Pareto-optimal trees and Pareto forest: a bi-objective optimization model for binary classification

Marianna De Santis* Daniele Patria† Justo Puerto‡

July 24, 2025

Abstract

As inherently transparent models, classification trees play a central role in interpretable machine learning by providing easily traceable decision paths that allow users to understand how input features contribute to specific predictions. In this work, we introduce a new class of interpretable binary classification models, named Pareto-optimal trees, which aim at combining the complementary strengths of Optimal Classification Trees (OCT) and Support Vector Machines (SVM). We formulate a bi-objective mixed integer quadratic optimization problem, whose nondominated solutions represent trade-offs between these two different classification techniques. To further enhance robustness and performance, we propose the Pareto Forest, an ensemble method based on the Pareto-optimal trees, aggregated through majority voting. Extensive experiments on benchmark datasets demonstrate that our models achieve competitive or superior accuracy compared to standard methods such as CART and OCT, underscoring the improvements gained through the bi-objective perspective. In particular, Pareto-optimal trees unify the ability of OCT and SVM within a single framework, resulting in enhanced classification performance relative to either method alone. Embracing a multiobjective perspective allows the construction of multiple "high-quality" trees. Our comparison between Pareto Forests and Random Forests shows that building shallow ensembles from a small number of such optimized trees outperforms relying on a large set of random trees with variable depth.

Key Words: Machine learning, Optimal classification trees, Support vector machines, Multiobjective Optimization.

Mathematics subject classifications (MSC 2020): 62H30, 90C90, 90C29.

1 Introduction

Interpretability, robustness and reproducibility are nowadays important requisites demanded to machine learning methods provoked by known biases and lack of transparency of some of the methodologies appearing in the last decade [16].

*Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Firenze, Via di Santa Marta 3, 50139 Firenze, Italia (marianna.desantis@unifi.it)

†DIAG, Sapienza, Universit'a di Roma, Via Ariosto 25, Roma, 00185, Italy. (daniele.patria@uniroma1.it)

‡IMUS, Universidad de Sevilla, 41012 Sevilla, Spain. (puerto@us.es)

To achieve interpretability, various techniques have been explored. One widely used approach is feature selection, which involves identifying a smaller subset of relevant features without compromising predictive performance. By reducing the number of parameters, the resulting models become more transparent and easier to understand, thus enhancing their explanatory power. Another approach involves using models that are inherently interpretable, meaning that significant parts of their prediction process can be independently understood. Additionally, some methods define interpretability as the ability for the model's construction to be fully replicated and understood by humans [7, 26]. A clear example is shallow Decision Trees, which, due to their limited depth, are easily visualized and interpreted—even by users without technical expertise in the underlying algorithms. A similar issue appears in bagging methods, as random forest, where using a large number of trees of different depth makes those methods hardly interpretable even for educated experts in machine learning.

Classification methods are designed to accurately predict the class of new observations based on a sample used to construct a corresponding classification rule. The use of Mathematical Programming in developing classification models is well-established see e.g. [6, 10, 5, 22]. Furthermore, Mathematical Programming has proven to be a versatile and effective framework for incorporating interpretability into classification models [21]. Usually, each methodology is based on a different rational trying to focus at (at times, to optimize) a different metric. In all cases, the metrics used are meaningful but each one adapt better to the corresponding methodology. For instance, maximizing margin in support vector machine (SVM) or misclassification rate in Classification Trees (CT) or bagging methods. Needless to say, combining these principles is an appealing idea worth to be explored.

In this paper, we want to elaborate into this direction combining two supervised classification methods based on different rationale, namely optimal classification trees and support vector machines. This is done by solving a newly defined bi-objective optimization model: we propose a new class of classification trees based on the non-dominated outcomes with respect to the above mentioned two methods, SVM and OCT, that we call Pareto-optimal trees. Furthermore, we exploit the coverage of our classifiers over the Pareto front, to present another new forest classifier, the Pareto forest, that exploits the non-dominance of our Pareto trees to produce shallow and interpretable forests. The goal is to gain from the characteristics of the two methods to assess the issues of robustness and interpretability.

Classification Trees (CT), first introduced by [7], use a hierarchical structure of nodes to guide observations from a root node to terminal leaves, where class labels are assigned. These paths are determined based on optimization criteria applied to the predictor variables in the training data. The resulting classification rules are intuitive and easily interpretable through the sequence of splits made at each node.

The original CART algorithm proposed in [7] constructs the tree using a greedy approach. Starting at the root with the entire training set, it iteratively splits the data by minimizing an impurity measure, forming two child nodes at each step. This process continues until a stopping condition is met. Similar greedy strategies are used in other well-known methods like ID3 [30] and C4.5 [31]. Traditional tree-based classification methods often produce overly deep and complex trees, which can lead to overfitting and reduced interpretability. This issue is typically addressed through pruning, which balances the reduction in impurity with the added complexity of the tree. Recent advances in optimization techniques, along with the flexibility of these models, have led to the successful application of optimization-based approaches in supervised classification

[3, 10]. Remarkably, [2] introduced the concept of Optimal Classification Trees (OCT), framing the tree construction process as a Mixed Integer Linear Programming problem to find globally optimal solutions. In contrast to the standard CART approach, OCT builds the tree by solving a single optimization problem taking into account (in the objective function) the complexity of the tree, avoiding post pruning processes. Moreover, every split is directly applied in order to minimize the misclassification errors on the terminal nodes.

Support Vector Machines (SVM), introduced by Cortes and Vapnik [11], builds classifiers by means of a separating hyperplane with large margin between classes. This hyperplane is obtained by solving a Non Linear Problem (NLP), in which the goal is to separate data by their two classes, maximizing the margin between them and minimizing the misclassification errors.

The main differences between SVM and Classification Trees: SVM accounts for misclassification errors based on distances (to the separating hyperplane), i.e., the closer to the correct side of the separating hyperplane, the better, whereas in CT all misclassified observations are equally penalized.

Recent work has focused on improving the efficiency of training Optimal Classification Trees (OCTs). Several algorithms based on dynamic programming have been proposed to construct OCTs and enhance them with additional features [14, 27, 25]. [15] introduced a bi-objective optimization approach that incorporates nonlinear classification metrics as alternatives to traditional accuracy, and also supports sparse models. Other optimization techniques have also been explored, including Constraint Programming [33] and SAT-based approaches [35, 24]. Additionally, [34] and [20] proposed alternative integer programming formulations that use significantly fewer decision variables compared to the original OCT model by [2].

Combining Support Vector Machines (SVM) with classification tree methods to build enhanced classifiers is not a new idea; a related approach was proposed by [1]. However, their method integrates margin maximization into the greedy CART framework by sequentially optimizing over fixed assignments of observations to tree leaves. More recently, [4] combines CT and SVM into a mathematical program to correct label noise and minimize misclassification error. In [17], OCT and SVM are also combined to incorporate maximum margin multivariate hyperplanes nested within a binary tree structure.

Our new approach is to incorporate in the combination of OCT with SVM a multiobjective point of view. We have exploited the properties of Pareto solutions of multiobjective optimization to develop novel supervised classification tools that enhance flexibility in classification and provide a range of robust classifiers based on non-dominance properties of Pareto outcomes of associated bi-objective programs. The classification tree is constructed to minimize the complexity of the tree (assuring interpretability) and also the misclassification risk (assuring predictive power) and support vector machines are applied at the root node aiming at helping the posterior classification at the branching nodes of the tree to globally improve the misclassification error of the different classes.

Our claim is that obtaining non-dominated classifiers with respect to the OCT and SVM principles one can get a range of different classifiers catching different insights of the sample data that will inherit the good properties of the two base classifiers.

This bunch of trees will be used to identify the "best" one to reduce misclassification or to be combined in a Pareto forest based ensemble classifier.

Ensemble methods in supervised classification boost performance by combining multiple models—called base learners—to make a stronger, more reliable overall model. Choosing be-

tween different methods, as bagging, boosting, stacking, or voting, depends on the base learners' bias, the available computational resources, and the desired model performance. Each technique brings its own trade-offs in accuracy, complexity, and interpretability. In our approach, we have chosen hard voting as the method to combine our Pareto-optimal trees into forests and making final predictions by majority vote (classification). Our goal is to reduce variance and to avoid overfitting.

The rest of the paper is organized as follows. The next subsection is devoted to highlight our contribution. In Section 2 a bi-objective mixed integer quadratic problem is presented, that integrates in a single mathematical program, the classification capabilities of OCT and SVM. Adopting a multi-objective optimization approach allows the generation of multiple "optimal" classification trees, referred to as Pareto-optimal trees, that in turn allow the definition of a more informative forest, presented in Section 3. Section 4 presents our numerical results. First, in Section 4.1, we report the performance of the Pareto-optimal trees on eight real-world binary classification datasets. Then, in Section 4.2, we evaluate the performance of the resulting Pareto forests, highlighting their strong performance in comparison to Random Forest. Finally, Section 5 concludes the paper.

1.1 Contribution of the paper

The contributions of the paper can be summarized as follows.

- Combining the rationale of two different classifiers, namely OCT and SVM, to obtain non-dominated classifiers with respect to them.
- Boosting the classification power of OCT by enforcing a maximum margin separation of data at the root node.
- Obtaining a meaningful representation of the Pareto front of OCT-SVM classifiers to catch insights reducing misclassification error. Thus proposing a new classification tree called *Pareto-optimal tree*.
- Proposing a new bagging method, namely the Pareto forest. This method builds a forest based on our Pareto-optimal trees that profits from their improved performance.
- Showing improved performance of our methods over existing ones on a number of standard datasets from the literature.
- Gaining insights on how shallow Pareto forests of limited depth can capture the data structure without resorting to randomization and large number of trees and variable depth.

2 The bi-objective optimal tree problem

It is the purpose of this work to build *Pareto-optimal* decision trees, being a trade-off among different classification techniques. A major advantage of decision trees lies in their interpretability, which is particularly valuable in fields like healthcare, where transparency is often favored over potentially more accurate but relatively uninterpretable models. Given a training dataset made of n observations $(\mathbf{x}_i, y_i), i = 1, \dots, n$, each with p features $\mathbf{x}_i \in \mathbb{R}^p$ and a label $y_i \in \{1, \dots, K\}$,

decision trees recursively partition the feature space and assign a label to each resulting partition. The tree is made of nodes, divided into branch and leaf nodes. In branch nodes, a split with certain parameters $\mathbf{a}_t \in \mathbb{R}^p$ and b_t, is applied. If $\mathbf{a}_t^T \mathbf{x}_i < b_t$, then sample i will follow the left branch from the node, otherwise it will follow the right one. In leaf nodes the samples are collected: all the samples that end up in the same leaf are classified with the same class label. Once the decision tree is built, namely once parameters $\mathbf{a}_t \in \mathbb{R}^p$ and $b_t \in \mathbb{R}$ are determined for each branch node and class labels are assigned to leaf nodes, the tree can be used to classify new samples according to the splits and labels: the new sample will follow a unique path within the tree based on the splitting rules, ending up in a leaf node that will predict its class label.

In this work, we aim at combining multivariate optimal classification trees and support vector machines, defining a bi-objective mixed-integer model for binary classification. Multi-objective optimization is an area of decision making concerned with mathematical programming problems that involve more than one conflicting objective function. Unlike single-objective optimization, where the goal is to detect a single optimal solution, multi-objective optimization typically looks for the set of non-dominated points, also called Pareto front. Each non-dominated point represents a different compromise, where improving one objective would lead to the deterioration of at least one other.

Starting from the mixed-integer linear model proposed in [2] - also called Optimal Classification Tree (OCT) problem - and the classical ℓ_2-regularized, ℓ_1-loss linear SVM problem [11], we propose a bi-objective mixed-integer convex quadratic model to train classification trees. Our model simultaneously minimizes the objective functions of the OCT and SVM problem and includes all the constraints of each single-objective model plus some linking constraints.

At a high level, the bi-objective model looks as follows:

$$\begin{aligned} \min \quad & (f_{OCT}(L,s), f_{SVM}(\omega,\xi)) \\ \text{s. t.} \quad & (L,s,A,b) \in X_{OCT} \\ & (\omega,\omega_0,\xi) \in X_{SVM} \\ & b_0 = \omega_0 \\ & \mathbf{a}_{0j} = \omega_j \qquad\qquad j = 1,\dots,p, \end{aligned} \tag{Pareto-OCT}$$

where $f_{OCT}(L,s)$ and $f_{SVM}(\omega,\xi)$ are the objective functions of the OCT and SVM problem and the sets X_{OCT} and X_{SVM} are the feasible set of the two models, respectively. Each non-dominated point of (Pareto-OCT) defines a *Pareto-optimal classification tree*. Let T be the set of nodes of a Pareto-optimal tree and let T_B be the subset of branch nodes. Variables $L_t,\ t \in T_L$ represent the optimal misclassification loss in each leaf node, while variable $s_{jt},\ j = 1,\dots p;\ t \in T_B$ represents the number of features used at each split. Matrix A has $|T_B|$ rows, each of them being variable $\mathbf{a}_t \in [-1,1]^p$ defining the splits at the branch nodes together with $b_t \in [0,1]$. Variables $\omega \in \mathbb{R}^p$ and $\omega_0 \in \mathbb{R}$ define the SVM hyperplane, while variables $\xi_i \in \mathbb{R},\ i = 1,\dots,n$ quantifies the degree of misclassification or margin violation from the SVM hyperplane.

The root node in (Pareto-OCT) is indicated by $t = 0 \in T_B$. The split applied at the root node of each Pareto-optimal classification is defined by $\mathbf{a}_{0j}^T x \le b_0$ with $b_0 = \omega_0$ and $\mathbf{a}_{0j} = \omega_j\ j = 1,\dots,p$. This implies that each non-dominated point of (Pareto-OCT) is a tree having the split at the root node being a compromise between the split applied by support vector machines and the one from optimal classification trees.

The complete formulation of our model is as follows, while a presentation of the OCT and SVM models is reported in the Appendix.

$$
\begin{aligned}
\min \quad & (f_{OCT}(L,s), f_{SVM}(\omega,\xi)) \\
\text{s. t.} \quad & L_t \geq N_t - N_{kt} - n(1-c_{kt}), && k=1,\dots,K, && \forall t \in T_L, && && \text{(1a)}\\
& L_t \leq N_t - N_{kt} + nc_{kt}, && k=1,\dots,K, && \forall t \in T_L, && && \text{(1b)}\\
& L_t \geq 0, && \forall t \in T_L, && && && \text{(1c)}\\
& N_{kt} = \sum_{i:\, y_i=k} z_{it}, && k=1,\dots,K, && \forall t \in T_L, && && \text{(1d)}\\
& N_t = \sum_{i=1}^{n} z_{it}, && \forall t \in T_L, && && && \text{(1e)}\\
& \sum_{k=1}^{K} c_{kt} = l_t, && k=1,\dots,K, && && && \text{(1f)}\\
& \mathbf{a}_m^T \mathbf{x}_i + \mu \leq b_m + (2+\mu)(1-z_{it}) && i=1,\dots,n, && \forall t \in T_B, && m \in \mathcal{L}(t) && \text{(2)}\\
& \mathbf{a}_m^T \mathbf{x}_i \geq b_m - 2(1-z_{it}) && i=1,\dots,n, && \forall t \in T_B, && m \in \mathcal{R}(t) && \text{(3)}\\
& \sum_{\forall t \in T_L} z_{it} = 1 && i=1,\dots,n && && && && \text{(4)}\\
& z_{it} \leq l_t && \forall t \in T_L && && && && \text{(5)}\\
& \sum_{i=1}^{n} z_{it} \geq N_{min} l_t && \forall t \in T_L, && && && && \text{(6)}\\
& \sum_{j=1}^{p} \hat{a}_{jt} \leq d_t, && \forall t \in T_B && && && && \text{(7a)}\\
& \hat{a}_{jt} \geq a_{jt}, && \forall t \in T_B, && j=1,\dots,p && && \text{(7b)}\\
& \hat{a}_{jt} \geq -a_{jt}, && \forall t \in T_B, && j=1,\dots,p && && \text{(7c)}\\
& -s_{jt} \leq a_{jt} \leq s_{jt}, && \forall t \in T_B && j=1,\dots,p && && \\
& s_{jt} \leq d_t, && \forall t \in T_B && j=1,\dots,p && && \\
& \sum_{j=1}^{p} s_{jt} \geq d_t, && \forall t \in T_B, && && && \\
& -d_t \leq b_t \leq d_t, && \forall t \in T_B, && && && \\
& d_t \leq d_{p(t)}, && \forall t \in T_B \backslash \{1\}, && && && \\
& z_{it}, l_t, c_{kt} \in \{0,1\}, && i=1,\dots,n, && k=1,\dots,K, && t \in T_L, && \\
& d_t, s_{jt} \in \{0,1\}, && j=1,\dots,p, && t \in T_B. && && \\
& y_i(\omega^T \mathbf{x}_i + \omega_0) \geq 1 - \xi_i && i=1,\dots,n && && && && \text{(SVM-1a)}\\
& \omega \in \mathbb{R}^p,\ \omega_0 \in \mathbb{R}, && && && && && \text{(SVM-1b)}\\
& \xi_i \in \mathbb{R}^+, && i=1\dots,n. && && && && \text{(SVM-1c)}\\
& b_0 = \omega_0 && && && && && \text{(split-root-1)}
\end{aligned}
$$

$$\mathbf{a}_{0j} = \omega_j \qquad j = 1, \dots, p \tag{split-root-2}$$

As already mentioned, our bi-objective mixed integer convex quadratic model needs to include all the constraints of both the OCT and the SVM models. The constraints from (1a) to (7c) are those from the OCT model (2), while the constraints (SVM-1a),(SVM-1b) and (SVM-1c) are those from the SVM model (3). The last two constraints, (split-root-1) and (split-root-2), are included to impose that the split applied at the root node is a compromise between the split applied by support vector machines and the one from optimal classification trees. Note that by these two constraints, the SVM split is defined having $\omega_0 \in [0, 1]$ and $\omega_j \in [-1, 1],\ j = 1, \dots, p$, while in the original model (3) the split is defined with $\omega_0 \in \mathbb{R}$ and $\omega \in \mathbb{R}^p$.

To compute the non-dominated points of a multi-objective optimization problem, various methods have been proposed in the literature [18]. In the specific case of multi-objective mixed integer quadratic problems exact approaches have been recently proposed (see e.g. [19, 13, 12]). However, our aim is not that of approximating the complete Pareto from of our model and these methods turn out to be too expensive in our context. Therefore, we opt to detect a finite number of Pareto-optimal decision trees employing the *weighted sum method*, a classical approach which belongs to the class of scalarization techniques. As the name suggests, in the weighted sum method, the objective functions are combined using non-negative weights, and the resulting scalar function is minimized over the feasible set. It can be shown that, if the weights are strictly positive, any optimal solution of the scalarized problem corresponds to a nondominated point of the original multi-objective problem (Theorem 3.6 in [18]).

Hence, in order to get a *Pareto-optimal tree*, we address a number of instances of the following single-objective model

$$\begin{aligned} \min \quad & \ell_1 f_{OCT}(L, s) + \ell_2 f_{SVM}(\omega, \xi) \\ \text{s. t.} \quad & (L, s, A, b) \in X_{OCT} \\ & (\omega, \omega_0, \xi) \in X_{SVM} \\ & b_0 = \omega_0 \\ & \mathbf{a}_{0j} = \omega_j \qquad\qquad j = 1, \dots, p, \end{aligned} \tag{Pareto-OCT$_\ell$}$$

being $\ell_1, \ell_2 > 0$. As further trees to consider, we also compute the trees obtained from Pareto-OCT$_\ell$, setting ℓ_i to zero, $i \in \{1, 2\}$. If we set $\ell_1 = 0$ (keeping $\ell_2 > 0$), we obtain a tree where at the root node, the split is taking into account only the influence of the SVM objective function, while if we set $\ell_2 = 0$ (keeping $\ell_1 > 0$) only the influence of the OCT objective function is taken into account. When addressing a bi-objective problem, choosing the weights in such a way, is equivalent to compute its ideal point. Furthermore, it can be shown (Theorem 3.4 in [18]) that these trees are *weakly* nondominated points for (Pareto-OCT), meaning that there is no other point that is strictly better in both objectives.

3 Building ensembles from Pareto-Optimal Classification Trees: *Pareto Forests*

Introduced in [8], the random forest algorithm has proven to be a highly effective and widely applicable method for both classification and regression tasks. By building an ensemble of randomized decision trees and combining their predictions, the technique delivers strong performance,

particularly in scenarios where the number of variables exceeds the number of observations. In addition to its robustness, the algorithm is scalable to large datasets, adaptable to specialized learning problems, and provides useful insights through variable importance measures.

In order to aggregate predictions from multiple decision trees a voting technique is used. Each tree in the ensemble independently predicts a class label for a given input, and if the *hard voting technique* is applied, the final classification is determined by the class that receives the highest number of votes across all trees. In case of a tie, where the two classes (or more, in case of general classification problems) receive the same number of votes, the algorithm typically resolves it by assigning the class label that appears most frequently in the training dataset. It has been shown that random forest algorithm leverages the collective decision-making of the trees, reducing the impact of individual errors and improving overall predictive accuracy. The diversity among trees, introduced through random feature selection, ensures that the ensemble benefits from varied perspectives, making majority voting a robust and effective strategy for classification tasks.

It is our purpose to build ensemble of trees using a number of Pareto-optimal trees. Given a finite set of nondominated points of the bi-objective model (Pareto-OCT), all possible combinations of $k \geq 2$ trees are considered, generating what we call *Pareto forests*. As for the random forest, the classical voting technique is adopted to aggregate the predictions of the Pareto-optimal trees considered.

4 Numerical Experiments

In this section, we report the results of the numerical experiments conducted in order to analyze the performances of the classification trees obtained addressing our bi-objective model. In particular, Section 4.1 focuses on analyzing the accuracy of the classifiers induced from a number of Pareto-trees, while Section 4.2 shows the results obtained by building ensembles of Pareto Trees, namely Pareto Forests, as described in Section 3.

Eight real-life datasets for binary classification taken from the UCI Repository [28] have been considered and we report their details (name, number of observations n and number of features p) in Table 1. For the datasets including categorical data, we treated ordinal attributes as numerical ones, while we applied the standard one-hot encoding for nominal features. Furthermore, we normalized the feature values of each dataset to the 0-1 interval. In our experiments, each dataset has been divided in training (80%) and test (20%) sets.

Dataset	Number of datapoints	Number of features
Breast Cancer (Diagnostic)	569	30
Breast Cancer (Original)	683	9
Heart Disease	270	13
Ionosphere	351	33
Parkinsons	195	22
Connectionist Bench (Sonar, Mines vs. Rocks)	208	60
SPECTF Heart	267	44
Tic-Tac-Toe Endgame	958	27

Table 1: Details on the UCI Repository [28] datasets used in our experiments.

Given a dataset and a classification rule, we define the out of sample accuracy A as

$$A = \frac{\text{number of well classified observations}}{\text{number of test observations}} \cdot 100.$$

We aim at showing that the computed Pareto-optimal trees or, in other words, combining OCT and SVM through our bi-objective model, yields classifiers that are more accurate with respect to optimal classification trees. Moreover, we want to show that building ensemble of Pareto-optimal trees may improve the performance of single Pareto-optimal trees and that the obtained Pareto Forests favorably compare - in terms of accuracy - to Random Forests, still gaining in interpretability.

All the experiments were run on a machine mounting an Intel(R) Xeon(R) Gold 5218 CPU running at 2.30GHz.

4.1 Pareto-optimal Trees' performances

In order to compute different Pareto-optimal trees, we considered eleven different values for the vector ℓ in the weighted-sum model (Pareto-OCT$_\ell$), namely we set $\ell_1 \in \{0, 0.1, 0.2, \dots, 0.9, 1\}$ and $\ell_2 = 1 - \ell_1$. The depth of the trees is fixed to $D = 2$ and the minimum number of observations per leaf node, N_{min}, is set to 5% of the number of datapoints.

For each ℓ, model (Pareto-OCT$_\ell$) requires the tuning of two different hyperparameters, specifically α for $f_{OCT}(L, s)$ and r for $f_{SVM}(\omega, \xi)$. The tuning process is carried out by means of a 4-fold cross-validation, considering $\alpha \in \{10^i : i = -5, \dots, 5\}$ and $r \in \{10^i : i = -5, \dots, 5\}$.

We compare the computed Pareto-optimal trees with other tree-like classifiers: CART, OCT and OCT-H. We use CART's implementation provided by the `scikit-learn` [29] Python library while OCT's and OCT-H's models have been coded in Python and solved using Gurobi [23]. The hyperparameter tuning for these classifiers is performed the same way as for Pareto-optimal trees. Furthermore, in order to achieve a fair comparison, a depth $D = 2$ has been considered also for the other classifiers. Where it applies (i.e. for all classifiers but CART) a time limit of 60 seconds has been set for the cross-validation and a time limit of 600 seconds has been imposed to solve the related optimization models.

In the following, for each dataset, we report in separate tables the accuracy of the Pareto-trees obtained, together with the time needed to solve (Pareto-OCT$_\ell$) and - in case the time limit is reached - the gap from the optimal solution. In particular, for each couple of weights (ℓ_1, ℓ_2), we report the accuracy obtained, the Time (in seconds), the GAP and the hyperparameters α and r. A dash (-) indicates that the time limit of 600 seconds was reached while solving (Pareto-OCT$_\ell$). The highest level of accuracy obtained for the specific dataset is highlighted in bold font. Note that a Pareto-optimal tree obtained with $(\ell_1, \ell_2) = (1.0, 0.0)$ is equivalent to the original OCT-H, i.e. the tree obtained solving (2), while the Pareto-optimal tree with $(\ell_1, \ell_2) = (0.0, 1.0)$ uses the SVM split at the root node, having $\omega \in [-1, 1]^p$ and $\omega_0 \in [0, 1]$.

In Table 2, we report the Pareto-optimal trees computed for the datasets *Breast Cancer (Diagnostic)* and *Breast Cancer (Original)*. The best accuracy attained for *Breast Cancer (Diagnostic)* is 99.12 and is obtained with the Pareto-optimal trees having $(\ell_1, \ell_2) = (1.0, 0.0)$, $(0.9, 0.1)$, $(0.5, 0.5)$. For *Breast Cancer (Original)* the best accuracy is 98.54 and is obtained for $(\ell_1, \ell_2) = (0.2, 0.8)$, $(0.1, 0.9)$. Note that in the case of *Breast Cancer (Original)* the best accuracy is obtained solving problem (Pareto-OCT$_\ell$) to optimality.

	Breast Cancer (Diagnostic)					Breast Cancer (Original)				
(ℓ_1, ℓ_2)	α	r	Accuracy	Gap	Time(s)	α	r	Accuracy	Gap	Time(s)
(1.0,0.0)	10^{-2}	10^{-5}	**99.12**	81.5	-	10^{-5}	10^{-5}	97.81	99.9	-
(0.9,0.1)	10^{-4}	10^{-2}	**99.12**	6.4	-	10^{-3}	10^{-2}	97.81	37.8	-
(0.8,0.2)	10^{-2}	10^{-2}	91.23	32.5	-	10^{-3}	10^{-2}	97.81	5.90	-
(0.7,0.3)	10^{-2}	10^{-5}	96.49	85.9	-	10^{-4}	10^{0}	97.81	0.00	136.3
(0.6,0.4)	10^{-2}	10^{-5}	96.49	84.9	-	10^{-5}	10^{-5}	97.81	95.4	-
(0.5,0.5)	10^{-2}	10^{-5}	**99.12**	90.0	-	10^{-4}	10^{-3}	97.8	32.49	-
(0.4,0.6)	10^{-3}	10^{-3}	98.25	27.8	-	10^{-5}	10^{0}	97.81	0.00	44.9
(0.3,0.7)	10^{-5}	10^{-3}	70.18	50.9	-	10^{-5}	10^{-4}	97.81	26.02	-
(0.2,0.8)	10^{-3}	10^{-3}	95.61	31.3	-	10^{-5}	10^{0}	**98.54**	0.0	125.2
(0.1,0.9)	10^{-3}	10^{-5}	96.49	75.3	-	10^{-5}	10^{-1}	**98.54**	0.0	155.1
(0.0,1.0)	10^{-5}	10^{-1}	64.04	0.0	68.4	10^{-5}	10^{4}	81.75	0.0	5.3

Table 2: Pareto-Optimal trees computed for the datasets Breast Cancer (Diagnostic) and Breast Cancer (Original)

In Table 3, we report the Pareto-optimal trees computed for the datasets *Heart Disease* and *Ionosphere*. The best accuracy attained for *Heart Disease* is 83.33 and it is obtained with the Pareto-optimal trees having $(\ell_1, \ell_2) = (0.7, 0.3)$ and $(\ell_1, \ell_2) = (0.1, 0.9)$. In both cases, the global optimum of model (Pareto-OCT$_\ell$) is reached within the time limit. For *Ionosphere* the best accuracy is 91.55, also obtained by solving to optimality model (Pareto-OCT$_\ell$) within the time limit, with weights $(\ell_1, \ell_2) = (0.9, 0.1)$ and $(0.7, 0.3)$.

In Table 4, we report the Pareto-optimal trees computed for the datasets *Parkinsons* and *Connectionist Bench (Sonar, Mines vs. Rocks)*. For *Parkinsons* the Pareto-optimal tree obtained with weights $(\ell_1, \ell_2) = (0.8, 0.2)$ could reach an accuracy of 100. For *Connectionist Bench (Sonar, Mines vs. Rocks)* the best accuracy reached by the Pareto-optimal trees computed is 71.43, obtained considering 6 different weights (ℓ_1, ℓ_2).

In Table 5, we report the Pareto-optimal trees computed for the datasets *SPECTF Heart* and *Tic-Tac-Toe Endgame*. Model (Pareto-OCT$_\ell$) for the *SPECTF Heart* dataset could be solved to global optimality for each setting of the weights (ℓ_1, ℓ_2) and all Pareto-optimal trees computed, could reach an accuracy of 85.19. For the *Tic-Tac-Toe Endgame* dataset, the best accuracy reached is 98.44, obtained for $(\ell_1, \ell_2) = (1.0, 0.0)$ and $(\ell_1, \ell_2) = (0.9, 0.1)$.

In Table 6, the best accuracy obtained out of the eleven Pareto-optimal trees computed is compared with the accuracy obtained with CART, OCT and OCT-H, for each dataset. We denote by PT any of the Pareto-Optimal trees able to reach the best accuracy and we highlight in bold font the highest level of accuracy reached. We can notice that PT is always giving a tree able to achieve the best accuracy among the approaches compared except for the *Connectionist Bench (Sonar, Mines vs. Rocks)* dataset.

4.2 Pareto Forests' performances

We build Pareto Forests considering all possible subsets of $2, 3, 4$ and 5 Pareto-optimal trees, computed as detailed in Section 4.2. In building our forests, we exclude the trees obtained from $\ell = (1, 0)$ and $\ell = (0, 1)$, representing the solutions of OCT-H's model and the Pareto-optimal

	Heart Disease					Ionosphere				
(ℓ_1, ℓ_2)	α	r	Accuracy	Gap	Time(s)	α	r	Accuracy	Gap	Time(s)
(1.0,0.0)	10^{-2}	10^{-5}	81.48	93.7	-	10^{-5}	10^{-5}	84.51	90.0	-
(0.9,0.1)	10^{-4}	10^{-4}	75.93	97.8	-	10^{-5}	10^{-5}	**91.55**	64.2	-
(0.8,0.2)	10^{-3}	10^{-4}	74.07	95.5	-	10^{-2}	10^{-4}	85.92	92.5	-
(0.7,0.3)	10^{-2}	10^{0}	**83.33**	0.4	-	10^{-3}	10^{-5}	**91.55**	90.8	-
(0.6,0.4)	10^{-2}	10^{-5}	79.63	95.9	-	10^{-2}	10^{-5}	90.14	92.4	-
(0.5,0.5)	10^{-4}	10^{-4}	81.48	88.0	-	10^{-2}	10^{-1}	88.73	1.2	-
(0.4,0.6)	10^{-4}	10^{-5}	81.48	97.7	-	10^{-2}	10^{0}	87.32	0.1	-
(0.3,0.7)	10^{-4}	10^{-4}	66.67	90.2	-	10^{-1}	10^{-1}	85.92	1.2	-
(0.2,0.8)	10^{-3}	10^{-3}	81.48	33.9	-	10^{-4}	10^{-5}	90.14	52.6	-
(0.1,0.9)	10^{-2}	10^{0}	**83.33**	0.0	-	10^{-2}	10^{-5}	88.73	89.3	-
(0.0,1.0)	10^{-5}	10^{1}	79.63	0.0	0.4	10^{-5}	10^{-5}	59.15	0.0	1.6

Table 3: Pareto-Optimal trees computed for the datasets Heart Disease and Ionosphere

	Parkinsons					Conn. Bench (Sonar, Mines vs. Rocks)				
(ℓ_1, ℓ_2)	α	r	Accuracy	Gap	Time(s)	α	r	Accuracy	Gap	Time(s)
(1.0,0.0)	10^{-3}	10^{-5}	92.31	95.4	-	10^{-2}	10^{-5}	64.29	91.9	-
(0.9,0.1)	10^{-5}	10^{-3}	87.18	94.3	-	10^{-5}	10^{-2}	**71.43**	0.1	-
(0.8,0.2)	10^{-3}	10^{-3}	**100.00**	64.2	-	10^{-5}	10^{-5}	**71.43**	64.9	-
(0.7,0.3)	10^{-2}	10^{-5}	92.31	94.4	-	10^{-5}	10^{0}	**71.43**	0.0	83.7
(0.6,0.4)	10^{-4}	10^{-1}	92.31	3.4	-	10^{-2}	10^{-3}	**71.43**	74.9	-
(0.5,0.5)	10^{-4}	10^{0}	92.31	0.3	-	10^{-4}	10^{-5}	**71.43**	64.5	-
(0.4,0.6)	10^{-5}	10^{-5}	89.74	98.9	-	10^{-3}	10^{-4}	69.05	57.6	-
(0.3,0.7)	10^{-2}	10^{-3}	89.74	61.8	-	10^{-4}	10^{1}	61.90	0.0	64.7
(0.2,0.8)	10^{-4}	10^{1}	92.31	0.0	25.2	10^{-5}	10^{-4}	**71.43**	0.5	-
(0.1,0.9)	10^{-2}	10^{-2}	94.87	4.4	-	10^{-2}	10^{-5}	64.29	90.5	-
(0.0,1.0)	10^{-5}	10^{-5}	71.79	0.0	0.1	10^{-5}	10^{-5}	50.00	0.0	4.85

Table 4: Pareto-Optimal trees computed for the datasets Parkinsons and Connectionist Bench (Sonar, Mines vs. Rocks)

	SPECTF Heart					Tic-Tac-Toe Endgame				
(ℓ_1, ℓ_2)	α	r	Accuracy	Gap	Time(s)	α	r	Accuracy	Gap	Time(s)
(1.0,0.0)	10^{-5}	10^{-5}	**85.19**	0.0	1.3	10^{-5}	10^{-5}	**98.44**	99.9	-
(0.9,0.1)	10^{-5}	10^{-5}	**85.19**	0.0	2.8	10^{-4}	10^{0}	**98.44**	0.1	-
(0.8,0.2)	10^{-5}	10^{-5}	**85.19**	0.0	2.9	10^{-5}	10^{-3}	97.92	24.0	-
(0.7,0.3)	10^{-5}	10^{-5}	**85.19**	0.0	4.4	10^{-5}	10^{0}	96.88	0.0	-
(0.6,0.4)	10^{-5}	10^{-5}	**85.19**	0.0	3.4	10^{-4}	10^{-4}	96.35	52.6	-
(0.5,0.5)	10^{-5}	10^{-5}	**85.19**	0.0	4.2	10^{-3}	10^{-5}	96.88	90.2	-
(0.4,0.6)	10^{-5}	10^{-5}	**85.19**	0.0	2.3	10^{-3}	10^{0}	65.10	0.1	-
(0.3,0.7)	10^{-5}	10^{-5}	**85.19**	0.0	2.6	10^{-4}	10^{0}	94.27	0.0	175.8
(0.2,0.8)	10^{-5}	10^{-5}	**85.19**	0.0	3.2	10^{-3}	10^{0}	97.92	0.0	241.7
(0.1,0.9)	10^{-5}	10^{-5}	**85.19**	0.0	3.3	10^{-5}	10^{0}	91.15	0.0	383.1
(0.0,1.0)	10^{-5}	10^{-5}	**85.19**	0.0	0.1	10^{-5}	10^{-5}	65.10	0.0	0.7

Table 5: Pareto-Optimal trees computed for the datasets SPECTF Heart and Tic-Tac-Toe Endgame

Dataset	Accuracy			
	CART	OCT	OCTH	PT
Breast Cancer (Diagnostic)	94.74	84.21	**99.12**	**99.12**
Breast Cancer (Original)	94.89	87.04	97.81	**98.54**
Heart Disease	**83.33**	64.81	79.63	**83.33**
Ionosphere	84.51	71.83	88.73	**91.55**
Parkinsons	89.74	87.18	91.67	**100.00**
Connectionist Bench (Sonar, Mines vs. Rocks)	**76.19**	66.67	71.43	71.43
SPECTF Heart	**85.19**	**85.19**	**85.19**	**85.19**
Tic-Tac-Toe Endgame	68.75	65.1	**98.44**	**98.44**

Table 6: Comparison on accuracy among CART, OCT, OCT-H and best Pareto-Optimal tree

tree adopting exclusively the SVM splits at the root node, respectively.

As a first comparison, we consider Random Forests of $2, 3, 4$ and 5 trees of depth $D = 2$. We report in Table (7) the results obtained on each dataset, considering the Pareto Forests attaining the highest level of accuracy. As before, we highlight in bold font the highest level of accuracy reached. In the last column of the table, we report the difference between the accuracy of the Pareto Forest and the highest accuracy achieved by any individual Pareto-optimal tree. A negative value therefore indicates that the ensemble performs worse than the best single tree. We observe that for the datasets *Breast Cancer (Diagnostic)*, *Breast Cancer (Origin)*, and *Parkinsons*, using an ensemble of Pareto-optimal trees can result in a lower accuracy than the best individual classifier. However, in all three cases, the accuracy of the ensemble remained above 97. For all the other datasets, there is at least one ensemble of Pareto-optimal trees that outperforms the best individual tree. Moreover, except for the *Connectionist Bench (Sonar, Mines vs. Rocks)* dataset, the Pareto Forests consistently achieve higher accuracy than the Random Forests.

As a second experiment, we compare our Pareto Forests, with Random Forests made of 100 trees, letting internal algorithm to decide the depth of the trees. The results of this experiment on each dataset are reported in Table 8, where we consider the best forest among the Pareto Forests obtained before. The best Pareto Forest consistently achieves the highest level of accuracy. It is worth noting that the Pareto-optimal trees used in our experiments have a fixed depth of $D = 2$. These results suggest that shallow forests composed of Pareto-optimal trees - even with limited depth - can capture the underlying data structure more effectively than Random Forests composed of a much larger number of trees with variable depths.

5 Conclusions

In this work, we introduced the Pareto-optimal trees, a new class of interpretable binary classification models that combine the strengths of Optimal Classification Trees (OCT) and Support Vector Machines (SVM). By formulating a bi-objective mixed integer quadratic optimization problem, we propose a way to compute classifiers being trade-offs of the two techniques. Indeed, each Pareto-optimal tree represents a nondominated solution of such bi-objective problem, where the split at the root node is a compromise between the split applied by SVM and the one from OCT. Our experiments across a range of benchmark datasets demonstrate that Pareto-optimal trees achieve performance comparable to or better than standard methods such as CART and OCT, while maintaining a shallow and interpretable structure. This shows the power of using Pareto solutions of our bi-objective model, able to include the ability of two classifiers. Furthermore, we proposed the Pareto Forest, an ensemble method that aggregates subsets of Pareto-optimal trees. Results show that Pareto Forests often outperform Random Forests - even those built with many more trees and variable depth - underscoring the benefit of optimizing for model quality rather than relying on randomization. We want to underline that our approach can be extended to deal with general classification problems, combining multiclass SVM with multiclass classification trees. Furthermore, the proposed multi-objective approach could be used combining OCT with any other classification model, as long as long as a suitable mathematical programming formulation models such technique.

Dataset	# of trees	Random Forest Accuracy	Pareto Forest Accuracy	Pareto Forest (Acc PF - Acc PT)
Breast Cancer (Diagnostic)	2	85.09	**99.12**	0
	3	94.74	**99.12**	0
	4	94.74	**99.12**	0
	5	95.61	98.25	-0.87
Breast Cancer (Original)	2	95.62	**98.54**	0
	3	97.08	97.81	-0.73
	4	94.89	97.81	-0.73
	5	94.89	97.81	-0.73
Heart Disease	2	77.78	83.33	0
	3	70.37	**87.04**	3.71
	4	74.07	**87.04**	3.71
	5	83.33	**87.04**	3.71
Ionosphere	2	87.32	91.55	0
	3	83.1	**95.77**	4.22
	4	84.51	**95.77**	4.22
	5	88.73	**95.77**	4.22
Parkinsons	2	87.18	**100**	0
	3	82.05	**100**	0
	4	89.74	**100**	0
	5	87.18	97.44	-2.56
Conn. Bench (Sonar, Mines vs. Rocks)	2	61.9	71.43	0
	3	64.29	76.19	4.76
	4	**78.57**	76.19	4.76
	5	66.67	76.19	4.76
SPECTF Heart	2	**85.19**	**85.19**	0
	3	**85.19**	**85.19**	0
	4	**85.19**	**85.19**	0
	5	**85.19**	**85.19**	0
Tic-Tac-Toe Endgame	2	67.19	98.44	0
	3	67.71	**99.48**	1.04
	4	67.19	**99.48**	1.04
	5	67.71	**99.48**	1.04

Table 7: Comparison between Pareto Forests and Random Forests, with 2,3,4 and 5 trees and depth $D = 2$.

Dataset	Accuracy	
	Random Forest	Pareto Forest
Breast Cancer (Diagnostic)	98.25	**99.12**
Breast Cancer (Original)	**97.81**	**97.81**
Heart Disease	83.33	**87.04**
Ionosphere	91.55	**95.77**
Parkinsons	89.74	**100**
Conn. Bench (Sonar, Mines vs. Rocks)	73.81	**76.19**
SPECTF Heart	**85.19**	**85.19**
Tic-Tac-Toe Endgame	73.44	**99.48**

Table 8: Comparison between the best Pareto Forest (among those with 2,3,4 and 5 trees and depth $D = 2$) and Random Forests with 100 trees and variable depth.

Acknowledgments

The research of the third author has been partially supported by the grants PID2020-114594GB-C21 funded by MICIU/AEI /10.13039/ 501100011033, PCI2024-155024-2 - "Optimization over Nonlinear Model Spaces: Where Discrete Meets Continuous Optimization" funded by AEI and EU funds; and IMUS-Maria de Maeztu grant CEX2024-001517-M - Apoyo a Unidades de Excelencia María de Maeztu.

6 Appendix

6.1 Multivariate Optimal Classification Trees

In [2] a mixed integer linear (MILP) model to train classification trees has been introduced. We present a concise introduction to the model, without attempting to cover all details. Given a maximum depth D, an optimal decision tree with such depth will have $T = 2^{(D+1)} - 1$ nodes, indexed by $t = 1, \ldots, T$ and divided -as already mentioned- into branch nodes (T_B) and leaf nodes (T_L). For each branch node $t \in T_B$, variables $\mathbf{a}_t \in [-1, 1]^p$ and $b_t \in [0, 1]$ are introduced to represent the multivariate split, with constraints (2) and (3) enforcing this structure. Constraints (7a)-(7c) are also related to the splits at a branch node and are modeling that $\sum_{j=1}^{p} |a_{jt}| \leq d_t$, being d_t a binary variable indicating whether node t applies a split. Note that this allows the option of not splitting at a branch node, by setting $\mathbf{a}_t = 0$ and $b_t = 0$. In the model, further indicator variables are introduced: z_{it} indicates whether the sample $\mathbf{x}_i$ is in node t, while l_t indicates whether leaf t contains any sample. Constraints (4), (5) and (6) are modeling the fact that each sample should be assigned to exactly one leaf node and that each leaf node should contain at least N_{min} samples. Binary variables c_{kt} are introduced to track the prediction of each node and the optimal misclassification loss in each node L_t is modeled by the constraints (1a)-(1f). Given a baseline accuracy $\hat{L}$, the objective function of the model minimizes a weighted combination - controlled by the parameter $\alpha > 0$ - of the misclassification error and the number of features used at each split, the latter represented by the variables s_{jt}.

$$\min \quad f_{OCT}(L,s) = \frac{1}{\hat{L}} \sum_{\forall t \in T_L} L_t + \alpha \sum_{t \in T_B} \sum_{j=1}^{p} s_{jt}$$

s. t.

$$L_t \geq N_t - N_{kt} - n(1-c_{kt}), \quad k=1,\dots,K, \quad \forall t \in T_L, \tag{1a}$$

$$L_t \leq N_t - N_{kt} + nc_{kt}, \quad k=1,\dots,K, \quad \forall t \in T_L, \tag{1b}$$

$$L_t \geq 0, \quad \forall t \in T_L, \tag{1c}$$

$$N_{kt} = \sum_{i:\, y_i = k} z_{it}, \quad k=1,\dots,K, \quad \forall t \in T_L, \tag{1d}$$

$$N_t = \sum_{i=1}^{n} z_{it}, \quad \forall t \in T_L, \tag{1e}$$

$$\sum_{k=1}^{K} c_{kt} = l_t, \quad k=1,\dots,K, \tag{1f}$$

$$\mathbf{a}_m^T \mathbf{x}_i + \mu \leq b_m + (2+\mu)(1-z_{it}) \quad i=1,\dots,n, \quad \forall t \in T_B, \quad m \in \mathcal{L}(t) \tag{2}$$

$$\mathbf{a}_m^T \mathbf{x}_i \geq b_m - 2(1-z_{it}) \quad i=1,\dots,n, \quad \forall t \in T_B, \quad m \in \mathcal{R}(t) \tag{3}$$

$$\sum_{\forall t \in T_L} z_{it} = 1 \quad i=1,\dots,n \tag{4}$$

$$z_{it} \leq l_t \quad \forall t \in T_L \tag{5}$$

$$\sum_{i=1}^{n} z_{it} \geq N_{min} l_t \quad \forall t \in T_L, \tag{6}$$

$$\sum_{j=1}^{p} \hat{a}_{jt} \leq d_t, \quad \forall t \in T_B \tag{7a}$$

$$\hat{a}_{jt} \geq a_{jt}, \quad \forall t \in T_B, \quad j=1,\dots,p \tag{7b}$$

$$\hat{a}_{jt} \geq -a_{jt}, \quad \forall t \in T_B, \quad j=1,\dots,p \tag{7c}$$

$$-s_{jt} \leq a_{jt} \leq s_{jt}, \quad \forall t \in T_B \quad j=1,\dots,p$$

$$s_{jt} \leq d_t, \quad \forall t \in T_B \quad j=1,\dots,p$$

$$\sum_{j=1}^{p} s_{jt} \geq d_t, \quad \forall t \in T_B,$$

$$-d_t \leq b_t \leq d_t, \quad \forall t \in T_B,$$

$$d_t \leq d_{p(t)}, \quad \forall t \in T_B \backslash \{1\},$$

$$z_{it}, l_t, c_{kt} \in \{0,1\}, \quad i=1,\dots,n, \quad k=1,\dots,K, \quad t \in T_L,$$

$$d_t, s_{jt} \in \{0,1\}, \quad j=1,\dots,p, \quad t \in T_B.$$

6.2 Support Vector Machines

Support Vector Machines (SVMs) are a class of supervised learning algorithms widely used for classification and regression (we refer to [9, 32] as classical references). In a binary classification

problem, an SVM attempts to find the hyperplane that best divides the given dataset into two classes. The optimal hyperplane is the one that maximizes the *margin*, i.e. the distance between the hyperplane and the nearest data points from each class, known as support vectors. To be more precise, given n observations $\{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \{-1, 1\}, i = 1, \ldots, n\}$, the classical ℓ_2-regularized, ℓ_1-loss linear SVM problem, looks for $(\omega^*, \omega_0^*) \in \mathbb{R}^n \times \mathbb{R}$ solution of the following convex quadratic optimization problem:

$$\begin{aligned}
\min \quad & f_{SVM}(\omega, \xi) = \frac{1}{2}||\omega||_2^2 + r \sum_{i=1}^{n} \xi_i & & \\
\text{s. t.} \quad & y_i(\omega^T \mathbf{x}_i + \omega_0) \geq 1 - \xi_i & i = 1, \ldots, n & \quad (3a) \\
& \omega \in \mathbb{R}^p, \ \omega_0 \in \mathbb{R}, & & \quad (3b) \\
& \xi_i \in \mathbb{R}^+, & i = 1 \ldots, n. & \quad (3c)
\end{aligned}$$

In this model, margin violations are penalized linearly through the ℓ_1-loss function $\sum_{i=1}^{n} \xi_i$, where each variable ξ_i quantifies the degree of misclassification or margin violation for sample i. Specifically, from constraint (3a), we observe that if $\xi_i \in (0, 1]$, the sample lies within the margin remaining correctly classified. If $\xi_i > 1$, the sample is misclassified. The regularization parameter $r > 0$ plays a crucial role in balancing two competing objectives in the optimization: maximizing the margin (via minimizing $||\omega||_2^2$) and minimizing the classification error (via penalizing $\sum_{i=1}^{n} \xi_i$). A larger value of r emphasizes error minimization, potentially at the cost of a smaller margin, while a smaller value of r prioritizes margin maximization, possibly allowing more misclassifications.

References

[1] Kristin P Bennett and JA Blue. A support vector machine approach to decision trees. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, volume 3, pages 2396–2401. IEEE, 1998.

[2] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Mach. Learn.*, 106:1039–1082, 2017.

[3] Dimitris Bertsimas and Jack Dunn. *Machine learning under a modern optimization lens*. Dynamic Ideas LLC Waltham, Belmont, Massachussets, 2019.

[4] V. Blanco, A. Japón, and J. Puerto. Robust optimal classification trees under noisy labels. *Advances in Data Analysis and Classification*, 16:155–179, 2022.

[5] Víctor Blanco, Alberto Japón, and Justo Puerto. A mathematical programming approach to binary supervised classification with label noise. *Comput. Ind. Eng.*, 172A:108611, 2022.

[6] Victor Blanco, Justo Puerto, and Antonio M Rodriguez-Chia. On lp-support vector machines and multidimensional kernels. *J. Mach. Learn. Res.*, 21:14–1, 2020.

[7] L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and regression trees, 1984.

[8] Leo Breiman. Random forests. *Mach. learn.*, 45:5–32, 2001.

[9] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[10] Emilio Carrizosa, Cristina Molero-Río, and Dolores Romero Morales. Mathematical optimization in classification and regression trees. *TOP*, 29(1):5–33, 2021.

[11] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. learn.*, 20(3):273–297, 1995.

[12] Philip J de Castro and Margaret M Wiecek. Pareto leap: An algorithm for biobjective mixed-integer programs. *Optimization Online*, 2025.

[13] Marianna De Santis, Gabriele Eichfelder, Daniele Patria, and Leo Warnow. Using dual relaxations in multiobjective mixed-integer convex quadratic programming. *J. Global Optim.*, 92:159–186, 2025.

[14] Emir Demirović, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J Stuckey. Murtree: Optimal decision trees via dynamic programming and search. *J. of Mach. Learn. Res.*, 23(26):1–47, 2022.

[15] Emir Demirović and Peter J Stuckey. Optimal decision trees for nonlinear metrics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3733–3741, 2021.

[16] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019.

[17] Federico D'Onofrio, Giorgio Grani, Marta Monaci, and Laura Palagi. Margin optimal classification trees. *Comput. Oper. Res.*, 161:106441, 2024.

[18] Matthias Ehrgott. *Multicriteria Optimization*. Springer-Verlag Berlin Heidelberg, Auckland, New Zealand, 2005.

[19] Gabriele Eichfelder and Leo Warnow. A hybrid patch decomposition approach to compute an enclosure for multi-objective mixed-integer convex optimization problems. *Math. Method. Oper. Res.*, 2023. DOI: 10.1007/s00186-023-00828-x.

[20] Murat Firat, Guillaume Crognier, Adriana F Gabor, Cor AJ Hurkens, and Yingqian Zhang. Column generation based heuristic for learning classification trees. *Comput. Oper. Res.*, 116:104866, 2020.

[21] Manlio Gaudioso, Enrico Gorgone, Martine Labbé, and Antonio M Rodríguez-Chía. Lagrangian relaxation for svm feature selection. *Comput. Oper. Res.*, 87:137–145, 2017.

[22] Oktay Günlük, Jayant Kalagnanam, Matt Menickelly, and Katya Scheinberg. Optimal decision trees for categorical data via integer programming. *arXiv preprint arXiv:1612.03225*, 2018.

[23] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.

[24] Hao Hu, Mohamed Siala, Emmanuel Hebrard, and Marie-José Huguet. Learning optimal decision trees with maxsat and its integration in adaboost. In *IJCAI-PRICAI 2020, 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence*, 2020.

[25] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. *Advances in Neural Information Processing Systems*, 32, 2019.

[26] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Ann. Appl. Stat.*, 9(3):1350–1371, 2015.

[27] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning*, pages 6150–6160. PMLR, 2020.

[28] Kolby Nottingham Markelle Kelly, Rachel Longjohn. The UCI Machine Learning Repository, accessed 2025.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

[30] J Quinlan. Machine learning and id3. *Los Altos: Morgan Kauffman*, 1996.

[31] Ross Quinlan. C4. 5. *Programs for machine learning*, 1993.

[32] Vladimir Vapnik. *The nature of statistical learning theory*. Springer, New York, NY, 2013.

[33] Hélene Verhaeghe, Siegfried Nijssen, Gilles Pesant, Claude-Guy Quimper, and Pierre Schaus. Learning optimal decision trees using constraint programming. *Constraints*, 25(3):226–250, 2020.

[34] Sicco Verwer and Yingqian Zhang. Learning optimal classification trees using a binary linear program formulation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1625–1632, 2019.

[35] Jinqiang Yu, Alexey Ignatiev, Peter J Stuckey, and Pierre Le Bodic. Computing optimal decision sets with sat. In *International Conference on Principles and Practice of Constraint Programming*, pages 952–970. Springer, 2020.