which becomes uninformative when the classes are not equally prevalent [59]. It is computed as:

$$F1 = \frac{PPV \times Sensitivity}{PPV + Sensitivity} + \frac{NPV \times Specificity}{NPV + Specificity},$$

where PPV stands for the *positive predictive value* or *precision*, and NPV stands for *negative predictive value*. In a two-class scenario, the PPV/NPV measure the proportion of positive/negative classifications which were actually correct. The sensitivity, also known as recall or *true positive rate*, and the *specificity*, also known as *true negative rate*, respectively measure the proposition of positive/negative cases were correctly retrieved by the model.

Characterizing the search tree

Using the proposed MDP, DT learning can be cast as a single-player game. The UCT-DT algorithm builds a partial *search tree*, where each node n contains:

- $s(n)$: a state or a DT;
- $a(n)$: the incoming action that generated n from its parent;
- $U(n)$: the set of candidate actions to be applied to $s(n)$. A new action is taken from $U(n)$, every time n is expanded.
- $U_0(n)$: the *initial* set of actions of n. When n is created, $U(n) = U_0(n)$. $U_0(n)$ remains unchanged thereafter, in order to be passed to the children of n.

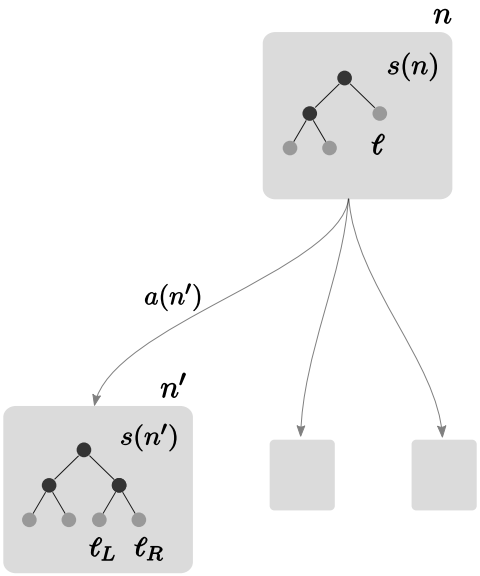


Figure 4.1: Two nodes of the search tree, n and n', with their states, $s(n)$ and $s(n')$. Executing the action $a(n')$ on $s(n)$ adds a split to leaf ℓ, resulting in two new leaves ℓ_L and ℓ_R. The depth of a search node corresponds to the number of upstream actions that led to it, and is equal to the number of splits of its DT. E.g. node n' has search depth equal to 3, and $s(n')$ has 3 internal nodes (in black) and 4 leaves (in gray).

- $N(n)$: the number of simulations started in n or one of its descendants;
- $V(n)$: the total reward of simulations started in n or its descendants. Each simulation returns the expected performance of the simulated final DT. $V(n)$ and $N(n)$ get incremented accordingly, such that the value of the state, $v\big(s(n)\big)$, can be estimated as $\frac{V(n)}{N(n)}$.

The *search tree* is not to be confused with the *decision tree*, present at each search-tree node. DT nodes are named using *calligraphic* font, i.e. ℓ instead of l.

Terminal and expandable states

Defining when a state is terminal allows us to impose restrictions on DT size. Just as in C4.5, we consider a DT node to be terminal if its number of training instances is smaller than $\eta_{instances}$. In addition, we restrict the depth of a DT branch to a maximum η_{depth}. A state $s(n)$ is terminal if all leaves of the its DT are terminal, or if the set of possible actions $U(n)$ is empty.

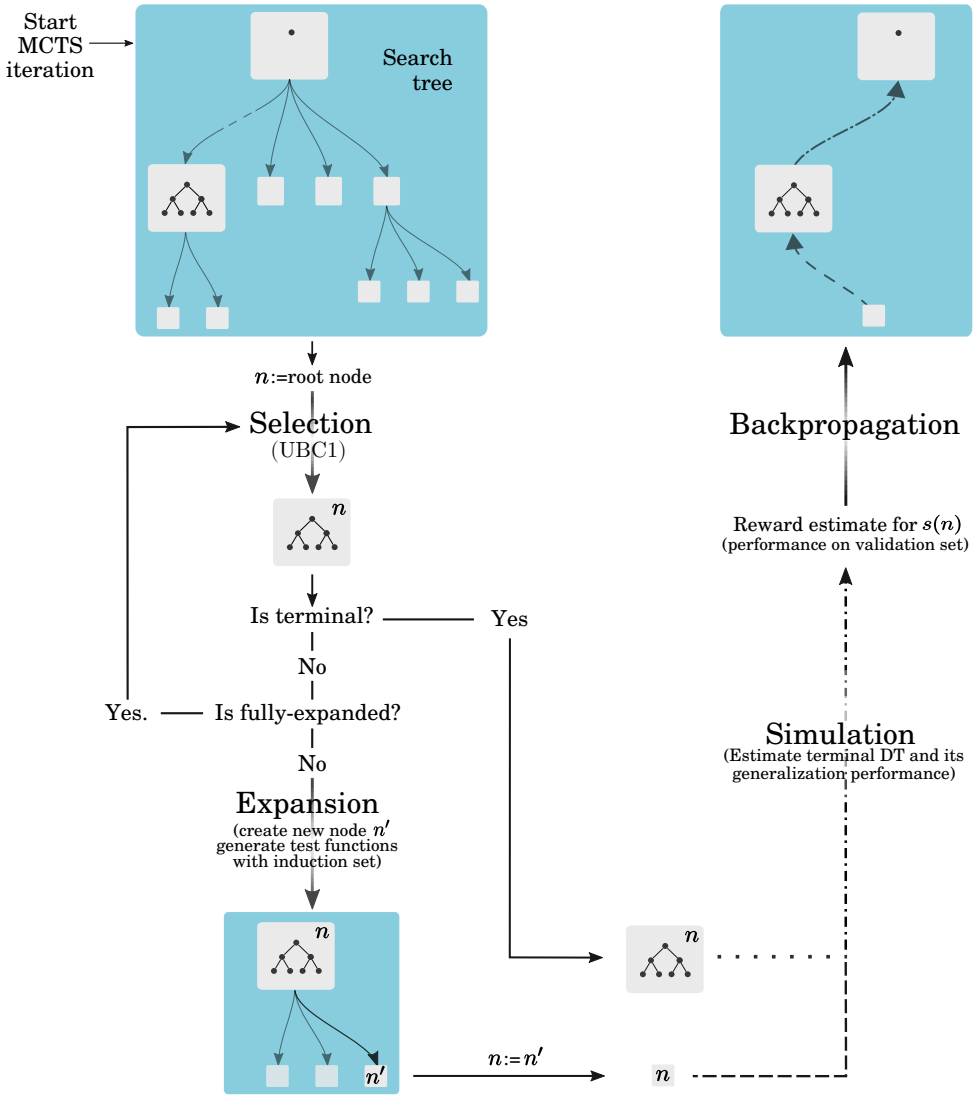


Figure 4.2: Illustration of one iteration of the UCT-DT approach.

Algorithm 2 Functions for the expansion of a node n.

function EXPAND(n, K)
Randomly pop an action $a = (\ell, t(\mathbf{x}))$ from $U(n)$.
Applying a to $s(n)$, creating the new state s' with new leaves ℓ_L and ℓ_R.
Initialize $U' \leftarrow U_0(n)$ except actions involving ℓ
if ℓ_L is splittable **then**
$U' \leftarrow U' \cup$ GETACTIONS(ℓ_L, K)
end if
if ℓ_R is splittable **then**
$U' \leftarrow U' \cup$ GETACTIONS(ℓ_R, K)
end if
Create child node n' with:
$s(n') \leftarrow s'$
$a(n') \leftarrow a$
$U_0(n'), U(n') \leftarrow U'$
$N(n'), Q(n') \leftarrow 0$
return n'
end function

function GETACTIONS(ℓ, K)
$T(\ell) \leftarrow$ top K functions $t(\mathbf{x})$ that maximize the information gain in subset $\mathcal{D}(\ell)$
return The set of new actions $\ell \times T(\ell)$
end function

4.4.2. Expansion

During selection, the most promising node is identified. If the selected node n is non-terminal or does not have enough statistics to apply selection, it gets expanded. As illustrated in Figure 4.1, a node n is expanded by taking an action a from its set of possible actions, $U(n)$. The action is executed in state $s(n)$, creating the new node n' and new state $s(n')$.

Algorithm 2 shows how the set of candidate actions of n', $U(n')$, is constructed. In n' and its descendants, ℓ is no longer a leaf, but an internal DT node with a test, and so no actions involving ℓ can be executed. As such, $U(n')$ is initially set to $U_0(n)$, excluding actions involving ℓ. When applying a, two new leaves ℓ_L and ℓ_R are added to $s(n)$. If ℓ_L and ℓ_R are not terminal, candidate test functions are generated for them. The combination of the new leaves with those functions composes the new candidate actions, which are added to $U(n')$.

Generating test functions

The procedure to generate functions for the new leaves consists in selecting the best functions for the local training sets at ℓ_L and ℓ_R. The method is outlined in Algorithm 2, and requires the specification of the K parameter, which determines how many functions to consider for each new leaf. The MCTS branching factor is thus $2K$ at each new search node, because we generate K actions for each of the two new DT leaves. The effective branching factor will be much lower, because of the search-pruning approaches described in Section 4.4.4. Larger values of K promote a more exploratory approach, while $K = 1$ falls back to locally-optimal search.

The function generation algorithm includes additional mechanisms to allow inheriting functions from the parent DT leaf, as well as to limit the number of functions generated for each input variable. The evaluation of these mechanisms was left out of the scope of this thesis.

4.4.3. Simulation

After selection and expansion, a node n is returned. From the state $s(n)$, a Monte Carlo simulator generates sequences of actions until a terminal state is reached, returning the reward of that state. In our case, the simulator estimates the average F1 of the completed DT we expect to find at the end of the search branch, having the DT in s as a starting point. Toward this aim, we considered several simulation policies:

1. **Validation-set performance (VS):** The simplest approach to estimating the performance of the expected DT is to consider the performance of the DT in state s on the validation set. This subset is not used in the generation of actions. The policy does not in fact perform any simulation.

2. **Complete the DT using C4.5 (C4.5):** This policy consists in completing the DT in n with C4.5 using the induction set, and evaluating it on the validation set. This gives us an estimate of the performance we would get, if we were to complete the DT with locally-optimal learning. As elaborated in Section 4.4.4, C4.5-based tree-completion can optionally be employed with the pessimistic pruning approach mentioned in Section 2.2.3.

3. **Complete the DT using C4.5 with bootstrap sampling (C4.5b):** This policy uses C4.5 as above but with bootstrapping, i.e. sampling the data with replacement. The idea is to use somewhat different data in each simulation, in an attempt to reduce the bias caused by

repeatedly using the same training data. Concretely, we complete the DT with C4.5 considering bootstrap samples of the induction and validation sets.

Since all policies are deterministic, except for C4.5b, a single rollout is done for each visited node. After the simulation, $N(n)$ is incremented 1 and the reward estimate is added to $V(n)$. The statistics are then back-propagated accordingly in the ancestor nodes.

4.4.4. Pruning the search tree

Search pruning techniques tackle the exponential growth of the search tree by removing suboptimal nodes, allowing more time to be employed on better choices [22]. UCT-DT performs checks to remove redundant states. In addition, the algorithm UCT-DT employs two optional search pruning approaches:

- **Decision tree-based pruning:** Pruning the MCTS tree is often done by including knowledge about the target application. DT pruning significantly improves locally-optimal learning, as a result of their approach to identifying unfavorable nodes during training. As a consequence, we expect these methods to be useful in removing bad actions from the search tree. The DT pruning algorithm described in Section 2.2.3 is integrated in UCT-DT, specifically in the simulation policy, as follows. Every time a simulation is done at a node, we check if the incoming action is present in the C4.5-completed DT. If the action is removed, the node is eliminated from the search tree.
- **Value-based pruning:** Since the DT search space can be highly redundant, we also investigate the benefit of a domain-independent search pruning approach based on the value estimates. Every n_{vp} iterations, where vp stands for value-based pruning, of all the search branches with depth greater than d_{vp}, only the branch with highest $\frac{V(n)}{N(n)}$ is kept. All nodes with depth smaller than d_{vp} are kept.

4.4.5. Output and post-search selection

MCTS is usually employed in games to find the next best move for a player, after which the search tree is discarded. Unlike the standard use, UCT-DT outputs the search tree. The performance of the DTs at each node is evaluated on an independent *test* set, unseen during the search. A single DT or a subset of DTs can be selected according to the performance on the validation set, or to specific application constraints.

4.5. Experimental setup

We performed several experiments to evaluate the UCT-DT approach, focusing on the role of its algorithmic components, and on how it compares to state-of-the-art alternatives. As a locally-optimal method we used C4.5 [159]. The employed evolutionary approach was the evolutionary tree-encoding (ETE) algorithm by Kretowski and Grzes [106].

4.5.1. Data and evaluation procedure

The experiments were done on 23 publicly-available and 8 synthetic datasets. Sources include the University of California Irvine (UCI) [112] and KEEL [1] repositories, and the MIMIC-II database [110]. The synthetic data were generated using an adaptation of the method by Guyon [81], as in Section 3.3.1. Each dataset was split into a training set (70%) and a test (30%) set. For UCT-DT, the training data were further divided into induction (70%) and validation (30%) sets. The test data are only used after algorithm completion.

Each UCT-DT experiment returns a search tree, from which we select the DT with best performance on the validation set. We focus on this DT and compute its performance on the test set. Improving performance consists in achieving better predictions with minimum DT complexity. This is assessed with the average F1 score (F1) over all classes and the number of leaves (L), respectively. DTs with a large number of nodes may be hard to interpret and may incur a higher cost of decision. And although minimizing DT size may not be critical in some practical scenarios, it is important to take into account the trade-off between prediction performance and model complexity. As such, we also consider the ratio of F1 per number of leaves (F1/L), as an estimate of the average performance gain achieved by the addition of one test function.

Hyperparameter setup

A cross-validated grid search was done on each training set to tune the DT pruning confidence factor. This parameter is kept constant for each dataset whenever employing the pruning strategy of Section 2.2.3, either in C4.5 or when using DT-based pruning in UCT-DT. We also tuned the α parameter of the ETE approach through cross-validation for each dataset.

Regarding the parameters of the MCTS expansion, the maximum depth of each DT branch, η_{depth}, is set to the maximum depth of the DT built by C4.5 on the training set, with the selected confidence factor. The η_{depth}

is therefore specific to each dataset. The minimum number of instances required for a split is set to $\eta_{instances} = 4$ for all datasets and UCT-DT experiments, based on the values of the reference implementation of C4.5 [163]. The number of candidate functions generated for each new DT leaf is set to $K = 3$ for all datasets and UCT-DT experiments.

It is unfeasible to optimize all the parameters of MCTS, as for many supervised learning algorithms, and so they are typically adjusted manually [22]. We therefore set the parameters to reasonable values, following several initial attempts. Specifically, the exploration-exploitation constant C_p was set to 1, and each experiment had a budget of 2×10^5 iterations. When employed, value-based pruning was performed every $n_{vp} = 3 \times 10^3$ iterations, at depth $d_{vp} = 5$. These parameters were the same in all experiments.

4.5.2. Effect of dataset properties algorithm behavior

The experiments are complemented by an analysis of how several dataset properties may determine the suitability of a locally-optimal or MCTS approach. Concretely, we investigate the effect of dataset size, the existence of strong feature interactions and whether the class variable can be well approximated by a small number of features.

To assess the importance of feature interactions when modeling the class, we compared the F1 performance obtained by a naïve Bayes (NB) classifier with the average F1 performance of a set of more expressive learners: k-nearest neighbors, linear and Gaussian-kernel support vector machines, Random forests, AdaBoost and logistic regression. If the NB model performs better, this means that its independence assumptions are suitable. In that case, we can infer that either the features have few significant interactions, or there is not enough data to allow modeling existing interactions.

In order to investigate if the target variable can be expressed by a small number of features, we computed the *importance* of each feature, using the random forest feature-importance procedure [18]. Subsequently, the number of variables with importance score (IS) above the median IS, and its proportion to the total number of variables, were reported.

4.6. Results and discussion

The results are presented in four parts. Section 4.6.1 focuses on the behavior of the search pruning approaches (Section 4.4.4), followed by

a comparison between distinct simulation policies (Section 4.6.2). Section 4.6.3 compares the best-performing variations of UCT-DT with the C4.5 and ETE algorithms, and Section 4.6.4 analyzes data properties that are likely to influence the performance of UCT-DT.

4.6.1. Impact of search-tree pruning

Table 4.1 displays the results of UCT-DT with the VS policy, focusing on the impact of search pruning. The differences in F1, number of leaves, and F1-leaf ratio among all pruning variations were statistically significant according to one-way within-subjects analysis of variance (ANOVA) employed on the normalized metrics ($p < 0.05$), after sufficiently confirmed normality assumptions[1]. The complementary Wilcoxon signed-ranks pairwise tests [185] are reported in Table 4.2.

UCT-DT without pruning had the highest F1 in only 10 out of the 31 datasets, as depicted in Table 4.1. In Figures 4.3a and 4.3c, we see that in most of the cases, the maximum search depth reached by MCTS was considerably smaller than the equivalent search depth of the C4.5 solution. We hypothesize that a better solution is likely to require more leaves, and therefore lie in a deeper state of the search tree. This is assuming that the number of leaves of a good solution is of the same order of magnitude as the locally-optimal one.

A deeper search was achieved by using value-based pruning, as illustrated in Figures 4.3b and 4.3d. Value-based pruning led to the best F1 in 14 of the datasets, compared to 10 without any pruning (see last row of Table 4.1). Similarly, combining value-based with DT-based pruning had an improvement in F1 from 7 to 16 wins, compared to using only the latter. Table 4.2 shows that the gain in F1 obtained by value-based pruning was statistically significant. This indicates that the method allowed more resources to be spent on the exploration of more promising states.

The effectiveness of value-based pruning seems to be related to dataset size. Figure 4.4 shows that the method was unable to improve F1 compared to C4.5 for datasets smaller than 730 instances. Possibly, the induction set was too small to learn generalizable test functions. It is also possible that the validation set was not large enough for the simulations to provide good estimates of DT performance, used in value-based pruning.

[1] Despite reluctance in the use of ANOVA in classification results [40], statisticians have argued that the test is more informative than its non-parametric alternative – the Friedman test – even if some of the normality assumptions are violated [172].

Table 4.1: Evaluation of search pruning methods, employed with the UCT-DT using the validation-set performance (VS) policy. The pruning variations are: (1) no pruning, (2) value-based pruning, (3) DT-based pruning, and (4) both. Metrics are average F1 over all classes (F1), number of DT leaves (L), and their ratio (F1/L). We also display the maximum depth reached by the search tree (D). The best result is highlighted in bold font, and the last row counts the number of times a variation provided the best result.

Dataset	(1) no pruning				(2) value-based pruning				(3) DT-based pruning				(4) both approaches			
	F1	L	F1/L	D	F1	L	F1/L	D	F1	L	F1/L	D	F1	L	F1/L	D
Balance Scale	**87.4**	12	7.3	13	**87.4**	29	3.0	43	**87.4**	12	7.3	13	84.4	**7**	**12.1**	35
Banknote auth.	**98.0**	**8**	**12.3**	11	**98.0**	**8**	**12.3**	21	**98.0**	**8**	**12.3**	11	**98.0**	**8**	**12.3**	17
BHP	85.8	**9**	**9.5**	9	95.2	26	3.7	31	85.8	**9**	**9.5**	9	**96.9**	28	3.5	30
Biodegradation	75.8	**9**	8.4	9	**80.1**	46	1.7	55	79.9	**9**	**8.9**	11	78.5	14	5.6	19
Breast cancer	**93.1**	5	18.6	6	**93.1**	5	18.6	6	91.9	**3**	**30.6**	5	91.9	**3**	**30.6**	5
Car evaluation	74.4	14	5.3	14	72.9	14	5.2	43	71.0	**12**	**5.9**	14	**78.1**	19	4.1	42
Contraceptive	51.2	9	5.7	11	48.4	60	0.8	60	51.6	**8**	**6.4**	10	**53.4**	28	1.9	53
Credit approval	81.3	**6**	**13.6**	10	**83.5**	22	3.8	36	80.6	9	9.0	12	81.3	10	8.1	18
Solar flare	58.8	12	**5.3**	13	62.2	39	1.6	54	58.8	**11**	**5.3**	13	**62.3**	40	1.6	55
German credit	**65.7**	**10**	**6.6**	11	63.7	73	0.9	90	63.4	12	5.3	13	63.9	22	2.9	31
Indian liver	**61.8**	**7**	**8.8**	10	59.0	55	1.1	61	60.9	8	7.6	10	58.5	21	2.8	40
Arterial catheter	87.7	8	11.0	9	**87.9**	9	9.8	18	86.8	7	12.4	12	87.0	**5**	**17.4**	13
Language	**66.8**	9	7.4	9	64.3	18	3.6	33	61.8	**2**	**30.9**	7	61.8	**2**	**30.9**	7
Localization	96.5	**8**	10.7	12	**97.0**	15	6.5	34	96.8	9	**10.8**	12	96.5	17	5.7	29
Mammographic	80.6	**8**	10.1	9	**80.8**	**8**	**10.1**	15	80.0	10	8.0	12	80.3	9	8.9	12
Diabetic	65.8	**9**	**7.3**	9	66.1	48	1.4	48	65.9	10	6.6	12	**66.6**	18	3.7	24
Phishing	74.7	**11**	**6.8**	12	**85.1**	56	1.5	88	76.2	12	6.4	12	81.3	21	3.9	32
Pima indians	72.9	**8**	9.1	10	76.1	30	2.5	34	74.0	**8**	**9.2**	11	**77.4**	17	4.6	28
Student	68.5	**7**	**9.8**	11	68.1	46	1.5	70	**72.5**	9	8.1	12	68.9	15	4.6	26
Synthetic 1	64.7	**8**	**8.1**	9	72.9	41	1.8	60	67.2	9	7.5	10	**73.7**	20	3.7	29
Synthetic 2	60.8	**10**	6.1	10	60.8	**10**	6.1	10	**62.7**	**10**	**6.3**	10	**62.7**	**10**	**6.3**	10
Synthetic 3	73.2	**8**	9.1	8	80.1	34	2.4	40	74.0	**8**	**9.3**	9	**80.9**	26	3.1	28
Synthetic 4	**73.4**	10	7.3	12	67.5	21	3.2	22	73.1	**5**	**14.6**	7	73.1	**5**	**14.6**	7
Synthetic 5	44.1	**8**	**5.5**	9	54.3	75	0.7	94	44.1	**8**	**5.5**	10	**55.3**	50	1.1	85
Synthetic 6	49.1	**7**	**7.0**	10	**52.1**	52	1.0	70	50.5	8	6.3	11	51.1	26	2.0	37
Synthetic 7	**52.1**	**10**	**5.2**	12	**52.1**	**10**	**5.2**	12	**52.1**	**10**	**5.2**	11	**52.1**	**10**	**5.2**	11
Synthetic 8	50.5	12	4.2	13	**56.0**	56	1.0	55	53.9	**10**	**5.4**	12	55.1	32	1.7	38
Tic-tac-toe	76.4	**13**	5.9	14	92.6	42	2.2	61	77.5	**13**	**6.0**	14	**92.9**	30	3.1	49
Titanic	**71.3**	**5**	**14.3**	12	**71.3**	**5**	**14.3**	12	**71.3**	8	8.9	9	**71.3**	**5**	**14.3**	9
Wine quality	**69.6**	**9**	**7.7**	9	**69.6**	**9**	**7.7**	9	**69.6**	**9**	**7.7**	9	**69.6**	**9**	**7.7**	9
Yeast	66.0	**8**	**8.2**	10	65.1	19	3.4	40	65.2	**8**	8.1	12	**68.2**	9	7.6	20
No. wins	10	22	16		14	6	5		7	18	18		16	10	10	

Table 4.2: Comparison between the search pruning methods employed with the VS policy: no pruning, value-based pruning (vp), DT-based pruning (dtp), and both (vp + dtp). Metrics are average F1 (F1), number of leaves (L), and F1/L ratio. A one-way within-subjects analysis of variance (ANOVA) was done on the normalized metrics, followed by Wilcoxon signed-ranks pairwise comparisons. The arrows indicate significant differences ($p < 0.05$), and point to the best result.

Metric		F1			L			F1/L		
Group comparison		$p < 0.001$			$p < 0.001$			$p < 0.001$		
Pairwise comparison ($p < 0.05$)	Search pruning method	vp	dtp	vp+dtp	vp	dtp	vp+dtp	vp	dtp	vp+dtp
	no pruning	↑		↑	←	←	←	←		←
	vp		←			↑	↑		↑	↑
	dtp			↑			←			←

In contrast, DT-based pruning did not significantly impact the search depth nor the F1 performance. Instead, its effect appears to be complimentary to value-based pruning, in the sense that, although it did not increase F1, it improved the trade-off between F1 and number of leaves. Using only DT-based pruning outperformed the other variations in terms of F1/L ratio in 18 cases, a moderate improvement compared to 16 cases without pruning. Similarly, when combined with value-based pruning, DT-based pruning had a higher F1/L ratio in 10 of the datasets, compared to only 5 wins without the method. In Table 4.2, we see that the improvements in F1/L obtained by DT-based pruning were statistically significant. The combination of the two approaches outperformed the other variations when taking into account both F1 and the F1/L ratio.

4.6.2. Comparison between simulation policies

When using the VS policy, the combination of both search pruning methods was superior to any other variation when considering both F1 and its trade-off with model size. We therefore compare the simulation policies using both pruning strategies. The results are shown in Table 4.3. There were no statistically significant differences in F1 among all policies according to an ANOVA on the normalized F1 scores, after sufficiently confirmed normality assumptions. On the contrary, the differences in normalized number of leaves and F1/L were significant ($p < 0.05$). The statistical comparisons are reported in Table 4.4.

In Table 4.3, we see that the C4.5 and C4.5b policies achieved the highest F1 in 14 and 19 out of 31 datasets, respectively, compared to 12 wins for the VS policy. The pairwise improvements were not however

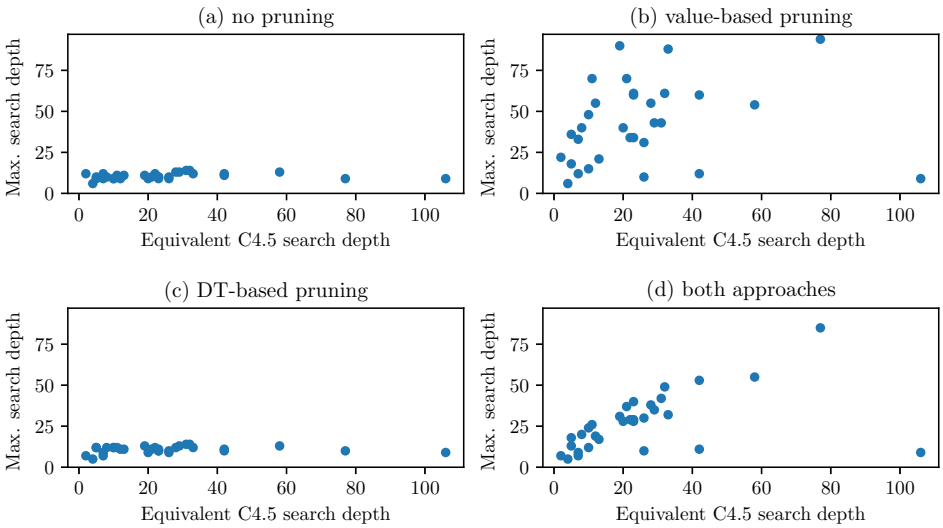


Figure 4.3: Maximum search depth of UCT-DT algorithm using the validation-set performance policy (a) without search pruning, (b) with value-based pruning, (c) with decision tree-based pruning, and (d) with both methods. Each dot represents a dataset.

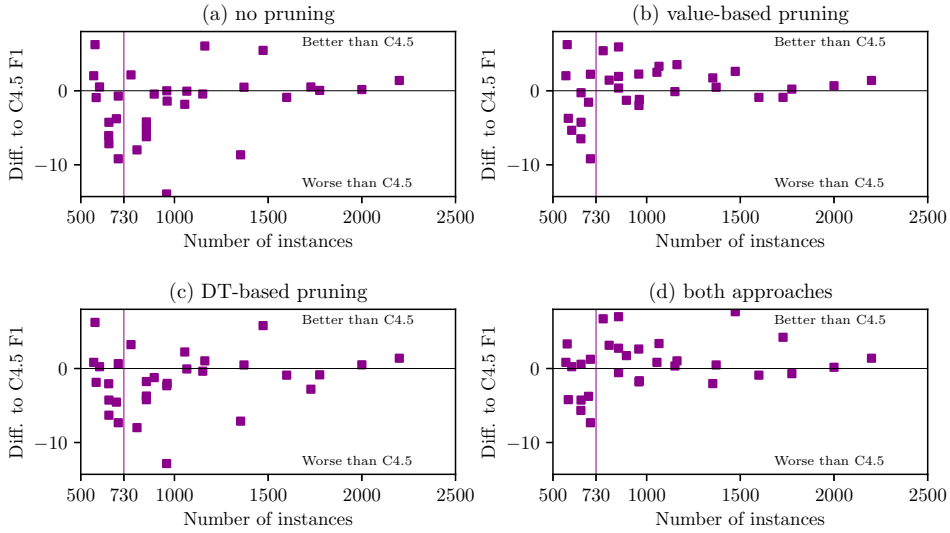


Figure 4.4: Difference between the F1 obtained with UCT-DT algorithm using the validation-set performance policy and the C4.5 algorithm. Pruning methods are (a) none, (b) value-based, (c) decision tree-based, and (d) both. Each point represents a dataset.

Table 4.3: Evaluation of the simulation policies used with both search-pruning methods. The policies are (1) the validation-set performance (VS), (2) completing the DT with C4.5 (C4.5), and completing the DT using C4.5 with bootstrapping (C4.5b). Metrics are average F1 over all classes (F1), number of leaves (L), and their ratio (F1/L). We report the time complexity of VS (T/hh:mm); the other run-times are omitted because they are identical. The best result is highlighted in bold font, and the last row counts the number of times a policy had the best result.

Dataset	(1) UCT-DT + VS				(2) UCT-DT + C4.5			(3) UCT-DT + C4.5b		
	F1	L	F1/L	T	F1	L	F1/L	F1	L	F1/L
Balance Scale	84.4	**7**	12.1	1:19	85.6	13	6.6	**86.8**	**7**	**12.4**
Banknote auth.	**98.0**	**8**	**12.3**	11:00	**98.0**	**8**	**12.3**	**98.0**	**8**	**12.3**
BHP	**96.9**	28	3.5	15:59	93.9	25	3.8	93.4	**23**	**4.1**
Biodegradation	78.5	14	5.6	46:43	**85.6**	13	6.6	79.4	**11**	**7.2**
Breast cancer	**91.9**	**3**	**30.6**	0:03	**91.9**	**3**	**30.6**	**91.9**	**3**	**30.6**
Car evaluation	78.1	**19**	**4.1**	3:19	**87.6**	33	2.7	83.3	31	2.7
Contraceptive	**53.4**	28	1.9	39:31	52.2	**25**	**2.1**	51.6	49	1.1
Credit approval	81.3	**10**	**8.1**	14:01	79.4	14	5.7	**82.3**	13	6.3
Solar flare	62.3	40	1.6	34:26	62.0	50	1.2	**63.0**	**32**	**2.0**
German credit	63.9	**22**	2.9	8:23	**65.5**	**22**	**3.0**	64.9	31	2.1
Indian liver	58.5	21	2.8	12:32	58.9	29	2.0	**61.8**	**7**	**8.8**
Arterial catheter	**87.0**	**5**	**17.4**	5:36	**87.0**	**5**	**17.4**	**87.0**	**5**	**17.4**
Language	**61.8**	**2**	**30.9**	1:30	**61.8**	**2**	**30.9**	**61.8**	**2**	**30.9**
Localization	96.5	17	5.7	23:26	**97.0**	19	5.1	96.8	**13**	**7.4**
Mammographic	80.3	9	8.9	1:07	80.0	10	8.0	**81.4**	**7**	**11.6**
Diabetic	66.6	18	3.7	43:26	65.4	9	7.3	**67.6**	**7**	**9.7**
Phishing	81.3	21	3.9	7:10	**87.6**	33	2.7	78.7	**15**	**5.2**
Pima indians	**77.4**	17	4.6	22:21	74.2	**11**	**6.7**	76.0	13	5.8
Student	68.9	**15**	**4.6**	12:49	69.8	17	4.1	**71.5**	18	4.0
Synthetic 1	**73.7**	**20**	**3.7**	25:56	69.7	29	2.4	68.6	26	2.6
Synthetic 2	**62.7**	10	6.3	33:48	45.7	**8**	5.7	61.3	**8**	**7.7**
Synthetic 3	80.9	26	3.1	21:37	81.4	24	3.4	**84.8**	**21**	**4.0**
Synthetic 4	**73.1**	**5**	**14.6**	0:05	**73.1**	**5**	**14.6**	**73.1**	**5**	**14.6**
Synthetic 5	55.3	**50**	**1.1**	58:53	**57.8**	74	0.8	51.4	67	0.8
Synthetic 6	51.1	**26**	**2.0**	42:18	48.4	29	1.7	**52.4**	33	1.6
Synthetic 7	52.1	10	5.2	39:45	53.4	**9**	**5.9**	**54.2**	10	5.4
Synthetic 8	55.1	**32**	1.7	7:12	**56.5**	**32**	**1.8**	55.1	**32**	1.7
Tic-tac-toe	**92.9**	**30**	**3.1**	4:59	91.6	42	2.2	**92.9**	43	2.2
Titanic	**71.3**	**5**	**14.3**	0:07	**71.3**	**5**	**14.3**	**71.3**	**5**	**14.3**
Wine quality	69.6	9	7.7	41:58	**70.7**	7	10.1	**70.7**	**6**	**11.8**
Yeast	68.2	**9**	**7.6**	9:07	66.1	14	4.7	**68.2**	10	6.8
No. wins	12	17	14		14	12	11	19	19	18

Table 4.4: Comparison between the simulation policies, employed with both pruning strategies. Metrics are average F1 (F1), number of leaves (L), and F1/L ratio. A one-way within-subjects analysis of variance (ANOVA) was done on the normalized metrics, followed by Wilcoxon signed-ranks pairwise comparisons. The arrows indicate significant differences ($p < 0.05$), and point to the best result.

Metric		F1		L		F1/L	
Group comparison		$p = 0.631$		$p = 0.044$		$p = 0.020$	
Pairwise comparison ($p < 0.05$)	Simulation policy	C4.5	C4.5b	C4.5	C4.5b	C4.5	C4.5b
	VS			←			
	C4.5				↑		↑

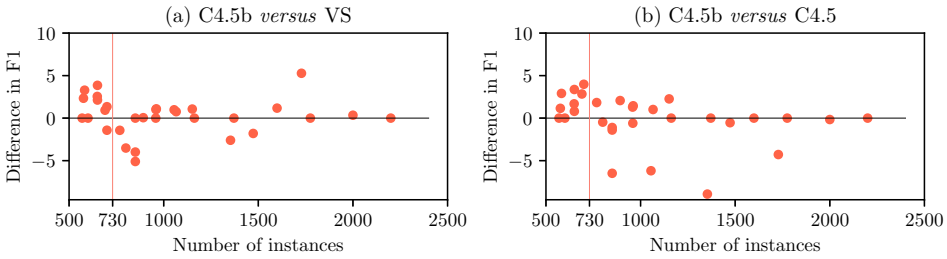


Figure 4.5: Effect of bootstrapping on the C4.5b policy as a function of dataset size. The plots display the difference in average F1 score between (a) the C4.5b and validation-set performance (VS) policies, and (b) the C4.5b and C4.5 policies. Each dot does represents a dataset. Bootstrapping the induction and validation sets during simulation helped improve the performance for datasets with less than 730 examples.

statistically significant, as can be observed in Table 4.4. The DTs learned with the C4.5 policy had a significantly larger number of leaves compared to VS. This indicates that the value estimates obtained through C4.5 tree-completion focus the search on a set of deeper states, resulting in larger DTs. This did not however lead to an improvement in F1, suggesting that C4.5-completion directed the search towards sub-optimal states. The C4.5 policy does not seem to be a better estimator of the expected DT performance compared to VS.

The C4.5b policy was proposed in an attempt to reduce the bias of the search towards the training data, as a consequence of using the same induction and validation sets numerous times for simulation. This policy led to statistically significant improvements in number of leaves and F1/L compared to C4.5. In particular, C4.5b improved the F1 performance of

the other polices for datasets under 730 instances, as can be observed in Figure 4.5.

4.6.3. Comparison between UCT-DT and other approaches

Since none of the simulation policies had significantly better F1, and the C4.5 policy was outperformed in terms of model size, we consider the VS and the C4.5b policies for comparison with locally-optimal and evolutionary learning. As before, both value-based and DT-based pruning were employed. The results are displayed in Table 4.5. Since the results did not pass the normality tests, a Friedman analysis of variance by ranks [66] was done, reported in Table 4.6.

For most of the datasets, the ETE approach resulted in larger DTs compared to the other approaches. The method led to an increased F1 in 12 datasets compared to C4.5, in most of which there was also a significant rise in DT size. E.g. in the *Car Evaluation* and *Solar flare* datasets, the number of leaves grew from 32 to 218 and from 59 to 121, respectively. Depending on the application, such DT sizes can hinder interpretability. Most of the 12 cases that benefited from the evolutionary approach were among the largest of the datasets, indicating that a more exhaustive search may be appropriate for those cases. ETE was outperformed by the UCT-DT variations.

Contrary to the evolutionary approach, the UCT-DT variations led to an overall reduction in DT size with maintained or improved accuracy. As depicted in Table 4.5, UCT-DT with the VS policy outperformed C4.5 in terms of F1 for 20 datasets, while UCT-DT with the C4.5b policy had a higher F1 performance 19 times compared to the locally-optimal algorithm. The UCT-DT variations achieved statistically significant improvements in the trade-off between F1 and DT size, as depicted in Table 4.6.

4.6.4. Data *versus* UCT-DT

In order to understand in which types of data a locally-optimal or MCTS approach may be more suitable, we propose the analysis in Table 4.7. Columns E and H state whether any version of UCT-DT achieved better F1 and F1/L than C4.5, respectively.

Greedy DT learning is known to be highly dependent on the particular sample of training data [51], specially when dealing with few examples. In the UCT-DT algorithm, dataset size is also an important factor in the behavior of search pruning and the simulation policy. Accordingly, dataset size appears to be important in determining the benefit of MCTS compared

Table 4.5: Comparison between the DT learned using C4.5 and the evolutionary tree-encoding (ETE) algorithm, and the best DT learned using UCT-DT with the VS and C4.5b policies. Metrics are average F1 over all classes (F1), number of DT leaves (L), and the F1/L ratio. The last rows count the number of times a variation outperformed the others, and the number of times it outperformed C4.5.

Dataset	C4.5			ETE			UCT-DT + VS			UCT-DT + C4.5b		
	F1	L	F1/L	F1	L	F1/L	F1	L	F1/L	F1	L	F1/L
Balance Scale	81.1	30	2.7	85.6	35	2.4	84.4	**7**	**12.1**	**86.8**	**7**	**12.4**
Banknote auth.	97.6	14	7.0	92.8	29	3.2	**98.0**	**8**	**12.3**	**98.0**	**8**	**12.3**
BHP	93.8	27	3.5	73.5	**3**	**24.5**	**96.9**	28	3.5	93.4	23	4.1
Biodegradation	77.6	13	6.0	74.0	21	3.5	78.5	14	5.6	**79.4**	**11**	**7.2**
Breast cancer	91.1	5	18.2	89.7	**3**	29.9	**91.9**	**3**	**30.6**	**91.9**	**3**	**30.6**
Car evaluation	73.8	32	2.3	**83.8**	218	0.4	78.1	**19**	**4.1**	83.3	31	2.7
Contraceptive	45.8	43	1.1	49.5	57	0.9	**53.4**	**28**	**1.9**	51.6	49	1.1
Credit approval	85.1	**6**	**14.2**	**86.9**	17	5.1	81.3	10	8.1	82.3	13	6.3
Solar flare	58.9	59	1.0	63.0	121	0.5	62.3	40	1.6	**63.0**	**32**	**2.0**
German credit	**65.7**	**20**	**3.3**	61.8	81	0.8	63.9	22	2.9	64.9	31	2.1
Indian liver	**62.7**	24	2.6	57.2	321	0.2	58.5	21	2.8	61.8	**7**	**8.8**
Arterial catheter	87.7	6	14.6	**87.9**	26	3.4	87.0	**5**	**17.4**	87.0	**5**	**17.4**
Language	60.7	8	7.6	57.8	9	6.4	**61.8**	**2**	**30.9**	**61.8**	**2**	**30.9**
Localization	96.3	23	4.2	96.5	**13**	7.4	96.5	17	5.7	**96.8**	**13**	**7.4**
Mammographic	**82.0**	11	7.5	78.5	22	3.6	80.3	9	8.9	81.4	**7**	**11.6**
Diabetic	66.2	11	6.0	67.6	**7**	9.7	66.6	18	3.7	**67.6**	**7**	**9.7**
Phishing	83.4	34	2.5	**89.7**	96	0.9	81.3	21	3.9	78.7	**15**	**5.2**
Pima indians	70.7	24	2.9	68.9	**3**	**23.0**	**77.4**	17	4.6	76.0	13	5.8
Student	**74.6**	12	6.2	73.5	**9**	**8.2**	68.9	15	4.6	71.5	18	4.0
Synthetic 1	70.9	24	3.0	59.8	29	2.1	**73.7**	**20**	**3.7**	68.6	26	2.6
Synthetic 2	**70.0**	27	2.6	61.8	43	1.4	62.7	10	6.3	61.3	**8**	**7.7**
Synthetic 3	80.3	21	3.8	76.6	**13**	**5.9**	80.9	26	3.1	**84.8**	21	4.0
Synthetic 4	72.8	3	24.3	60.5	9	6.7	**73.1**	**5**	**14.6**	**73.1**	**5**	**14.6**
Synthetic 5	48.3	78	0.6	46.9	**15**	**3.1**	**55.3**	50	1.1	51.4	67	0.8
Synthetic 6	49.8	22	2.3	49.8	**17**	**2.9**	51.1	26	2.0	**52.4**	33	1.6
Synthetic 7	56.4	43	1.3	**58.6**	23	2.5	52.1	**10**	5.2	54.2	**10**	**5.4**
Synthetic 8	**55.6**	**29**	**1.9**	52.2	95	0.5	55.1	32	1.7	55.1	32	1.7
Tic-tac-toe	90.3	33	2.7	74.8	193	0.4	**92.9**	**30**	**3.1**	**92.9**	43	2.2
Titanic	70.0	8	8.7	**71.9**	**3**	**24.0**	71.3	5	14.3	71.3	5	14.3
Wine quality	70.5	107	0.7	**71.3**	117	0.6	69.6	9	7.7	70.7	**6**	**11.8**
Yeast	66.4	**9**	7.4	65.8	23	2.9	**68.2**	**9**	**7.6**	**68.2**	10	6.8
No. wins	6	4	3	7	10	7	11	12	11	13	16	16
No. better C4.5				12	11	11	20	20	21	19	20	21

Table 4.6: Comparison between C4.5, ETE, and UCT-DT with the VS and C4.5b policies and both search-pruning methods. Metrics are average F1 (F1), DT size (L), and F1/L ratio. A Friedman analysis of variance by ranks was done, where the null hypothesis is rejected for $\chi^2_F \geq \chi^2_{0.05} = 5.99$. Wilcoxon signed-ranks tests compared each pair of methods. The arrows indicate significant differences ($p < 0.05$), and point to the best result.

Metric		F1			L		
Group comparison		$\chi^2_F = 8.22$			$\chi^2_F = 11.72$		
	Algorithm	ETE	UCT VS	UCT C4.5b	ETE	UCT VS	UCT C4.5b
Pairwise comparison ($p < 0.05$)	C4.5				←	↑	↑
	ETE		↑	↑		↑	↑
	UCT VS						

Metric		F1/L		
Group comparison		$\chi^2_F = 15.07$		
	Algorithm	ETE	UCT VS	UCT C4.5b
Pairwise comparison ($p < 0.05$)	C4.5		↑	↑
	ETE			↑
	UCT VS			

to greedy learning. In Table 4.7, we see that for all datasets with more than 1000 instances, at least one of the UCT-DT variations outperformed C4.5 in terms of F1 and its trade-off with number of leaves. This suggests that heuristic search that relies on estimates of generalization performance, such as UCT-DT, is data hungry. The correctness of the estimates is crucial to guide the search, and those estimates are more likely to be inaccurate if they are based on a small sample of data.

Another aspect of DT learning is the impact of datasets arising from complex distributions, such as those with strong feature interactions, or with many *weak* features contributing to the class. Data complexities become particularly challenging in the small-dataset range.

To assess the importance of feature interactions, column D of Table 4.7 compares the NB classifier with a set of more expressive learners. For the datasets with less than 1000 instances, when the NB outperformed the other models, the chances of UCT-DT outperforming C4.5 are lower. In other words, it seems that when the variables have moderate interactions, or they are not relevant for predicting the class, MCTS is less likely to bring benefit compared to locally-optimal search. For larger datasets, this property does not appear to be as relevant.

Table 4.7: Analysis of dataset properties and their relation with UCT-DT performance, specifically dataset size, feature interactions and proportion of strong features. The table focuses on the number of instances, whether or not the naïve Bayes (NB) classifier outperformed a set of more expressive classifiers (NB F1 $>$ avg. classif. F1?), the proportion of variables above the second quartile of importance to the class prediction ($\frac{\text{No. ISs}>Q_2}{\text{No. vars}}$), and whether or not UCT-DT outperformed C4.5. "No" is omitted to facilitate visualization.

A	B	C	D	E	F	G	H
Dataset	Instances	Instances $>$ 1000?	NB F1 $>$ avg. classif. F1?	UCT-DT F1 $>$ C4.5 F1?	No. ISs $> Q_2$	$\frac{\text{No. ISs}>Q_2}{\text{No. vars}}$	UCT-DT F1/L $>$ C4.5 F1/L?
Breast cancer	569		yes	yes	3	0.10	yes
German credit	959		yes		3	0.13	
Synthetic 4	600		yes	yes	2	0.20	
Credit approval	690		yes		3	0.20	
Student	649		yes		7	0.23	
Synthetic 6	700		yes	yes	5	0.25	
Pima indians	768		yes	yes	2	0.25	yes
Synthetic 1	850		yes	yes	6	0.30	yes
Synthetic 2	700		yes		6	0.30	yes
Indian liver	582		yes		3	0.33	yes
Synthetic 3	650		yes	yes	5	0.33	yes
Synthetic 5	850		yes	yes	7	0.35	yes
Synthetic 7	650		yes		7	0.47	yes
Balance Scale	576		yes	yes	2	0.50	yes
Mammographic	961		yes		2	0.50	yes
Synthetic 8	850			yes	5	0.25	
Yeast	892			yes	2	0.25	yes
BHP	800			yes	6	0.29	yes
Tic-tac-toe	958			yes	6	0.67	yes
Biodegradation	1055	yes		yes	6	0.15	yes
Solar flare	1066	yes		yes	2	0.18	yes
Diabetic	1151	yes		yes	4	0.21	yes
Language	1162	yes	yes	yes	4	0.07	yes
Phishing	1353	yes		yes	2	0.22	yes
Bankte auth.	1371	yes		yes	1	0.25	yes
Contraceptive	1473	yes		yes	2	0.22	yes
Wine quality	1599	yes	yes	yes	3	0.27	yes
Car evaluation	1728	yes	yes	yes	2	0.33	yes
Arterial catheter	1775	yes	yes	yes	4	0.09	yes
Localization	2000	yes	yes	yes	2	0.29	yes
Titanic	2200	yes	yes	yes	1	0.33	yes

DT algorithms have also been described as inadequate for data with many weak inputs. To study this property, the number of variables with IS above the median, and its proportion to the total number of variables, are reported in columns F and G of Table 4.7. We expect datasets with a small proportion of relevant predictors to benefit less from the exploration of many variables as candidate splits. Moreover, such data should require less pruning and be less dependent on accurate estimates of generalization performance. Column H of Table 4.7 seems to support this hypothesis. In the datasets with less than 1000 examples, most cases with IS ratio smaller or equal to 0.25 had a higher F1/L with C4.5 than UCT-DT. The procedure to estimate generalization performance is distinct in the two algorithms, and their pruning approaches. In C4.5, it consists of a statistical test on the training subsets. In UCT-DT it depends on the simulation policy and pruning method. It seems that the simulation policy was better in the many-weak-input scenario, while the statistical pruning of C4.5 was sufficient for less complex data.

4.7. Conclusions

This chapter describes a novel approach for learning DTs based on MCTS. The DT learning problem is modeled as an MDP, and the search is treated as Monte Carlo planning. Based on the UCT algorithm, we propose problem-specific expansion strategies, simulation policies and search pruning approaches. The proposed approach, called UCT-DT, is general and can be employed with other procedures to generate actions and estimate of DT value.

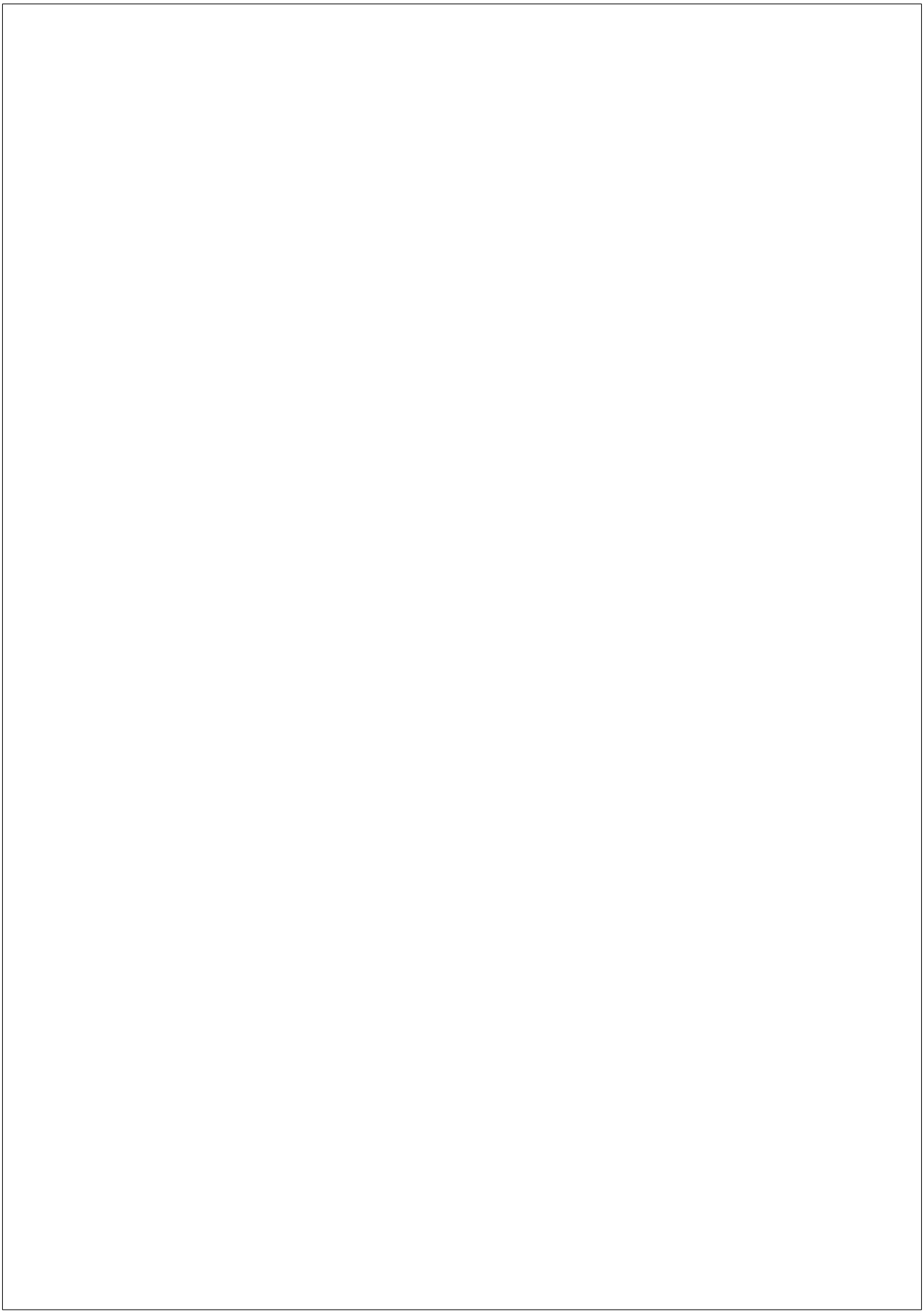
Experiments on 31 classification datasets show that the UCT-DT algorithm had statistically significant improvements in the trade-off between performance and model complexity, compared to locally-optimal learning. It achieved higher prediction performance and reduced model sizes for in 20 out of the 31 datasets. We also evaluated an evolutionary learning approach which outperformed locally-optimal search in 12 of the datasets, at the expense of building larger DTs.

The two proposed search pruning strategies showed complementary effects. Value-based pruning allowed deeper and more promising states to be explored, leading to significant increases in prediction performance. This was particularly true for datasets with more than 730 examples, for which it is possible to build larger generalizable DTs. Conversely, DT-based pruning benefits the trade-off between prediction performance and model size. This result makes sense, as the method is used for regularizing greedy DT learning. The combination of both pruning approaches

offered the best compromise between prediction performance, and it trade-off with model complexity.

Regarding the simulation policies, using C4.5 to complete the current DT at the node was not superior to using the performance that tree on the validation set. This indicates that, although DT-based pruning was useful in predicting overfitting, locally-optimal induction was not a good estimator of the performance of the expected DT at the end of the search branch. The bootstrapped version of the C4.5 policy appears to ameliorate overfitting to the training data, as it improved the other policies in the small-data range. We therefore recommend using this policy for datasets with less than 730 cases, and the VS policy otherwise.

As final remarks, we add that UCT-DT brings benefit compared to locally-optimal search for datasets with more than 1000 instances. This is probably caused by the need for accurate estimates of the value of each state, i.e. performance of the expected DT, in order to guide the search. Obtaining such estimates is more challenging when few samples are available. In the small-dataset range, greedy learning seems to do well for datasets presenting a small degree of feature interactions, or whose class can be well approximated by a small number of features. Future work directions include a study of better simulation policies, in order to provide more robust estimates of performance. In addition, the employment of more effective search pruning strategies would allow a reduction in the algorithm runtime.



Chapter 5

COMPARING THE STRUCTURE OF DECISION TREES

Despite being used for their interpretability, decision trees (DTs) are almost always evaluated based exclusively on prediction performance. The need for comparing the structure of DT models arises frequently in practice, and is usually dealt with by assessing the models manually. Nonetheless, literature on measures for comparing or representing DTs based on their structure is limited. We attempt to fill this gap by proposing an edit distance for comparing the structure of DTs, and a procedure for computing it. The procedure comprises (1) an algorithm for sorting the nodes of a DT, and (2) cost configurations for the editing operations, weighted by the similarity between the tests the DT nodes. The proposed distances were able to accurately model DT structure similarity. Taking into consideration the similarity between test functions allowed decision patterns that are relevant for a prediction task to be expressed by a distance measure.

5.1. Introduction

Chapter 4 elaborated on an alternative induction approach, which performed a search on the space of possible decision trees (DTs). While the method constructed a multitude of DTs, the experiments focused on the performance of a single model, selected based on prediction performance. As a consequence, we asked ourselves if there would be an alternative way of selecting DTs, which can take advantage of their decision structure, as well as the information contained in the forest.

In general, although the popularity of DTs is attributed to their interpretability, their merit is almost always assessed by measures of prediction performance, as indeed for most machine learning (ML) models [61]. However, the task of qualitatively comparing DTs arises frequently in practical scenarios, and is usually dealt with by manually inspecting a small number of models.

Many statistical learning methods train models by optimizing over a set of parameters that live in an inner product space, making some assumptions about the data. On the contrary, DT algorithms learn a graph whose nodes specify rules that partition the input domain. Such a model is not easily embedded in a vector space. As a consequence, although DTs allow an intuitive understanding of the predictions, it is not trivial to quantitatively describe their decision structure. The qualitative comparison of DTs works for a small number of models, but it is not scalable to the output of large ensembles.

To the best of our knowledge, measures for comparing DTs based on their structure, or attempts to find such a representation, have not yet been proposed. Literature on the topic is limited and usually consists on reporting DT size, or relies on the predictions made on a sample of data. The evaluation of DT models often combines prediction performance with measures of model size [61]. The correlation between the DTs in a Random Forest is defined in terms of the consistency of the predicted classes of a data sample [18]. However, using model size or performance metrics based on data may not be descriptive enough to characterize the structure of a DT.

We attempt to fill this gap by proposing a measure of DT dissimilarity that focuses on the organization of the decision functions, rather than on how they perform on a sample of data. We also propose a procedure for computing this dissimilarity, composed by:

1. an algorithm for sorting the nodes of a DT, and

2. a cost configuration for the editing operations of DT nodes.

The approach then makes use of an existing algorithm for efficiently computing the distance between ordered labeled tree graphs, called All Path Tree Edit Distance (APTED) [143, 146]. The sorting algorithm and the cost configuration rely minimally on the data used to build the DTs, and are independent of the learning algorithm or evaluation method. The DT edit distance can be used with any distance-based learning approach. In particular, it is apt for embedding DTs in a vector space, e.g. using dissimilarity representation approaches, successfully employed to other types of graphs.

5.2. Related Work

Vector representations use entities like vectors, strings or graphs to individually characterize objects and are the building block of statistical learning and pattern recognition. Graphs are useful in domains best represented by structural relations. Extensive literature exists on graph embedding, allowing statistical learning techniques to be used in domains modeled as graphs [74].

One such approach is dissimilarity representation, a family of methods that characterize objects by a measure of the pairwise dissimilarities instead of a set of features [154]. They construct a vector-space embedding based on the dissimilarities. The difference between dissimilarity and kernel representations, is that the former do not need to be an inner product. One approach to dissimilarity representation consists on constructing the dissimilarity space. Given a set of input objects $X = \{x_1, ..., x_n\}$ and a dissimilarity function or data, $d(x_i, x_j)$, the dissimilarity space is characterized by the mapping $D(x, R) : X \to \mathbb{R}^k$ [150]:

$$D(x, R) = \big(d(x, p_1), ..., d(x, p_k)\big) \tag{5.1}$$

The set $R \subset X$ contains k representative objects that characterize the variability of the domain, known as prototypes, p_i, $i = 1, ..., k$. Each dimension in the dissimilarity space denotes the distance between the object and a prototype. Any distance $d(x_i, x_j)$ can be used, which makes this framework suitable for non-metric problems.

Several graph embedding methods have been proposed based on dissimilarity representation [165, 23]. One of such approaches introduces a graph edit distance as a measure of dissimilarity between two graphs [23]. The graph edit distance is a generalization of the string edit distance, which estimates the minimum amount of distortion needed to transform a graph into another. Comparing graphs based on edit distances is flexible, as

the formulation can handle any type of graphs and vertex/edge labels. Algorithms for computing graph edit distances are however computationally demanding.

A tree is an undirected graph in which every two vertices are connected by exactly one path. Specifically, a DT is a rooted directed tree, where the vertex labels represent split functions for a given domain. The edit distance between ordered labeled trees has been defined as the sequence of edit operations that transform one tree into another, with minimum cost [11]. Computing the minimum tree edit distance has interest in multiple domains, and has received considerable attention from the database and graph communities [4, 37, 71]. In particular, the problem has a solution that decomposes the trees into subtrees, and follows a recursive strategy to compare and decompose the resulting trees and forests. Several dynamic programing algorithms implement this approach [193, 39], of which the Robust Algorithm for the Tree Edit Distance (RTED) stands out by outperforming its competitors in terms of runtime complexity [144]. Compared to the other existing methods, the algorithm performance is independent of the shape of the trees and runs in $O(n^3)$ time and $O(n^2)$ space, with n the number of nodes of two compared trees. The All Path Tree Edit Distance (APTED) algorithm supersedes the RTED by placing an upper bound on memory requirements with runtime improvements [143, 146].

5.3. Proposed approach

We propose an edit distance to compare the decision structure of a pair of DTs, and a procedure to compute it. The approach consists in adapting an existing algorithm for computing the edit distance between pairs of ordered labeled trees, the APTED algorithm [143, 146]. Since DTs have no inherent ordering, we first propose a methodology for obtaining sorted DTs, described in Section 5.3.1. Secondly, in Section 5.3.2 we propose a cost configuration for the node edit operations, which takes into consideration the properties of nodes' test functions.

As before, consider the input $\mathbf{X}$ with dimensions X_j, $j = 1, ..., M$, and the class variable Y. As described in Section 2.2.2, each DT node n displays a binary univariate test function of the form:

$$t(\mathbf{x}) = \mathbf{1}_{(\tau,\infty)}(x_j)$$

if the variable X_j is continuous, and:

$$t(\mathbf{x}) = \mathbf{1}_{S_L}(x_j)$$

if X_j is nominal. S_L and S_R represent the subsets of values of X_j corresponding to the left and right child branches of n. Let us make the assignment $t(\mathbf{x})$ to node n explicit by indexing on n, as $t_n(\mathbf{x})$.

As a worked-example throughout this chapter, we employ the *Breast Cancer Wisconsin Diagnostic* dataset from the UCI Machine Learning Repository [112]. The dataset contains 30 variables, describing certain properties of cell nuclei present in images of breast mass aspirates. Each sample is labeled as malignant (M) or benign (B). The samples were divided into a 397-instance training set and a 172-instance test set, with maintained class proportions.

5.3.1. Sorting decision tree nodes

An ordered tree is one where the children of each node have an inherent ordering. To employ existing edit distance algorithms for ordered trees, we first need to ensure that the nodes of the input DTs are sorted. In this Section, we propose a procedure for ordering the nodes of DTs. The application of the method to the Breast Cancer data is depicted in italicized font.

To sort the nodes of a classification DT, we:

- Consider the class proportions on the training set.
  The proportion of B and M classes in the Breast Cancer training set is 0.63 and 0.37, respectively.
- Rank the classes from the most prevalent to the least prevalent in the training set. Label them according to their rank, $r = 1, 2, ..., k$.
  The classes are ranked $r(B) = 1$ and $r(M) = 2$.
- For each internal node n, let n_L and n_R denote the left and right child nodes:
  - Swap n_L and n_R such that the child on the *left* of n contains the greatest proportion of the most prevalent class, i.e. the class with $r = 1$. If n_L and n_R get swapped, this requires changing the test function t_n accordingly. I.e. replace $t_n(\mathbf{x}) = \mathbf{1}(x_{j_n} < \tau_n)$ by $t_n(\mathbf{x}) = \mathbf{1}(x_{j_n} \geq \tau_n)$.
    Place the child with greatest proportion of class B on the left of n.
  - If the class with $r = 1$ is equally distributed in n_L and n_R, move on to the class with $r = 2$. Order n_L and n_R such that the class with $r = 2$ is most prevalent on the *right* side of n. Adapt test function t_n accordingly.

If the proportions of class B are equal in both child nodes, place the child with greatest proportion of class M on the right of n.

- If needed, continue ordering the child nodes according to the subsequent classes in the rank, alternating between placing the child with most prevalent *odd*-ranked class on the left side ($r = 3, 5, ...$), and the child with most prevalent *even*-ranked class on the right ($r = 4, 6, ...$).

The motivation behind this strategy is that nodes should be rearranged to make test functions of different DTs are as comparable as possible. Since the functions stratify distinct classes, the sorting algorithm arranges the nodes according to a fixed order of the most prevalent classes. The nodes of a regression tree can be sorted by rearranging them by ascending order of the target variable predicted from the training set.

The use of this procedure for DTs built with different data samples assumes that the class ranking is maintained across the datasets. If the training data is not available, statistics from related studies may be used, as long as the ordering criterion is consistent in all edit distance computations.

5.3.2. Cost configuration of edit operations

The minimum edit distance between two strings is defined as the set of editing operations that transform one string into another with minimum cost [98]. The operations are the deletion, insertion or substitution of a symbol. The well-known *Levenshtein distance* attributes a cost of 1 to each operation [111]. Alternatively, Levenshtein proposed penalizing substitutions by assigning them a cost of 2. In this section, we describe a possible cost configuration for DTs.

Deletion and insertion cost

As with strings, we consider the cost of deleting or inserting a DT node n to be $c_d(n) = 1$ and $c_i(n) = 1$, respectively.

Substitution cost

The substitution cost reflects how this operation is penalized compared to insertions and deletions. A cost of 2 determines that one substitution is equivalent to one deletion followed by one insertion, which we will denote as type-A cost configuration. A substitution cost of 1 considers the operation to be a unitary transformation, which we shall refer to as a type-B

configuration. The appropriateness of either configuration depends on the specific application. We therefore consider and evaluate both type-A and type-B configurations.

In the case of DT nodes, the relative importance of substitutions becomes more complex, because the similarity between the test functions comes into play. While the cost of replacing a character a by b can be expressed as deleting a and inserting b, it is less straightforward how the cost of replacing $t_n(\mathbf{x}) = \mathbf{1}(x_{j_n} < 0.5)$ by $t_v(\mathbf{x}) = \mathbf{1}(x_{j_v} < 0.4)$ should be quantified. As such, we define substitution costs that take into account how similar the test functions are. In all proposed configurations, the cost of replacing a node by one with the same test function is 0.

Type-A substitution cost Using a type-A cost, each substitution has a cost closer to two unitary operations. Concretely, assume that we have two DT nodes n and v, with test functions t_n and t_v. We define the cost $c_s^{A1}(n, v)$ of replacing t_n by t_v as:

$$c_s^{A1}(n, v) = \begin{cases} 2 & \text{if } j_n \neq j_v \\ 1 + min(1, \frac{|\tau_n - \tau_v|}{IQR(X_{j_n})}) & \text{if } j_n = j_v \text{ and } X_{j_n} \text{ continuous} \\ 1 + J_{min}(n, v) & \text{if } j_n = j_v \text{ and } X_{j_n} \text{ categorical} \end{cases} \tag{5.2}$$

where j_n and j_v are the indexes of the variables of the functions at nodes n and v, X_{j_n} and X_{j_v}. $IQR(X_{j_n})$ is the interquartile range of X_{j_n}, estimated from the training sample. For categorical variables, we define $J_{min}(n, v)$ as the minimum average Jaccard distance between the two subsets of values L_n, R_n and L_v, R_v:

$$J_{min}(n, v) = 1 - max \begin{cases} \frac{1}{2}\big(J(L_n, L_v) + J(R_n, R_v)\big) \\ \frac{1}{2}\big(J(L_n, R_v) + J(R_n, L_v)\big) \end{cases}$$

and where the Jaccard index $J(A, B)$ is a measure of the similarity between the sets A and B, with $0 \leq J(A, B) \leq 1$:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Ideally, $IQR(X_{j_n})$ or $J(A, B)$ are estimated from the training data used to built the DTs. If those data are unavailable, the statistics can be obtained from related studies, provided they reflect the variables with good enough accuracy, and are consistent throughout all edit distance computations.

The cost $c_s^{A1}(n, v)$ determines that if the functions t_n and t_v have distinct variables, a substitution has a cost of 2. Alternatively, the cost of a substitution when $j_n \neq j_v$ could also take into consideration the similarity between X_{j_n} and X_{j_v}. We therefore propose a second type-A substitution cost c_s^{A2}:

$$c_s^{A2}(n, v) = \begin{cases} 2 - |\hat{\rho}_{j_n j_v}| & \text{if } j_n \neq j_v \\ 1 + min(1, \frac{|\tau_n - \tau_v|}{IQR(X_{j_n})}) & \text{if } j_n = j_v \text{ and } X_{j_n} \text{ num.} \\ 1 + J_{min}(n, v) & \text{if } j_n = j_v \text{ and } X_{j_n} \text{ cat.} \end{cases} \tag{5.3}$$

with $\hat{\rho}_{j_n j_v}$ an estimate of the correlation between X_{j_n} and X_{j_v} in the training data, such as the Pearson correlation coefficient, the Kendall rank correlation coefficient, or the Spearman rank correlation coefficient. While the former measures linear relationships, the latter two statistics assess monotonic relationships. The type-A substitution costs, c_s^{A1} and c_s^{A2}, lie in the interval $[1, 2]$.

Type-B substitution cost It may be preferable to have a substitution cost that is closer to the cost of a single insert/delete operation. By replacing a DT node by one with a identical test function, we expect the resulting DT to have a similar way of partitioning the domain. It would thus make sense to have less costly substitutions. To express this idea, we define the $c_s^{B1}(n, v)$ substitution cost, which is identical to type-A costs but increased by 1:

$$c_s^{B1}(n, v) = \begin{cases} 1 & \text{if } j_n \neq j_v \\ min(1, \frac{|\tau_n - \tau_v|}{IQR(X_{j_n})}) & \text{if } j_n = j_v \text{ and } X_{j_n} \text{ num.} \\ J_{min}(n, v) & \text{if } j_n = j_v \text{ and } X_{j_n} \text{ cat.} \end{cases} \tag{5.4}$$

As before, we can take into account the correlation between distinct variables X_{j_n} and X_{j_v}, leading to $c_s^{B2}(n, v)$:

$$c_s^{B2}(n, v) = \begin{cases} 1 - |\hat{\rho}_{j_n j_v}| & \text{if } j_n \neq j_v \\ min(1, \frac{|\tau_n - \tau_v|}{IQR(X_{j_n})}) & \text{if } j_n = j_v \text{ and } X_{j_n} \text{ num.} \\ J_{min}(n, v) & \text{if } j_n = j_v \text{ and } X_{j_n} \text{ cat.} \end{cases} \tag{5.5}$$

Type-B substitution costs, c_s^{B1} and c_s^{B2}, lie in the interval $[0, 1]$. Although the relative cost of the operations may change the editing sequence of

operations found by the algorithm, type-A distances are strictly greater than their equivalent type-B.

After sorting the input DTs and specifying the editing costs, the last step of the procedure consists on employing the APTED algorithm[1]. Depending on the substitution cost, the resulting distances are denoted as d_{A1}, d_{A2}, d_{B1} and d_{B2}.

Normalizing the decision tree edit distance

It has been shown that the string edit distance cannot be normalized by first obtaining the conventional edit distance and then dividing it by the length of the corresponding editing path [120]. The computation of the normalized edit distance is non-trival and requires a specialized algorithm, yet to be developed for trees. Obtaining an estimate of the normalized distance by dividing the absolute distance by a function of (string) size has however shown usefulness in several applications [182]. We take this approach and approximate a normalization of the DT edit distance $d(t_1, t_2)$ as:

$$d'(t_1, t_2) = \frac{d(t_1, t_2)}{|t_1| + |t_2|} \tag{5.6}$$

with $|t_1|$ and $|t_2|$ the number of nodes of trees t_1 and t_2.

5.4. Application to the Breast Cancer dataset

To perform an empirical assessment of the four proposed distances, we consider the Breast Cancer data. This dataset was chosen because some of its variables have a significant degree of correlation. They represent properties of three cell nuclei, labeled 1, 2 and 3. Variables *area 1*, *radius 1* and *perimeter 1* refer to the properties of the cell nucleus labeled 1. Likewise, variables *area 3*, *radius 3* and *perimeter 3* refer to nucleus 3. This allows us to investigate the effect of the substitution costs defined in Equations 5.3 and 5.5 when there are distinct variables that are highly-correlated. The data were parted into a training and a test dataset.

A hierarchy of decision trees An method based on Monte Carlo tree search (MCTS) has been proposed for DT learning [138]. The algorithm outputs a forest that represents a hierarchy of DTs, setting a pertinent

[1] Implementation under MIT License [145], adapted to parse DTs, take as input the statistics about the correlations between the variables and class distributions, and to perform the sorting of the trees.

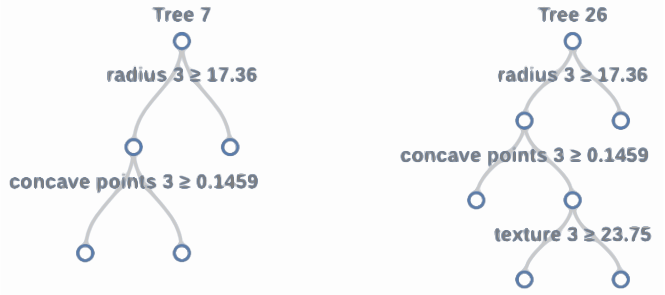


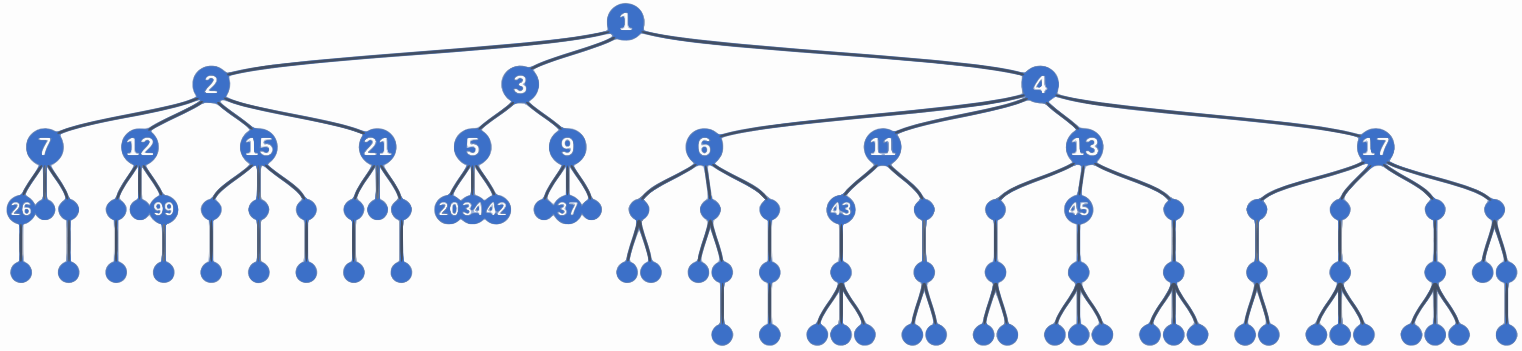
Figure 5.1: Example of a parent (7) and child (26) DTs. Tree 26 is obtained by adding one test function to tree 7.

context for the study of DT similarity. Each edge of the hierarchy connects two parent-child DTs, and corresponds to adding a new split function to the parent. Consequently, trees that are close in the hierarchy are expected to be more similar than trees that are far. There may also be similarities between DTs that are far.

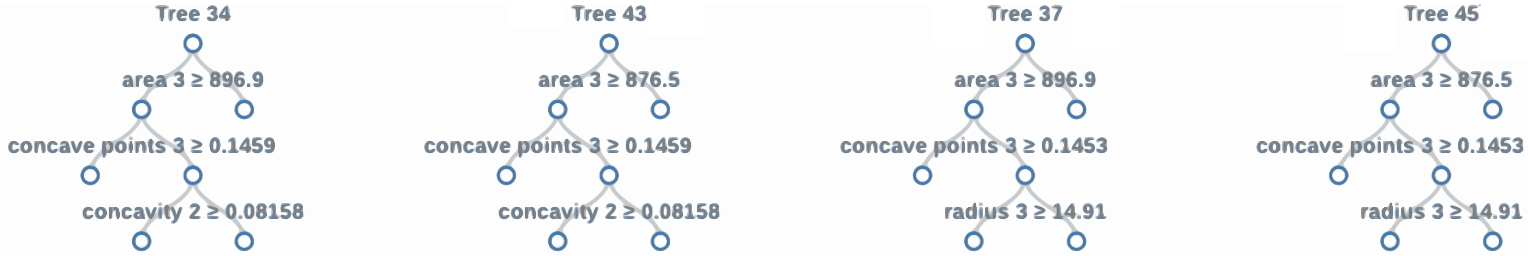
Using the MCTS approach, we constructed a forest of 104 DTs for the Breast Cancer training set. The forest is displayed in Figure 5.2a, where e.g. we see that the tree labeled 7 is the parent of the one labeled 26. As shown in Figure 5.1, tree 26 is the result of adding the split function $t(\mathbf{x}) = \mathbf{1}(\textit{texture 3} \geq 23.75)$ to tree 7, where *texture 3* is the name of a variable. The prediction performance of each DT was then evaluated on the test set.

Computing the distances The proposed distances d_{A1}, d_{A2}, d_{B1} and d_{B2} were computed between each pair of DTs of the forest of Figure 5.2a, using the Pearson, Kendall rank, and Spearman rank correlation coefficients in c_s^{A2} and c_s^{B2}. The resulting distance matrices $\mathbf{D}_{A1}, \mathbf{D}_{A2}, \mathbf{D}_{B1}, \mathbf{D}_{B2}$ are displayed in Figure 5.3, where the Spearman correlation was used for d_{A2} and d_{B2}. The 104 trees at each axis are sorted by their number of nodes, and their ordering is maintained across all plots. The bottom-left corner corresponds to the smallest DTs with only 3 nodes, while the top-right corner corresponds to the largest DTs with 13 nodes.

In Figure 5.3, type-A distances show on average larger values than the equivalent type-B distances. The values of d_{A1} lie in the interval $[0, 9]$, while d_{B1} lie in $[0, 5]$. Distances d_{A2} are contained in the interval $[0, 6.3]$, while d_{B2} lie in $[0, 4.1]$. These results are expected, since the substitution costs make type-A distances strictly greater than type-B distances, despite possibly leading to different sequences of editing operations. The same reasoning applies when comparing d_{A1} with d_{A2}, and d_{B1} with d_{B2}.

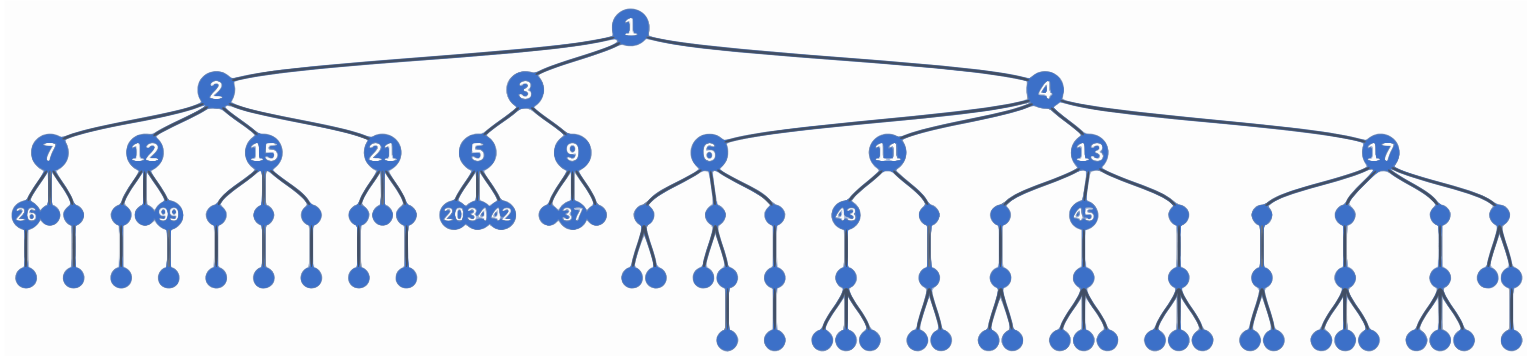


(a) Hierarchical forest of DTs. Each edge represents the addition of a new test function. White labels identify the DTs.

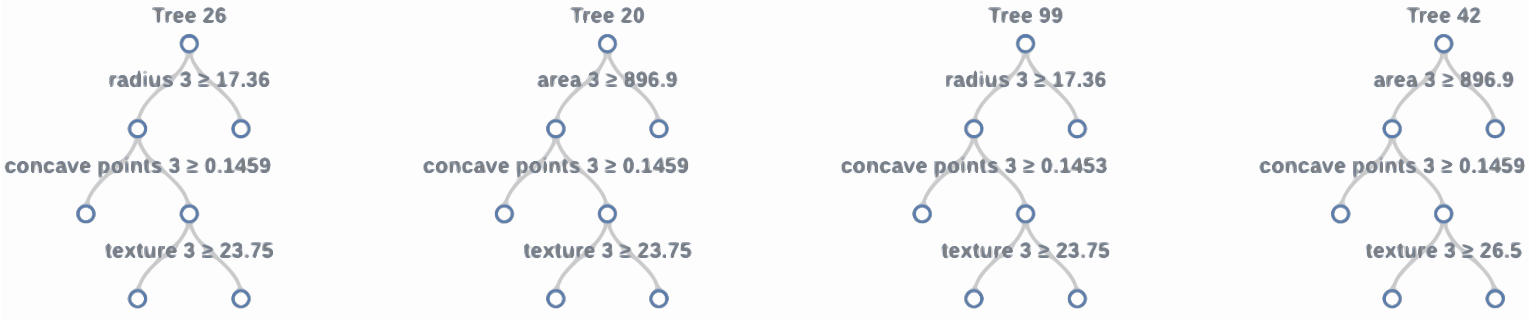


(b) DTs with substitutions of same-variable functions.

Figure 5.2: Hierarchical forest of decision trees (DTs) constructed for the Breast Cancer Diagnostic dataset, using the Monte Carlo tree search (MCTS) approach for learning DTs [138]. Examples of the DTs used in the qualitative criteria described in Section 5.4.2.



(a) Hierarchical forest of DTs. Each edge represents the addition of a new test function. White labels identify the DTs.



(c) DTs with substitutions of functions of distinct-but-correlated variables.

Figure 5.2: (cont.) Hierarchical forest of decision trees (DTs) constructed for the Breast Cancer Diagnostic dataset, using the Monte Carlo tree search (MCTS) approach for learning DTs [138]. Examples of the DTs used in the qualitative criteria described in Section 5.4.2.

A pair of DTs with many nodes is more likely to require a large number of substitutions in the distance computation. In the Figures, we observe that all the edit distances indeed appear to be correlated with DT size. Type-B distances do however appear to have lower values towards the diagonal of the distance matrix. The way in which DT size affects the distances and their ability to capture similarity is further discussed in Section 5.4.2.

5.4.1. Comparison to distances based on prediction similarity

DTs models are usually evaluated based on their prediction performance. And although a performance metric is unlikely to capture DT structure, identical DTs are expected to lead to similar predictions. As a consequence, we compare the proposed distances with a measure of classification similarity. Concretely, we focus the predictions made for the 172 test instances of the Breast Cancer data. We consider two closely-related measures: (1) one based on the class probabilities estimated by each DT, and (2) another based on the actual class predictions.

Class-probability distance

Let $p_t(y|\mathbf{x})$ be the distribution of class y given instance $\mathbf{x}$, estimated by the tree t. For a test dataset with m samples, $\mathbf{x}_1, .., \mathbf{x}_m$, the *class-probability distance* between two DTs t_1 and t_2 can be estimated as:

$$d_{KL}(t_1, t_2) = \frac{1}{m} \sum_{i=1}^{m} D_{KL}\big(p_{t_1}(y|\mathbf{x}_i) \parallel p_{t_2}(y|\mathbf{x}_i)\big) \tag{5.7}$$

with $D_{KL}\big(p \parallel q\big)$ the *Kullback-Leibler divergence* between distributions p and q[2].

Class-prediction distance

The *class-prediction distance* between trees t_1 and t_2 is defined as:

$$d_p(t_1, t_2) = \frac{1}{m} \sum_{i=1}^{m} L\big(y_{t_1}(\mathbf{x}_i), y_{t_2}(\mathbf{x}_i)\big) \tag{5.8}$$

where $L(\cdot, \cdot)$ is the 0-1 loss and $y_t(\mathbf{x})$ the class predicted by tree t for $\mathbf{x}$.

[2] In order to handle inconsistencies with the conventions for the supports of p and q, $S_p \subset S_q$, additive smoothing was applied to the estimates of q. Specifically, $\hat{q} = \epsilon U_k + (1 - \epsilon)q$, where U_k is the uniform distribution and ϵ was set to 10^{-5},

Table 5.1: Matrix norm of difference between the DT edit distance matrices to the prediction-based distance matrices, d_{KL} and d_p. The normalization was approximated using Equation 5.6.

Matrix norm	Unnormalized distances	Normalized distances
$\|\|\mathbf{D}_{A1} - \mathbf{D}_{KL}\|\|$	466.5	15.1
$\|\|\mathbf{D}_{A2} - \mathbf{D}_{KL}\|\|$	374.6	11.3
$\|\|\mathbf{D}_{B1} - \mathbf{D}_{KL}\|\|$	267.2	9.5
$\|\|\mathbf{D}_{B2} - \mathbf{D}_{KL}\|\|$	178.0	12.1
$\|\|\mathbf{D}_{A1} - \mathbf{D}_{p}\|\|$	480.4	25.5
$\|\|\mathbf{D}_{A2} - \mathbf{D}_{p}\|\|$	388.5	19.9
$\|\|\mathbf{D}_{B1} - \mathbf{D}_{p}\|\|$	281.2	13.7
$\|\|\mathbf{D}_{B2} - \mathbf{D}_{p}\|\|$	191.3	8.7

The rightmost plots of Figure 5.3 display the $\mathbf{D}_{KL}$ and $\mathbf{D}_p$ matrices, containing the distances d_{KL} and d_p for each pair of DTs built for the Breast Cancer dataset. In order to compare the edit distances with d_{KL} and d_p, we computed the norm of the difference between the corresponding distance matrices. Since edit distances and prediction-based distances represent distinct quantities, we also consider the normalized edit distances. The resulting matrix norms are reported in Table 5.1.

We first observe that the d_{A2} and the d_{B2} distances, in particular their normalized counterparts, are indeed closer to d_{KL} and d_p. This suggests that the taking the correlation between distinct variables into account is more likely capture similarity in the class predictions of the DTs than otherwise. It appears that the normalization attenuated the effect of DT size, as the norms are much smaller for normalized distances. The impact of the normalization on the ability to express tree structure is investigated in the following Section. Finally, a qualitative observation of Figure 5.3 suggests that the patterns of the edit-distance matrices show some coherence with the $\mathbf{D}_{KL}$ and $\mathbf{D}_p$ matrices. The d_{B2} measure seems to be the least affected by the number of leaves, as the diagonal of corresponding matrix holds the lowest values.

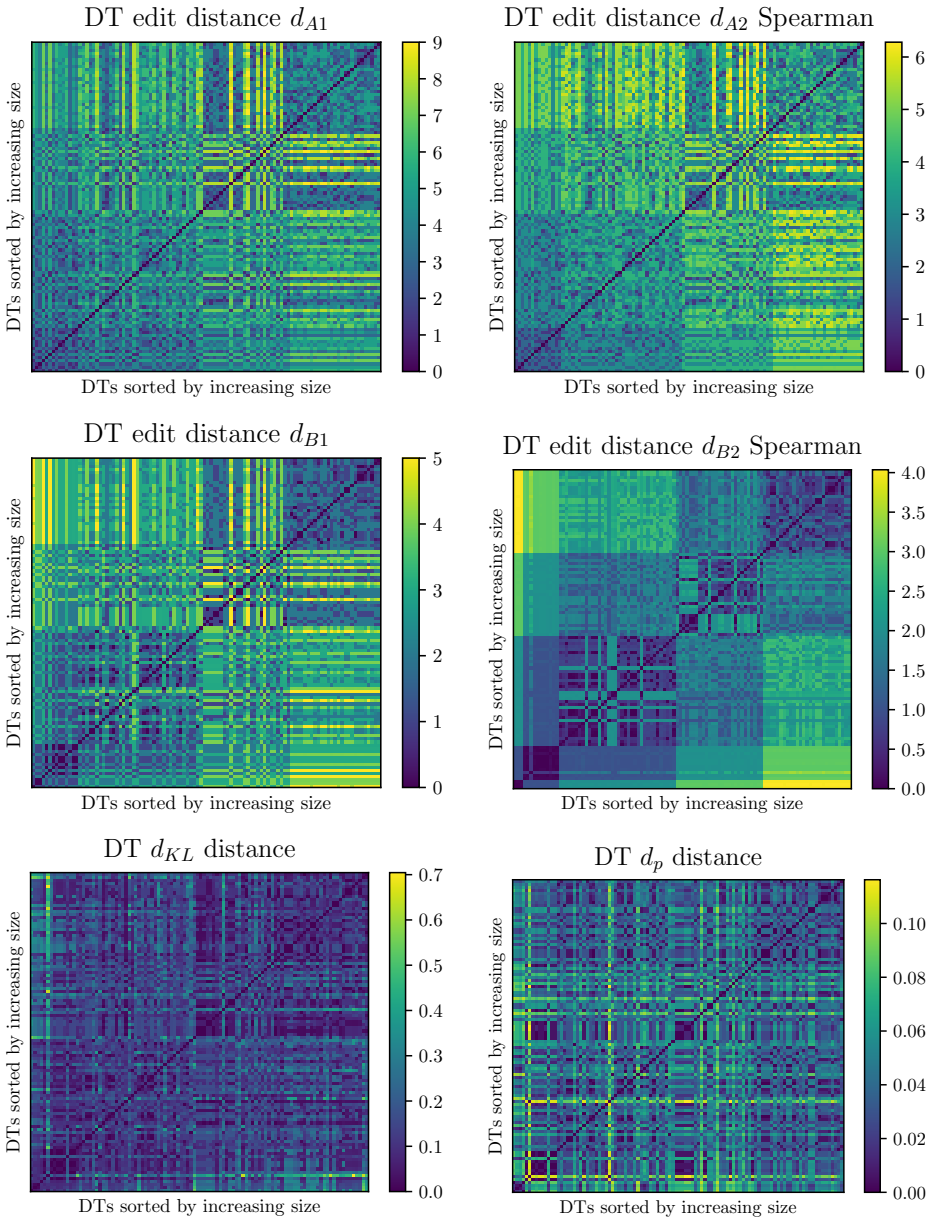


Figure 5.3: Distance matrices comparing the 104 DTs constructed with the Breast Cancer Diagnostic data, using the Monte Carlo tree search approach [138]. The DTs at each axes are sorted according to their number of nodes, with consistent sorting in all plots. Distances d_{A2} and d_{B2} obtained using the Spearman correlation coefficient.

5.4.2. Decision tree cluster analysis

In order to evaluate the distances on their ability to capture DT structure, we perform a cluster analysis. A Laplacian Eigenmaps decomposition was used to obtain an embedding based on the edit distances [134]. The k-means algorithm was then used to find clusters of DTs, using $k = 4$. Figure 5.4 displays the clusters retrieved using the d_{B2} distance with the Spearman rank correlation. The visualization shows the two first modes of t-SNE dimensionality reduction [118].

Assessment criteria

We consider two quantitative criteria to assess the clustering results:

1. *Silhouette coefficient:* The silhouette coefficient evaluates clustering consistency by measuring how similar each DT is to its assigned cluster, compared to other clusters.

2. *Normalized mutual information between the cluster labels and tree size*. This quantity measures the mutual information between the assigned cluster labels and the number of nodes of each DT. It assesses whether or not the DTs get clustered according to their number of nodes, and thus the influence of tree size on the edit distance.

The evaluation is complemented by three qualitative criteria based on a few selected examples of DTs:

1. **Insertions/deletions:** Each branch of the MCTS forest of Figure 5.2a represents the addition of a new split function. In other words, it represents an operation of insertion or deletion. Therefore, parent-child DTs can be converted into one another by a single insertion/deletion. If two parent-child DTs are clustered together, it means that these operations were deemed comparatively less costly than substitutions by the distance algorithm. To investigate this effect, we observe whether or not the trees 2, 3 and 4 get clustered together with their child trees:

   - Children of 2: 7, 12, 15 and 21 (4 operations),
   - Children of 3: 5, 9 and 14 (3 operations),
   - Children of 4: 6, 11, 13 and 17 (4 operations)

   This corresponds to a total of 11 insertions/deletions. Taking these as an example, the *insertion/deletion score* is the fraction of these 11 pairs of trees that got clustered together. The score does not assess

the quality of a distance measure, nor its clustering result. It aims at describing the relative importance insertions/deletions compared to substitutions.

2. **Same-variable substitutions:** We observe whether or not each proposed distance was able to cluster together DTs with substitutions of functions of the same variable, but with distinct split values. In particular, we evaluate whether or not the trees 34, 37, 43 and 45 were assigned the same cluster. As displayed in Figure 5.2b, these trees have two variables in common in the same nodes, *area 3* and *concave points 3*, with distinct split thresholds. The relevant pairwise transformations are:

   - Single same-variable substitution in top node: 34 to 43, and 37 to 45 (2 transformations)
   - Single same-variable substitution in level-2 node: 34 to 37 (1 transformation)
   - Two same-variable substitutions: 34 to 45, and 37 to 43 (2 transformations with 2 substitutions each).

   Each time a pair of DTs corresponding to a transformation is clustered together, the score gets incremented by $1/5$. As such, if the four DTs are clustered together, the score is $5/5$.

3. **Distinct-but-correlated-variable substitutions:** This criterion assesses the ability to group together DTs whose distance computation is likely to require a substitution by a function of a different variable, but which is significantly correlated to the original variable. E.g. the trees 26, 99, 20 and 42 shown in Figure 5.2c are extremely similar, except for their first split. In trees 26 and 99, the first split variable is *radius 3*. In 20 and 42, the first split uses *area 3*. The two variables have a Kendall and Spearman rank correlations of approximately 1. There are two relevant transformations:

   - Substitution of *radius 3* by *area 3*: tree 26 to 20.
   - Substitution of *radius 3* by *area 3*: tree 99 to 42.

   The *distinct-but-correlated-variable substitution score* is equal to 1 if the four trees are clustered together, $1/2$ if three of them are clustered to together, and 0 otherwise.

Table 5.2 summarizes the application of these criteria to the clustering result obtained using each variation of the proposed edit distance.

Table 5.2: Application of the criteria of Section 5.4.2 to evaluate the edit distances and their clustering outcome. The quantitative criteria assess (1) clustering consistency, (2) the effect of tree size on the edit distance. The qualitative criteria analyze (3) the extent to which insertions/deletions are captured by the edit distance relative to substitutions scores, (4) the ability to cluster together pairs of DTs with same-variable substitutions, and (5) the ability to cluster together pairs of DTs with distinct-but-highly-correlated variable substitutions

Distance	(1) Silhouette coeficient	(2) Mutual info. cluster labels & no. nodes	(3) Insertions/ deletions	(4) Same-variable substitution	(5) Substitution of distinct-correlated variables
A1	0.195	0.318	10/11	3/5	0
A1 normalized	0.357	0.288	2/11	2/5	0
A2 Pearson	0.154	0.245	11/11	2/5	1
A2 Pearson normalized	0.309	0.230	1/11	1/5	0
A2 Kendall	0.149	0.232	8/11	2/5	1/2
A2 Kendall normalized	0.320	0.216	1/11	0	0
A2 Spearman	0.150	0.207	7/11	2/5	1/2
A2 Spearman normalized	0.298	0.238	0	0	0
B1	0.417	0.399	7/11	5/5	0
B1 normalized	0.549	0.451	2/11	0	0
B2 Pearson	0.597	0.745	9/11	4/5	1
B2 Pearson normalized	0.474	0.745	4/11	0	0
B2 Kendall	0.594	0.745	9/11	5/5	1
B2 Kendall normalized	0.467	0.730	3/11	0	0
B2 Spearman	0.639	0.745	10/11	5/5	1
B2 Spearman normalized	0.489	0.745	4/11	0	0

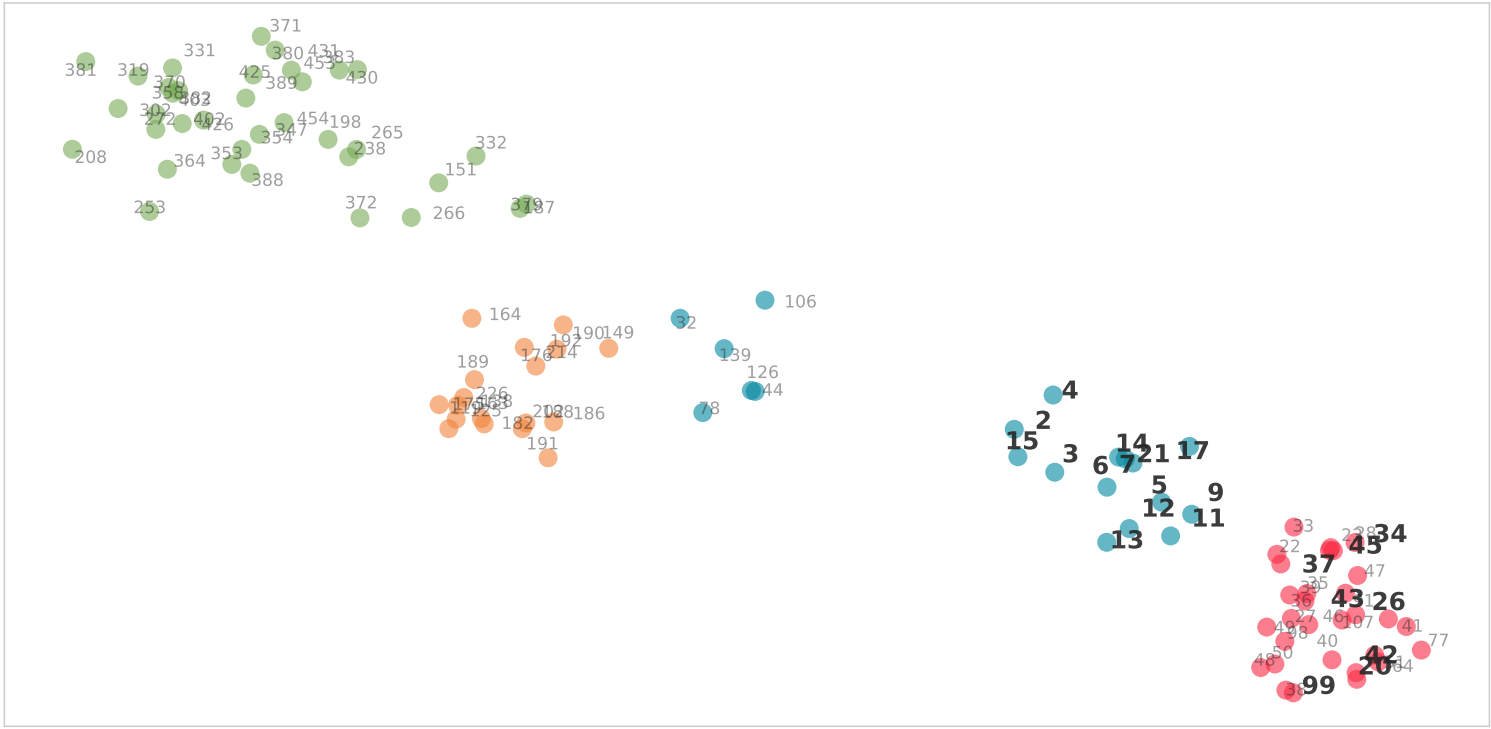


Figure 5.4: Low-dimensional visualization using t-SNE based on the d_{B2} distance with the Spearman rank correlation. The colors represent the clusters found using k-means on the result of spectral embedding, with $k = 4$. Each dot represents a decision tree (DT) of the hierarchical forest constructed with the Monte Carlo tree search (MCTS) approach for learning DTs [138], depicted in Figure 5.2a. Labels in bold font highlight DTs considered by the qualitative criteria defined in Section 5.4.2. Some jitter was added to the points to avoid overlapping labels.

Discussion of the clustering results

In Table 5.2, the first observation is that, although the normalized counterparts of each distance lead to higher Silhouette coefficients in some cases, the qualitative criteria indicate that they are unable to express structural similarities. The normalized distances did considerably worse in capturing same-variable or distinct-variable substitutions. The normalized distances were also unable to identify insertions/deletions. The *insertion/deletion* score was lower for all normalized counterparts. Although the edit distances seem to be dependent on the size of the DTs, this relationship seems to be more complex than a factor of the sum of their numbers of leaves.

The mutual information between the assigned cluster labels and DT size is on average greater for type-B distances. Although in Section 5.4.1 we saw that type-A distances have on average higher values than type-B distances, DT similarity attributed to its size appears to be more accurately expressed by type-B distances.

As expected, the d_{A1} and d_{B1} distances were not able to group together the trees labeled as 20, 99, 26 and 42, as the *distinct-but-correlated* score for these distances was equal to 0. However, d_{A2} and d_{B2} were able to cluster these DTs together, showing that taking the correlation between the variables into account is essential in comparing DT structure that is relevant for the classification task.

The insertion/deletion criterion did not show significant differences between type-A and type-B distances. However, type-B did considerably better at identifying similarities between DTs with same-variable or distinct-but-correlated variable substitutions. This result suggests that modeling substitutions as unitary operations, with a cost that is at most equal to that of an insertion/deletion, does better at capturing relevant DT structure. In other words, if two different test functions lead to an identical partition of the domain, we would want DTs using those functions to be considered similar. Type-B distances seem to achieve this better than type-A. This is confirmed by the values of the Silhouette coefficients, which indicate greater consistency of the clusters found for type-B distances.

Regarding the type of correlation statistic for the d_{A2} and d_{B2} distances, no significant differences were found in the clustering results. The best result was achieved using d_{B2} with the Spearman rank correlation coefficient.

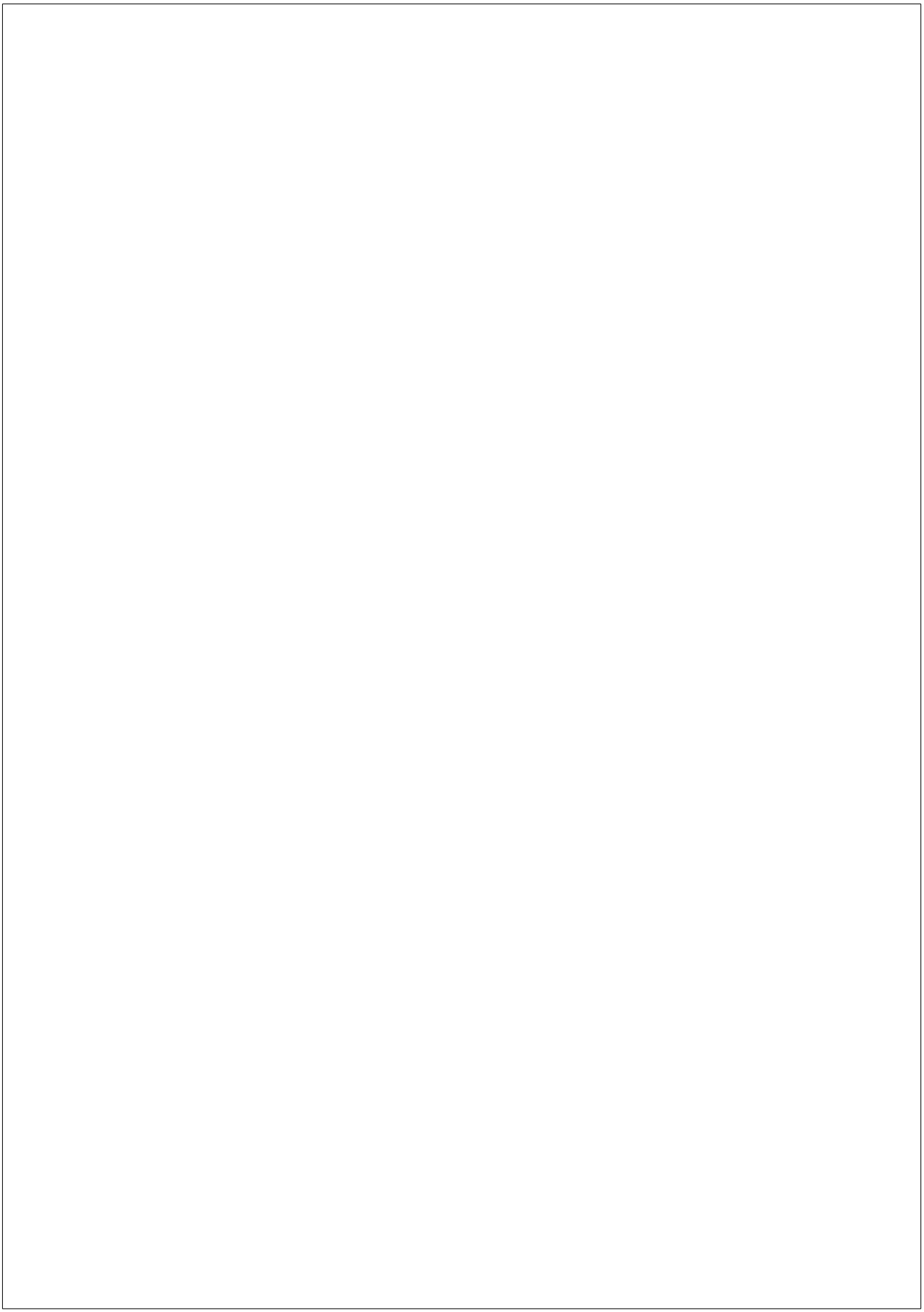
5.5. Conclusions

The present chapter proposes a novel edit distance for comparing the structure of DTs, and a procedure for computing it. The procedure is first composed by an algorithm for sorting the nodes of a DT, and cost configurations for the editing operations. These elements are combined with the All Path Tree Edit Distance (APTED) algorithm [143, 146], an efficient method for computing the edit distance between ordered label tree graphs. Node editing operations are insertion, deletion and substitution. The proposed substitution costs take into consideration the similarity between the test functions, present at each compared DT node. We investigate the effect of modeling a substitution as a combination of two unitary operations (type-A), or as single unitary operation (type-B). Finally, since the interpretation of the edit distance depends on the size of the input sequence, we also proposed approximated normalized versions of the distances. Our method relies minimally on statistics of the data, and can be employed regardless of the algorithm used to build the DTs.

In order to assess the proposed measures, we perform a comparison with distances based on classification similarity, followed by a cluster analysis. A hierarchical forest of DTs was built for the Breast Cancer Diagnostic dataset, and the distances were computed for each pair of DTs. The best of proposed distances successfully clustered together DTs with identical structures. The results indicate that taking into account the similarity between test functions allowed the properties of DT structure that are relevant in the context of prediction to be captured by the distance measure. Considering a substitution as a unitary operation was superior in expressing properties of DT structure. In other words, the d_{B2} distance, with the Spearman rank correlation coefficient was the best measure of DT similarity according to our evaluation criteria. The measure was also better at expressing similarity due to DT size. The approach used for normalizing the edit distances was unable to preserve their ability to represent DT structure similarity.

The proposed approach represents a first step towards a more complete assessment of DT models, in combination with prediction performance. It is generic and can be adapted to different types of DTs. An immediate direction of improvement is a more formal evaluation based on more DTs. It is also necessary to assess the scalability to larger forests, as well as the definition of a useful normalized distance measure. A possible method of achieving a normalized distance could be computing the Euclidean distance in a DT embedding constructed with a set of prototypes, through the mapping of Equation 5.1. The prototypes could be selected

based on the DT clusters obtained with the proposed edit distance, for example by selecting a several trees from each cluster. Applications of this work include the selection of DTs based on their structure. E.g. the error rate of a random forest model is bounded by the weighted correlation of the base models. The DT edit distance can be used to discard models with redundant structures, thus increasing the variability within the ensemble.



Chapter 6

DECISION TREES IN THE RESEARCH AND MANAGEMENT OF AORTIC STENOSIS

The previous chapters elaborated on how to improve the search and evaluation of decision tree (DT) models. The present chapter takes a different turn by attempting to assess the value of DTs and their praised interpretability on a specific clinical decision problem.

6.1. Introduction

As introduced in Chapter 1, clinical guidelines summarize research findings and express expert consensus regarding thresholds of normality and recommendations for intervention. In this chapter, we briefly detail the merits and limitations of the current guidelines for the management of aortic stenosis (AS), the most common cardiac valvular pathology. To overcome some of these limitations, we investigate the potential of DT learning to generate guidelines directly based on (location-specific) data. Our approach consists in learning a model for risk stratification, investigating how it ponders the natural history of AS, and evaluating the expected improvement with aortic valve intervention (AVI) predicted by the model.

6.1.1. Aortic stenosis

Aortic stenosis (AS) is a narrowing of the aortic root, mostly attributed to age-related calcification of the aortic valve. As a consequence of the obstruction of blood flow across the valve, the disease progression encompasses a variety of pathways [133]. In particular, the reduction in cardiac output can be compensated by left-ventricular (LV) pressure overload, resulting in compensatory concentric LV remodeling. Although initial LV remodeling is physiological, the mechanism eventually becomes insufficient, leading to cell damage and LV dysfunction. Aortic stenosis is therefore characterized by a long asymptomatic period, until later stages where patients usually present with angina, syncope and dyspnea. An accurate assessment of AS severity and related functional descriptors has critical prognostic implications [6]. Evaluating the progression of AS is however challenging, owing to the variety of disease pathways and the long asymptomatic period [14].

6.1.2. Assessment of aortic stenosis severity

The diagnostic workup of AS primarily relies on echocardiography to characterize valve anatomy and hemodynamics, and LV function. The key descriptors of AS severity are aortic valve area (AVA), peak transvalvular velocity (Vmax) and mean transaortic pressure gradient (ΔPm), obtained through Doppler echocardiography. In theory, AVA is the ideal way of quantifying AS but its measurement is operator-dependent and less robust than velocity or gradient estimates [6]. Vmax and ΔPm assess LV ejection afterload. V_{max} and ΔPm are closely related. But ΔPm provides additional information about the distribution of transaortic velocities throughout the ejection, important in the identification of prolonged LV contraction caused by dysfunction. A reduction in AVA results in an increase in jet velocity, and consequent increase in the pressure gradient between the left ventricle and the aorta.

Based on these main indicators, the European Society of Cardiology (ESC) and the American College of Cardiology/American Heart Association (ACC/AHA) stratify AS in four main categories [6, 136]:

- High-gradient AS: AVA< 1 cm^2, $V_{max} \geq 4$ m/s and $\Delta Pm \geq 40$ mmHg. These patients are considered to have severe AS.
- Low-flow, low-gradient AS with reduced ejection fraction: AVA< 1 cm^2, $\Delta Pm < 40$ mmHg, left-ventricular ejection fraction (LVEF) $< 50\%$ and stroke volume index (SVi) $\leq 35\%$. These patients can be severe or

pseudosevere, if an increase to AVA> 1 cm^2 is seen in dobutamine stress echocardiography, with implications in prognostic value.

- Low-flow, low-gradient AS with preserved ejection fraction: AVA$<$ 1 cm^2, $\Delta Pm < 40$ mmHg, LVEF $\geq 50\%$ and SVi $\leq 35\%$. This group usually consists of elderly patients with smaller ventricular dimensions. This category is known paradoxical low-flow low-gradient AS, and its assessment of severity and prognosis remains challenging with contrasting results in the literature. This stenosis type has been found to have increased survival compared to reduced LVEF [119]. For those patients, calcium scoring is predictor of severe AS [7] as well as mortality [133].

- Normal-flow, low-gradient AS with preserved ejection fraction: AVA$<$ 1 cm^2, $\Delta Pm < 40$ mmHg, LVEF $\geq 50\%$ and SVi $> 35\%$, which mostly corresponds to moderate AS.

A comprehensive diagnosis requires combining these descriptors with others such as symptomatic status, advanced age, coronary heart disease, as well as indicators of flow rate and LV function, including the presence of hypertrophy and assessment of the longitudinal function.

6.1.3. Indications for aortic valve intervention

Aortic valve intervention (AVI) is a class-I indication for high-gradient AS (ΔPm $\geq$ 40 mmHg) in the presence of symptoms or LV dysfunction (LVEF $< 50\%$) [6, 136]. The majority of patients with severe AS present with high transaortic pressures and largely benefit from AVI. Most low-gradient AS patients are expected to benefit from intervention, following careful confirmation through multislice computerized tomography (MSCT) calcium scoring and exclusion of a pseudosevere pattern.

Lack of effectiveness for all sub-groups Although, aortic valve replacement or repair are effective for the majority of AS patients, there are sub-groups for whom the benefit of AVI remains unclear. Uncertainty persists regarding the application of current recommendations, especially in symptomatic low-gradient or asymptomatic patterns. Predicting the long-term outcome of intervention in these patients remains particularly challenging, despite significant advancements on their characterization [13, 12, 119]. In particular, it is necessary to improve the detection of patients with rapidly progressing stenosis but which are likely benefit from prophylactic aortic valve replacement (AVR) [133].

Discordant grading and unresolved thresholds Assessing low-gradient AS is complicated by the occurrence of a discording grading pattern, characterized by deficient valvular area but physiological velocities and pressure gradients. A ΔPm of 40 mmHg actually correlates with an AVA of 0.8 cm^2, rather than the 1 cm^2 threshold observed in the guidelines [123]. This discordant classification may affect up to 40% of the AS patients [31], and is mostly attributed to errors in the measurement of transaortic velocities, which are flow-dependent, and to the underestimation of AVA [123]. The latter is caused by assuming a spherical left-ventricular outflow tract (LVOT) in the estimation of LVOT diameter from 2D-echo [178].

Threshold values at variance with the guidelines were also proposed for LVEF, in patients with asymptomatic AS. While AVI is currently recommended in this context for LVEF $< 50\%$, a 55% threshold was recently found to be a significant marker of poor outcome [13]. Overall, research on the prognostic value of current descriptors and their corresponding thresholds of AS has yet to be exhausted.

Optimal timing for intervention On another note, the optimal timing for intervention remains a topic of research. Intervening too early exposes patients to unnecessary risk of complications [56]. On the other hand, performing surgery too late is associated with adverse outcomes, because of advanced LV dysfunction [56, 107]. Moreover, each center needs to take into account location/surgeon-specific risk factors. The optimal timing for intervention in severe high-gradient asymptomatic AS with preserved LVEF remains controversial. If the patients have low surgical risk, it has been shown that an early intervention has significantly increased survival compared to conservative management despite the absence of symptoms [12].

6.2. Objectives and methodology

As explained in previous chapters, interpretable machine learning (ML) approaches construct models whose predictions are accompanied by meaningful explanations. DTs are a type of interpretable model, which displays a hierarchy of tests on the input descriptors. For this reason, DT have the potential for providing insight in the workup of AS, whose assessment is complicated by the existence of numerous predictors, discordant grading patterns, a variety of disease mechanisms, and an unclear progression in time. Moreover, DT models have the potential for easy integration into the guidelines, as existing recommendations are often organized in tree structures. The objective of the study described in this chapter is to evaluate

the merit of interpretable ML, specifically DT models, in the risk stratification of AS. In particular, we focus on a retrospective assessment of the expected improvement with AVI.

6.2.1. Study population

We conducted a retrospective analysis of 2761 patients (1347 men and 1414 women, between 23 and 100 years old, median 79), enrolled in a single-center AS registry between 2000 and 2016, in Cliniques Universitaires Saint-Luc in Brussels, Belgium. The registry was stored in an electronic database and composed of 45 baseline demographics, clinical features and echocardiographic data obtained at the time of inclusion. Follow-up information, including cardiac events such as AVI and death from cardiovascular causes, were provided by the investigators until December 2017.

6.2.2. Decision tree learning

DT learning was employed with the aim of predicting AS natural-history mortality from cardiovascular causes, using baseline measurements at the time of inclusion as predictors. The task was modeled as the classification of cardiovascular mortality within six years (6y) of inclusion in natural-history. The class labels were survival and death from cardiovascular causes within six years of inclusion. For training and testing the classification models, patients who underwent AVI were censored at the date of intervention, while patients who did not undergo surgery and were alive or lost to follow-up were censored at the date last seen alive. After censoring and performing class-stratified sampling, the training set consisted of 411 records, while the test dataset contains 138 records. Redundant descriptors were removed and body surface area (BSA) was added, so that a total of 39 features were used in the model as described in Table 6.1.

The DT algorithm was optimized for maximum prediction performance and easy interpretability, by favoring models with high average $F1$ score over the two classes (survival and death) and minimum number of leaves. The average $F1$ is defined in Equation 4.4.1. This was done using 10-fold cross validation (CV) on the training dataset. The employed algorithm was the C4.5, using binary splits for all variables, the information gain criterion for split selection, and error-based pruning (EBP) with the Laplace correction, as described in Chapter 2. In addition to setting the extent of pruning, the parameters of the algorithm also controlled the repetition of variables along a branch, and early stopping criteria. The repetition of variables was

controlled by reducing the information gain of a candidate test function by a constant factor, if an upstream node contained a test on the same variable. The early stopping criterion was requiring a minimum number of instances on the both resulting nodes of a split. On top of CV, the tuning of these parameters also took into consideration feedback from a expert in cardiology. The finally selected values for the pruning confidence factor, repetition-avoidance factor and minimum number of split instances were 0.87, 0.7, and 20, respectively. The DT constructed with these parameters was finally evaluated based on the predictions obtained for the test set.

6.2.3. Statistical analysis

Each leaf of a DT represents a sub-group of the population. Therefore, the constructed DTs was used to stratify all individuals of the registry. Two distinct analyses were done in this regard.

Natural-history survival The first analysis was a characterization of the natural history of AS for the sub-populations at each leaf of the DT. The main aim is to investigate if the DT makes sense for the assessment of stenosis severity, in light of existing knowledge. First, survival estimates were obtained for the complete study population and for the sub-groups at each DT node, using the Kaplan-Meier (KM) estimator. The data was right-censored for intervention and death. The survival of the patient groups at each node was compared to the survival of the entire population. Nodes presenting significantly different survival estimates were selected for discussion. Two survival curves were considered significantly different when presenting $p<0.005$ in a pairwise logrank test. We also display the estimated survival of the general population at each node, using statistics matched for age and gender. A univariate and multivariate Cox regression analysis was then performed considering each leaf as an independent categorical variable.

Survival after intervention The second analysis consisted in a retrospective evaluation of the survival following AVI, as predicted by the tree models. For each subgroup within the DT, the survival of patients that were and were not referred for intervention within six months of inclusion (VI and no-VI, respectively) was compared using the KM method. An intervention included surgical or transcatheter replacement or repair. We observed for which nodes the survival curves were significantly different for the VI and no-VI groups, using pairwise logrank tests ($p \geq 0.005$).

Table 6.1: Description of variables of the AS registry

Variable	Type	Description
Gender	Nominal categorical	0 (female), 1 (male)
Age	Continuous	years
BSA	Continuous	m^2
New York Heart Association (NYHA) class	Ordinal categorical	I, II, III, IV
Angina	Nominal categorical	0 (no), 1 (yes)
Syncope	Nominal categorical	0 (no), 1 (yes)
Diabetes	Nominal categorical	0 (no), 1 (yes)
Arterial hypertension	Nominal categorical	0 (no), 1 (yes)
Coronary heart disease	Nominal categorical	0 (no), 1 (yes)
Myocardial infarction	Nominal categorical	0 (no), 1 (yes)
History of arrhythmia/atrial fibrillation	Nominal categorical	0 (no), 1 (yes)
Charlson comorbidity index	Continuous	
Brain natriuretic peptide	Continuous	BNP
Serum creatinine	Continuous	μmol/l
Systolic blood pressure	Continuous	mmHg
Diastolic blood pressure	Continuous	mmHg
Heart rate	Continuous	bpm
LV end-diastolic diameter	Continuous	mm
LV end-systolic diameter	Continuous	mm
Posterior wall thickness (PWT) in diastole	Continuous	mm
Interventricular septum wall thickness in diastole	Continuous	mm
LV outflow tract diameter	Continuous	mm
AVA	Continuous	cm^2
Indexed aortic valve area (AVAi)	Continuous	cm^2/m^2
Stroke volume (SV)	Continuous	ml/beat
$\triangle$Pm	Continuous	mmHg
Vmax	Continuous	cm/s
Pulmonary artery pressure	Continuous	mmHg
LV end-diastolic volume	Continuous	ml
LV end-systolic volume	Continuous	ml
LVEF (Simpson)	Continuous	%
E/e’ ratio	Continuous	
Left-atrial diameter	Continuous	mm
Left-atrial volume (biplane,)	Continuous	ml
Aortic regurgitation	Nominal categorical	0 (no), 1 (yes)
Mitral regurgitation	Nominal categorical	0 (no), 1 (yes)
Tricuspid regurgitation	Nominal categorical	0 (no), 1 (yes)
Calcium score	Continuous	
Associated bypass	Nominal categorical	0 (no), 1 (yes)

6.3. Results

The selected DT is displayed in Figure 6.1 and its prediction performance is reported in Table 6.2. The classification accuracy of the tree on the test population was approximately 71% with sensitivity and PPV of 73.3% and 73.0%, respectively. The top predictor variables were AVA, Vmax, age, LV PWT in diastole, and creatinine. Additional selected criteria were presence of arterial hypertension, ΔPm, and SV.

Natural-history survival The estimated natural-history survival is reported in Table 6.3. The sub-groups selected for having significantly different survival compared to the population average were the nodes labeled 4, 9, and 10, corresponding to the patient groups:

- Group 4: AVA $\geq$ 0.9 cm^2, age $<$ 73 years, and creatinine $<$ 1.6 μmol/l
- Group 9: AVA $\geq$ 0.9 cm^2, age $<$ 88 years, and creatinine $\geq$ 1.6 μmol/l
- Group 10: AVA $\geq$ 0.9 cm^2, and age $\geq$ 88 years

The survival curves of these nodes are displayed in Figure 6.2. Node 4 displayed significantly longer survival compared to the population average, while in nodes 9 and 10 the opposite was observed.

Survival after intervention The comparison between the survival of patients who were or were not referred to valve intervention within six months of inclusion is summarized in Table 6.4. For the groups in nodes 2, 3 and 5, there was a significant increase in survival for the patients who underwent intervention ($p<0.005$). The difference in median survival consisted of $>$8.0, 5.7, and $>$5.7 years, respectively. The respective hazard ratios were nearly halved from 2.0 to 1.1, from 1.8 to 0.9, and from 1.2 to 0.3. In nodes 7 and 10, intervention also seemed correlated to increased survival but without enough statistical support. For the remaining nodes (1, 4, 6, 8 and 9), a significant improvement in survival following AVI could not be ascertained. The survival curves for these nodes are displayed in Figure 6.3. The patient groups at those nodes are characterized as:

- Group 1: AVA $<$ 0.9 cm^2 and Vmax $<$ 325.0 cm/s
- Group 4: AVA $\geq$ 0.9 cm^2, age $<$ 73 years, and creatinine $<$ 1.6 μmol/l

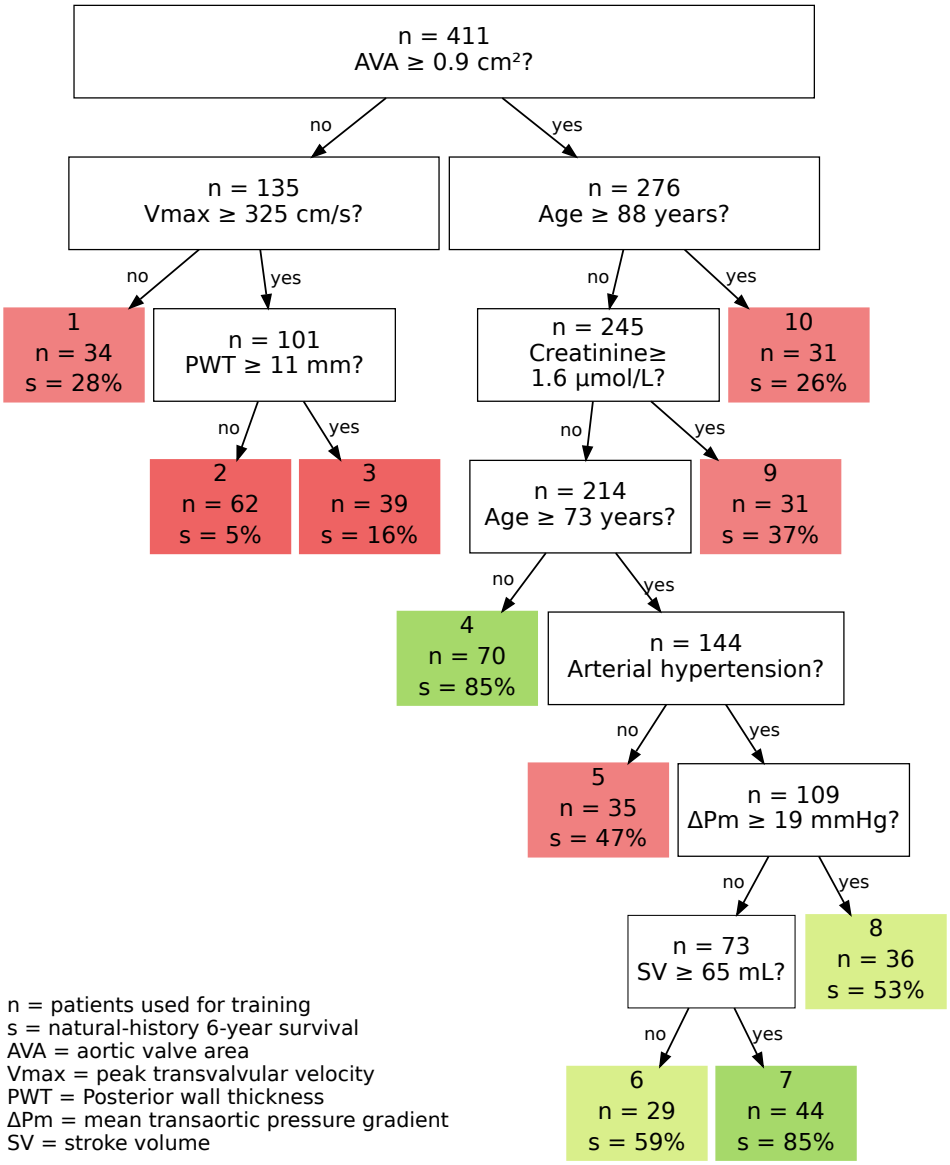


Figure 6.1: Selected decision tree for the prediction of 6-year survival from death due to cardiovascular causes in natural-history of aortic stenosis. The number of patients and survival rate at each node refer to the population used for training. The leaf nodes are labeled from 1 to 10, where s estimated the probability of death due to cardiovascular causes within 6 years of admission.

Table 6.2: Performance of the decision tree (DT) constructed for the prediction of six-year natural-history (NH) survival of aortic stenosis (AS), estimated on the test dataset described in section 6.2.2.

Metric	Estimate (%)	95% CI
Accuracy	70.7	63.2 - 78.3
Survival recall	67.7	56.1 - 79.2
Survival precision	68.1	56.5 - 79.6
Survival F-score	67.9	49.6 - 86.1
Death recall	73.3	63.3 - 83.3
Death precision	73.0	62.9 - 83.0
Death F-score	73.2	57.3 - 89.0

- Group 6: AVA $\geq$ 0.9 cm^2, age between 73 and 88 years, creatinine $<$ 1.6 μmol/l, hypertension, and ΔPm $<$ 19 mmHg, and SV $<$ 65 ml
- Group 8: AVA $\geq$ 0.9 cm^2, age between 73 and 88 years, creatinine $<$ 1.6 μmol/l, hypertension, and ΔPm $\geq$ 19 mmHg
- Group 9: AVA $\geq$ 0.9 cm^2, age $<$ 88 years and creatinine $\geq$ 1.6 μmol/l

6.4. Discussion

Two of the top predictors selected by the DT algorithm, AVA, Vmax and age, were consistent with their central role in the AS diagnostic workup and recommended cut-off values. The selected thresholds for AVA and Vmax were respectively 0.9 cm^2 and 325 cm/s, while the values currently used to classify severe AS are 1.0 cm^2 and 400 cm/s. In the DT, Vmax was primarily selected in favor of ΔPm. The former appeared in level 2 affecting nearly 33% of the population, while the latter appeared at level 6, applied to 27% of the patients. Incidentally, Vmax is considered the strongest predictor of clinical outcome [140]. It is measured directly from the continuous-wave Doppler flow signal, while ΔPm is computed from the velocities using the simplified Bernoulli equation. For hypertensive patients, the threshold value found for ΔPm was 19 mmHg, suggesting that a lower cut-off on ΔPm may be of value in this scenario. This is expected, as these ventricles contract against an increased afterload.

Another predictor selected by the DT algorithm was LV PWT, for patients with small valvular area and elevated transvalvular velocities. The pattern described by nodes 2 and 3 seems to coincide with the fact that patients with larger PWT, and therefore more likely to have compensated

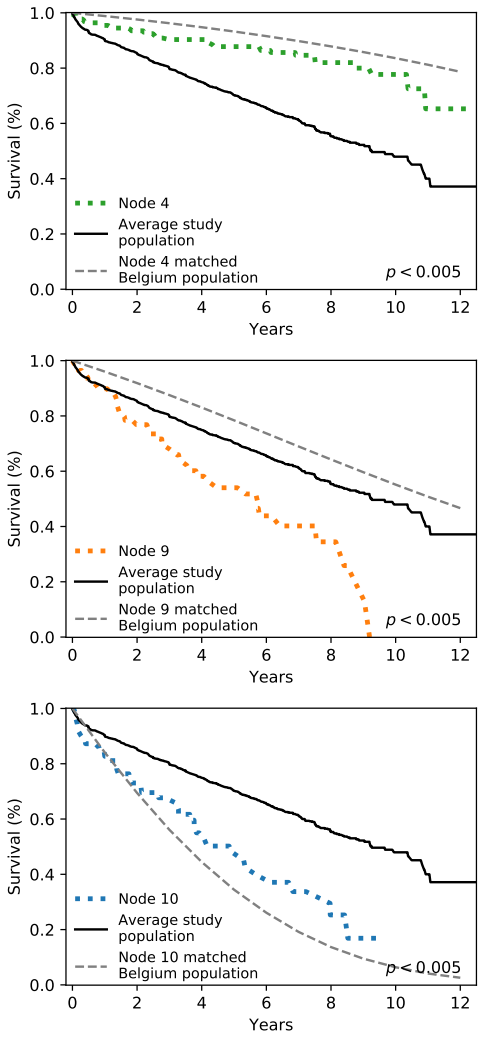


Figure 6.2: Kaplan-Meier cardiovascular survival estimates for selected nodes of the decision tree in Figure 6.1. Nodes were selected for having significantly different survival compared to the average study population ($p<0.005$).

LV function, have better outcome compared to patients with thinner ventricles. In addition, this threshold is coherent with findings of a recent survey, highlighting the prognostic value of imaging biomarkers of remodeling and myocardial fibrosis [133]. Nonetheless, the decision to perform AVI in the patients of groups 2 and 3 is less challenging, as they approach the high-gradient regime.

Table 6.3: Kaplan-Meier (KM) estimates for the natural-history of aortic stenosis in all individuals of the registry, estimated at each leaf of the decision tree ($p<0.005$ in the multivariate logrank test comparison). The KM confidence intervals are taken at the last point of the curve. The survival of the population at each node is compared to the average survival of the study population.

Group		Survival estimates (years)					Univariate analysis		Multivariate analysis*	
		Q1	Q2	Q3	95%CI	p-value	HR (95%CI)	p-value	HR (95%CI)	p-value
All individuals		4.0	9.3	12.5	0.2					
Node 1	No significant difference	2.7	7.2	11.1	0.2	0.199	1.2 (0.4)	0.142	1.0 (0.3)	0.898
Node 2		3.2	8.3	11.7	0.3	0.021	1.3 (0.2)	0.001	1.2 (0.2)	0.014
Node 3		4.0	9.9	12.5	0.4	0.948	1.0 (0.2)	0.848	1.0 (0.4)	0.825
Node 5		4.5	8.1	10.9	0.3	0.944	1.0 (0.4)	0.932	1.0 (0.2)	0.797
Node 6		5.1	10.8	11.4	0.6	0.389	0.8 (0.4)	0.353	0.5 (0.3)	0.011
Node 7		7.5	10.9	10.4	0.3	0.010	0.5 (0.3)	0.007	0.4 (0.2)	<.001
Node 8		3.6	10.5	12.0	0.4	0.981	1.0 (0.3)	0.916	0.9 (0.3)	0.307
Node 4	Significant difference	10.3	>12.4	>12.4	0.3	<.001	0.3 (0.1)	<.001	0.9 (0.4)	0.512
Node 9		2.4	5.7	8.5	0.4	<.001	1.8 (0.6)	<.001	2.1 (0.7)	<.001
Node 10		1.7	4.9	8.4	0.3	<.001	2.2 (0.7)	<.001	1.0 (0.3)	0.784

HR=hazard ratio, CI=confidence interval, Q1, Q2, Q3=25%, 50%, 75% quantiles.
*Multivariate analysis adjusted for gender, age, body surface area, hypertension, coronary heart disease, myocardial infarction, and history of atrial fibrillation

Natural-history survival The nodes whose populations stood out as having significantly different natural-history survival compared to the general population meet the expectations regarding the encompassed subgroups. The patients at node 4 had significantly greater survival compared to the average of the study population, which is explained by their comparatively younger age. Indeed, their survival was close to the survival of the matched Belgian population. Nodes 9 and 10 respectively represent populations with renal dysfunction or advanced age, whose prognosis is predictably worse than the entire-population average.

The patients of group 10 seem to have identical survival when compared to the matched Belgian population. The matched population estimate is likely to reflect other factors affecting such an elderly population. The difference between the survival of matched Belgian population and the study average in node 4 needs further investigation, since these patients appear to have moderate stenosis.

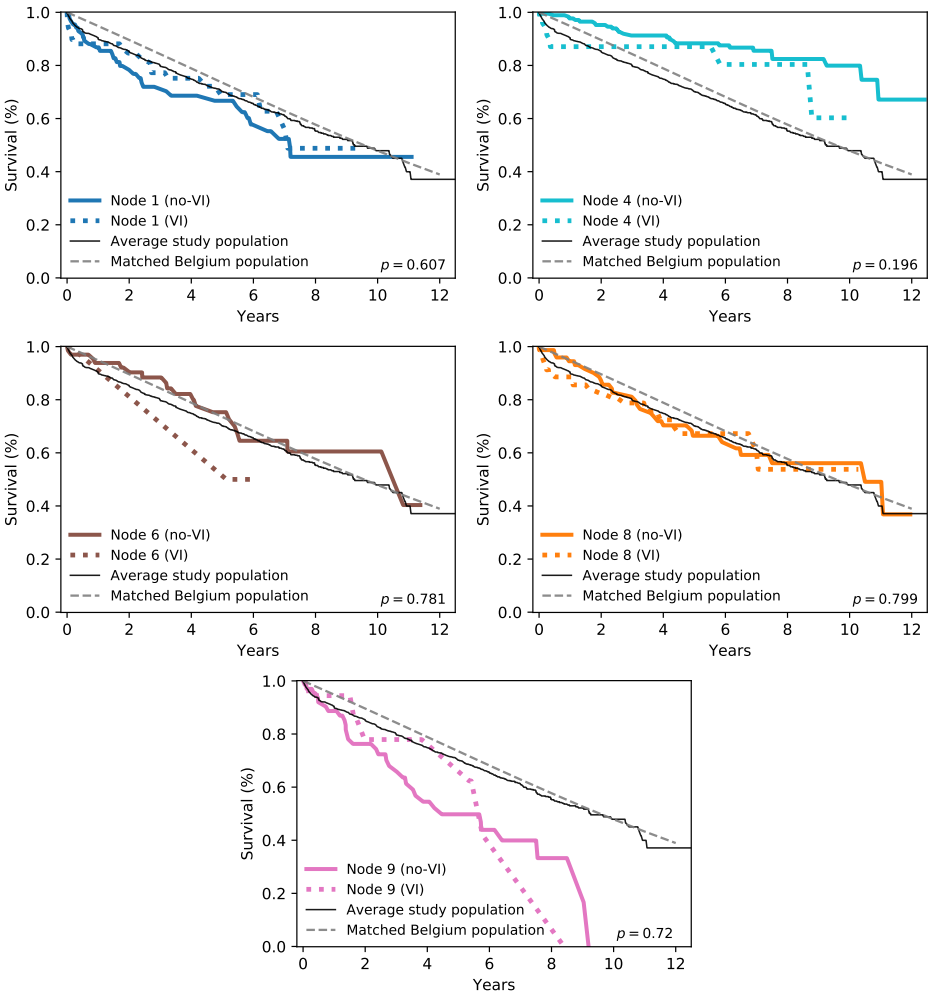


Figure 6.3: Comparison of the Kaplan-Meier survival estimates for the patients who did not undergo aortic valve intervention within six months of inclusion (no-VI), and those who did (VI), for the nodes where the two populations did not have significantly different outcomes ($p \geq 0.005$).

Survival after intervention When assessing the survival of patients were or were not referred to intervention within 6 months of inclusion, we saw that the patients with AVA$<$0.9 cm^2 and Vmax$\geq$325.0 cm/s were associated with positive outcomes. This goes in agreement with the recommendations for intervention in high-gradient AS. In the presence of this AS pattern, the decision to intervene focuses on surgical risk, timing and type of intervention.

Table 6.4: Difference in Kaplan-Meier (KM) survival estimates between individuals who were referred to aortic valve intervention (VI) and individuals who were not (no-VI). The confidence of the differences was estimated by propagating the uncertainty of the confidence intervals taken at the last point of each KM curve.

Group		Difference in survival (years)				Univariate analysis no-VI		Univariate analysis VI	
		ΔQ1	ΔQ2	95%CI	p-value	HR (95%CI)	p-value	HR (95%CI)	p-value
All individuals		4.0	9.3	12.5	0.2				
Node 1	No significant difference	1.9	-0.1	0.3	0.607	1.2 (0.4)	0.414	1.2 (0.6)	0.347
Node 4		-1.7	-	0.4	0.196	0.2 (0.1)	$<.001$	0.6 (0.5)	0.161
Node 6		-5.0	-5.7	0.5	0.781	0.7 (0.3)	0.104	1.5 (5.1)	0.709
Node 7		-1.5	>1.5	-	0.591	0.5 (0.2)	0.001	0.0 (-)	0.991
Node 8		0.3	>1.9	0.3	0.799	0.8 (0.3)	0.391	1.2 (0.5)	0.405
Node 9		1.6	1.3	0.4	0.720	1.6 (0.6)	0.008	1.7 (1.5)	0.198
Node 10		7.8	>8.2	0.4	0.305	2.0 (0.7)	$<.001$	1.0 (3.3)	0.963
Node 2	Significant difference	2.9	>8.0	0.2	<0.001	2.0 (0.5)	$<.001$	1.1 (0.2)	0.386
Node 3		3.5	5.7	0.3	<0.001	1.8 (0.6)	$<.001$	0.9 (0.2)	0.598
Node 5		6.7	>5.7	0.4	0.003	1.2 (0.5)	0.364	0.3 (0.4)	0.035

HR=hazard ratio, CI=confidence interval, ΔQ1, ΔQ2 = Difference in the 25% and 50% KM quantiles between the survival of VI and no-VI populations at each node.

One of the patient groups for which intervention was not associated with improved survival was that of node 1. This group corresponds to a discordant grading pattern, characterized by physiological velocities despite reduced AVA. Indeed, according to the guidelines, the decision in favor of intervention in the low-gradient AS is not straightforward. The proposed DT does not provide enough stratification of this patient group, given the small number of available training samples at the node (n = 34). For this reason, and given the complexity of this AS pattern, we propose that another DT model be constructed specifically for this group, e.g. by changing the prediction task from 6 to 3 years. This would allow a reduction in the number of censored patients and achieve a larger training dataset.

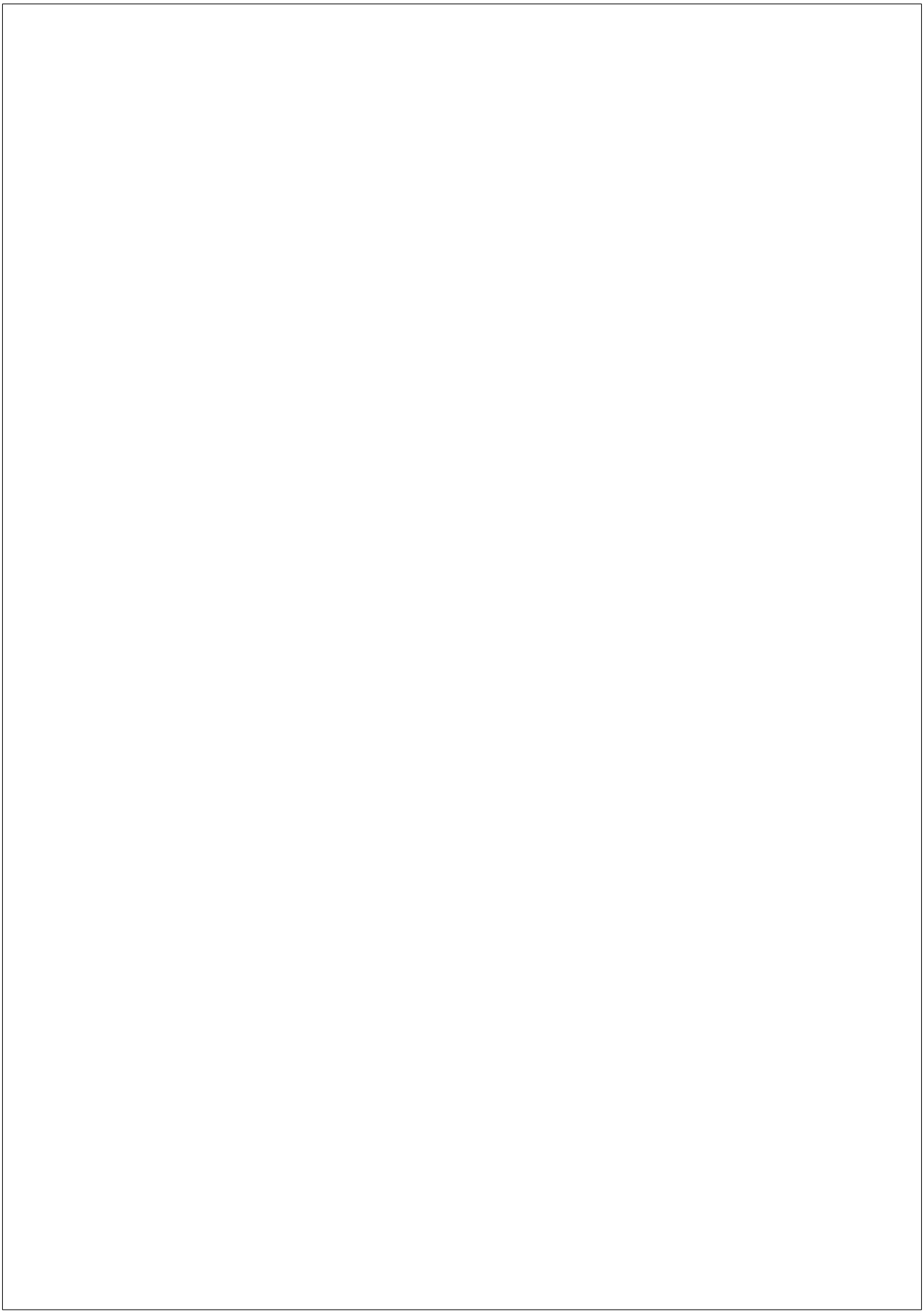
The results for subgroups 8 and 9 suggest a negative association of AVI with hypertension and renal dysfunction, respectively. Both pressure overload and kidney failure are important parameters in the characterization of cardiac state, suggesting that the patients in group 9 may be at an advanced stage of LV dysfunction. The results for these populations call for further investigation into the prognostic value of blood pressure and serum creatinine is characterizing cardiovascular health associated with positive or negative outcomes of aortic valve intervention.

On a last note, SV appears to be an important indicator of the outcome of intervention for hypertensive patients with normal AVA and ΔPm. This can be observed by contrasting the survival of the populations at nodes 6 and 7. Despite the limited statistical support, the difference between the survival of operated patients with SV$<$65% and SV$\geq$65% was of more the 6 years.

6.5. Conclusions

The predictors and corresponding threshold values learned by the DT algorithm were coherent with existing recommendations. AVA and Vmax were prioritized over ΔPm, which was the chosen descriptor for hypertensive patients. The cut-off values selected for valve area and peak velocity were consistent but slightly lower than currently-employed values. Specifically, the ΔPm threshold selected for hypertensive patients was significantly lower the 40 mmHg value, currently considered in the guidelines. The other tests at the nodes focuses on PWT, serum creatinine and history of hypertension and SV, reinforcing the importance of the characterization of ventricular remodeling and function in the assessment of the progression of stenosis.

The data suggest that valve intervention may not improve the survival of certain AS subgroups, specifically in the presence of a discordant grading pattern, or in association with renal dysfunction or hypertension. These patient groups therefore deserve further investigation. Since the perspective on valve intervention is retrospective, the associations found between the DT paths and AS survival cannot be interpreted as causal. They do however identify several AS subgroups, where the pathology profile and effectiveness of intervention deserve further study. The analysis demonstrates the ability of DT learning approaches to build explainable models grounded on data, and which could be used as clinical research or decision-support tools with potential for integration in the guidelines.



Chapter 7

CONCLUSIONS

This thesis set out to improve decision tree (DT) learning algorithms, motivated by the growing need for interpretable machine learning (ML), the challenges presented by medical data, and the potential for application of ML in clinical decision scenarios. The following lines summarize the findings of the preceding chapters.

7.1. Overview and contributions

The application of ML to the medical domain is a field whose advent has lingered for more than 20 years. In agreement with recent reflections by the community, this thesis assumes that shifting the focus towards learning approaches that can be understood, such as DTs, and that can incorporate and show coherence with existing knowledge is the way forward to change this reality. In other words, prediction performance, interpretability, and complementarity with existing practice/knowledge are three of the most important requisites for prediction models to be accepted by experts and be useful in practice. It is also crucial to understand the behavior of the models when handling data with different properties. The present thesis can be therefore regarded from these four perspectives:

Prediction performance As in other domains, mining medical data is sometimes complicated by the limited size of the databases, the uncertainty of the measurements, or the heterogeneity of the variables. Noisy measurements can reduce DT performance, as interpretable trees prompt decisions based on strict thresholds. Existing approaches to deal with this problem either did not account for the uncertainty during the training phase, or they did so with significant increases in learning time, or they

made no distinction between the noise model used for training and evaluation. Moreover, there was no conclusive evaluation of the impact of noise on DT performance. Chapter 3 proposed a probabilistic DT learning approach that models measurement noise as a distribution in the distinct phases of learning: (1) searching for the split functions, (2) propagating the training instances down the tree, and (3) generating predictions for unseen instances.

The soft training approaches showed consistent improvements in prediction performance in the experiments, approximately in the range of 1% to 4% for increasing noise levels. On one hand, this could be explained by the effect of smoothing of the information gain criterion during the search, that pushes the algorithm to select ”less extreme" cut-offs. On the other hand, it could be be explained by the smoothing of the class probability estimates at the leaves, as a consequence of performing soft splits on the training data. While both methods can lead to the selection of different split thresholds compared to hard search, the soft training propagation seems more likely to impact the pruning approach, causing it to cut more branches that would otherwise not generalize well. Compared to an existing approach, the proposed soft search kept the computational time under control.

The algorithm proposed in Chapter 4 approached the improvement of prediction performance from a different perspective. Standard DT algorithms follow a locally-optimal strategy which, despite proving themselves useful, has no guarantees regarding the optimality of the output. We therefore hypothesized that it might be possible to trade-off some computational efficiency for performance. Existing non-greedy methods either construct multivariate trees, or they impose a fixed structure *a priori*. As such, we proposed a heuristic DT learning algorithm based on Monte Carlo tree search (MCTS), called UCT-DT, which performs a broader exploration of the space of univariate DTs. The main components of the algorithm are the (1) Markov decision process used to model the DT construction, (2) the approach to generate candidate actions, (3) the simulation policy, and (4) the search-pruning strategy. In our MDP, each state represents a DT, and the value of each state therefore represents the expected performance of the DT we would build by starting at the current state and following a given policy.

In order to compare the proposed algorithm with locally-optimal search, we focused on the performance of a single DT. Overall, the UCT-DT algorithm was able to outperform the prediction performance of greedy search in many of the datasets, although without sufficient statistical support. Its most significant win was the reduction in DT size. The method was su-

perior to C4.5 in both regards whenever the dataset had more than 1000 instances. The search-pruning strategy proved itself essential in reaching states leading to better DTs. Specifically, pruning states based on value estimates seems to have benefited prediction performance. On the other hand, removing states based on the C4.5 DT-pruning algorithm prioritized the trade-off between performance and DT size. The best option appears to be a combination of both strategies. Value-based pruning was only useful for larger datasets, which could be explained by the otherwise insufficient size of the data subset used to estimate the value. Computing the prediction performance on an independent data sample was also a better way of simulating the performance of a given state, compared to completing the DT with C4.5.

The results obtained with the proposed soft learning approaches and UCT-DT bring us back to the central challenge of learning a model: that of correctly estimating its ability to generalize. In DT learning, such estimations take place in two moments: when selecting the test functions and deciding upon removing nodes. Expressing less confidence in the data on one hand, and performing a broader search in the space of DTs, on the other, seem to have contributed to more assertive estimates of the performance of a given test function, branch or tree when extrapolated to unseen data.

Model explainability One of the dimensions of model explainability is the size of the model, as ”smaller" models are more likely to be easily understood. In this regard, one of the main achievements of the proposed soft training methods described in Chapter 3 was the significant reduction in the size of the output decision trees. As discussed above, this was likely caused by the smoothing effect on the information gain and class-probability estimates, which allowed cutting off more branches which did not generalize well. It therefore appears that the way in which the uncertainty model was realized allowed the algorithm to express our confidence in the data. Both the soft search and the soft propagation of the training instances led to improvements in the number of leaves while maintaining or improving accuracy. This result could not be obtained by changing the extent of pruning, which is the current way of letting DT algorithms known how much we (do not) trust the data. The UCT-DT approach also led to improvement in the number of leaves for most of the datasets, suggesting that it was able to direct the search towards more optimal states.

Although minimizing the size of a model usually correlates with increased interpretability, it is a common oversimplification of the concept. For this reason, Chapter 5 described a measure to quantify the similar-

ity between DTs. According to a validation on a forest of trees generated through UCT-DT, the method appears to have been able to capture DT structure. This contribution represents a modest step in the direction of a more pertinent assessment of interpretable models, and away from accuracy-oriented design.

Integration of existing knowledge and acceptance by experts Medical training, research and practice is an immense source of implicit and explicit knowledge. For example, clinicians have plenty of experience on judging the reliability of a given measurement and predicting its impact on the final decision. One of the desirable properties of learning algorithms is therefore the ability to include knowledge about the reliability of the measurements, coming from the experience or clinical studies. In this regard, the the soft training approaches offer the possibility of specifying any distribution of noise for the input variables. As a matter of fact, in the experiments of Chapter 3, we observed some correlation between the uncertainty distributions incorporated into the learning algorithm and the distributions used to corrupt the data used from training/evaluation. As a consequence, the method is likely to yield better results if more realistic distributions of noise are employed. It is thus recommended that each clinical measurement be accompanied by an assessment of its reliability.

In addition to prediction performance and interpretability, coherence with background knowledge is another important factor in the acceptance of the models by experts. As such, we constructed and investigated the pertinence of a DT for risk assessment in aortic stenosis, a condition whose diagnosis and therapeutic decision-making are intricate. The analysis indicated that the test functions selected as the nodes were consistent with currently employed predictions, and a stratification of the population according to the tree discerned groups at higher or lower risk, as expected. A survival analysis based on the constructed DT further identified some disease groups whose decision for valve intervention did not bring about a clear benefit. The analysis illustrates the potential role of DT learning as an asset for clinical research and decision-making.

Understanding decision tree algorithms It is also important to know what to expect when training a model on given dataset, depending on its properties. In particular, there was no previous conclusive account on how the uncertainty in the measurements used for training or testing impacts DT algorithms. In Chapter 3, experiments with increasing noise levels confirmed that DT learning is much more robust to noise in the training data than noise in the test data, as had been suggested in earlier experiments.

Indeed, the probability of learning an accurate model from several noisy instances is greater than the probability of generating correct predictions for individually-noisy examples, since the noise in the training dataset may average out. One could expect this to be the case for any learning algorithm, which could be clarified by further experiments.

When employing a learning algorithm, it is also important to understand the impact of dataset size on the expected result. The number of instances used for training was an important factor in determining the value of the UCT-DT algorithm compared to greedy search. The method relies on accurate estimates of DT performance to select candidate test function, guide the search, and remove unpromising states. It appears that if the data sample used to obtain those estimates is too small, the estimates are less likely to be generalizable. In that case, C4.5 is more likely to provide better results at a cheaper cost. Two other dataset properties that were briefly investigated were the presence of interactions between the input variables and the many-weak-input scenario, where the class variable cannot be approximated by a function of a small number of features. It appears that the C4.5 algorithm is not easily outperformed by alternative search approaches whenever the data has a small degree of feature interactions, or those interactions are not relevant for predicting the class. Similarly, the experiments indicate that datasets with a small proportion of relevant predictors are also less likely to benefit from the proposed MCTS approach. To conclude, the analysis indicates that top-down induction is well-suited for less complex domains or when only small datasets are available, while MCTS-based DT learning is otherwise more likely to find smaller, more accurate models.

7.2. Limitations and future perspectives

The above considerations are not without awareness about limitations and directions of improvement.

The soft approach proposed to generate predictions from DTs was not successful in handling noise. Smoothing of the class probabilities estimated by the DT caused more test instances to be incorrectly classified on average. The method may provide users with a class probability weighted according to the uncertainty, which can be useful in applications where we would like to see how our confidence in the data is reflected in the prediction. However, the method will not make more accurate predictions compared to hard evaluation. Another shortcoming of the proposed soft training/test approaches was that they did not account for the noise in categorical variables or the class variable. While the extension to categorical

variables only involves the specification of an appropriate noise distribution using the same framework, the extension to class noise would require another probabilistic context for expressing the uncertainty about its value. A final critique at the experiments of Chapter 3 is that they focused on accuracy as a metric of prediction performance, which can lead to falsely optimistic results in the presence of unbalanced classes.

On the side of efficiency, the UCT-DT approach leaves something to be desired. This issue is closely related to the number of iterations and, since the algorithm is anytime, a smaller number of iterations could have been chosen. This is a limitation of the evaluation described in Section 4.6, where the impact of the number of iterations and several other hyper-parameters was not assessed. Running time also calls for better heuristics to guide and prune the search. Indeed, MCTS algorithms have many variations. In UCT-DT, MCTS was employed with the Upper Confidence Bound (UCB) policy for node selection, based on the Upper Confidence Bound for Trees (UCT) algorithm. Perhaps it would be beneficial to experiment with the simultaneous optimistic optimization algorithm [128] or best-arm identification MCTS [101], recent improvements over UCT. A related thought is that the method possibly remains too tightly connected to the greedy approach, namely in the generation of actions which was done by selecting a set of locally-best test functions. Alternatively, a more exploratory approach could have been tried e.g. the random feature sampling approach used in random forests. Finally, although we have focused on the performance of a single DT, we did not take advantage of the entire search tree output by the UCT-DT algorithm. Between a single DT and an ensemble of numerous models, it could be useful to select a small set of DTs of interest for a specific application.

The distance measure proposed in Chapter 5 takes a first step in that direction, but the evaluation should be taken a step further to a concrete application. Envisioned applications include selecting a subset of models that represent distinct decision patterns, for example through the selection of DTs from different clusters. This would yield alternative DTs which may be distinctly suited for certain patient groups. Another possible application is the integration of structure-based DT selection with bagging or boosting. It has been shown that the error rate of a random forest is bounded by the correlation between the base models, in that case measured in terms of prediction coherence. We may thus hypothesize that selecting DTs in order to minimize the correlation between their structure would contribute to improving the performance of the ensemble.

Bibliography

[1] J. Alcalá-Fdez et al. “KEEL data-mining software tool”. In: *Journal of Multiple-Valued Logic and Soft Computing* 17.2-3 (2011), pp. 255–287.

[2] J. Angwin et al. *Machine Bias: There’s software used across the country to predict future criminals. And it’s biased against blacks.* 2016. URL: propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. “Finite-time analysis of the multiarmed bandit problem”. In: *Machine learning* 47.2-3 (2002), pp. 235–256.

[4] N. Augsten et al. “TASM: Top-k approximate subtree matching”. In: *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*. IEEE. 2010, pp. 353–364.

[5] R. C. Barros et al. “A survey of evolutionary algorithms for decision-tree induction”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.3 (2012), pp. 291–312.

[6] H. Baumgartner et al. “2017 ESC/EACTS guidelines for the management of valvular heart disease”. In: *European heart journal* 38.36 (2017), pp. 2739–2791.

[7] H. Baumgartner et al. “Recommendations on the echocardiographic assessment of aortic valve stenosis: a focused update from the European Association of Cardiovascular Imaging and the American Society of Echocardiography”. In: *European Heart Journal-Cardiovascular Imaging* 18.3 (2016), pp. 254–275.

[8] K. Bennett. “Global Tree Optimization: A Non-greedy Decision Tree Algorithm”. In: *Computing Science and Statistics*. 1994, pp. 156–160.

[9] G. L. Betow. “Equal Credit Opportunity Developments”. In: *Bus. Law.* 45 (1989), p. 1821.

[10] M. Biehl, B. Hammer, and T. Villmann. “Prototype-based models in machine learning”. In: *Wiley Interdisciplinary Reviews: Cognitive Science* 7.2 (2016), pp. 92–111.

[11] P. Bille. “A survey on tree edit distance and related problems”. In: *Theoretical computer science* 337.1-3 (2005), pp. 217–239.

[12] Y. Bohbot et al. “Asymptomatic Severe Aortic Stenosis With Preserved Ejection Fraction: Early Surgery Versus Conservative Management”. In: *Journal of the American College of Cardiology* 72.23 Part A (2018), pp. 2938–2939.

[13] Y. Bohbot et al. “Relationship between left ventricular ejection fraction and mortality in asymptomatic and minimally symptomatic patients with severe aortic stenosis”. In: *JACC: Cardiovascular Imaging* 12.1 (2019), pp. 38–48.

[14] R. O. Bonow et al. “2008 focused update incorporated into the ACC/AHA 2006 guidelines for the management of patients with valvular heart disease: a report of the American College of Cardiology/American Heart Association Task Force on Practice Guidelines”. In: *Journal of the American College of Cardiology* 52.13 (2008), e1–e142.

[15] B. E. Boser, I. M. Guyon, and V. N. Vapnik. “A training algorithm for optimal margin classifiers”. In: *Proceedings of the fifth annual workshop on Computational learning theory*. ACM. 1992, pp. 144–152.

[16] J. P. Bradford et al. “Pruning decision trees with misclassification costs”. In: *European Conference on Machine Learning*. Springer. 1998, pp. 131–136.

[17] L. Breiman. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pp. 123–140.

[18] L. Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[19] L. Breiman et al. *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group, 1984.

[20] L. A. Breslow and D. W. Aha. “Simplifying decision trees: A survey”. In: *The Knowledge Engineering Review* 12.1 (1997), pp. 1–40.

[21] C. E. Brodley and P. E. Utgoff. “Multivariate decision trees”. In: *Machine learning* 19.1 (1995), pp. 45–77.

[22] C. B. Browne et al. “A survey of Monte Carlo tree search methods”. In: *IEEE Transactions on Computational Intelligence and AI in games* 4.1 (2012), pp. 1–43.

[23] H. Bunke and K. Riesen. “Graph classification based on dissimilarity space embedding”. In: *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer. 2008, pp. 996–1007.

[24] W. Buntine and T. Niblett. “A further comparison of splitting rules for decision-tree induction”. In: *Machine Learning* 8.1 (1992), pp. 75–85.

[25] M. A. Carreira-Perpinán and P. Tavallali. “Alternating optimization of decision trees, with application to learning sparse oblique trees”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 1211–1221.

[26] R. Caruana. *Intelligible Machine Learning Models for HealthCare*. Talk at the Allen Institute for Artificial Intelligence (AI2). Accessed: April 2019. 2016. URL: youtube.com/watch?v=ezSG9GORF54.

[27] R. Caruana and A. Niculescu-Mizil. “An empirical comparison of supervised learning algorithms”. In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 161–168.

[28] A. J. Champandard. *Monte-Carlo tree search in TOTAL WAR: ROME II’s campaign AI*. AIGameDev.com. 2014. URL: aigamedev.com/open/coverage/mcts-rome-ii.

[29] G. Chaslot et al. “Monte-Carlo Tree Search: A New Framework for Game AI.” In: *Artificial Intelligence and Interactive Digital Entertainment*. 2008.

[30] K. J. Cios and G. William Moore. “Uniqueness of medical data mining”. In: *Artificial Intelligence in Medicine* 26.1-2 (2002), pp. 1–24. ISSN: 09333657.

[31] M.-A. Clavel, J. Magne, and P. Pibarot. “Low-gradient aortic stenosis”. In: *European heart journal* 37.34 (2016), pp. 2645–2657.

[32] D. A. Clifton et al. “Health Informatics via Machine Learning for the Clinical Management of Patients”. In: *Yearb Med Inform* 10.1 (2015), pp. 38–43.

[33] C. Clopper and E. Pearson. “The use of confidence or fiducial limits illustrated in the case of the Binomial”. In: *Biometrika* 26.4 (1934), p. 404.

[34] A. J. S. Coats and L. G. Shewan. *Inconsistencies in the development of the ESC Clinical Practice Guidelines for Heart Failure*. 2013.

[35] Committee for Practice Guidelines of the European Society of Cardiology. “Recommendations for Guidelines Production”. In: (2012).

[36] G. F. Cooper et al. “Predicting dire outcomes of patients with community acquired pneumonia”. In: *Journal of biomedical informatics* 38.5 (2005), pp. 347–366.

[37] T. Dalamagas et al. “A methodology for clustering XML documents by structure”. In: *Information Systems* 31.3 (2006), pp. 187–228.

[38] R. L. De Mántaras. “A distance-based attribute selection measure for decision tree induction”. In: *Machine learning* 6.1 (1991), pp. 81–92.

[39] E. D. Demaine et al. “An optimal decomposition algorithm for tree edit distance”. In: *ACM Transactions on Algorithms (TALG)* 6.1 (2009).

[40] J. Demsar. “Statistical Comparison of Classifiers over Multiple Data Sets”. In: *Journal of Machine Learning Research* 7.7 (2006), pp. 1–30. ISSN: 15324435.

[41] R. C. Deo. “Machine learning in medicine”. In: *Circulation* 132.20 (2015), pp. 1920–1930.

[42] J. D’hooge et al. “Two-dimensional speckle tracking echocardiography: standardization efforts based on synthetic ultrasound data”. In: *Eur Heart J Cardiovasc Imaging* 17.6 (2016), pp. 693–701.

[43] T. G. Dietterich et al. “Ensemble learning”. In: *The handbook of brain theory and neural networks* 2 (2002), pp. 110–125.

[44] P. Domingos. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. Basic Books, 2015.

[45] M. S. Donaldson, J. M. Corrigan, L. T. Kohn, et al. *To err is human: building a safer health system*. Vol. 6. National Academies Press, 2000.

[46] F. Doshi-Velez and B. Kim. “Towards a rigorous science of interpretable machine learning”. In: *arXiv preprint:1702.08608* (2017).

[47] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, 2012.

[48] J. Dvorák and P. Savický. “Softening Splits in Decision Trees Using Simulated Annealing”. In: *Adaptive and Natural Computing Algorithms, 8th International Conference, ICANNGA 2007, Warsaw, Poland, April 11-14, 2007, Proceedings, Part I*. 2007, pp. 721–729.

[49] K. Dwyer and R. Holte. “Decision tree instability and active learning”. In: *European Conference on Machine Learning*. Springer. 2007, pp. 128–139.

[50] D. M. Eddy. “Practice policies: where do they come from?” In: *Jama* 263.9 (1990), pp. 1265–1275.

[51] J. Elith, J. R. Leathwick, and T. Hastie. “A working guide to boosted regression trees”. In: *Journal of Animal Ecology* 77.4 (2008), pp. 802–813.

[52] *Equal Credit Opportunity Act*. 12 X C.F.R. § 1002.9(b)(2). Revised 2011.

[53] European Society of Cardiology. *ESC Pocket Guidelines App*. Accessed: April 2019. URL: escardio.org/Guidelines/Clinical-Practice-Guidelines/Guidelines-derivative-products/ESC-Mobile-Pocket-Guidelines.

[54] European Union. *Data Protection Directive*. Directive 95/46/EC of the European Parliament and of the Council on the Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of Such Data, Official Journal L 281, 23/11/1995 :0031–0050. 1995.

[55] European Union. *General Data Protection Regulation*. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016, repealing Directive 95/46/EC, Official Journal L 119, 4.5.: 1–88. 2016.

[56] R. J. Everett et al. “Timing of intervention in aortic stenosis: a review of current and future strategies”. In: *Heart* 104.24 (2018), pp. 2067–2076.

[57] F. Fischer et al. “Barriers and strategies in guideline implementation—a scoping review”. In: *Healthcare*. Vol. 4. 3. Multidisciplinary Digital Publishing Institute. 2016, p. 36.

[58] P. Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

[59] P. Flach. “The geometry of ROC space: understanding machine learning metrics through ROC isometrics”. In: *Proceedings of the 20th international conference on machine learning (ICML-03)*. 2003, pp. 194–201.

[60] T. Foley et al. “Measuring left ventricular ejection fraction-techniques and potential pitfalls”. In: *European Cardiology* 8.2 (2012), pp. 108–114.

[61] A. A. Freitas. “Comprehensible classification models: a position paper”. In: *ACM SIGKDD explorations newsletter* 15.1 (2014), pp. 1–10.

[62] A. A. Freitas. *Data mining and knowledge discovery with evolutionary algorithms*. Springer, 2002.

[63] J. H. Friedman. “A recursive partitioning decision rule for nonparametric classification”. In: *IEEE Transactions on Computers* 4 (1977), pp. 404–408.

[64] J. H. Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* (2001), pp. 1189–1232.

[65] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[66] M. Friedman. “The use of ranks to avoid the assumption of normality implicit in the analysis of variance”. In: *Journal of the American Statistical Association* 32.200 (1937), pp. 675–701.

[67] J. Gama. “Discriminant trees”. In: *In practice* 1 (1999), p. 4.

[68] X. Gan, Y. Bao, and Z. Han. “Real-Time Search Method in Nondeterministic Game-Ms. Pac-Man”. In: *International Computer Games Association Journal* 34.4 (2011), pp. 209–222.

[69] L. M. García et al. “Strategies for monitoring and updating clinical practice guidelines: a systematic review”. In: *Implementation Science* 7.1 (2012), p. 109.

[70] A. X. Garg et al. “Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review”. In: *Jama* 293.10 (2005), pp. 1223–1238.

[71] M. Garofalakis and A. Kumar. “XML stream processing using tree-edit distance embeddings”. In: *ACM Transactions on Database Systems (TODS)* 30.1 (2005), pp. 279–332.

[72] T. S. Genders, B. S. Ferket, and M. M. Hunink. “The quantitative science of evaluating imaging evidence”. In: *JACC: Cardiovascular Imaging* 10.3 (2017), pp. 264–275.

[73] B. Goodman and S. Flaxman. “European Union regulations on algorithmic decision-making and a “right to explanation””. In: *AI Magazine* 38.3 (2017), pp. 50–57.

[74] P. Goyal and E. Ferrara. “Graph embedding techniques, applications, and performance: A survey”. In: *Knowledge-Based Systems* 151 (2018), pp. 78–94.

[75] L. W. Green. “Closing the chasm between research and practice: evidence of and for change”. In: *Health Promotion Journal of Australia* 25.1 (2014), pp. 25–29.

[76] R. Grilli et al. “Practice guidelines developed by specialty societies: the need for a critical appraisal”. In: *The Lancet* 355.9198 (2000), pp. 103–106.

[77] J. Grimshaw et al. “Effectiveness and efficiency of guideline dissemination and implementation strategies.” In: (2004).

[78] R. Guidotti et al. “A survey of methods for explaining black box models”. In: *ACM Computing Surveys (CSUR)* 51.5 (2018), p. 93.

[79] V. Gulshan et al. “Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs”. In: *Jama* 316.22 (2016), pp. 2402–2410.

[80] H. Guo and S. B. Gelfand. “Classification trees with neural network feature extraction”. In: *IEEE Transactions on Neural Networks* 3.6 (1992), pp. 923–933.

[81] I. Guyon. *Design of experiments for the NIPS 2003 variable selection benchmark*. Accessed: 2016. 2003. URL: clopinet.com/isabelle/Projects/NIPS2003.

[82] S. de Haan et al. “Assessment of left ventricular ejection fraction in patients eligible for ICD therapy: Discrepancy between cardiac magnetic resonance imaging and 2D echocardiography”. In: *Netherlands Heart Journal* 22.10 (2014), pp. 449–455. ISSN: 1568-5888.

[83] H. Haenssle et al. “Man against machine: diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists”. In: *Annals of Oncology* 29.8 (2018), pp. 1836–1842.

[84] J. Han, M. Kamber, and J. Pei. *Data mining concepts and techniques, third edition*. Waltham, Mass.: Morgan Kaufmann Publishers, 2012.

[85] S. Hara and K. Hayashi. “Making tree ensembles interpretable”. In: *arXiv preprint: 1606.05390* (2016).

[86] D. Heckerman. “Bayesian networks for data mining”. In: *Data mining and knowledge discovery* 1.1 (1997), pp. 79–119.

[87] J. A. Hoeting et al. “Bayesian model averaging: a tutorial”. In: *Statistical science* (1999), pp. 382–401.

[88] T. Hothorn, K. Hornik, and A. Zeileis. "Unbiased recursive partitioning: A conditional inference framework". In: *Journal of Computational and Graphical statistics* 15.3 (2006), pp. 651–674.

[89] L. Hu et al. "An observational study of deep learning and automated evaluation of cervical images for cancer screening". In: *JNCI: Journal of the National Cancer Institute* (2019).

[90] L. Hyafil and R. L. Rivest. "Constructing optimal binary decision trees is NP-complete". In: *Information Processing Letters* 5.1 (1976), pp. 15–17.

[91] J. Iavindrasana et al. "Clinical data mining: a review". In: *Yearbook of medical informatics* 18.01 (2009), pp. 121–133.

[92] O. Irsoy, O. T. Yildiz, and E. Alpaydin. "Soft decision trees". In: *International Conference on Pattern Recognition*. 2012.

[93] A. Jakulin et al. "Nomograms for visualizing support vector machines". In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM. 2005, pp. 108–117.

[94] D. Jensen and T. Oates. "The effects of training set size on decision tree complexity". In: *Proceedings of the 14th International Conference on Machine Learning*. 1999, pp. 254–262.

[95] J. L. Jesneck et al. "Optimized approach to decision fusion of heterogeneous data for breast cancer diagnosis". In: *Medical physics* 33.8 (2006), pp. 2945–2954.

[96] W. G. 1. Joint Committee for Guides in Metrology. "Evaluation of measurement data - Guide to the expression of uncertainty in measurement". In: *Tech. Rep. JCGM 100: 2008 (BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP and OIML*. 2008.

[97] M. I. Jordan and R. A. Jacobs. "Hierarchical mixtures of experts and the EM algorithm". In: *Neural computation* 6.2 (1994), pp. 181–214.

[98] D. Jurafsky. *Speech & language processing*. Pearson Education India, 2000.

[99] A. F. Karr, A. P. Sanil, and D. L. Banks. "Data quality: A statistical perspective". In: *Statistical Methodology* 3.2 (2006), pp. 137–173. ISSN: 15723127.

[100] G. V. Kass. "An exploratory technique for investigating large quantities of categorical data". In: *Applied statistics* (1980), pp. 119–127.

[101] E. Kaufmann and W. M. Koolen. "Monte-carlo tree search by best arm identification". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4897–4906.

[102] M. Kearns et al. "An experimental and theoretical comparison of model selection methods". In: *Machine Learning* 50 (1997), pp. 7–50.

[103] J. H. Keffer. "Guidelines and algorithms: Perceptions of why and when they are successful and how to improve them". In: *Clinical Chemistry* 47.8 (2001), pp. 1563–1572.

[104] L. Kocsis and C. Szepesvári. "Bandit based monte-carlo planning". In: *ECML* 6 (2006), pp. 282–293.

[105] R. Kohavi. "The power of decision tables". In: *European conference on machine learning*. Springer. 1995, pp. 174–189.

[106] M. Kretowski and M. Grzes. "Global learning of decision trees by an evolutionary algorithm". In: *Information Processing and Security Systems* (2005), pp. 401–410.

[107] P. Lancellotti et al. "Clinical outcome in asymptomatic severe aortic stenosis: insights from the new proposed aortic stenosis grading classification". In: *Journal of the American College of Cardiology* 59.3 (2012), pp. 235–243.

[108] N. Lavrač et al. "Intelligent data analysis for medical diagnosis: Using machine learning and temporal abstraction". In: *AI Communications* 11.3 (1998), pp. 191–218. ISSN: 09217126 (ISSN).

[109] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning". In: *nature* 521.7553 (2015), p. 436.

[110] J. Lee et al. "Open-access MIMIC-II database for intensive care research". In: *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*. IEEE. 2011, pp. 8315–8318.

[111] V. I. Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*. Vol. 10. 8. 1966, pp. 707–710.

[112] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: archive.ics.uci.edu/ml.

[113] B. Liu, Y. Xia, and P. S. Yu. "Clustering through decision tree construction". In: *Proceedings of the ninth international conference on Information and knowledge management*. ACM. 2000, pp. 20–29.

[114] W.-Y. Loh and Y.-S. Shih. “Split selection methods for classification trees”. In: *Statistica sinica* (1997), pp. 815–840.

[115] W.-Y. Loh and N. Vanichsetakul. “Tree-structured classification via generalized discriminant analysis”. In: *Journal of the American Statistical Association* 83.403 (1988), pp. 715–725.

[116] J. R. Lopez-Minguez et al. “Comparison of imaging techniques to assess appendage anatomy and measurements for left atrial appendage closure device selection.” In: *The Journal of invasive cardiology* 26.9 (Sept. 2014), pp. 462–467.

[117] G. Luo. “Automatically explaining machine learning prediction results: a demonstration on type 2 diabetes risk prediction”. In: *Health information science and systems* 4.1 (2016), p. 2.

[118] L. v. d. Maaten and G. Hinton. “Visualizing data using t-SNE”. In: *Journal of machine learning research* 9.Nov (2008), pp. 2579–2605.

[119] F. Maes et al. “Natural History of Paradoxical Low Gradient Severe Aortic Stenosis”. In: *Circulation: Cardiovascular Imaging* (2014), CIRCIMAGING–113.

[120] A. Marzal and E. Vidal. “Computation of normalized edit distance and applications”. In: *IEEE transactions on pattern analysis and machine intelligence* 15.9 (1993), pp. 926–932.

[121] H. Mercier and D. Sperber. *The enigma of reason*. Harvard University Press, 2017.

[122] J. Mingers. “An empirical comparison of selection measures for decision-tree induction”. In: *Machine learning* 3.4 (1989), pp. 319–342.

[123] J. Minners et al. “Inconsistencies of echocardiographic criteria for the grading of aortic valve stenosis”. In: *European heart journal* 29.8 (2007), pp. 1043–1048.

[124] B. Mittelstadt, L. Floridi, and S. Wachter. “Why a Right to Explanation of Automated Decision-Making Does Not Exist in the General Data Protection Regulation”. In: *International Data Privacy Law* 7.2 (2017), pp. 76–99.

[125] J. N. Morgan and J. A. Sonquist. “Problems in the analysis of survey data, and a proposal”. In: *Journal of the American statistical association* 58.302 (1963), pp. 415–434.

[126] M. Možina et al. “Nomograms for visualization of naive Bayesian classifier”. In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer. 2004, pp. 337–348.

[127] S. Mukherjee. *The laws of medicine: field notes from an uncertain science*. Simon and Schuster, 2015.

[128] R. Munos. “Optimistic optimization of a deterministic function without the knowledge of its smoothness”. In: *Advances in neural information processing systems*. 2011, pp. 783–791.

[129] S. K. Murthy, S. Kasif, and S. Salzberg. “A system for induction of oblique decision trees”. In: *Journal of artificial intelligence research* 2 (1994), pp. 1–32.

[130] S. A. Nashef et al. “European system for cardiac operative risk evaluation (Euro SCORE)”. In: *European journal of cardio-thoracic surgery* 16.1 (1999), pp. 9–13.

[131] S. A. Nashef et al. “Euroscore ii”. In: *European Journal of Cardio-Thoracic Surgery* 41.4 (2012), pp. 734–745.

[132] National Heart, Lung, and Blood Institute. *Data Science Bowl Cardiac Challenge Data*. 2015. URL: kaggle.com/c/second-annual-data-science-bowl.

[133] A. Nchimi et al. “Predicting Disease Progression and Mortality in Aortic Stenosis: A Systematic Review of Imaging Biomarkers and Meta-Analysis”. In: *Frontiers in cardiovascular medicine* 5 (2018), p. 112.

[134] A. Y. Ng, M. I. Jordan, and Y. Weiss. “On spectral clustering: Analysis and an algorithm”. In: *Advances in neural information processing systems*. 2002, pp. 849–856.

[135] T. Niblett and I. Bratko. “Learning Decision Rules in Noisy Domains”. In: *Proceedings of Expert Systems ’86, The 6Th Annual Technical Conference on Research and Development in Expert Systems III*. Brighton, United Kingdom: Cambridge University Press, 1987, pp. 25–34. ISBN: 0-521-34145-X.

[136] R. A. Nishimura et al. “2014 AHA/ACC guideline for the management of patients with valvular heart disease: a report of the American College of Cardiology/American Heart Association Task Force on Practice Guidelines”. In: *Journal of the American College of Cardiology* 63.22 (2014), e57–e185.

[137] M. Norouzi et al. “Efficient non-greedy optimization of decision trees”. In: *NIPS* (2015), pp. 1–9.

[138] C. Nunes et al. “A Monte Carlo tree search approach to learning decision trees”. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2018, pp. 429–435.

[139] Organisation for Economic Cooperation and Development. *Guidelines on the Protection of Privacy and Transborder Flows of Personal Data*. 1980.

[140] C. M. Otto. “Calcific aortic valve disease: new concepts”. In: *Seminars in thoracic and cardiovascular surgery*. Vol. 22. 4. Elsevier. 2010, pp. 276–284.

[141] A. Papagelis and D. Kalles. “Breeding decision trees using evolutionary techniques”. In: *ICML*. Vol. 1. 2001, pp. 393–400.

[142] V. L. Patel et al. “The coming of age of artificial intelligence in medicine”. In: *Artificial Intelligence in Medicine* 46.1 (2009), pp. 5–17.

[143] M. Pawlik and N. Augsten. “Efficient computation of the tree edit distance”. In: *ACM Transactions on Database Systems (TODS)* 40.1 (2015), p. 3.

[144] M. Pawlik and N. Augsten. “RTED: a robust algorithm for the tree edit distance”. In: *Proceedings of the VLDB Endowment* 5.4 (2011), pp. 334–345.

[145] M. Pawlik and N. Augsten. *Tree edit distance*. 2017. URL: `tree-edit-distance.dbresearch.uni-salzburg.at/` (visited on 07/25/2018).

[146] M. Pawlik and N. Augsten. “Tree edit distance: Robust and memory-efficient”. In: *Information Systems* 56 (2016), pp. 157–173.

[147] M. Pazzani. “Comprehensible knowledge discovery: gaining insight from data”. In: *First Federal Data Mining Conference and Exposition*. 1997, pp. 73–82.

[148] M. J. Pazzani, S. Mani, and W. R. Shankle. “Acceptance of rules generated by machine learning among medical experts”. In: *Methods of information in medicine* 40.05 (2001), pp. 380–385.

[149] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[150] E. Pekalska, P. Paclik, and R. P. Duin. “A generalized kernel approach to dissimilarity-based classification”. In: *Journal of machine learning research* 2.Dec (2001), pp. 175–211.

[151] M. F. Piepoli et al. “2016 European Guidelines on cardiovascular disease prevention in clinical practice: The Sixth Joint Task Force of the European Society of Cardiology and Other Societies on Cardiovascular Disease Prevention in Clinical Practice”. In: *European heart journal* 37.29 (2016), pp. 2315–2381.

[152] P. Ponikowski et al. “ESC Guidelines for the diagnosis and treatment of acute and chronic heart failure 2012”. In: *European Heart Journal* (2016).

[153] F. Provost and P. Domingos. “Well-trained PETs: Improving probability estimation trees”. In: (2000).

[154] D. R. PW and E. Pekalska. *Dissimilarity Representation For Pattern Recognition, The: Foundations And Applications*. Vol. 64. World scientific, 2005.

[155] L. D. Pyeatt, A. E. Howe, et al. “Decision tree function approximation in reinforcement learning”. In: *Proceedings of the third international symposium on adaptive systems: evolutionary computation and probabilistic graphical models*. Vol. 2. 1/2. Cuba. 2001, pp. 70–77.

[156] J. R. Quinlan. “Simplifying decision trees”. In: *International journal of man-machine studies* 27.3 (1987), pp. 221–234.

[157] J. R. Quinlan. “Some elements of machine learning”. In: *International Conference on Inductive Logic Programming*. Springer. 1999, pp. 15–18.

[158] J. Quinlan et al. “Interactive Dichotomizer, ID3”. In: *Eds. Morgan Kauffmann, Springer-Verlag* (1979).

[159] R. Quinlan. *C4.5: Programs for Machine Learning*. CA: Morgan Kaufmann, 1993.

[160] R. Quinlan. “Decision trees as probabilistic classifiers”. In: *Proceedings of the 4th International Workshop on Machine Learning*. Morgan Kauffman, 1987, pp. 31–37.

[161] R. Quinlan. “Induction of decision trees”. In: *Machine Learning* 1.1 (1986), pp. 81–106.

[162] R. Quinlan. “Probabilistic decision trees”. In: *Machine learning: an artificial intelligence approach* 3 (1990), pp. 140–152.

[163] R. Quinlan. *Ross Quinlan’s personal homepage: C4.5 Release 8*. rulequest.com/Personal/. Accessed: 2018-01-30. 1992.

[164] M. T. Ribeiro, S. Singh, and C. Guestrin. “Why should i trust you?: Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2016, pp. 1135–1144.

[165] K. Riesen, M. Neuhaus, and H. Bunke. “Graph embedding in vector spaces by means of prototype selection”. In: *International Workshop on Graph-Based Representations in Pattern Recognition*. 2007, pp. 383–393.

[166] J. Roddick, P. Fule, and W. Graco. “Exploratory medical knowledge discovery: Experiences and issues”. In: *ACM SIGKDD Explorations Newsletter* (2003), pp. 2–7. ISSN: 19310145.

[167] L. Rokach and O. Maimon. “Top-down induction of decision trees classifiers - A survey”. In: *IEEE Transactions on Systems and Cybernetics* 35.4 (2005), pp. 476–487.

[168] R. E. Schapire. “A brief introduction to boosting”. In: *IJCAI International Joint Conference on Artificial Intelligence*. Vol. 2. 1999, pp. 1401–1406.

[169] M. Schwabacher, P. Langley, and P. Norvig. “Discovering communicable scientific knowledge from spatio-temporal data”. In: (2001).

[170] A. Segatori, F. Marcelloni, and W. Pedrycz. “On Distributed Fuzzy Decision Trees for Big Data”. In: *IEEE Transactions on Fuzzy Systems* (2017), pp. 1–1.

[171] T. M. Shaneyfelt, M. F. Mayo-Smith, and J. Rothwangl. “Are guidelines following guidelines?: The methodological quality of clinical practice guidelines in the peer-reviewed medical literature”. In: *Jama* 281.20 (1999), pp. 1900–1905.

[172] D. J. Sheskin. *Handbook of parametric and nonparametric statistical procedures*. CRC Press, 2003.

[173] D. Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *Nature* 529.7587 (2016), pp. 484–489.

[174] K. Singh et al. “Intra- and interobserver variability in the measurements of abdominal aortic and common iliac artery diameter with computed tomography. The Tromsø study”. In: *European Journal of Vascular and Endovascular Surgery* 25.5 (2003), pp. 399–407.

[175] The New York Times. *Uber’s Self-Driving Cars Are Set to Return in a Downsized Test*. Accessed: April 2019. 2018. URL: nytimes.com/2018/12/05/technology/uber-self-driving-cars.html.

[176] S. Tsang et al. *Decision Trees for Uncertain Data*. 2011.

[177] R. J. Urbanowicz and J. H. Moore. “Learning classifier systems: a complete introduction, review, and roadmap”. In: *Journal of Artificial Evolution and Applications* (2009).

[178] H. Utsunomiya et al. “Underestimation of aortic valve area in calcified aortic valve disease: effects of left ventricular outflow tract ellipticity”. In: *International journal of cardiology* 157.3 (2012), pp. 347–353.

[179] A. Van Assche and H. Blockeel. “Seeing the forest through the trees: Learning a comprehensible model from an ensemble”. In: *European Conference on Machine Learning*. Springer. 2007, pp. 418–429.

[180] S. Wang and R. M. Summers. “Machine learning and radiology”. In: *Medical image analysis* 16.5 (2012), pp. 933–951.

[181] X. Wang et al. “On the optimization of fuzzy decision trees”. In: *Fuzzy Sets and Systems* 112.1 (May 2000), pp. 117–125.

[182] A. Weigel and F. Fein. “Normalizing the weighted edit distance”. In: *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International.Conference on Pattern Recognition*. Vol. 2. IEEE. 1994, pp. 399–402.

[183] S. F. Weng et al. “Can machine-learning improve cardiovascular risk prediction using routine clinical data?” In: *PLoS ONE* 12.4 (2017).

[184] F. H. Wians. “Clinical laboratory tests: which, why, and what do the results mean?” In: *Laboratory Medicine* 40.2 (2009), pp. 105–113.

[185] F. Wilcoxon. “Individual Comparisons by Ranking”. In: *Biometrics Bulletin* 1.6 (1945), pp. 80–83.

[186] Wired. *California Unanimously Passes Historic Privacy Bill*. Acces -sed: April 2019. 2018. URL: wired.com/story/california-unani-mously-passes-historic-privacy-bill/.

[187] X. Wu et al. “Top 10 algorithms in data mining”. In: *Knowledge and information systems* 14.1 (2008), pp. 1–37.

[188] C. W. Yancy et al. “2013 ACCF/AHA guideline for the management of heart failure: a report of the American College of Cardiology Foundation/American Heart Association Task Force on Practice Guidelines”. In: *Journal of the American College of Cardiology* 62.16 (2013), e147–e239.

[189] O. T. Yıldız and E. Alpaydın. “Linear discriminant trees”. In: *International Journal of Pattern Recognition and Artificial Intelligence* 19.03 (2005), pp. 323–353.

[190] Y. Yuan. “Induction of fuzzy decision trees”. In: *Fuzzy Sets and Systems* 69.2 (1995), pp. 125–139.

[191] H. Zantema and H. L. Bodlaender. *Finding small equivalent decision trees is hard*. Vol. 1999. Utrecht University: Information and Computing Sciences, 1999.

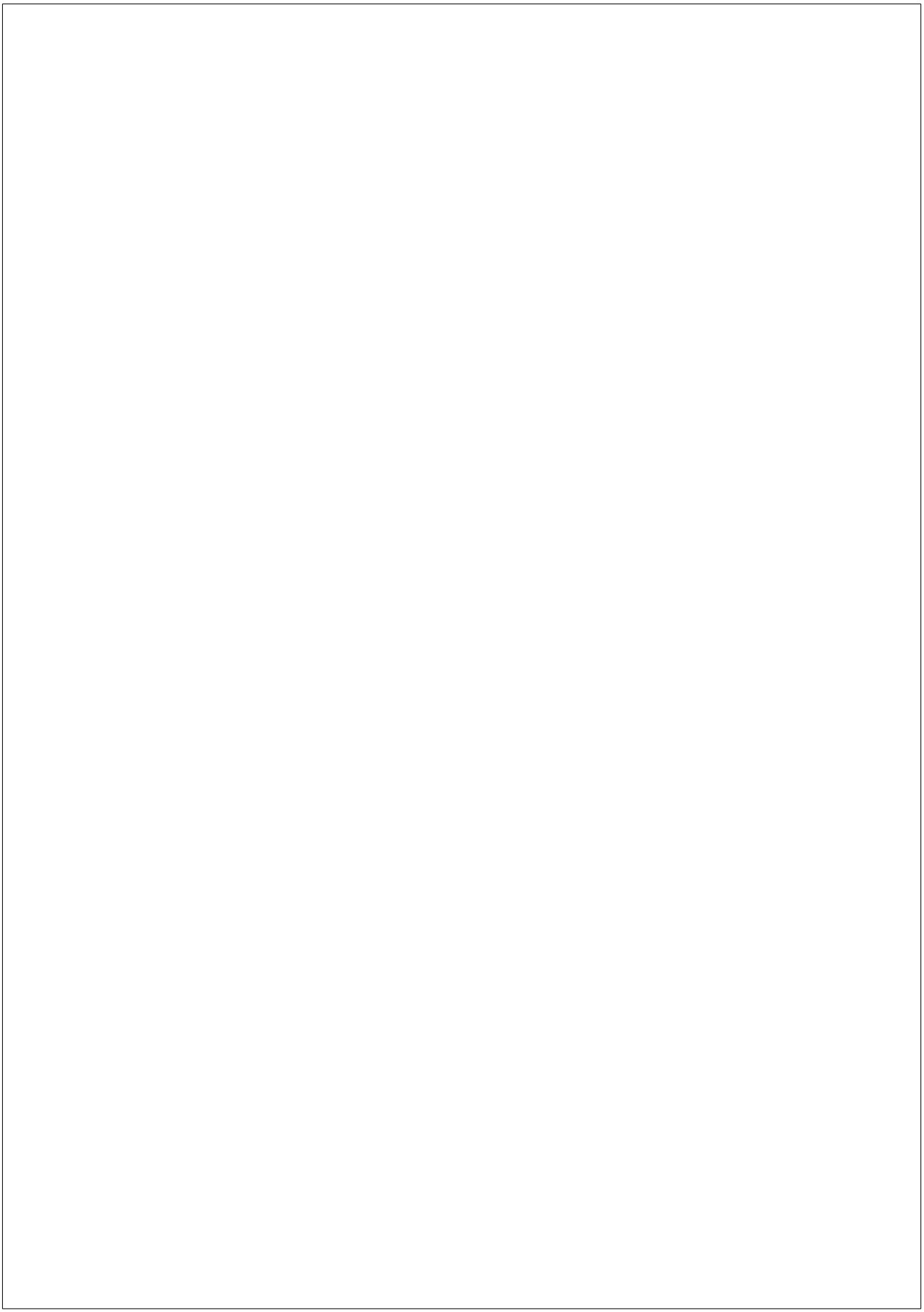
[192] C. Zhang and S. Zhang. *Association rule mining: models and algorithms*. Springer-Verlag, 2002.

[193] K. Zhang and D. Shasha. “Simple fast algorithms for the editing distance between trees and related problems”. In: *SIAM journal on computing* 18.6 (1989), pp. 1245–1262.

[194] Q.-s. Zhang and S.-C. Zhu. “Visual interpretability for deep learning: a survey”. In: *Frontiers of Information Technology & Electronic Engineering* 19.1 (2018), pp. 27–39.

Publications

- C. Nunes, H. Langet, M. De Craene, O. Camara, B. Bijnens, A. Jonsson. ”Decision Tree Learning for Uncertain Clinical Measurements" (2018) – journal article submitted for review in *IEEE Transactions on knowledge and data engineering*
- C. Nunes, A. Jonsson, O. Camara, B. Bijnens. ”Learning from uncertain data: a decision tree approach". In: *Women in Machine Learning (WiML) Workshop, Conference on Neural Information Processing Systems* (2016) – abstract accepted in conference workshop
- C. Nunes, M. De Craene, H. Langet, O. Camara, A. Jonsson. ”Learning decision trees through Monte Carlo tree search: An empirical evaluation" (2019) – journal article submitted for review in *WIREs Data Mining and Knowledge Discovery*
- C. Nunes, M. De Craene, H. Langet, O. Camara, and A. Jonsson. “A Monte Carlo tree search approach to learning decision trees”. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2018, pp. 429–435 – article in conference proceedings
- C. Nunes, O. Camara, B. Bijnens, M. De Craene, A. Jonsson ”Monte Carlo decision tree search for clinical recommendation". In: *CASEIB Congreso anual de la Sociedad Española de Ingeniería Biomédica* (2017) – abstract accepted in conference
- C. Nunes, H. Langet, M. De Craene, O. Camara, A. Jonsson. ”Comparing the Structure of Decision Trees" – conference article under preparation
- C. Nunes, H. Langet, M. De Craene, O. Camara, A. Jonsson., J. L. Vanoverschelde, B. L. M. Gerber, B. Bijnens ”Predicting post-intervention survival improvement in aortic stenosis through interpretable machine learning" (2019) – abstract submitted for review at *EuroEcho Annual congress of the European Association of Cardiovascular Imaging (EACVI)*
- C. Nunes, H. Langet, M. De Craene, O. Camara, A. Jonsson., J. L. Vanoverschelde, B. L. M. Gerber, B. Bijnens. ”Decision trees in the management of aortic stenosis" – journal article under preparation



Funding information

This work was supported by:

- European Union Horizon 2020 research and innovation programme under grant agreement 642676 (Cardiofunxion)
- The Spanish Ministry of Economy and Competitiveness (grant TIN2014-52923-R; Maria de Maeztu Units of Excellence Programme - MDM-2015-0502)
- The European Union FP7 for research, technological development and demonstration under grant agreement VP2HF (611823)