Bridging Interpretability and Accuracy using Optimization: A Computational Comparison of Optimal Classification Trees

Nils Kulmbacher

July 31, 2024
UvA Supervisor: *** ***
Student number ********

Thesis Bsc Business Analytics
University of Amsterdam

Statement of Originality

This document is written by Student Nils Kulmbacher who declares to take full responsibility for the contents of this document. I declare that the text and the work presented in this document are original and that no sources other than those mentioned in the text and its references have been used in creating it. I have not used generative AI (such as ChatGPT) to generate or rewrite text. UvA Economics and Business is responsible solely for the supervision of completion of the work and submission, not for the contents.

Abstract

For high-stakes decisions such as loan applications or healthcare, algorithms can improve decision-making. However, often black-box models are used, which are difficult to explain. Models need to be explainable, such that practitioners, regulators and consumers can be convinced to use them. New research disagrees with the notion of an Interpretability/Accuracy trade-off suggesting that interpretable models can achieve comparable performance. One of these easy-to-interpret models are classification trees, whose structure allows for an intuitive understanding of decisions. Currently-used heuristic methods to generate them provide low performance for complex problems.

This research aims to carefully navigate accuracy and interpretability by constructing optimal trees using mathematical optimization formulations, a novel technique to generate decision trees. While many methods exist in literature, no comprehensive comparison of the main ones has been done. Formulations are benchmarked on various datasets. Ultimately, when comparing them to heuristic models, the optimal trees seem to perform similarly or better than the heuristic models in terms of performance and have a smaller size, making them easier to explain.

1 Introduction

Imagine your bank denies you a credit application, your doctor diagnoses a disease, or your job application gets rejected - for all these important decisions people are interested in an explanation. These are high-stakes decisions, they have a significant impact on people and/or capital and thus require consideration. An explanation is not always straightforward, especially if machine learning models make those decisions. This is problematic especially when companies prefer to use difficult-to-explain black-box models (Bertsimas et al., 2019). In the worst case, they would be unable to comply with current regulations such as the EU's right to explanation introduced in the EU's General Data Protection Regulation (European Union, 2016), which gives individuals the right to obtain meaningful explanations about decisions that significantly affect them. Similarly, companies also risk making biased decisions in one of these high-stakes cases, as happened for example at Amazon, where a hiring algorithm ended up proportionally discriminating against female applicants (Dasting, 2018). Potentially, this could also have significant legal consequences, besides the loss of public image.

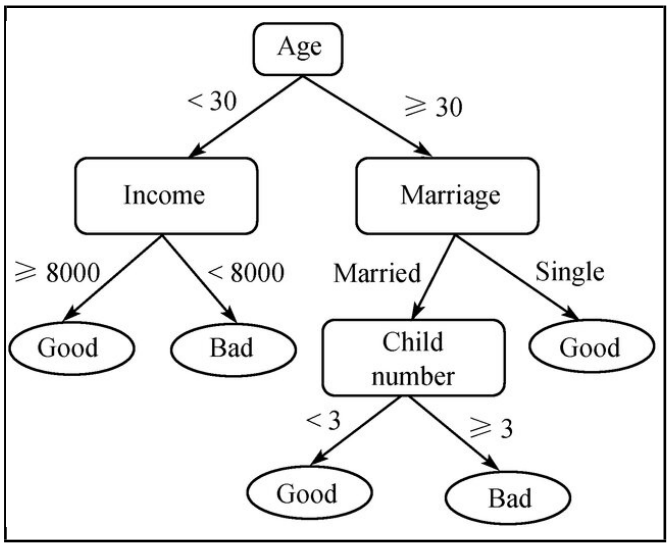


Figure 1: Example of a classification tree for credit risk scoring. Taken from Kin Keung (2012)

Explainable artificial intelligence (XAI) has been a growing field of research in recent years (Schmidt & Biessmann, 2019). While there are no clear general definitions yet, most scholars agree that there is a difference between inherently explainable models (also interpretable or glass-box) and post-hoc explanations of black-box models (Doran et al., 2017). Where a lack of transparency and accountability can have negative consequences for society (Rudin, 2019), interpretable models can provide a fundamental understanding of the underlying reasoning of the model. For example, as can be seen in Figure 1, the rules that a classification tree operates on can be easily understood and replicated by a human interpreter.

Rudin (2019) also identifies a knowledge gap in research on interpretable AI, mentioning there is plenty of ongoing and past research about explaining various black-box models (e.g. deep learning) but comparatively less research about interpretable models. With AI models being used more and more, there is a recent push from regulators and

consumers to control the use of AI models (European Parliament, 2023) to limit the potentially harmful consequences of bad algorithmic decisions (Kuźniacki, 2023). Hence, there is a growing need for explainable solutions that are also attractive for businesses in terms of model performance to still profit from using them. Rudin (2019) makes a strong case for why interpretable models should be preferred, as the accuracy-interpretability trade-off described in the literature is a "myth", according to her. With the right tools, one can develop naturally interpretable models, like linear regressions or classification trees that are comparable in terms of accuracy to black-box models (Lamberti, 2023; Rudin, 2019).

Unfortunately, commonly used greedy heuristic methods to generate trees (e.g. CART) are not optimal and have poor performance (Alès et al., 2024; Verwer & Zhang, 2019). With suboptimal trees, the preconception among many practitioners exists that decision trees are not sufficiently accurate to replace black-box models. Provided that certain computational challenges (e.g. time-intensive global optimization) are overcome, an optimal classification tree can be as accurate as many black-box models (Rudin, 2019). Going even further, adding additional loss terms or constraints for added interpretability enables the generation of more explainable trees. For example, one might choose to add a penalty term to the objective function for tree depth or stability of explanations. (Rosenfeld, 2021; Zhou et al., 2018). Regularizing using the depth of the tree, one could generate more explainable trees that still perform with high accuracy.

The problem with the most relevant optimal tree algorithms is that they differ substantially in complexity, the type of datasets they accept and the type of problems they can solve. For example, BinOCT (Verwer & Zhang, 2019) is based on OCT (Bertsimas & Dunn, 2017) and cannot handle numerical continuous features but only binary ones, while OCT can. While tests are comparing some of these models' performances (Tang & Lin, 2021), no wide-reaching analysis has been done to identify the fastest, most performant method under different conditions.

Thus, this research aims to comprehensively compare and analyse the most relevant proposed algorithms to implement optimal decision trees to answer: How do current algorithms to train optimal decision trees perform in terms of accuracy and interpretability?

A short literature review has been carried out and the methods used for comparison and other related work will be discussed in Section 2. Then, the set-up of the computational study is described in Section 3. Finally, results are reported and discussed in Sections 4, 5 and 6.

2 Literature review

There are two fundamental dimensions to explore for this paper: Explainable AI (XAI) and Optimal Decision Trees. Therefore, it was chosen to first look at every dimension on its own and then later combine the two. This framework can be seen in Figure 2. The main interest within the XAI dimension is to define what XAI is and clarify any terms that are commonly used. Next, literature focussing on measuring interpretability in general and for decision trees specifically is compared. Furthermore, Optimal Decision Trees need to be defined and their usefulness evaluated. To achieve this, the most relevant methods to generate optimal trees are researched and existing comparisons are evaluated. To also measure the performance of trees, specific metrics on classification are defined. Finally, both topics are combined in one comparison to compare several optimal tree techniques using several metrics for performance and interpretability against reference models.

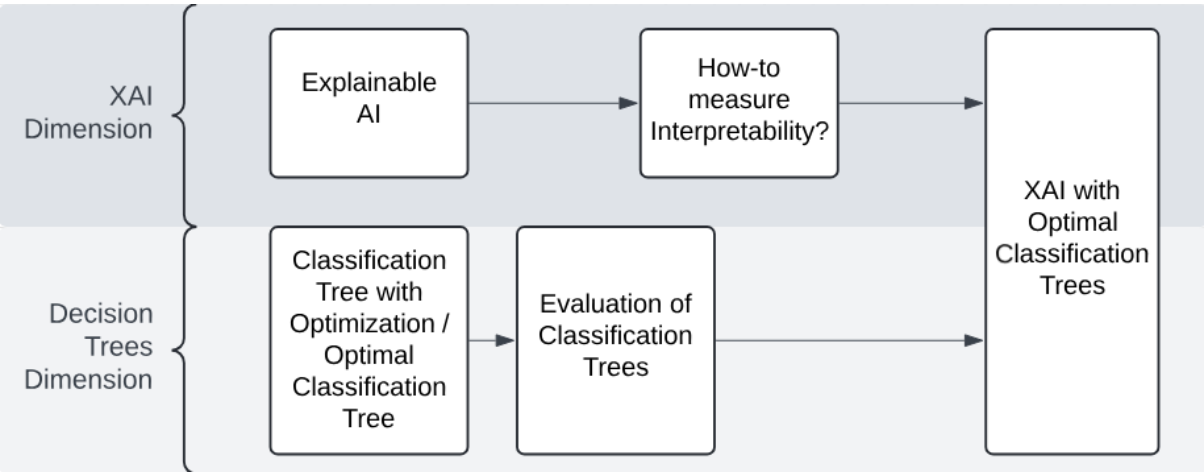


Figure 2: Proposed research strategy to combine topics

2.1 XAI

With the earliest work on Explainable Artificial Intelligence (XAI) being done in the 1970s and 80s (Xu et al., 2019), the notion of explaining a system that makes decisions has been around for more than 50 years. Models and market penetration increased over this time and governments introduced regulations for making algorithmic decisions while increasing spending on research (Xu et al., 2019). Therefore, making these decisions transparent is a “fundamental issue” (Longo et al., 2020, p.2) of AI. Considering that the actual use in public organizations (de Bruijn et al., 2021) and for high-stakes decisions (Rudin, 2019) has increased as well, a lack of aforementioned transparency and accountability, can have severe consequences (Rudin, 2019).

XAI Terminology

As XAI is a developing field, there exists no common terminology and a lot of ambiguous terms (de Bruijn et al., 2021; Doran et al., 2017; Longo et al., 2020). Nevertheless, there are some commonly agreed terms and definitions, that may have different names but refer to the same concept. In general, research suggests a difference between explainable and interpretable models (Lamberti, 2023). While explainable models help elements of a model to be understood and described, interpretable models have concrete meaning, they can be understood with little additional methods assuming the right knowledge. Most articles bring forward that difficult-to-explain models are usually explained using a post-hoc method, employed after a black-box model is trained to extract explanations (Rudin, 2019). These explanations can be either of global nature, describing the overall models (e.g. feature importance) or local nature, explaining a single decision being made (Longo et al., 2020). This stands in contrast to interpretable or glass-box models, which are interpretable on their own (Doran et al., 2017; Longo et al., 2020; Xu et al., 2019).

XAI on its own has multiple definitions, depending on the approach. For example Lamberti (2023) states that XAI provides insights into models, whereas de Bruijn et al. (2021) makes a point that XAI should explain the workings of an AI system to the general public. Notably, research agrees that XAI should especially be employed for so-called high-stakes decisions, meaning decisions with a significant effect on people, money or other ethical aspects (Rudin, 2019; Xu et al., 2019).

Research also suggests that there is an unused potential to use AI algorithms for more applications where an explanation is currently not possible, but required to have model assurance (Longo et al., 2020). Model assurance is the ability of a model to be generalizable, and to perform on data it was not trained on (Lamberti, 2023). These explanations of how decisions are derived are needed for trust and to evaluate the ethical side of the decision taken, such that it becomes human-understandable (Doran et al., 2017; Hoffman et al., 2018). Xu et al. (2019) suggests that transparency and trust are needed in the case of critical decisions when people are affected directly. Furthermore, it can be used to identify flaws and better controllability of algorithmic decisions (Longo et al., 2020). This is relevant when problems are prone to inaccuracy. For example, imbalanced data or a high number of classes to categorize can lead to wrong conclusions about actual model performance, which is when models need to be interpretable to verify their validity (Lamberti, 2023)

The Accuracy/Interpretability Trade-off

What researchers disagree on, is the trade-off between model accuracy and interpretability. They point out that this trade-off inevitability limits the use of interpretable models (Longo et al., 2020), that there is an inverse correlation between explainability and accu-

racy. However, Rudin (2019) describes this trade-off as a "myth". Furthermore, Lamberti (2023) describes an image problem with interpretable models offering inferior performance, while in reality, they can outperform black-box models on structured data while offering interpretability. They describe the wrong preconception that black-box models are necessary because practitioners assume that the underlying decisions are opaque (Lamberti, 2023). Black-box models and the subsequent post-hoc explanation attempts drive human error and are inherently flawed (Rudin, 2019). Logically, Rudin (2019) explains that if black-box model explanations would offer perfect explanations, then they would be the same as the agent model, hence interpretable.

Nevertheless, researchers acknowledge that XAI cannot always provide accurate explanations (de Bruijn et al., 2021). There is a multitude of different XAI methods, each used for a different model class with varying degrees of interpretability and limitations (Longo et al., 2020). Another challenge lies in the very definition of explanations, which are subjective (Hoffman et al., 2018; Zhou et al., 2018). While some challenges can be overcome, such as the lack of expertise in this field (de Bruijn et al., 2021; Rudin, 2019), others need to be addressed with future research. These challenges can be seen in Table 1.

However, with the right techniques and expertise, these problems can be overcome. Generally, low-stakes decisions do not require an interpretable model, reducing the effort to create interpretable models (Doran et al., 2017). A possible solution for high-stakes decisions however, could be optimized interpretable models that are human-readable, such as decision trees, that are both interpretable and explainable (Lamberti, 2023; Xu et al., 2019), meaning that they allow each element to be understood as well as having concrete physical meaning. For decision trees, the global and local explanations can be derived from the same output. For a specific data point, the tree can be traversed to see how the prediction is made, for the tree as a whole, the most important features and characteristics can be explored.

2.2 Evaluation

While heuristic methods use metrics such as entropy and information gain (Hossin & Sulaiman, 2015), optimal trees need to be considered using a variety of methods to understand the specifications of a model (Vujovic, 2021). Evaluation metrics can be used to evaluate generalizability (in conjunction with train-test split), model selection and selecting an optimal model.

Evaluation of classification

Accuracy seems to be the most popular metric (Bertsimas & Dunn, 2017; Hossin & Sulaiman, 2015; Vujovic, 2021). Accuracy describes the proportion of correctly identified

Problem with	Description
Global explanations	Providing feature importance might not take into account interrelated features (de Bruijn et al., 2021).
Local explanations	Unable to provide an overview of the entire model and the local explanation can be an outlier (de Bruijn et al., 2021).
Profit incentive	Companies could make profits from intellectual property by using proprietary black-box models. There is a strong profit incentive not to use transparent "open source" models that could be copied (Rudin, 2019).
High number of features and dimensionality reduction	For problems with many features (e.g. natural language data), it can become difficult to provide manageable explanations. While dimensionality reduction can help to reduce the number of features here, it tremendously reduces explainability (Lamberti, 2023).
Causality	Humans might perceive decisions as the result of causality, while the model is not based on causality (de Bruijn et al., 2021; Longo et al., 2020).
Effort	Constrained problems are harder to solve than unconstrained ones. Therefore models that aim to be both performant and interpretable require more effort to create (Rudin, 2019).

Table 1: Overview of Challenges with XAI identified by literature

samples among all samples. Unfortunately, it does not represent the model's characteristics when the underlying dataset is imbalanced (Hossin & Sulaiman, 2015). In this case, the AUC metric (Area under the ROC curve) is a more accurate indicator for model performance (Hossin & Sulaiman, 2015). However, it is challenging to compute for multi-class classification. Other common metrics include measures calculated using the confusion matrix such as precision and recall or composite measures such as the F-Score (Vujovic, 2021). Precision is the proportion of all correctly positive classified samples among all positive classified samples (also wrongly positive classified). On the other hand, recall is the proportion of all correctly positive classified samples among all samples that are actually positive. The F1-score is derived using the harmonic mean of these two values. For multi-class classification, a micro-average can be used to compute the F1-Score. Important to note is that for trees specifically specialized metrics exist that affect accuracy but more importantly interpretability (Bertsimas & Dunn, 2017).

Evaluating Interpretability

Understanding AI decisions is necessary to convince executives and justify their effectiveness to regulators and shareholders (Hoffman et al., 2018). Still, studies have shown that there is a distrust of predictive models by these decision-makers, despite AI models being more accurate than humans (Bertsimas et al., 2019). Hence, interpretability has been a major research focus (Schmidt & Biessmann, 2019). Expert systems, one of the first forms of AI systems also implemented explanations to make them more transparent (Hoffman et al., 2018). Even back in the 1980s, there was a need for metrics to evaluate and quantify the benefit of these systems, which extends to XAI today (Rosenfeld, 2021). Without quality explanations, there is a greater risk of models making unexpected decisions (Moraffah et al., 2020).

While there are many metrics to evaluate the performance of AI models, there is little research on quantifying interpretability (Zhou et al., 2018). This is exacerbated by the fact that many XAI methods cannot be compared in standardized benchmarks, because of their specific nature (Schmidt & Biessmann, 2019).

Another issue is that interpretability alone does not guarantee fairness, as decisions can be understandable but still exhibit a bias (Moraffah et al., 2020). As Jo et al. (2023) explains for high-stakes decisions, models should be interpretable and fair, as the decisions affect people substantially.

Unfortunately, many practitioners are hesitant to give up black-box models, as they perceive them to have a higher accuracy (Bertsimas et al., 2019). Research suggests that surveys are often used to quantify the goodness of explanations (Zhou et al., 2018), which suffers from confirmation bias (Rosenfeld, 2021). This makes sense as the very nature of explanations is that they are subjective (Zhou et al., 2018), and affected by people's knowledge. This context-dependence (Hoffman et al., 2018) makes it harder to navigate the trade-off between explaining truthfully and explaining in simple terms to make them more accessible (Schmidt & Biessmann, 2019). Vilone and Longo (2021) summarizes three requirements for good explanations: (i) *Correct Representation* (ii) *Non-overlap* (iii) *Understandability*. In general, more complex models are said to be more difficult to explain (Zhou et al., 2018). For Classifiers, Zhou et al. (2018) describe interpretability as the ability to understand the logic behind predictions made.

In conclusion, surveys, as opposed to metrics are widely used to measure interpretability, despite their inherent flaws. A potential solution to overcome the lack of common metrics for interpretability could be to select an interpretable model from the beginning, assuming that it produces good enough explanations. The drawback would be the limited choice of models and the potentially worse performance as a classifier. This is called the "price of interpretability" (Bertsimas et al., 2019; Jo et al., 2023) and has to be navigated straightforwardly.

2.3 Classification Trees

Decision trees are a supervised machine learning technique, introduced by Morgan and Sonquist (1963) that can perform regression (regression trees) or classification (classification trees). Their structure is simple to understand, interpret and visualize (scikit-learn, 2009). When a tree is built, a classification is derived by traversing the tree from the root node to one of the leaf nodes. Each branch includes a decision (e.g. Feature 1 is smaller than 10) that can be binary (true & false) or multi-valued. Based on the last decision, the following branch node is chosen and the next node is reached until the tree has been traversed. The leaf node a data point lands in assigns a prediction for all data points that reach it. While this is the basic structure of a classification tree, there are multiple extensions possible. For example, with an oblique tree, each branch is allowed to split on a combination of features rather than a single feature, as with axis-aligned trees (Alès et al., 2024).

In her talk, Rudin (2021) suggests that sparse trees can be just as accurate as bigger black-box models if one finds an optimal tree. While optimization might not scale with the number of features or dataset size (Bertsimas et al., 2019), even a suboptimal tree can offer greater performance than one made through traditional approaches. In the past, this has been done using greedy heuristics (Alès et al., 2024; Verwer & Zhang, 2019) such as CART (Breiman et al., 1984) or C5.0 (Quinlan, 1992). Trees are grown using a purity criterion that tries to maximize the purity in the resulting leaves for each step. These methods offer no indication of how optimal a tree is while making more complicated trees than necessary (McTavish et al., 2021). In addition to that, the splitting is done without taking into account the resulting splits and so on. The tree is not grown individually without looking at it as a whole. Another limitation is the pruning necessary for top-down models (Bertsimas & Dunn, 2017). Pruning removes part of the tree to avoid overfitting because trees are grown using a two-step process, first growing the tree and then pruning using some misclassification criterion (Bertsimas & Dunn, 2017). When growing a decision tree using one step that aims to minimize misclassification and penalize overfitting, it is possible to achieve optimal trees. While this idea to optimize trees has been around for longer, it was usually thought of as an NP-hard problem (Alès et al., 2024; Verwer & Zhang, 2019). Thanks to increased computational speeds, better solvers and new optimization formulations, it is possible to solve these problems in a reasonable amount of time (Alès et al., 2024). While decision trees can also be used to make continuous predictions (regression trees), this research focuses on classification trees specifically.

Optimal trees

One of the first applicable formulations for Optimal Classification Trees, named OCT is the Mixed-Integer Optimization (MIO) formulation proposed by Bertsimas and Dunn (2017). They argue that by using MIO, the formulation is less rigid, such that constraints and objectives can be adapted to account for additional factors (e.g. sparsity). Bertsimas and Dunn (2017) archive better performance than the heuristic methods (here: CART), while still performing worse than ensemble models such as Random Forest (RF). Interestingly, for datasets with a smaller amount of target classes and features, OCT is as accurate as RF.

A key limitation of OCT is that the complexity depends on the training size, as constraints and variables are added for each data point (Verwer & Zhang, 2019). Alès et al. (2024) aim to extend the formulation using a quadratic objective function, resulting in fewer constraints making it easier to solve. Another new formulation (BinOCT), also based on MIO (Verwer & Zhang, 2019), aims to combine constraints using Big-M notions, also resulting in a much smaller optimization problem, with size independent of the training size. An obvious limitation, is that features need to be binary, requiring bucketing to transform continuous features into categorical features to which one-hot encoding is then applied to create binary features. While performance is similar to OCT, the main benefit of BinOCT is that it can be trained faster than OCT, making it better for larger datasets (Verwer & Zhang, 2019).

Aghaei et al. (2021) aim to resolve this limitation by utilizing a max-flow formulation (FlowOCT), where they convert a decision tree with a fixed depth to an acyclic graph. They argue that BinOCT uses fewer decision variables than OCT but has a weaker linear optimization relaxation, therefore offering no performance improvement. A linear optimization relaxation (LO relaxation) is a technique to reduce constraints to enlarge the feasible solution space and find a solution faster. Besides the novel model formulation, they propose to add constraints for fairness, imbalanced data and sparsity, as well as an additional mathematical formulation to improve processing times (Aghaei et al., 2021). While they did not compare their model with BinOCT directly, the performance results indicate that FlowOCT performs better than both OCT and BinOCT (because OCT and BinOCT have similar performance).

The previously mentioned models define characteristics of the tree to be generated beforehand (e.g. fixed depth), then try to find optimal trees given these assumptions. Therefore, they require parameter tuning to try out different hyperparameters and choose the one that gives the best results. Hu et al. (2019) suggests that this produces suboptimal trees which cannot be imbalanced (meaning have to have a constant depth). Therefore, they suggest a novel formulation (OSDT) based on dynamic programming and branch and bound (Hu et al., 2019). Basically, they try to find the optimal trees according to a

regularized function among all possible trees of all possible depths. Using selective cuts to the search area to reduce the number of possibilities that increase exponentially otherwise with the number of features. In comparison with BinOCT and CART, OSDT offers similar processing times as BinOCT while performing better on accuracy and sparsity of trees. It was expected that OSDT would perform better than CART, as they use a warm-start to initialize their model using the solution found by CART. Recently, OSDT has been extended (GOSDT) by McTavish et al. (2021) introducing better bucketization for features, depths constraints and tighter lower bounds, as well as "smart guesses" suggesting better optimal solutions derived from ensemble models.

In summary, multiple methods were developed to derive optimal decision trees, each with its own limitations and comparisons run. While some of the formulations build on top of each other and compare the results with their respective predecessor, it is hard to determine the most effective method without a comprehensive comparison.

Therefore, a computational study comparing these methods with each other, on different types of datasets using different evaluation metrics for performance as well as interpretability is a beneficial addition to this field. An overview of recently proposed algorithms relevant to this field is shown in Table 2. Note that GOSDT is only a faster version of OSDT and therefore, not a separate new formulation.

Table 2: Comparison of Formulations

	OCT (Bertsimas & Dunn, 2017)	**BinOCT** (Verwer & Zhang, 2019)	**FlowOCT** (Aghaei et al., 2021)	**(G)OSDT** (Hu et al., 2019; McTavish et al., 2021)
Type of formulation	mixed integer programming	mixed integer programming	max-flow formulation	dynamic programming/branch and bound
Feature limitations		only binary features*	only binary features*	
Critiques	Number of constraints and variables based on dataset size, fixed tree depth	Weaker LO relaxation because of big-M constraints used, fixed tree depth	Only possible to make trees with fixed depth	Search space could increase with the number of features
Implementation used	IAI (Interpretable AI, 2023)	Optimal Classification Trees (Tang & Lin, 2021)	BendersOCT (Aghaei et al., 2021)	GOSDT (Seltzer et al., 2024)
Extensions	Warm starts using CART solution	Warm starts using CART solution	Possibility to add objectives for fairness and robustness, formulation for imbalanced trees	Faster version GOSDT introducing "smart guesses" using Reference Ensembles
Optimization solver	Gurobi**	Gurobi**	Gurobi**	Custom python (GOSDT: Custom C++)

*Continuous and categorical features can be transformed during preprocessing.
**Gurobi Solver version 11.0.1

3 Methodology

In this computational study, the previously identified methods from Section 2 are applied to multiple datasets and their performance is compared in terms of classification accuracy (Hossin & Sulaiman, 2015) and different notions of interpretability (Rosenfeld, 2021).

3.1 Datasets

For this study, public datasets will be used. This will include both binary and multi-class classification problems, imbalanced datasets, as well as datasets with varying sizes. Using diverse datasets ensures a stronger comparison (Hossin & Sulaiman, 2015).

In total, 29 datasets are being used for the experiments. There is a great variety in datasets concerning size (178 up to 58,509 data points), target class imbalance, number of features and number of target classes. While most datasets have been accessible in a pre-processed format, meaning few or no missing values and correct labels, some adjustments have been made. First, one-hot encoding is applied to categorical features. Secondly, for the algorithms requiring binary features, binarization according to Lawless et al. (2021) has been applied to generate binary versions of the continuous features within each dataset.

To later analyse the results, a dataset summary has been generated, which can be seen in Table 3. This includes the name, number of samples, number of features, number of features after binarization, number of target classes and imbalance ratio. The imbalance ratio is calculated by dividing the minority class by the majority class. In practice, a dataset with a 100% imbalance ratio has perfectly equally distributed target classes, while a dataset with a 25% imbalance ratio indicates that the majority class has four times more occurrences than the minority class. The motivation to extract such a summary of the datasets is to compare the algorithms not only in terms of classification-specific metrics but also by different features of the underlying dataset. Ultimately, the dataset summary can be joined on the problem name to have all the information together to analyse the results also in terms of e.g. dataset size, number of features and imbalance.

3.2 Metrics

Every algorithm loops through all the available datasets. For any of these runs, metrics are extracted and (if available) a tree is plotted and saved for qualitative analysis. The metrics extracted fall into two categories: classification metrics (e.g. accuracy, F1-score) or Tree-specific metrics (e.g. maximum depth, number of leaves), which can be used to judge interpretability. Additionally, the time for fitting and predicting has been measured. Important to note is that not all metrics are applicable for multi-class classification, as they are too complex or not informative enough to compute. In that case,

Table 3: List of all datasets with characteristics

Dataset Name	Sample Size	Features	Binary Features	Target Classes	Imbalance Ratio
ADULT	32,561	107	262	2	31.72%
ATTRITION	1,470	57	354	2	19.22%
BANK_MKT	11,162	51	180	2	90.06%
BANKNOTE	1,372	4	72	2	80.05%
COMPAS	3,518	7	26	2	97.09%
DEFAULT	30,000	32	316	2	28.40%
DIABETES_PIMA	768	8	134	2	53.60%
ECOLI	336	7	94	8	1.40%
GLASS	214	9	138	6	11.84%
HEARTS	303	27	128	2	62.90%
ILPD	579	10	158	2	39.86%
IONOSPHERE	351	34	566	2	56.00%
LAW	22,386	5	44	4	0.03%
LIVER	345	6	104	2	72.50%
MAGIC	19,020	10	180	2	54.23%
MAMMOGRAPHY	11,183	6	70	2	2.38%
MUSHROOM	8,124	117	234	2	93.06%
MUSK	6,598	166	2,922	2	18.22%
NURSERY	12,960	8	52	5	0.05%
OILSPILL	937	48	772	2	4.58%
PHONEME	5,404	5	90	2	41.54%
SEEDS	210	7	126	3	100.00%
SENSORLESS	58,509	48	864	11	100.00%
SKINNONSKIN	245,057	3	54	2	26.19%
STUDENT	647	32	164	5	41.21%
TICTACTOE	958	27	54	2	53.04%
TRANSFUSION	748	4	64	2	31.23%
WDBC	569	30	540	2	59.38%
WINE	178	13	234	3	67.61%

the corresponding metric column is left empty for this run. All the used metrics can be found in Table 4.

Table 4: List of metrics extracted for the comparison

Metric	**Definition**	**Function**	**Applicable**
accuracy	Proportion of correctly classified instances among all.	accuracy_score()*	Binary and multiclass
f1	The harmonic mean of precision and recall.	f1_score()* using micro average for multi-class	Binary and multiclass
auc	Area under graph plotting Recall, FP-Rate (1 - specificity)	roc_auc_score()*	Binary classification
precision	True positives over all positives = $\frac{TP}{TP+FP}$	precision_score()*	Binary classification
recall	True positive Rate = $\frac{TP}{TP+FN}$	recall_score()*	Binary classification
specificity	True negative rate = $\frac{TN}{TN+FP}$	specificity_score()*	Binary classification
#leaves	Number of leaf nodes in decision tree	Custom	Tree
#nodes	Total number of nodes in decision tree	Custom	Tree
#rules	Number of branch nodes = #nodes - #leaves	Custom	Tree
max_depth	Maximum depth of the decision tree.	Custom	Tree
avg_depth	Average depth of nodes in decision tree.	Custom	Tree
features_used	Number features used in the final tree	Custom	Tree
imbalanced	Decision tree is imbalanced if avg_depth $\neq$ max_depth	Custom	Tree
Fitting time	Time for fitting including parameter tuning and validation	Custom	Tree
Predict time	Time for predicting using values given	Custom	Tree

*function taken from scikit-learn https://scikit-learn.org/stable/

3.3 Algorithms

To compare the optimal tree models to traditional decision trees, as well as ensemble models (like Random Forest), four different algorithms are benchmarked. This is done to view this novel optimization approach from the perspective of supposedly more accurate (but less interpretable) black-box models and offers a glimpse into its actual usefulness. The heuristic methods are compared to show that optimal classification trees offer similar or even better performance while being more interpretable (trees of smaller size or depth).

As reference algorithms, ClassificationTree (Tree) and RandomForest (RF) from the scikit-learn package[1] are used. These will be compared to four different optimal tree algorithms from the literature: Optimal Classification Trees (OCT), Binary Optimal Classification Trees (BinOCT), Max-flow Optimal Classification Trees (FlowOCT) with Benders-cuts and Generalized Optimal Sparse Decision Trees (GOSDT).

3.4 Implementation

Additionally, to foster objective comparisons and model assurance (Lamberti, 2023) several methods will be used to better compare the different models and outrule random errors. This includes resampling methods (such as cross-validation and train-test-split), regularisation, and parameter tuning. The regularisation will likely be a potential strategy to increase interpretability as more complex models tend to be more difficult to understand (Zhou et al., 2018). The final implementation of all formulations can be found on GitHub [2].

In general, every implementation is trained on a training set (80% of samples) and validated (and benchmarked) on a test set containing the remaining 20% percent of the samples. Throughout the experiments, the random state is set to a constant value to ensure reproducibility. More importantly, the maximum depth of the generated trees is capped at five. This is done to only compare interpretable trees, as trees with a lower depth are more interpretable (Aghaei et al., 2021) as well as to keep training times lower. However, different implementations (through different methods) perform hyperparameter tuning on the depth to find the optimal depth within the bounds (range 2 to 5).

The hardware used to run the algorithms has the following specifications:

Processor	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx @ 2.10 GHz
Installed RAM	8,00 GB
System	64-bit operating system, Windows 11 Home

[1] scikit-learn tree module documentation: https://scikit-learn.org/stable/modules/tree.html
[2] GitHub repository: https://github.com/NilsKulmbacher/BAN-thesis-optimal-trees

Classification Trees (Tree)

The implementation of ClassificationTree is straightforward. A grid search (GridSearchCV) is done with three folds to tune parameters for depth, minimum samples in a leaf and minimum samples for splitting a node. Here, the depth is varied from two to the maximum depth (=5). The minimum samples to split are varied from two to four, and the minimum samples in a leaf from one to two. Finally, the resulting trees are plotted using the matplotlib package[3].

RandomForest (RF)

Similarly, the implementation of RandomForest uses GridSearchCV for the same parameters with the addition of 'number of estimators' (number of trees), which is varied between 10, 20, 50 and 100. The resulting trees will not be plotted, as this would involve plotting a large number of trees and will not be useful to compare the model manually. The metrics are calculated (if possible) across all resulting trees. Such that the average depth and number of nodes is calculated as the average across all estimators.

Optimal Classification Trees (OCT)

For OCT, the same train-test-split is used, but hyper-parameter tuning (varying tree depth) is done using the built-in function from the IAI package [4]. Extracting tree metrics and plotting the trees are similarly done using the built-in functions, including some list comprehensions to extract special metrics (e.g. average depth). Importantly, OCT is written in the Julia programming language, with a Python wrapper available that was used to call functions from a Python script.

Binary Optimal Classification Trees (BinOCT)

For BinOCT the binary transformed datasets are used. A reference model (ClassificationTree) is used to find the optimal depth and the minimum samples to split, as the implementation used has no capacity for simple hyperparameter tuning. While doing the same steps manually would be possible, limited hardware and training time did not allow for this approach. The reference model is trained using the same steps as are taken during the training of the ClassificationTree. However, the reference model is trained on the same binary dataset used for the final fitting of BinOCT, which is run using a warm start with a time limit of 600 seconds. A warm start uses a heuristic model as the initial solution for the optimization problem solved by the Gurobi solver. As with the other optimal tree models, custom functions are used to extract the metrics, if not

[3]Matplotlib package: https://matplotlib.org/
[4]IAI package: https://docs.interpretable.ai/stable/OptimalTrees/

provided by the implementation itself. Unfortunately, this specific implementation did not allow plotting trees. The implementation by Tang and Lin (2021) is not the official implementation accompanying the paper that introduced this method (Verwer & Zhang, 2019). The officially available code is not commented or easily accessible to allow for a good comparison.

Max-flow Optimal Classification Trees (FlowOCT)

Continuing, the FlowOCT implementation uses the same binary datasets, and the same reference model to find the optimal depth with the same time limit of 600 seconds. To solve the optimization problem, the Gurobi solver is used here as well. As described by Aghaei et al. (2021), using Benders decomposition speeds up computations. The parameter for the objective function is set to accuracy-based, which tries to maximize classification accuracy across all target classes. The resulting trees are plotted using the provided method.

Generalized Optimal Sparse Decision Trees (GOSDT)

Lastly, the implementation for GOSDT uses a train-test-split, which is applied after finding thresholds for features. This processing is part of the algorithm that makes GOSDT so efficient. This transformation is done on the whole dataset such that it can be split afterwards into a train and test set. When computing these thresholds per feature (which is a built-in method), one has to choose a depth for the feature importance algorithm used when finding these thresholds. It was observed that the documentation gives this depth to be one at all times. For multiclass datasets, however, this leads to errors as the underlying model cannot predict thresholds for more than two classes. Therefore, this depth was set to the minimum depth needed to have y_unique possible leaves, with y_unique being the number of target classes in the dataset.

$$d_{threshholds} = \max(1, \lceil \log_2(\text{y\_unique}) \rceil)$$

The next step in the implementation is to compute the warm labels to further speed up the model. The pre-computed values, coming from a scikit-learn GradientBoostingClassifier are then used to provide lower bounds for the optimization problem. Afterwards, the optimization problem is solved using a custom C++ solver. The same time limit (600s) and maximum depth (five) are applied here. The speciality of GOSDT is that the depth is not set as a static parameter but rather as a depth budget, such that the same run will also try out and compare all different trees with depth smaller than the maximum depth, not requiring an intensive cross-validation.

The hyperparameter "regularization", which is a weight for the penalty term in the optimization model has to be chosen in advance, normally requiring hyperparameter tuning. For the sake of this comparison, this parameter was set to 0.05. In their documentation, Seltzer et al. (2024) describe that this parameter should not be below $\frac{1}{n_{samples}}$, and as a default value, their package gives 0.05. Doing additional hyperparameter tuning for this regularization parameter for a specific dataset might result in increased performance and/or interpretability. However, this is outside this project's scope but is briefly explored in Appendix A.

$$\begin{aligned} \text{ComplexityPenalty} &= n_{\text{leaves}} \times \text{regularization} \\ &\text{with regularization} = 0.05 \end{aligned}$$

The implementation has no plotting function build-in or a way to generate all metrics needed for the comparison. Therefore, both of these functions are implemented as a custom function using the JSON format tree output from the implementation

4 Results

While executing the comparison, problems that resulted in errors for specific algorithms were skipped, which resulted in the unavailability of metrics for certain problems (see Table 5). Especially, with the larger multiclass datasets, many algorithms exhausted the hardware available resulting in an "Out of memory" error or slow convergence. This is apparent with the binary MIP formulations (BinOCT and FlowOCT) where only 17 and 19 problems out of 29 could be executed. For OCT, RF and Tree all problems could be executed, while almost all could be executed for GOSDT. The detailed results for all datasets are reported in Tables 8, 9, 10 and 11.

Table 5: Number of datasets successfully executed by model

model	datasets executed
BinOCT	17
FlowOCT	19
GOSDT	26
OCT	29
RF	29
Tree	29

4.1 Metrics analysis

Before analysing the results, it is important to see which of the metrics extracted should be used primarily for comparing performance and interpretability. Looking at Figure 3,

which is showing the min-max scaled mean values of metrics per algorithm, it is clear how the number of nodes, number of rules and the number of leaves are highly correlated. Logically, this makes sense as more branch or leaf nodes also result in more nodes overall. Hence, it is sufficient to use one of these measures to compare the interpretability of trees.

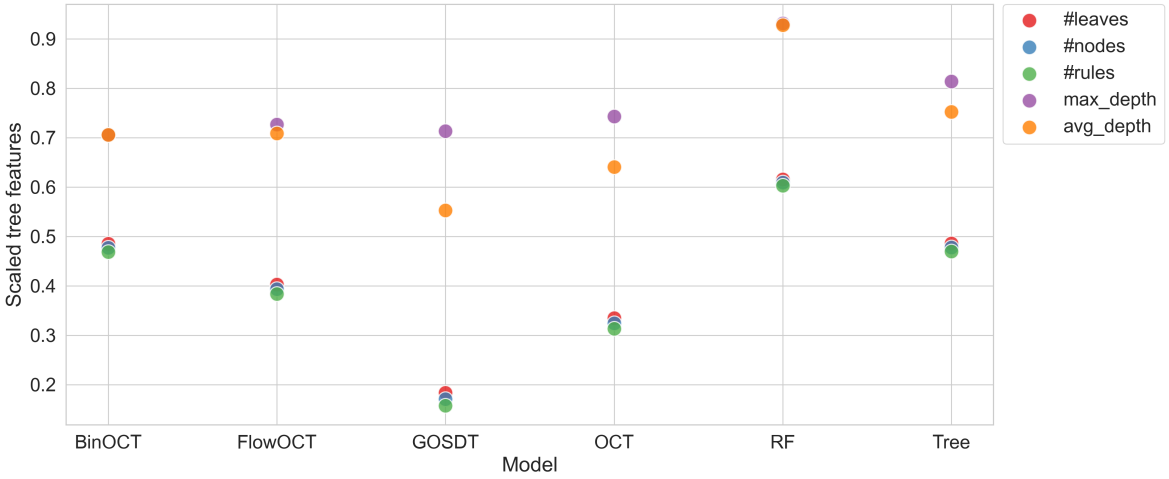


Figure 3: Scaled mean values of tree metrics by algorithm.

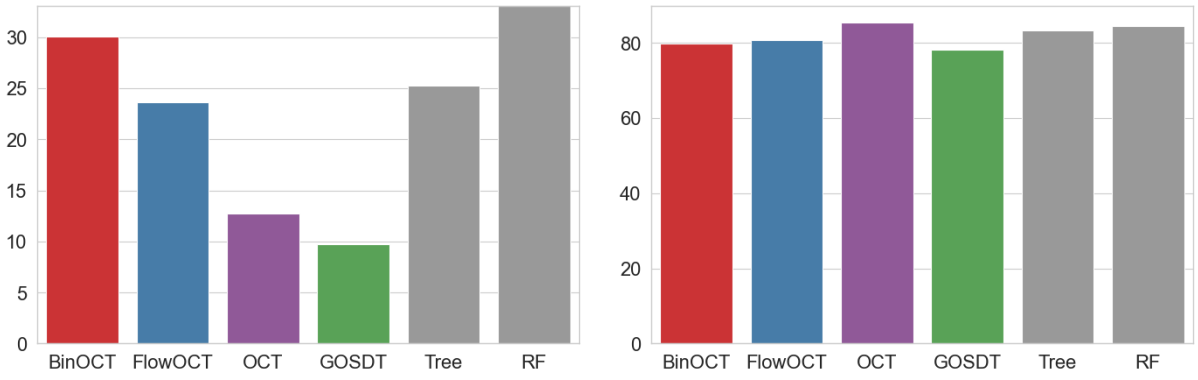
On the other hand, visible differences between the maximum and average depth can be observed. Both the RandomForest, BinOCT and FlowOCT algorithms are almost precisely equal, as with current settings these algorithms cannot produce imbalanced trees. FlowOCT has slight differences due to post-pruning of leaves.

For ClassificationTree, the value for average depth lies slightly below maximum depth, thanks to post-pruning being done by the CART algorithm.

Interestingly, both for OCT and GOSDT, the average depth lies lower than the maximum depth, indicating that the optimization formulation found more imbalanced trees. For this reason, average depth should be taken as the primary metric among the two. The maximum depth does not take into account the imbalance of the trees and therefore underestimates how interpretable they are.

For the performance metrics, accuracy is the most popular metric. As AUC has not been calculated across all datasets, the F1-Score appears to be another good composite score. Especially, in high-stakes situations, where False Positives and False Negatives are important, the F1-Score can bring better clarity on how the classification has performed. This also applies to imbalanced datasets. Therefore, both accuracy and (micro) F1-Score will be used as performance metrics.

4.2 Accuracy/Interpretability Tradeoff



(a) Mean number of nodes by model

(b) Mean accuracy by model

Figure 4: Barplots showing means of accuracy and number of nodes (interpretability) metrics across different algorithms used

To see whether an accuracy/interpretability tradeoff holds within our comparison, the average accuracies and number of nodes for each algorithm are plotted. As can be seen from Figure 4, the optimal trees tend to have fewer nodes but seem to have similar or better accuracy on average, which also holds true when using the average tree depth or F1-Score as metrics. This contradicts the supposed accuracy/interpretability tradeoff. Importantly, these averages are calculated across 17 datasets that have been run on all algorithms, such that the comparison is equal. GOSDT followed by OCT had the most sparse trees, while also offering better or comparable accuracy compared to the reference models. The binary-based models BinOCT and FlowOCT have slightly larger trees and are behind OCT in terms of performance.

4.3 Imbalanced datasets

To compare the results between balanced and imbalanced datasets, the variable 'is_imbalanced' is introduced. A dataset is categorized as imbalanced $if\ imbalance\_ratio < 0.2$, meaning an imbalanced dataset is a dataset where the value count of the minority class represents 20% or less than that of the majority class. From 29 datasets, eight fulfil this requirement and are categorized as an "imbalanced dataset". In Table 6, the main mean metrics are shown across all datasets for each algorithm by imbalance categorization.

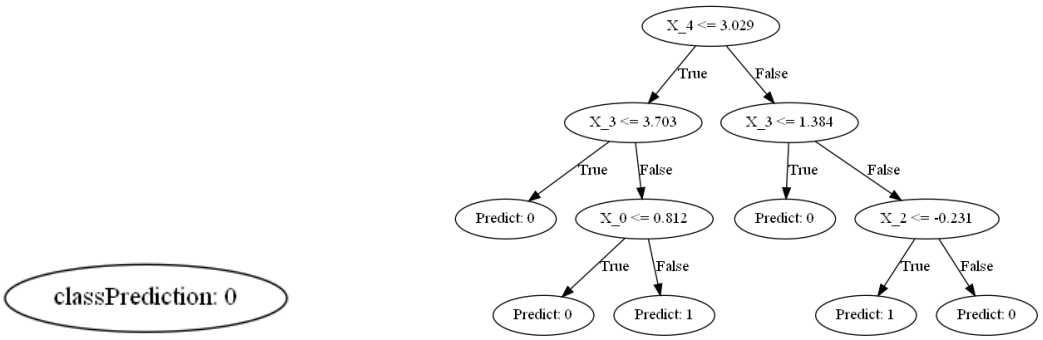
Generally, the accuracy increases for all algorithms besides GOSDT when running an imbalanced dataset. Meanwhile, the F1-Score decreases across all algorithms. Looking at the tree complexity metrics, they seem to decrease across algorithms, when compared to the non-imbalanced datasets, while for RF the values stay more constant.

Table 6: Mean metrics for each algorithm for (not) imbalanced datasets.

Algorithm	**BinOCT**		**FlowOCT**		**GOSDT**		**OCT**		**RF**		**Tree**	
imbalanced	-	+	-	+	-	+	-	+	-	+	-	+
Accuracy	80.86	78.01	78.52	86.54	81.82	80.58	84.87	90.02	**85.07**	**90.09**	83.47	88.21
F1	71.57	60.64	67.37	49.77	74.16	71.02	**77.36**	**71.65**	76.56	68.75	76.08	66.97
#leaves	15.27	16	13.5	11.2	**5.45**	**5.33**	10.67	9.62	19.78	19.44	17.1	11.5
#nodes	29.55	31	26	21.4	**9.9**	**9.67**	20.33	18.25	38.56	37.89	33.19	22
#rules	14.27	15	12.5	10.2	**4.45**	**4.33**	9.67	8.62	18.78	18.44	16.1	10.5
Max Depth	3.55	3.5	3.79	3.2	**3.4**	**2.83**	3.48	3.88	4.57	4.88	4.19	3.75
Avg Depth	3.55	3.5	3.69	3.13	**2.62**	**2.35**	3.05	3.19	4.55	4.88	3.89	3.43
Fit Time (s)	563.16	611.69	594.68	602.15	11.94	9.06	527.05	88.28	120.18	46.74	**4.57**	**1.74**
Count (problems)	11	6	14	5	20	6	21	8	21	8	21	8

Note: Best value for each category in bold.

In summary, looking at the F1-Score which is a better performance indicator when dealing with imbalanced datasets, OCT has the highest value both for balanced and imbalanced datasets. When comparing tree complexity measures, GOSDT has the smallest trees.



(a) GOSDT (b) Tree

Figure 5: Generated trees for dataset MAMMOGRAPHY

Additional parameter tuning is necessary for some problems and algorithms. For example the problem MAMMOGRAPHY, which is a very imbalanced binary problem (imbalance ratio 2.3%), results in underfitting for GOSDT. This can be clearly seen in the tree output (Figure 5) with a depth of zero (root node), which GOSDT found to be the optimal tree. When compared, to the tree generated by Tree we can hypothesize that the regularization parameter was too high. As GOSDT uses misclassification error and tree sparsity in the objective function, with a high regularization, the algorithm seemed to prefer to only predict the majority class which makes up 92% of the dataset. By tuning the regularization parameter, the objective can be navigated such that the model does not underfit (see Appendix A).

4.4 Multiclass classification

Similarly, to the comparison for imbalanced datasets, it is also interesting to compare how the different algorithms performed on binary vs multiclass classification problems. For that, the variable 'is_multi_class' is introduced, which is defined as a dataset having more than two target classes. Of the 29 datasets, eight are multi-class classification problems. The imbalanced and multiclass datasets are not mutually exclusive, four datasets are both imbalanced and multi-class. As can be seen in Table 7, the accuracy metric is decreasing for multi-class datasets across all algorithms, while the F1-Score increases for multi-class datasets. While the number of nodes decreases across all algorithms, the average depth does not follow an increasing or decreasing trend.

Table 7: Mean metrics for each algorithm for binary and multiclass datasets.

Algorithm	**BinOCT**		**FlowOCT**		**GOSDT**		**OCT**		**RF**		**Tree**	
multiclass	**-**	+	**-**	+	**-**	+	**-**	+	**-**	+	**-**	+
Accuracy	82.85	75.57	79.59	82.88	85.68	75.84	**87.68**	85.75	86.54	86.25	**86.11**	81.26
F1	62.21	75.57	53.44	82.88	**77.49**	75.84	75.59	85.75	69.89	**86.25**	70.64	81.26
#leaves	14.8	16.57	13.08	12.5	**6**	**5.5**	11.15	9.62	20.75	16.89	17	11.75
#nodes	28.6	32.14	25.15	24	**11**	**10**	21.3	18.25	40.51	32.78	33	22.5
#rules	13.8	15.57	12.08	11.5	**5**	**4.5**	10.15	8.62	19.75	15.89	16	10.75
Max Depth	**3.4**	3.71	3.69	3.5	3.71	**3.17**	3.7	3.75	4.67	4.62	4.05	4.12
Avg Depth	3.4	3.71	3.59	3.43	**2.82**	**2.6**	3.21	3.19	4.65	4.61	3.8	3.66
Fit Time (s)	569.5	595.71	594.32	601.69	14.2	**7.07**	314.55	684.96	80.11	151.91	**2.26**	7.79
Count (problems)	10	7	13	6	17	6	20	8	21	8	21	8

Note: Best value for each category in bold.

Table 8: Results of accuracy measure in % for each algorithm and dataset.

	Accuracy (%)					
Algorithm	**BinOCT**	**FlowOCT**	**GOSDT**	**OCT**	**RF**	**Tree**
Problem						
ADULT	x	x	84.28	**86.18**	84.22	85.05
ATTRITION	x	x	x	85.37	**87.07**	85.37
BANKNOTE	97.82	94.55	96.36	97.45	**98.55**	96.73
BANK_MKT	x	x	70.35	**80.74**	80.07	79.49
COMPAS	68.32	67.33	67.61	67.61	68.47	**68.89**
DEFAULT	x	x	**99.85**	82.12	81.67	82.1
DIABETES_PIMA	74.03	72.08	72.73	77.27	77.27	**78.57**
ECOLI	76.47	83.82	61.76	79.41	**88.24**	80.88
GLASS	74.42	55.81	58.14	**83.72**	76.74	72.09
HEARTS	**98.36**	**98.36**	**98.36**	**98.36**	**98.36**	**98.36**
ILPD	63.79	62.93	**67.24**	62.93	63.79	61.21
IONOSPHERE	x	71.83	**91.55**	88.73	**91.55**	88.73
LAW	98.5	98.5	98.62	**100**	99.98	**100**
LIVER	66.67	60.87	66.67	65.22	**75.36**	66.67
MAGIC	x	x	82.39	**84.07**	83.78	82.28
MAMMOGRAPHY	98.48	97.76	x	**98.88**	98.79	98.66
MUSHROOM	x	x	98.46	**100**	99.08	**100**
MUSK	x	x	91.36	**96.89**	94.32	93.94
NURSERY	22.84	x	76.77	**78.51**	77.7	77.89
OILSPILL	97.34	96.81	96.81	97.34	**97.87**	96.81
PHONEME	x	73.54	79.93	**84.55**	82.98	81.68
SEEDS	85.71	**88.1**	85.71	**88.1**	85.71	80.95
SENSORLESS	x	x	x	87.97	**90.1**	71.53
SKINNONSKIN	x	x	91.55	**99.54**	98.74	98.42
STUDENT	**73.85**	**73.85**	46.92	**73.85**	71.54	72.31
TICTACTOE	**91.67**	68.75	75	88.02	82.81	91.15
TRANSFUSION	72	73.33	76	**78.67**	76	**78.67**
WDBC	x	**96.49**	**96.49**	**96.49**	**96.49**	95.61
WINE	97.22	97.22	88.89	94.44	**100**	94.44

Note: x indicates that the data is not available. The bolded values represent the highest accuracy achieved for each problem.

Table 9: Results of F1-Score measure in % for each algorithm and dataset.

	F1-Score (%)					
Algorithm	**BinOCT**	**FlowOCT**	**GOSDT**	**OCT**	**RF**	**Tree**
Problem						
ADULT	x	x	62.49	**65.93**	55.84	62.6
ATTRITION	x	x	x	4.44	**13.64**	4.44
BANKNOTE	97.62	93.93	96.03	97.21	**98.43**	96.33
BANK_MKT	x	x	69.83	**79.85**	78.72	78.78
COMPAS	67.45	65.26	**67.52**	**67.52**	66.87	67.36
DEFAULT	x	x	**92.68**	46.43	41.18	46.94
DIABETES_PIMA	61.54	49.41	62.5	66.02	66.67	**67.33**
ECOLI	76.47	83.82	61.76	79.41	**88.24**	80.88
GLASS	74.42	55.81	58.14	**83.72**	76.74	72.09
HEARTS	**98.25**	**98.25**	**98.25**	**98.25**	**98.25**	**98.25**
ILPD	19.23	0	**20.83**	0	8.7	11.76
IONOSPHERE	x	77.78	93.18	90.7	**93.33**	90.91
LAW	98.5	98.5	98.62	**100**	99.98	**100**
LIVER	73.56	70.33	72.29	75	**81.72**	71.6
MAGIC	x	x	87.09	88.16	**88.43**	87.43
MAMMOGRAPHY	63.04	10.71	x	**70.59**	64.94	59.46
MUSHROOM	x	x	98.54	**100**	99.12	**100**
MUSK	x	x	68.33	**89.83**	78.75	78.49
NURSERY	22.84	x	76.77	**78.51**	77.7	77.89
OILSPILL	28.57	0	62.5	**66.67**	50	62.5
PHONEME	x	31.25	64.83	**74.19**	69.74	69.91
SEEDS	85.71	**88.1**	85.71	**88.1**	85.71	80.95
SENSORLESS	x	x	x	87.97	**90.1**	71.53
SKINNONSKIN	x	x	94.57	**99.71**	99.2	99
STUDENT	**73.85**	**73.85**	46.92	**73.85**	71.54	72.31
TICTACTOE	**93.65**	79.31	80.49	91.39	88.17	93.33
TRANSFUSION	19.23	23.08	5.26	**44.83**	30.77	42.86
WDBC	x	**95.45**	95.24	95.12	95.24	94.12
WINE	97.22	97.22	88.89	94.44	**100**	94.44

Note: *x* indicates that the data is not available. The bolded values represent the highest F1 score achieved for each problem.

Table 10: Number of nodes for each algorithm and dataset.

	Number of Nodes					
Algorithm	**BinOCT**	**FlowOCT**	**GOSDT**	**OCT**	**RF**	**Tree**
Problem						
ADULT	x	x	**13**	41	55.44	55
ATTRITION	x	x	x	**5**	48	7
BANKNOTE	63	61	**11**	19	30.4	35
BANK_MKT	x	x	**3**	41	54.84	57
COMPAS	63	53	**11**	**11**	15	61
DEFAULT	x	x	**5**	15	60.9	15
DIABETES_PIMA	31	29	9	**7**	46.72	51
ECOLI	31	31	**3**	9	24.96	27
GLASS	63	47	**17**	21	32.44	23
HEARTS	7	7	**5**	**5**	26.2	**5**
ILPD	7	7	7	**1**	7	21
IONOSPHERE	x	31	13	**5**	17.14	**5**
LAW	7	7	**5**	11	27.44	11
LIVER	15	15	13	**5**	37.88	31
MAGIC	x	x	**23**	37	58.86	61
MAMMOGRAPHY	**15**	**15**	x	19	45.2	**15**
MUSHROOM	x	x	**7**	15	28.4	23
MUSK	x	x	**19**	57	49.5	53
NURSERY	63	x	**9**	19	44.76	33
OILSPILL	7	7	**5**	**5**	30.8	7
PHONEME	x	31	**13**	51	56.2	53
SEEDS	31	29	**7**	**7**	18.68	**7**
SENSORLESS	x	x	x	57	44.9	**35**
SKINNONSKIN	x	x	**5**	35	55.56	57
STUDENT	15	15	15	**11**	52.9	31
TICTACTOE	63	25	**15**	35	56.36	43
TRANSFUSION	15	15	**3**	9	48.2	15
WDBC	x	31	13	**9**	22.04	23
WINE	15	15	**7**	11	16.2	13

Note: x indicates that the data is not available. The bolded values represent the lowest number of nodes achieved for each problem.

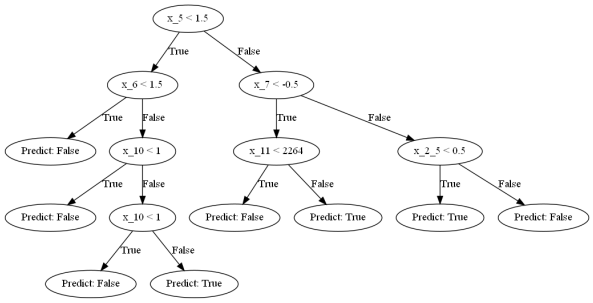
Table 11: Average tree depth for each algorithm and dataset.

	Average Depth					
Algorithm	**BinOCT**	**FlowOCT**	**GOSDT**	**OCT**	**RF**	**Tree**
Problem						
ADULT	x	x	**3.14**	4.57	5	4.89
ATTRITION	x	x	x	**1.67**	5	2
BANKNOTE	5	4.97	**3.33**	3.5	5	4.44
BANK_MKT	x	x	**1**	4.57	5	4.9
COMPAS	5	4.81	3	**2.67**	3	4.97
DEFAULT	x	x	**1.67**	3.12	5	3
DIABETES_PIMA	4	3.93	2.8	**2**	5	4.77
ECOLI	4	4	**1**	2.4	4	3.86
GLASS	5	4.67	**3.44**	3.73	5	3.67
HEARTS	2	2	**1.67**	**1.67**	4.7	**1.67**
ILPD	2	2	2.25	**0**	2	3.64
IONOSPHERE	x	4	3.71	**1.67**	4	**1.67**
LAW	2	2	**1.67**	3	5	3.33
LIVER	3	3	3.29	**1.67**	5	4
MAGIC	x	x	**3.83**	4.58	5	4.97
MAMMOGRAPHY	**3**	**3**	x	4.1	5	**3**
MUSHROOM	x	x	**2.25**	3.12	5	3.92
MUSK	x	x	**3.7**	4.9	5	4.85
NURSERY	5	x	**2.6**	4.1	5	4.76
OILSPILL	2	2	**1.67**	**1.67**	5	2
PHONEME	x	5	**3.71**	4.77	5	4.85
SEEDS	4	3.93	2.25	**2**	4	**2**
SENSORLESS	x	x	x	4.93	5	**4.67**
SKINNONSKIN	x	x	**1.67**	4.39	5	4.9
STUDENT	3	3	3.62	**2.67**	5	4
TICTACTOE	**5**	4	**3**	4.33	5	4.64
TRANSFUSION	3	3	**1**	2.8	5	**3**
WDBC	x	5	3.29	**2.4**	4	3.75
WINE	3	3	**2**	2.67	3.85	3

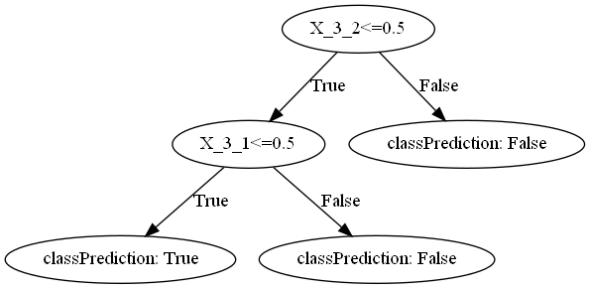
Note: x indicates that the data is not available. The bolded values represent the lowest average depth achieved for each problem.

5 Discussion

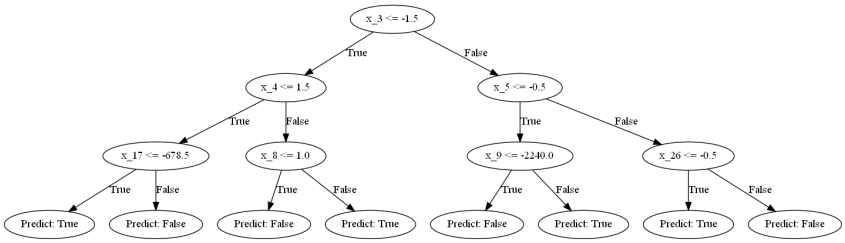
Looking at these results, it is apparent how optimal trees can compete in terms of performance against traditionally generated trees and ensemble models. For 20 out of 29 datasets, one of the optimal tree algorithms archives the same or better accuracy. Looking at the F1-Score, this number climbs to 23 out of 29. At the same time, optimal tree algorithms produced smaller or the same-sized trees for all datasets, making them more interpretable. For some datasets, this difference is more apparent than others. For example, the DEFAULT dataset sees GOSDT have a superior accuracy (and F1-Score) while also producing the smallest tree for this dataset among all methods tested. As can be seen in Figure 6, GOSDT produces a tree that is more comprehensive, imbalanced and uses only two features. Looking at the basic Tree (Figure 6c), or even at the OCT tree (Figure 6a), they produce bigger trees using more features, involving more decisions. For this specific problem, the "price of interpretability" seems to be non-existent - a smaller model can outperform more complex models in terms of classification performance. This highlights the tremendous value of global optimality through mathematical optimization.



(a) OCT



(b) GOSDT

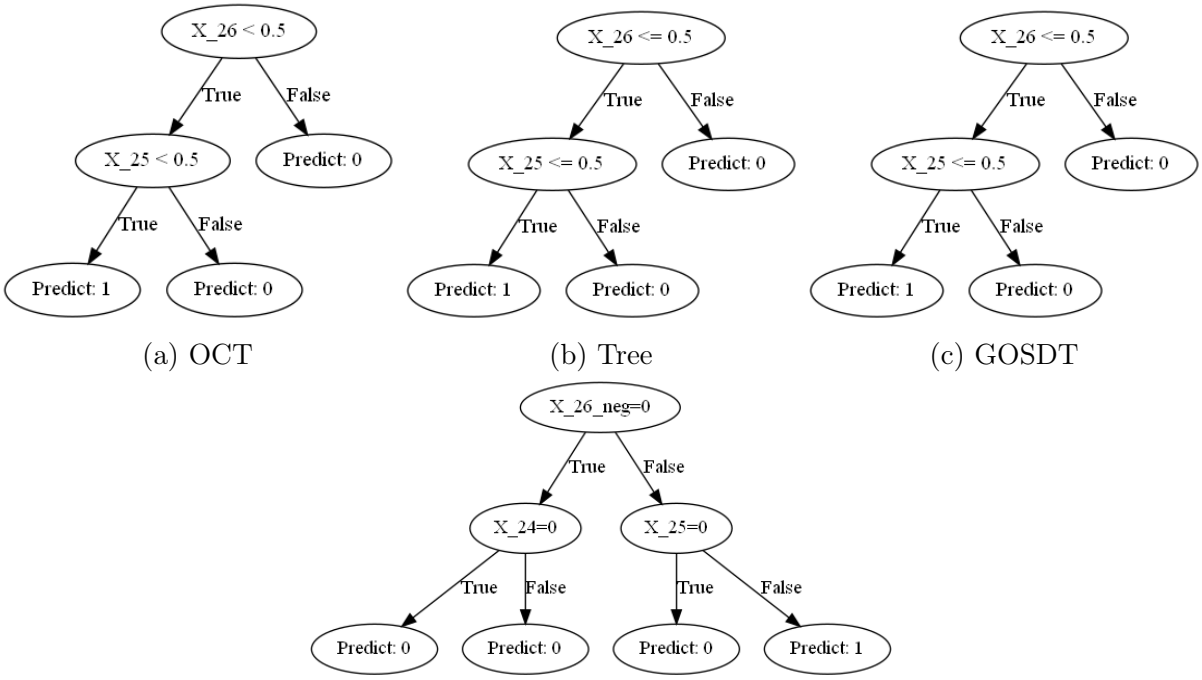


(c) Tree

Figure 6: Generated trees for dataset DEFAULT.

However, as the detailed results (Table 8,9,10 and 11) already suggest, this is not the case for all problems. For problem HEARTS all algorithms produce the same result in terms of performance (accuracy and F1-Score) with only slight differences in trees (Figure 7). Here, OCT, GOSDT and Tree produce the same tree with the same features, highlighting that CART can also find the optimal tree. FlowOCT and BinOCT produce slightly bigger trees, possibly due to the binarization of features for this dataset. Compared to a Random Forest model, which is much harder to explain, all methods (including

the basic Tree) perform the same while being much more interpretable, using a single tree to derive decisions.



(a) OCT (b) Tree (c) GOSDT

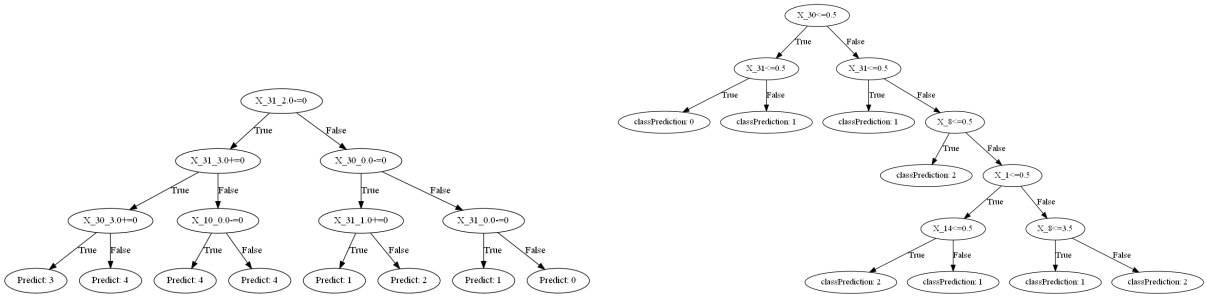
(d) FlowOCT

Figure 7: Generated trees for dataset HEARTS.

As mentioned in Subsection 4.3, imbalanced datasets can lead to optimal models underfitting due to the prediction of the majority class to maximize accuracy. For the STUDENT dataset, the imbalance does not lead to underfitting, still, the GOSDT tree does not predict all existing target labels, while FlowOCT and OCT do. It is remarkable how FlowOCT manages to achieve a better performance using only binary trees. This poor fit could be because GOSDT, as opposed to the other methods uses weighted samples importance for the accuracy objective. This means that it prefers the tree that predicts most classes correctly, no matter if entire target classes are misclassified. As the F1-Score is a more informative measure for imbalanced datasets, GOSDT has a more than 30%-point lower score than other methods for this dataset. This could be mitigated by using a balanced setting for GOSDT, which equalizes the importance of each target class.

In terms of interpretability, Figure 8 visualizes how different metrics for interpretability are useful for judging a tree's interpretability. Both trees have the same amount of nodes but FlowOCT has a lower average depth. Connecting back to the notion of explainability, trees with lower depth are more interpretable because they require to keep less previous decisions in mind when traversing the tree.

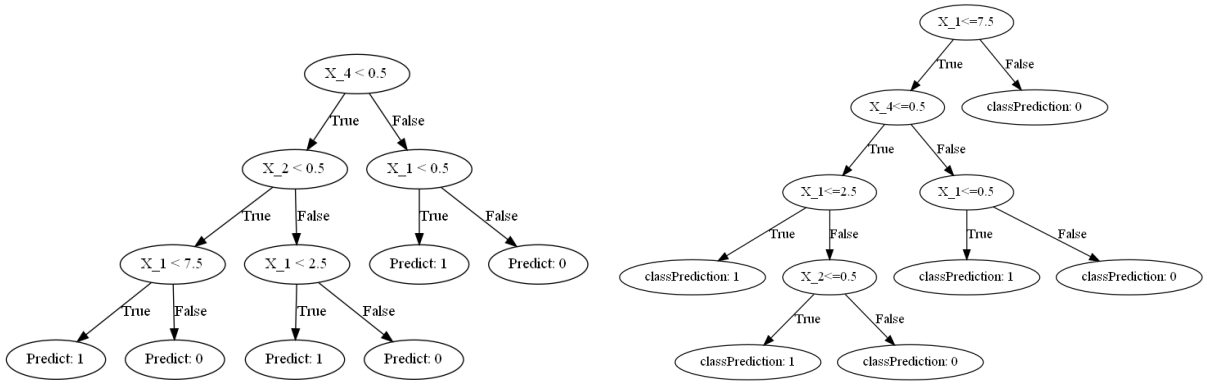
Another interesting observation is the slight difference in optimal trees found by the different methods. As BinOCT and FlowOCT are harder to compare, due to their binary decisions the focus here will be on comparing OCT and GOSDT. For the dataset



(a) FlowOCT (b) GOSDT

Figure 8: Generated trees for dataset STUDENT

COMPAS, both OCT and GOSDT use the same features with the same splits, but with a slightly different structure (Figure 9). The sequence in which the splits are organized within the trees varies. Besides a slight difference in average depth, they score equally on performance and interpretability metrics. Therefore, there are likely multiple optimal solutions for this specific dataset in this case.



(a) OCT (b) GOSDT

Figure 9: Generated trees for dataset COMPAS

5.1 Training time

A significant factor for the feasibility of optimal tree methods is a relatively fast training time, allowing practitioners to iterate and improve the model faster. When looking at all training times side by side across all datasets, there are huge differences in training times across algorithms and datasets. In Figure 10, the datasets have been sorted by sample size in ascending order, such that it is easier to see whether some algorithms have higher training times for bigger datasets.

Firstly, FlowOCT and BinOCT mostly hit the time limit of 600 seconds and did not produce results for the bigger datasets. Still, a time limit is necessary as these optimization problems can take a long time to converge. Mostly, after the cut-off time, trees are still generated, but they might not be optimal. Judging from the optimization

outputs, which display the current optimality gap these two methods take a long time to get started and then converge quite fast, hence the timeout occurring for most datasets. Multiple datasets could not be executed due to memory errors and non-convergence to any basic solution likely due to limited hardware. This occurs more and more with datasets with a higher number of samples.

Secondly, OCT, RF and Tree positively correlate with increasing sample size. For OCT this can be expected as the number of constraints for the optimization formulation depends directly on the number of samples in the dataset. For Tree and RF, the computational complexity with these heuristic models increases with sample size (Sani et al., 2018).

- **Decision Tree** Train Time Complexity: $O(n \cdot \log(n) \cdot m)$
- **Random Forest** Train Time Complexity: $O(k \cdot n \cdot \log(n) \cdot m)$

Thirdly, the training times using the GOSDT algorithms are the most similar to the standard ClassificationTree algorithm. Of all the optimal tree formulations, GOSDT is the fastest. This exemplifies how effective the branch & bound method, paired with reference ensemble models is. However, for specific problems, GOSDT takes notably more time to train the model, which can be attributed to the pre-processing of thresholds for different splits that are done.

5.2 Limitations

Hardware: Looking at similar research (Tang & Lin, 2021), the hardware used for such comparisons was more advanced with better processors, and much more RAM. This would allow to run all the different datasets, without running out of memory such that all optimization problems converge and the comparison is based on globally optimal trees only.

Parameter tuning: As can be seen in Figure 5 and Figure 8, where GOSDT underfitted or did not predict all target classes, more tuning is needed. This is necessary to address the different nature of datasets such as different imbalance ratios, multi-class classification and number of features. Ideally, every model would be tested using the same cross-validation with the same parameters to tune such as depth and regularization. For example, this would mean not using a reference model to determine the optimal depth but running BinOCT and FlowOCT several times on different folds of the dataset to figure out the optimal depth. This would make a stronger comparison but was out of the scope of this project due to time constraints.

Implementations: The implementations used to execute the different optimal tree formulations are mostly the official implementations provided by the authors of the paper. However, due to a lack of documentation for BinOCT, a third-party implementation was

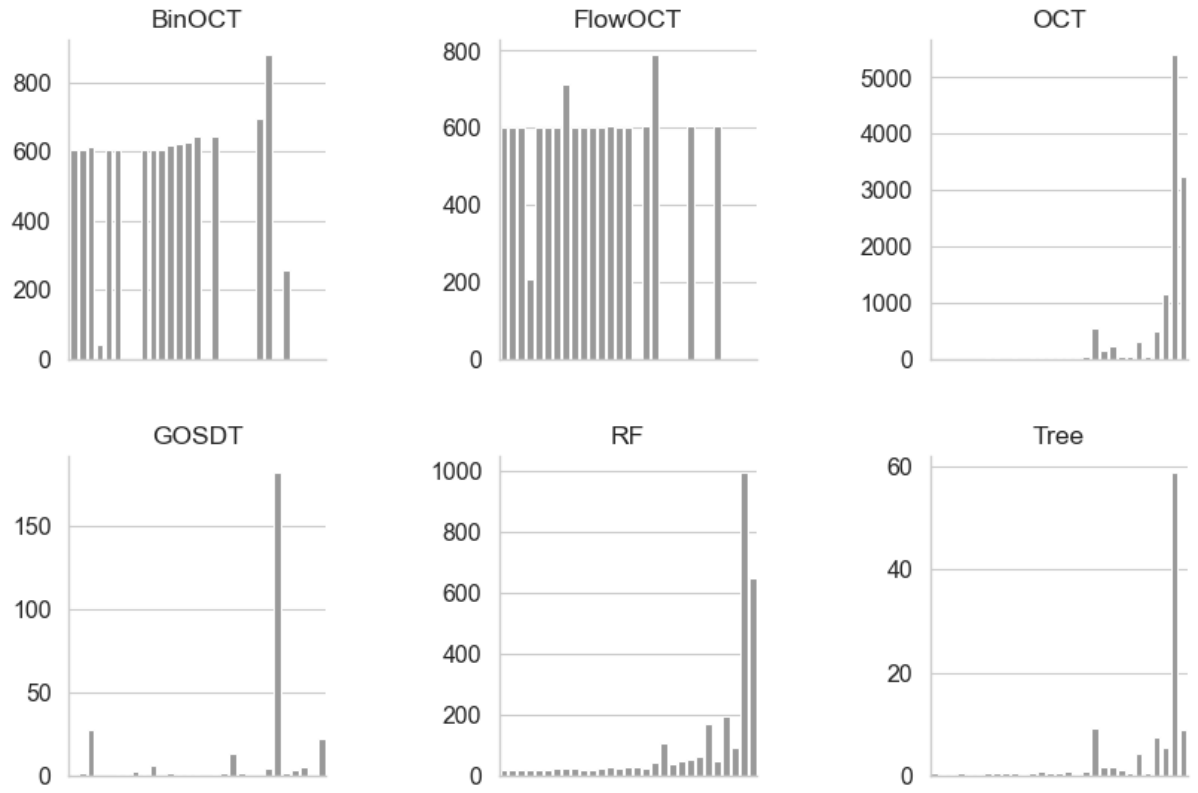


Figure 10: Comparison of training times in seconds for each optimal tree algorithm. Each vertical bar represents the training time for one dataset, a gap indicates a dataset was not executed successfully.

used here. Therefore, the execution issues (only able to run on 17 out of 29 datasets) could be attributed to the implementation rather than the formulation. Furthermore, the implementation of OCT is written in a different programming language than the others, namely Julia with a wrapper for Python compatibility. It is unclear how this impacts performance and results. Likewise, the custom C++ solver might impact results compared to the other models that use the Gurobi solver. For GOSDT specifically, it might also be interesting to look further into why computing thresholds takes so long for bigger datasets.

5.3 Future Research

Stability analysis: Especially when multiple solutions seem to be optimal (Figure 9), the question of the stability of the solution arises. Therefore, it might be insightful to further look into the stability of these optimal tree methods to see what happens when different permutations of the dataset are used for training. A stable tree that stays optimal could be arguably more desirable than when the underlying tree changes easily, for example when outliers affect the resulting tree disproportionately.

Bigger comparison: To truly judge when and if interpretable optimal trees could be

used instead of black-box models, a wider comparison with more types of black-box models would be needed. For example, to find out how optimal trees perform against SVMs or neural networks on structured datasets.

Regularization: Ideally, optimal trees allow for better accuracy while being more sparse. In some instances, accuracy drops when making increasingly smaller trees. When deciding on a regularization parameter to decide between accuracy and sparsity objectives when optimizing trees, for each model different outcomes can be seen. Even if hyperparameter tuning is applied, it is hard to select a parameter for regularization without a metric that encompasses accuracy and interpretability (or tree sparsity). For example, it is unclear what would be preferred if a much smaller tree would result in a slight decrease in accuracy - Where do you draw the line? Hence, a methodology to quantify the best trade-off between potential accuracy loss and added interpretability needs to be researched further.

6 Conclusion

Between all 29 datasets benchmarked, in 19 cases the optimal methods generate a tree that is both more accurate and more interpretable, based on accuracy and number of nodes measured. This highlights the value of using interpretable optimal trees as opposed to a black-box model.

As Rudin (2019) indicated, the supposed Interpretability/Accuracy trade-off does not seem to hold. Considering that optimal trees are more interpretable than a black-box model such as a Random Forest while still performing similarly or even better, there is no indication that this trade-off always stands, as suggested in the literature.

There is no doubt however that interpretable models are needed for high-stakes decisions. There are numerous cases where the lack of interpretability leads to problems, that presumably could be avoided. Nevertheless, unstructured datasets (such as SENSORLESS) still see a black-box model outperforming optimal trees.

Considering the outcomes of this comparison, GOSDT seems like the most promising choice for generating optimal trees. Its performance is comparable to reference models, even if accuracy is slightly lower compared to OCT. On the other hand, it generates the smallest trees and runs importantly with fast training times allowing for many iterations and more tuning, which is necessary to bridge interpretability and accuracy.

References

Aghaei, S., Gómez, A., & Vayanos, P. (2021). Strong optimal classification trees. *arXiv (Cornell University)*. https://doi.org/10.48550/arXiv.2103.15965

Alès, Z., Huré, V., & Lambert, A. (2024). New optimization models for optimal classification trees. *Computers & operations research*, *164*. https://doi.org/10.1016/j.cor.2023.106515

Bertsimas, D., Delarue, A., Jaillet, P., & Martin, S. (2019). The price of interpretability. *arXiv.org*. https://doi.org/10.48550/arXiv.1907.03419

Bertsimas, D., & Dunn, J. (2017). Optimal classification trees. *Machine Learning*, *106*, 1039–1082. https://doi.org/10.1007/s10994-017-5633-9

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. Belmont, Calif. Wadsworth International Group.

Dasting, J. (2018). *Amazon scraps secret ai recruiting tool that showed bias against women* [Reuters]. https://www.reuters.com/article/idUSKCN1MK0AG

de Bruijn, H., Warnier, M., & Janssen, M. (2021). The perils and pitfalls of explainable ai: Strategies for explaining algorithmic decision-making. *Government Information Quarterly*, *39*, 101666. https://doi.org/10.1016/j.giq.2021.101666

Doran, D., Schulz, S., & Besold, T. R. (2017). What does explainable ai really mean? a new conceptualization of perspectives. *ArXiv*, *abs/1710.00794*. https://doi.org/https://doi.org/10.48550/arXiv.1710.00794

European Parliament. (2023). Eu ai act: First regulation on artificial intelligence. Retrieved June 20, 2024, from https://www.europarl.europa.eu/topics/en/article/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence

European Union. (2016). Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation) (text with eea relevance).

Hoffman, R. R., Mueller, S. T., Klein, G., & Litman, J. A. (2018). Metrics for explainable ai: Challenges and prospects. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1812.04608

Hossin, M., & Sulaiman, M. (2015). A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process*, *5*, 01–11. https://doi.org/10.5121/ijdkp.2015.5201

Hu, X., Rudin, C., & Seltzer, M. I. (2019). Optimal sparse decision trees. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1904.12847

Interpretable AI. (2023). Interpretable ai documentation. https://www.interpretable.ai

Jo, N., Aghaei, S., Gómez, A., & Vayanos, P. (2023). Learning optimal fair classification trees: Trade-offs between interpretability, fairness, and accuracy. *arXiv.org*. https://doi.org/10.1145/3600211.3604664

Kin Keung, L. (2012). An example of classification tree. *researchgate*. https://www.researchgate.net/publication/233842752_Benchmarking_Binary_Classification_Models_on_Data_Sets_with_Different_Degrees_of_Imbalance

Kuźniacki, B. (2023). De toeslagenaffaire toont aan dat we uitlegbare ai-regels nodig hebben. *Universiteit van Amsterdam*. Retrieved June 20, 2024, from https://www.uva.nl/shared-content/faculteiten/nl/faculteit-der-rechtsgeleerdheid/nieuws/2023/02/de-onthulling-van-het-kinderopvangtoeslagschandaal-kan-betekenen-dat-nederland-vooroploopt.html

Lamberti, W. F. (2023). An overview of explainable and interpretable ai. *AI Assurance Towards Trustworthy, Explainable, Safe, and Ethical AI*, 55–123. https://doi.org/10.1016/b978-0-32-391919-7.00015-9

Lawless, C., Dash, S., Günlük, O., & Wei, D. (2021). Interpretable and fair boolean rule sets via column generation. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2111.08466

Longo, L., Goebel, R., Lecue, F., Kieseberg, P., & Holzinger, A. (2020). Explainable artificial intelligence: Concepts, applications, research challenges and visions. *Lecture Notes in Computer Science*, 1–16. https://doi.org/10.1007/978-3-030-57321-8_1

Marutho, D., Hendra Handaka, S., Wijaya, E., & Muljono. (2018). The determination of cluster number at k-mean using elbow method and purity evaluation on headline news. *2018 International Seminar on Application for Technology of Information and Communication*, 533–538. https://doi.org/10.1109/ISEMANTIC.2018.8549751

McTavish, H., Zhong, C., Achermann, R., Karimalis, I., Chen, J., Rudin, C., & Seltzer, M. (2021). Fast sparse decision tree optimization via reference ensembles. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2112.00798

Moraffah, R., Karami, M., Guo, R., Raglin, A., & Liu, H. (2020). Causal interpretability for machine learning - problems, methods and evaluation. *ACM SIGKDD Explorations Newsletter*, *22*, 18–33. https://doi.org/10.1145/3400051.3400058

Morgan, J. N., & Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, *58*, 415–434. https://doi.org/10.1080/01621459.1963.10500855

Quinlan, J. R. (1992). Learning with continuous classes. https://api.semanticscholar.org/CorpusID:1056674

Rosenfeld, A. (2021). Better metrics for evaluating explainable artificial intelligence. *Proceedings of the 20th International Conference on Autonomous Agents and Multi-Agent Systems*, 45–50.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*, 206–215. https://doi.org/10.1038/s42256-019-0048-x

Rudin, C. (2021). Please stop explaining black box models and use interpretable models instead, *YouTube*. Retrieved June 1, 2024, from https://www.youtube.com/watch?v=sl78EgrT4TY

Sani, H. M., Lei, C., & Neagu, D. (2018). Computational complexity analysis of decision tree algorithms. *Lecture Notes in Computer Science*, 191–197. https://doi.org/10.1007/978-3-030-04191-5_17

Schmidt, P., & Biessmann, F. (2019). Quantifying interpretability and trust in machine learning systems. *arXiv (Cornell University)*. https://doi.org/https://doi.org/10.48550/arXiv.1901.08558

scikit-learn. (2009). 1.10. decision trees — scikit-learn 0.22 documentation. *Scikit-learn.org*. https://scikit-learn.org/stable/modules/tree.html

Seltzer, T., Seltzer, M., & McTavish, H. (2024). Implementation of general optimal sparse decision tree. *PyPI*. Retrieved May 30, 2024, from https://pypi.org/project/gosdt/

Tang, B., & Lin, B. (2021). Optimal decision tree - mip formulations, solution methods, and stability. *GitHub*. Retrieved May 30, 2024, from https://github.com/LucasBoTang/Optimal_Classification_Trees

Verwer, S., & Zhang, Y. (2019). Learning optimal classification trees using a binary linear program formulation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 1625–1632. https://doi.org/10.1609/aaai.v33i01.33011624

Vilone, G., & Longo, L. (2021). Notions of explainability and evaluation approaches for explainable artificial intelligence. *Information Fusion*, *76*, 89–106. https://doi.org/10.1016/j.inffus.2021.05.009

Vujovic, Ž. Đ. (2021). Classification model evaluation metrics. *International Journal of Advanced Computer Science and Applications*, *12*. https://doi.org/10.14569/ijacsa.2021.0120670

Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., & Zhu, J. (2019). Explainable ai: A brief survey on history, research areas, approaches and challenges. *Natural Language Processing and Chinese Computing*, *11839*, 563–574. https://doi.org/10.1007/978-3-030-32236-6_51

Zhou, Q., Liao, F., Mou, C., & Wang, P. (2018). Measuring interpretability for different types of machine learning models. *Lecture notes in computer science*, 295–308. https://doi.org/10.1007/978-3-030-04503-6_29

Appendices

A GOSDT: parameter tuning

As mentioned before, the regularization parameter for running GOSDT depends on the problem, so a constant value will not lead to optimal results. With using a constant value, some trees overfit while others underfit. Therefore, picking a regularization parameter is not as straightforward, especially considering interpretability at the same time. This analysis tries to highlight how this parameter could be potentially tuned and help answer the following questions:

For a specific dataset, can GOSDT generate a smaller tree that offers the same performance or comparable performance? Where can potential accuracy loss justify added interpretability?

A.1 Tuning graph

Similar to how the optimal number of clusters is found using the 'Elbow' Method (Marutho et al., 2018), a graph with varying regularization and two curves for classification performance and interpretability could be formed. This would allow a professional to decide on a case-to-case basis which regularization should be applied. For example, if a slight loss in accuracy results in a much more interpretable tree, they could decide to choose that over the maximum accuracy possible.

Therefore, the methodology for creating such a graph will be looping through many possible values for the regularization parameter and plot the results accordingly. As GOSDT runs quite fast, this will not be a big computational effort. For imbalanced problems, two graphs can be drawn with different settings for 'balance' which equates to the target class importance and supposedly helps to deal with imbalanced datasets. In summary, the following key metrics will be considered:

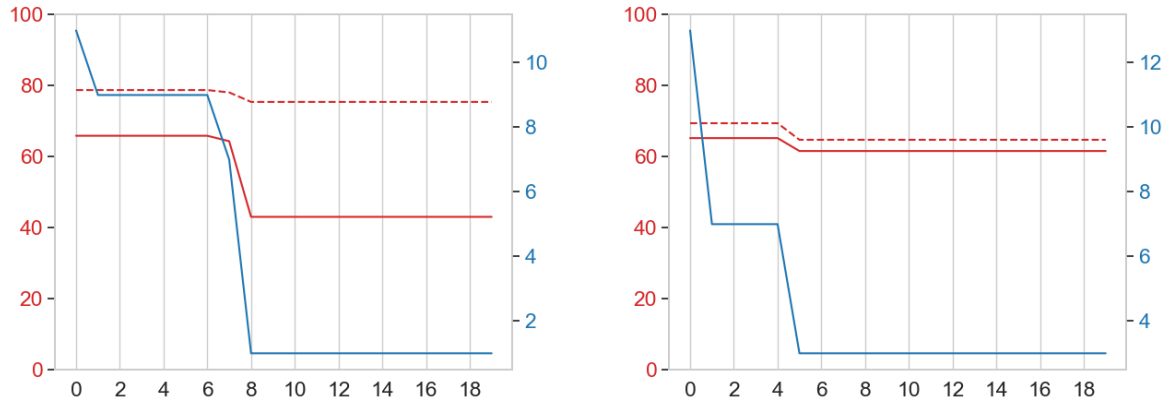
- **Performance:** Accuracy + F1-Score
- **Interpretability:** Number of nodes
- **Regularization:** Ranges from 0 to 50 multiplied by $\frac{1}{n\_samples}$
- **Balance setting:** GOSDT parameter equalizing the importance of each target class (*True* or *False*)

These metrics will provide insights into both the performance and interpretability of the model, as well as how regularization impacts its behaviour. For performance, we plot Accuracy and F1-Score on the y-axis, to also see the difference for imbalanced datasets. For interpretability, we plot the number of nodes as the penalty term for

sparsity directly includes this metric, such that increasing regularization decreases the number of nodes. Finally, the regularization parameter is varied from 0 to 50 times $\frac{1}{n\_samples}$. We also test zero (meaning no regularization), even if $\frac{1}{n\_samples}$ is the lowest regularization suggested by McTavish et al. (2021) for sparse trees and efficient training.

A.2 Results

In Figure 11, we can see two tuning graphs of the dataset TRANSFUSION, with different "balance" settings. This problem is a binary classification that is quite imbalanced (ratio 31.23%). The previously generated GOSDT tree performs acceptable in terms of accuracy, but insufficient in terms of F1-Score. As previously shown with MAMMOGRAPHY (Figure 5), the tree is underfitted, with a depth of zero such that only the majority class is always predicted. This can be observed at a regularization of 8 or more in (Figure 11a). When enabling the target class balance, this does not happen anymore. Furthermore, the drop in F1-Score is much smaller using the balanced setting, while the decline in accuracy stays the same.



(a) Balance: *False*

(b) Balance: *True*

Figure 11: Tuning graphs for dataset TRANSFUSION showing accuracy (dotted red, left axis), F1-Score (red, left axis), and number of nodes (blue, right axis) for each regularization parameter with different balance settings.

When looking at the plots of the smallest trees for either setting (in this case Regularization=8), F1-Scores improve when using the balanced mode. However, the improvement is drastically more when compared to the F1-Score achieved during the prior comparison using a constant regularization. The balanced GOSDT tree results in the highest F1-Score (61.50%) across all methods for this dataset, while also being the most sparse tree. This shows how tuning the GOSDT model can improve performance on imbalanced datasets. On the other hand, the balanced setting has little effect, when applied to a balanced dataset. When comparing results for balanced datasets such as COMPAS or BANKNOTE, there is little difference in performance or interpretability.

This makes sense as GOSDT assigns importance based on target class distribution. Hence, when the distribution is relatively equal, the balanced setting has little effect. Also, for other imbalanced datasets such as NURSERY and STUDENT, the balanced setting has little effect on the tuning graph. One possible reason for that could be that they are multi-class classifications whereas TRANSFUSION is binary. What is remarkable to see, is how optimal trees often do not lose accuracy while making smaller and smaller trees with increasing regularization. This can be seen in Figure 12, while in Figure 13 there are clear drops in accuracy for smaller trees, still allowing to pick the desired interpretability and accuracy.

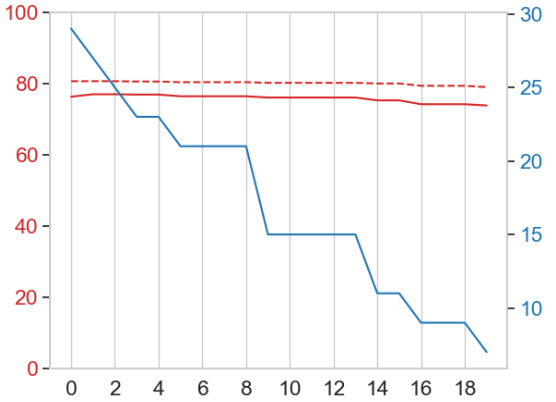


Figure 12: PHONEME

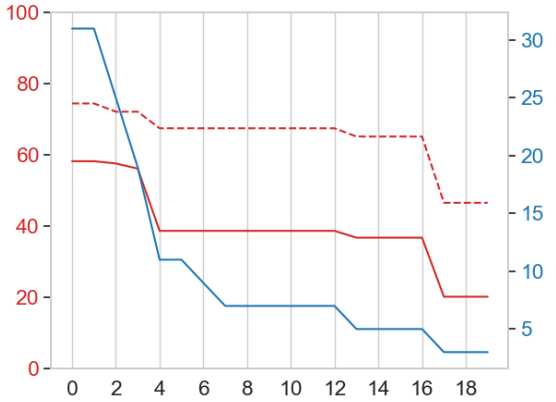


Figure 13: GLASS

Tuning graphs showing accuracy (dotted red, left axis), F1-Score (red, left axis), and number of nodes (blue, right axis) for different regularization parameters. Balanced: *False*

Ultimately, for many datasets, tuning allows to address imbalance and underfitting. More important, however, is the ability to allow for custom tuning by a professional. As with every high-stakes decision problem, the desired outcome is highly subjective. Therefore, a professional might have a certain tree size in mind and want to find the most accurate tree that is this size. Conversely, the most interpretable tree could be found that meets a certain accuracy threshold. Hence, through hyperparameter tuning GOSDT can be even more effective and useful.