Optimal Sparse Survival Trees

Rui Zhang Duke University
Rui Xin Duke University
Margo Seltzer University of British Columbia
Cynthia Rudin Duke University

Abstract

Interpretability is crucial for doctors, hospitals, pharmaceutical companies and biotechnology corporations to analyze and make decisions for high stakes problems that involve human health. Tree-based methods have been widely adopted for *survival analysis* due to their appealing interpretablility and their ability to capture complex relationships. However, most existing methods to produce survival trees rely on heuristic (or greedy) algorithms, which risk producing sub-optimal models. We present a dynamic-programming-with-bounds approach that finds provably-optimal sparse survival tree models, frequently in only a few seconds.

1 INTRODUCTION

Interpretability is essential for high stakes decisions (Rudin et al., 2022), particularly in healthcare. Thus, when a machine learning model estimates the answers to critical questions such as "how long is this patient expected to survive?", the reasoning process of the model must be understandable to humans. There has been little overlap between the fields of *interpretable machine learning* and *survival analysis*, despite the importance of survival analysis problems in healthcare and beyond. Survival analysis is used in reliability modeling for mechanical failure of equipment (where it is called reliability analysis), customer churn prediction, and questions in the social sciences such as predicting economic events. Early work in survival analysis either did not use covariates, such as Kaplan-Meier curves (Kaplan and Meier, 1958), or used a (non-sparse) linear combination of covariates, such as Cox proportional hazard models (Cox, 1972). Interpretable machine learning tools introduce the ability to use *sparse* and *nonlinear* combinations of variables to form accurate survival models. This makes survival analyses more powerful, while maintaining the interpretability necessary for use in practice.

In a classical setup for survival analysis, we might wish to predict the time at which an event ("death") will occur. However, estimating time-to-event is problematic, because some of the data are *censored*, meaning that the time of the event is not known; all we know is that the individual survived beyond the last observation time. Because we cannot distinguish between samples who have different death times when neither time is observed, we instead estimate the probability that a sample with variables $\mathbf{x}$ survives past time y, which is defined as the survival function $S_{\mathbf{x}}(y)$. Since $S_{\mathbf{x}}(y)$ is a probability estimate, it suggests that techniques for classification or regression might work, but the loss function is completely different, so these techniques do not apply directly. Some machine learning techniques can adapt to many loss functions to achieve *high performing* models but not *interpretable* models. Specifically, it is easy to adapt standard machine learning techniques to produce black box survival models, simply by changing the loss function to a survival loss (Che et al., 2018; Ching et al., 2018; Giunchiglia et al., 2018; Hothorn et al., 2006; Ishwaran and Kogalur, 2007; Katzman et al., 2018; Ripley and Ripley, 2001). Since the survival curves created by these methods are black box, they are unlikely to be useful in practice. Another easy way to get survival models is to use greedy methods for minimizing the loss function; greedy tree-based models for the survival function $S_{\mathbf{x}}(y)$ are popular (Ciampi et al., 1988; Davis and Anderson, 1989; Gordon and Olshen, 1985; Hothorn et al., 2006; Jin et al., 2004; Keles and Segal, 2002; LeBlanc and Crowley, 1992, 1993; Molinaro et al., 2004; Segal, 1988; Therneau et al., 1990; Zhang, 1995). These methods generally choose and fix the top split first according to heuristic splitting rules, and continue to split according to heuristics until the tree is formed, perhaps using additional heuristics to prune the tree afterwards to prevent overfitting. However, their performance is often limited as a bad split cannot be fixed once it has been made.

Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

To get interpretable models that achieve good performance requires optimization. As soon as the requirement of sparsity is added to any machine learning problem, the problem becomes computationally extremely hard, and techniques must be tailored to the specific loss functions in order to maintain performance. Specialized techniques have been developed for classification and regression for sparse decision trees (Grubinger et al., 2014; Dunn, 2018; Lin et al., 2020; Nijssen et al., 2020; Zhang et al., 2023). We have learned from these modern methods that their single – very sparse – trees are often as accurate as black box models for tabular datasets. We have also learned that each problem requires whole new algorithms. However, fully optimizing a survival tree is much harder than a classification or regression tree.

The only previous attempt to construct optimized sparse survival trees that we know of is that of Bertsimas et al. (2022) who attempted to solve the problem using Mixed Integer Programming (MIP) and local search. However, their method assumes that the ratio of the hazard functions for any two individuals is constant over time. Their search method also can get stuck at local optima. Their code is proprietary and tends to crash very often (about 90% of runs).

The method presented in this paper is first algorithm for optimal sparse survival trees with public code. We chose a dynamic-programming-with-bounds framework, where theorems based on the survival loss – in this case, the Integrated Brier Score – are used to narrow the search space. The tighter these bounds are, the better they prune the search space and reduce computation. Our bounds are tight enough that optimal sparse survival trees can be found in seconds or minutes for all public survival analysis datasets we know of. We call our algorithm **Optimal Sparse Survival Trees** (OSST). In Section 2.1, we present notation and the survival objective, and Sections 2.2 and 2.3 present our theorems that make training optimal sparse survival trees possible. Section 3 contains the experimental results. An extended related work section appears in the appendix.

2 METHODOLOGY

2.1 Notation and Objective

We denote the training dataset $(\mathbf{X}, \mathbf{c}, \mathbf{y})$ as $\{(\mathbf{x}_i, c_i, y_i)\}_{i=1}^N$, where $c_i \in \{0,1\}$ is a binary variable indicating whether the sample has an observed death ($c_i = 1$) or is censored ($c_i = 0$), and $y_i \in \mathbb{R}$ is the time of last observation for sample i. $\mathbf{x}_i \in \{0,1\}^M$ is a binary feature vector, where real-valued variables are converted to a set of binary variables using either all possible splits or a subset of splits chosen by a black box model (McTavish et al., 2022). The union of $\{\mathbf{x}_i\}$ is the set of possible splits for any decision tree.

The Integrated Brier Score (IBS) for censored observations proposed by Graf et al. (1999) is used as our performance metric. We denote the loss of tree t on the training dataset as

$$\mathcal{L}(t, \mathbf{X}, \mathbf{c}, \mathbf{y}) := \frac{1}{y_{\max}} \int_0^{y_{\max}} BS(y) dy \tag{1}$$

where $y_{\max} = \max\{y_i\}_{i=1}^N$ is the latest time point of all observed samples, and $BS(y)$ is the Brier Score of tree t at given time y, which can be interpreted as the mean square error between the data and the predicted survival function $\hat{S}_{\mathbf{x}_i}(y)$ by tree t, weighted by Inverse Probability of Censoring Weights (IPCW):

$$\begin{aligned} BS(y) \quad = \quad & \frac{1}{N} \sum_{i=1}^{N} \frac{(\hat{S}_{\mathbf{x}_i}(y) - 0)^2}{\hat{G}(y_i)} \cdot \mathbf{1}_{y_i \le y, c_i = 1} \\ & + \frac{(\hat{S}_{\mathbf{x}_i}(y) - 1)^2}{\hat{G}(y)} \cdot \mathbf{1}_{y_i > y} \end{aligned} \tag{2}$$

where $\hat{S}_{\mathbf{x}_i}(\cdot)$ is estimated by a non-parametric Kaplan–Meier estimator at the leaf node to which $\mathbf{x}_i$ gets assigned, and $\hat{G}(\cdot)$ is the Kaplan–Meier estimate of the censoring distribution $\mathbf{c}$, which is assumed to be independent of any covariates (Molinaro et al., 2004). The Kaplan–Meier estimator, known as the product limit estimator, is given by

$$\hat{S}(y) = \prod_{i: y_i \le y} \left(1 - \frac{d_i}{n_i}\right) \tag{3}$$

where d_i is the number of deaths at time y_i, and n_i is the number of samples known to have survived up to time y_i. The first term in Equation 2 applies to non-censored data and encourages the survival function to be close to 0 after the death event. The second term applies to censored and non-censored data, encouraging the survival function to be 1 up until the censoring (for censored data) or up until a death event is observed (for non-censored data).

We define the objective function $R(t, \mathbf{X}, \mathbf{y})$ of tree t, as a combination of the tree loss defined above and a complexity penalty: $\mathcal{L}(t, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda \cdot \text{complexity}(t)$, where the complexity penalty is H_t, the number of leaves in tree t:

$$R(t, \mathbf{X}, \mathbf{c}, \mathbf{y}) := \mathcal{L}(t, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_t. \tag{4}$$

In addition to the soft constraint on complexity, we also add an optional hard constraint:

$$\mathcal{L}(t, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_t, \text{ s.t. } \text{depth}(t) \le d. \tag{5}$$

The hard constraint on depth makes computation substantially easier, because each increase in depth creates an exponentially larger search space. The soft complexity constraint is always used, because it excludes unnecessary leaves. That is, the optimal tree with a depth limit of 6 might need only 8 leaves, but the hard constraint would permit trees having up to 64 leaves. Our algorithm finds optimal trees that minimize Equation 5 in seconds. It can also minimize Equation 4 without a depth constraint but takes more time to find the globally optimal solution.

2.2 Dynamic Programming

We use dynamic-programming-with-bounds (Algorithm 1) for optimization (see Hu et al., 2019; Lin et al., 2020; McTavish et al., 2022; Zhang et al., 2023); the question here is how to derive the bounds for survival analysis so that the search space is pruned effectively (Line 35-38). If the bounds determine that a partially constructed tree cannot be part of an optimal solution, then the part of the search space it extends to can be eliminated.

We start with a single leaf node to which all sample points are assigned (Line 1-5), and then try splitting this single leaf by all possible features and all possible ways of splitting those features (Line 12-17). This process produces more leaf nodes, each containing only a subset of the data; recursive splitting creates a large number of nodes/subsets. Each node/subset represents a *sub-problem* for which we want to find the corresponding optimal sub-tree. Each *sub-problem* is identified by a support set $s = \{s_1, s_2, \ldots, s_N\}$, where s_i is a Boolean value indicating whether sample i is in the support set s. With each sub-problem, the algorithm records and updates the lower bound and upper bound (current best score) of its corresponding sub-tree objective (Line 19). The sub-problem is considered *solved* when these two bounds converge (Line 9, 22). If the bounds determine that a sub-problem cannot benefit from splitting (Line 37), then we can mark it as solved without further exploration of that part of the search space. We store all sub-problems and the parent/child relationships between them in a *dependency graph*. Since a sub-problem can have more than one parent problem (i.e., a sub-problem can arise by multiple different sequences of splits), the dependency graph lets us avoid duplicate computations. A priority queue is maintained to store the copies of all unsolved subproblems, and there can be multiple copies of a specific sub-problem in the queue at the same time (i.e., the subproblem can be pushed by its child problems, Line 21; or it can be pushed by its parent problem, Line 29-30). We use the difference between the fraction of captured points and the objective lower bound as the priority value for ordering the queue. Once all subproblems in the graph are solved, the optimization is complete, and we can extract all optimal survival trees from the graph.

2.3 Bounds

Using notation similar to Lin et al. (2020), we represent a tree t as a set of H_t distinct leaves: $t = \{l_1, l_2, \ldots, l_{H_t}\}$. It can also be written as:

$$t = (t_{\text{fix}}, \delta_{\text{fix}}, t_{\text{split}}, \delta_{\text{split}}, K, H_t),$$

where $t_{\text{fix}} = \{l_1, l_2, \ldots, l_K\}$ are a set of K **fixed leaves** that are not allowed to be further split in this part of the search space, $\delta_{\text{fix}} = \{\hat{S}_{l_1}, \hat{S}_{l_2}, \ldots, \hat{S}_{l_K}\}$ are predicted survival functions for the fixed leaves, $t_{\text{split}} = \{l_{K+1}, l_{K+2}, \ldots, l_{H_t}\}$ are $H_t - K$ **splitting leaves** that can be further split in this part of the search space, and their predicted survival functions are $\delta_{\text{split}} = \{\hat{S}_{l_{K+1}}, \hat{S}_{l_{K+2}}, \ldots, \hat{S}_{H_t}\}$.

A child tree $t' = (t'_{\text{fix}}, \delta'_{\text{fix}}, t'_{\text{split}}, \delta'_{\text{split}}, K', H_{t'})$ can be created by splitting a subset of splitting leaves t_{split} in tree t. t'_{fix} is a superset of t_{fix}. We denote $\sigma(t)$ as the set of all child trees of t. *Note that proofs for all theorems are in Appendix B.*

2.3.1 Lower Bounds

A large portion of the search space is reduced through leveraging lower bounds of the survival tree objective. Specifically, as we will show, if the objective lower bound of tree t exceeds the current best objective so far, R^c, then neither tree t nor any of its children $t' \in \sigma(t)$ can be an optimal tree. This is typically called the hierarchical objective lower bound, but we cannot prove this without some work. Let us start by deriving some principles of the objective.

Theorem 2.1. *The loss of a survival tree is an additive function of the observations and leaves.*

Because of Theorem 2.1 and that fixed leaves are not allowed to be further split in this part of the search space, the loss of observations in fixed leaves provides a lower bound for tree t:

$$R(t, \mathbf{X}, \mathbf{c}, \mathbf{y}) \geq \mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_t, \tag{6}$$

where $\mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y})$ is the sum of losses for the fixed leaves:

$$\mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) = \frac{1}{y_{\max}} \frac{1}{N} \sum_{i=1}^{N} \Bigg\{ \int_0^{y_i} \frac{(\hat{S}_{\mathbf{x}_i}(y) - 1)^2}{\hat{G}(y)} dy + c_i \int_{y_i}^{y_{\max}} \frac{(\hat{S}_{\mathbf{x}_i}(y) - 0)^2}{\hat{G}(y_i)} dy \Bigg\} \cdot \mathbf{1}_{\text{cap}(t_{\text{fix}}, \mathbf{x}_i)},$$

where $\mathbf{1}_{\text{cap}(t_{\text{fix}}, \mathbf{x}_i)}$ is 1 if a leaf in t_{fix} captures $\mathbf{x}_i$ (when $\mathbf{x}_i$ falls into one of the fixed leaves of t), 0 otherwise.

Algorithm 1 dynamic-programming-with-bounds $(\mathbf{X}, \mathbf{c}, \mathbf{y}, \lambda, T, R)$

// Optional input: reference model T and initial risk score R

$Q \leftarrow \emptyset$ *// priority queue*
$G \leftarrow \emptyset$ *// dependency graph*
$s_0 \leftarrow \{1, \ldots, 1\}$ *// support set for root problem*
$p_0 \leftarrow$ find_or_create_node(G, s_0) *// root problem*
$Q.push(s_0)$ *// add to priority queue*
while $p_0.lb \neq p_0.ub$ **do**
  $s \leftarrow Q$.pop()
  $p \leftarrow G$.find(s)
  if $p.lb = p.ub$ **then**
    break *// problem already solved*
  $lb', ub' \leftarrow (\inf, \inf)$ *// initialize starting bounds*
  for each feature $j \in \{1, \ldots, M\}$ **do**
    // support sets for child problems
    $s_l, s_r \leftarrow$ split$(s, j, \mathbf{X})$
    $p_l^j \leftarrow$find_or_create_node(G, s_l)
    $p_r^j \leftarrow$find_or_create_node(G, s_r)
    // create bounds as if j were chosen
    // for splitting
    $lb' \leftarrow \min(lb', p_l^j.lb + p_r^j.lb)$
    $ub' \leftarrow \min(ub', p_l^j.ub + p_r^j.ub)$
  // signal the parents if an update occurred
  if $p.lb \neq lb'$ **or** $p.ub \neq ub'$ **then**
    $(p.lb, p.ub) \leftarrow (lb', ub')$
    for $p_\pi \in G$.parent(p) **do**
      // propagate information upwards
      Q.push$(p_\pi$.id, priority $= count(s)/N - p.lb)$
  if $p.lb \geq p.ub$ **then**
    continue *// problem solved just now*
  // loop, enqueue all children that are dependencies
  for each feature $j \in [1, M]$ **do**
    repeat line 14-16
    $lb' \leftarrow p_l^j.lb + p_r^j.lb$
    $ub' \leftarrow p_l^j.ub + p_r^j.ub$
    if $lb' < ub'$ **and** $lb' \leq p.ub$ **then**
      Q.push$(s_l$, priority $= count(s_l)/N - p_l.lb)$
      Q.push$(s_r$, priority $= count(s_r)/N - p_r.lb)$
return

subroutine: **find_or_create_node**(G, s)
// p not yet in dependency graph
if G.find(s) = NULL
  $p.id \leftarrow s$ *// identify p by s*
  // compute initial lower and upper bounds
  $p.lb \leftarrow$ get_lower_bound$(s, \mathbf{X}, \mathbf{c}, \mathbf{y})$
  $p.ub \leftarrow$ get_upper_bound$(s, \mathbf{X}, \mathbf{c}, \mathbf{y})$
  if fails_bounds(p) **then**
    $p.lb = p.ub$ *// no more splitting allowed*
  G.insert(p) *// put p in dependency graph*
return G.find(s)

Theorem 2.2. *(Hierarchical Objective Lower Bound). Any tree $t' = (t'_{\text{fix}}, \delta'_{\text{fix}}, t'_{\text{split}}, \delta'_{\text{split}}, K', H_{t'}) \in \sigma(t)$ in the child tree set of tree $t = (t_{\text{fix}}, \delta_{\text{fix}}, t_{\text{split}}, \delta_{\text{split}}, K, H_t)$ obeys:*

$$R(t', \mathbf{X}, \mathbf{c}, \mathbf{y}) \geq \mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_t.$$

Theorem 2.2 indicates that the objective lower bound of the parent tree also holds for all its child trees, which means that if the parent tree can be pruned via the lower bound then so can its child trees.

Sometimes even if the parent tree t satisfies $\mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_t \leq R_c$, we still can prune all of its child trees, which is shown in Theorem 2.3.

Theorem 2.3. *(Objective Lower Bound with One-step Lookahead). Let $t = (t_{\text{fix}}, \delta_{\text{fix}}, t_{\text{split}}, \delta_{\text{split}}, K, H_t)$ be a tree with H_t leaves. If $\mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_t + \lambda > R^c$, even if its objective lower bound obeys $\mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_t \leq R^c$, then for any child tree $t' \in \sigma(t)$, $R(t', \mathbf{X}, \mathbf{c}, \mathbf{y}) > R^c$.*

This bound shows that although the parent tree cannot be pruned via the lower bound, we should not explore any of its child trees if $\mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_t + \lambda > R^c$, because all of them are sub-optimal.

To make the lower bound tighter, we leverage the property that some points cannot be partitioned into different leaves in any survival tree, by introducing the concept of equivalent points.

Equivalent Points

We denote the loss from sample $\{\mathbf{x}_i, c_i, y_i\}$ with prediction $\hat{S}_{\mathbf{x}_i}(\cdot)$, as $\mathcal{L}(\hat{S}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i)$, where

$$\begin{aligned}\mathcal{L}(\hat{S}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i) &= \frac{1}{y_{\max}} \frac{1}{N} \int_0^{y_i} \frac{(\hat{S}_{\mathbf{x}_i}(y) - 1)^2}{\hat{G}(y)} dy \\ &+ c_i \int_{y_i}^{y_{\max}} \frac{(\hat{S}_{\mathbf{x}_i}(y) - 0)^2}{\hat{G}(y_i)} dy. \quad (7)\end{aligned}$$

In the simplest case, $\mathcal{L}(\hat{S}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i) = 0$ if the leaf node contains only $\{\mathbf{x}_i, c_i, y_i\}$. More formally, $\mathcal{L}(\hat{S}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i)$ can be zero only if two conditions are satisfied: (1) there is no sample in the same leaf that died prior to sample $\{\mathbf{x}_i, c_i, y_i\}$, i.e., $\{\mathbf{x}_j, c_j = 1, y_j < y_i\}$ (here, the first term in 7 is nonzero because the survival function will be < 1 at y_i), and (2) if $c_i = 1$, then all the points in i's leaf that were observed to survive up to time y_i must also die at y_i. To explain, if $c_i = 1$ and there are (censored or uncensored) points in i's leaf node whose observation time is later than y_i, i.e., $\{\mathbf{x}_j, c_j, y_j > y_i\}$, the second term in 7 will be nonzero (those points will force the survival curve to be nonzero even after i dies at time y_i). Also if $c_i = 1$, we

cannot have censored points with the same observation time as i, i.e., $\{\mathbf{x}_j, c_j = 0, y_j = y_i\}$, since it would mean that when i dies at y_i, another point does not, so the survival function cannot go to 0 at y_i.

If each sample point can be placed in a leaf by itself, the tree t can achieve zero loss, but this tree is clearly overfitted. Further, such a tree may not be possible if there exist two samples $\{\mathbf{x}_i, c_i, y_i\}$ and $\{\mathbf{x}_j, c_j, y_j\}$ such that $\mathbf{x}_i = \mathbf{x}_j$, but $c_i \neq c_j$ or $y_i \neq y_j$; we call such points equivalent points. There is no tree that can partition these samples into different leaves, so the loss contributed from these samples must be non-zero if the two conditions above cannot be satisfied.

Let us generalize this argument. Let u be a set of equivalent points where samples have exactly the same feature vector $\mathbf{x}$, such that $\forall j_1, j_2, ...j_{|u|} \in u$:

$$\mathbf{x}_{j_1} = \mathbf{x}_{j_2} = \cdots = \mathbf{x}_{j_{|u|}}. \tag{8}$$

Assume set u does not satisfy the two conditions, which means loss is non-zero:

$$\sum_{k=1}^{|u|} \mathcal{L}(\hat{S}_{\mathbf{x}_{j_k}}(\cdot), \mathbf{x}_{j_k}, c_{j_k}, y_{j_k}) > 0. \tag{9}$$

Let us derive the lower bound for Equation 9. Recall that samples in a leaf share the same predicted survival function.

Lemma 2.4. *(Equivalent Loss). Let u be a set of equivalent points defined as in (8). We denote ${}^*S$ as the optimal step function that minimizes IBS loss only for set u (leaf contains set u only), such that:*

$${}^*S = \operatorname*{argmin}_S \sum_{k=1}^{|u|} \mathcal{L}(S, \mathbf{x}_{j_k}, c_{j_k}, y_{j_k}).$$

*We define **Equivalent Loss** for set u as $\mathcal{E}_u = \sum_{k=1}^{|u|} \mathcal{L}({}^*S, \mathbf{x}_{j_k}, c_{j_k}, y_{j_k})$. Then, any leaf l that captures set u in a survival tree has loss $\mathcal{L}(l, \mathbf{X}, \mathbf{c}, \mathbf{y}) \geq \mathcal{E}_u$.*

Lemma 2.5. *Let l be a leaf node that captures n equivalent sets: $\{u_i\}_{i=1}^n$ and corresponding $\mathcal{E}_{u_i}$. The loss of l: $\mathcal{L}(l, \mathbf{X}, \mathbf{c}, \mathbf{y}) \geq \sum_{i=1}^n \mathcal{E}_{u_i}$.*

That is, the lower bound of a leaf is the sum of equivalent losses of the equivalent sets it captures.

Theorem 2.6. *(Equivalent Points Lower Bound). Let $t = (t_{\text{fix}}, \delta_{\text{fix}}, t_{\text{split}}, \delta_{\text{split}}, K, H_t)$ be a tree with K fixed leaves and $H_t - K$ splitting leaves. For any child tree $t' = (t'_{\text{fix}}, \delta'_{\text{fix}}, t'_{\text{split}}, \delta'_{\text{split}}, K', H_{t'}) \in \sigma(t)$:*

$$\begin{aligned} R(t', \mathbf{X}, \mathbf{c}, \mathbf{y}) \quad &\geq \quad \mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_t \\ &+ \quad \sum_{u \in U} \mathcal{E}_u \cdot \mathbf{1}_{cap(t_{\text{split}}, u)}, \end{aligned} \tag{10}$$

where U is the set of equivalent points sets in the training dataset $(\mathbf{X}, \mathbf{c}, \mathbf{y})$ and $\mathbf{1}_{cap(t_{\text{split}}, u)}$ is 1 when t_{split} captures set u, 0 otherwise. Combining with Theorem 2.3, we have a tighter bound: for any child tree t':

$$\begin{aligned} R(t', \mathbf{X}, \mathbf{c}, \mathbf{y}) \quad &\geq \quad \mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_t + \lambda \\ &+ \quad \sum_{u \in U} \mathcal{E}_u \cdot \mathbf{1}_{cap(t_{\text{split}}, u)}. \end{aligned} \tag{11}$$

Lower Bound from Reference Models

Given that the hardness of the optimization problem to solve varies, the Equivalent Points Lower Bound may not be tight enough in some cases. When it is not tight enough to sufficiently prune the search space, calculating the bound adds overhead and slows the optimization procedure. Moreover, some datasets do not have many equivalent points, resulting in a looser lower bound.

To efficiently prune the search space, we adopt the *guessing* technique of McTavish et al. (2022). Specifically, we use a reference survival model that we believe will make errors that will also be made by an optimal survival tree and use the errors made by the reference model as a lower bound in our branch-and-bound method. We denote the reference model as T and let $\hat{S}^T_{\mathbf{x}_i}(\cdot)$ be its predicted survival function for sample i. Define s_a as the subset of training samples that satisfy a boolean assertion a, such that: $s_a := \{i : a(\mathbf{x}_i) = \text{True}, i \in \{1, 2, \cdots, N\}\}$, $\mathbf{X}(s_a) := \{\mathbf{x}_i : i \in s_a\}$, $\mathbf{c}(s_a) := \{c_i : i \in s_a\}$, and $\mathbf{y}(s_a) := \{y_i : i \in s_a\}$, which corresponds to a subproblem defined in Section 2.2 (e.g., the boolean value at index i in s is True for $i \in s_a$). We define our guessed lower bound of subproblem s_a as the error made by the reference model T on these samples plus a complexity penalty λ:

$$lb_{\text{guess}}(s_a) = \mathcal{L}(\hat{\mathbf{S}}^T, \mathbf{X}(s_a), \mathbf{c}(s_a), \mathbf{y}(s_a)) + \lambda \tag{12}$$

where $\hat{\mathbf{S}}^T = \{\hat{S}^T_{\mathbf{x}_i}(\cdot) : i \in s_a\}$ and

$$\mathcal{L}(\hat{\mathbf{S}}^T, \mathbf{X}(s_a), \mathbf{c}(s_a), \mathbf{y}(s_a)) = \sum_{i \in s_a} \mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{X}(s_a), \mathbf{c}(s_a), \mathbf{y}(s_a)),$$

which is the sum of sample losses defined in Equation 7. We set the guessed lower bound lb_{guess} as the initial lower bound for each subproblem. If the initial upper bound is less than or equal to lb_{guess}, then we consider this subproblem solved without further exploration of that portion of search space. Even when the initial lower bound is less than the initial upper bound, lb_{guess} still improves runtime. Recall that during the optimization process, the lower bound of a subproblem never decreases and the upper bound of it never

increases. Since the updates of a subproblem's lower bound rely on the lower bounds of its children, tighter initial lower bounds of children help the parent's lower bound converge to its upper bound.

It can happen that we miss the true optimal solution of the subproblem s_a if $lb_{\text{guess}} > R^*(s_a)$, where $R^*(s_a)$ is the optimal objective of subproblem s_a. This could impact the optimality of the optimization problem. We next quantify how much performance we might sacrifice by using lb_{guess}. Theorem 2.7 shows that the distance to the true optimal solution depends on the performance of the reference model, and, under certain circumstances, we do not lose optimality at all.

Theorem 2.7. *(Guarantee on guessed survival tree performance). Given dataset $\{\mathbf{X}, \mathbf{c}, \mathbf{y}\}$, depth constraint d, leaf penalty λ and a reference model T, let t_{guess} be the tree returned using lb_{guess} defined in Equation 12 and $R(t_{guess}, \mathbf{X}, \mathbf{c}, \mathbf{y})$ be its objective. Let t^* be the true optimal tree on the training set according to the regularized objective defined in Equation 4. We have:*

$$R(t_{guess}, \mathbf{X}, \mathbf{c}, \mathbf{y}) \leq \lambda H_{t^*} + \sum_{i=1}^{N} \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i), \mathcal{L}(\hat{S}^{t^*}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i)\right\} \quad (13)$$

where $\hat{S}^T_{\mathbf{x}_i}(\cdot)$ is the predicted survival function of reference model T on sample i, and $\hat{S}^{t^*}_{\mathbf{x}_i}(\cdot)$ is the predicted survival function of optimal tree t^*. That is, the objective of the guessed tree is no worse than the union of errors made by the reference model and the optimal tree.

Under certain circumstances, the tree returned does not lose optimality.

Corollary 2.7.1. *Let T, t_{guess}, t^* defined as in Theorem 2.7. If the reference model T performs no worse than t^* on each sample, the tree returned using lb_{guess} is still optimal.*

In other words, if the reference model has good training performance (even if it is overfitted), the tree returned after using this optional guessing technique is still optimal.

3 EXPERIMENTS

We ran experiments on 17 datasets (11 real-world survival datasets and 6 synthetic datasets from regression tasks), whose details are described in Appendix D. Our experiments answer the following questions. 1) How far from optimal are existing survival tree methods (Section 3.1)? 2) How well do optimal sparse survival trees generalize (Section 3.2)? 3) How long does OSST take to find the optimal survival trees (Section 3.4)? 4) How does OSST scale on large datasets (Section 3.5)? 5) What do optimal survival trees look like (Section 3.6)?

We used Conditional Inference Trees (CTree) (Hothorn et al., 2015), Recursive Partitioning and Regression Trees (RPART)(Therneau and Atkinson, 2019), and the SurvivalTree model in Scikit-survival (SkSurv) (Pölsterl, 2020) as baselines. Interpretable AI (IAI) implements the OST algorithm proposed by Bertsimas et al. (2022). We were given a license to the (proprietary) software, but source code was unavailable. When we tried to run the experiments, it frequently crashed. As a result, we were unable to include results from it below. We discuss Interpretable AI further in Appendix L.

We use several metrics to evaluate the quality of our survival trees, discussed in Appendix C. This includes the Integrated Brier Score Ratio (IBS Ratio – higher is better). The IBS Ratio of tree t is:

$$\text{IBS Ratio}(t) = 1 - \frac{\mathcal{L}(t, \mathbf{X}, \mathbf{c}, \mathbf{y})}{\mathcal{L}(t_0, \mathbf{X}, \mathbf{c}, \mathbf{y})} \quad (14)$$

where $\mathcal{L}$ is the IBS loss defined in Equation 1 and t_0 is a single node containing all samples (equivalently a KM estimator ignoring all features). *All IBS scores in our experiments below refer to IBS Ratios.*

We also use other metrics, namely Harrell's C-index and Uno's C-index (which are concordance metrics, higher is better), and Cumulative-Dynamic-AUC (higher is better); see Appendix C. Note that these metrics are not additive, which means that if we want to optimize them, then we must construct the entire survival tree. Therefore we can use these metrics only in the evaluation stage, not for training. We found that even though we optimize the IBS, our optimal survival trees often perform better on the other metrics as well, which we show in Section 3.3 and Appendix G.

3.1 Optimality

For our method and baselines, we used different hyperparameters to generate trees of various sizes and show the relationship between loss and sparsity. Figure 1 demonstrates that OSST *produces better performance (IBS Ratio) than all other methods.* Compared with other methods, the trees returned by OSST have higher performance and fewer leaves, which means that the trees found by OSST have *higher quality in both test performance and sparsity.* Importantly, *since OSST is the only method to produce optimal trees, other methods cannot quantify closeness to optimality without OSST.* More extensive results appear in Appendix E.

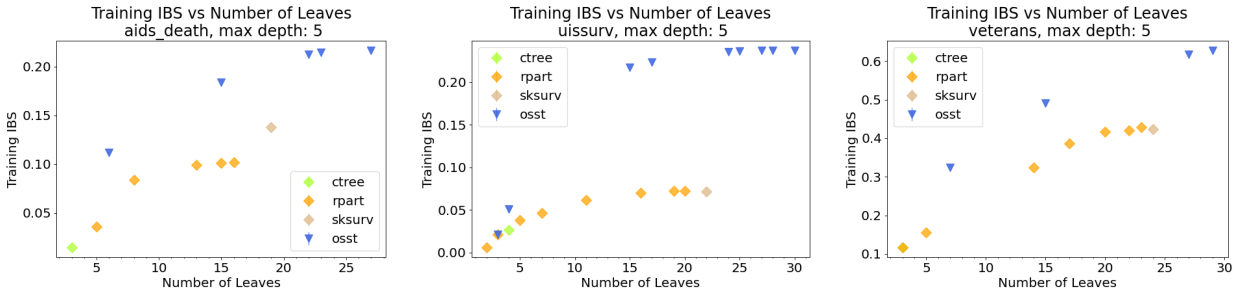

Figure 1: Training Score (IBS Ratio) of CTree, RPART, SkSurv and OSST on datasets: aids_death, uissurv, veterans, max depth 5.

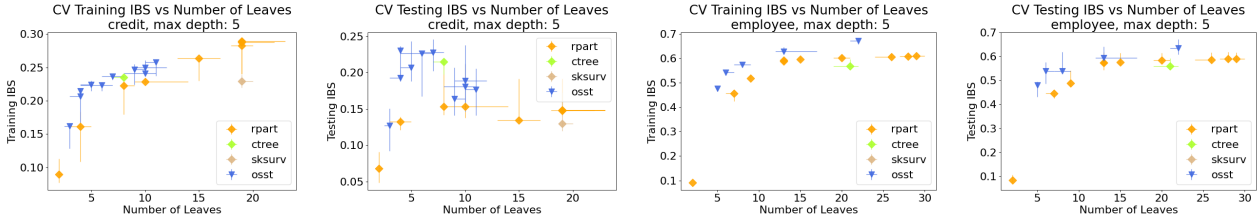

Figure 2: 5-fold cross validation of CTree, RPART, SkSurv and OSST on datasets: credit, employee, max depth 5.

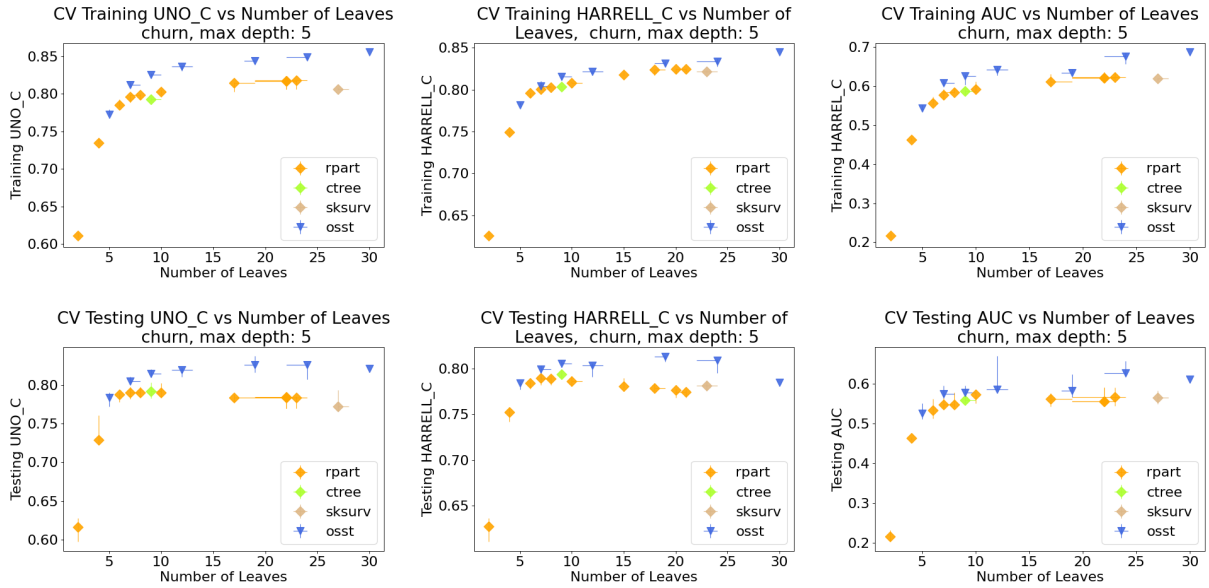

Figure 3: Testing performance of CTree, RPART, SkSurv and OSST on churn dataset, max depth 5, using different metrics. Cross-validation was used for confidence intervals.

3.2 Generalization

We ran 5-fold cross-validation experiments on various datasets; the results show that OSST *generalizes well*. Figure 2 shows that optimal sparse survival trees *also obtain higher testing performance.* More cross-validation results appear in Appendix F.

3.3 Comprehesive Quality

We evaluated trees returned by all methods again, using metrics defined in Appendix C. Figure 3 shows that the trees found by OSST also have higher Uno's C-index, Harrell's C-index and Cumulative-Dynamic-AUC, which indicates *better overall quality.*

3.4 Running Time

Table 1 shows the 60-trial average run time and standard deviation of OSST on each dataset, using different configurations. The trees vary in complexity between 4 and 64 leaves. *OSST is often able to find the optimal survival trees within a few seconds.* More details appear in Appendix H.

Dataset	running time(s)
aids	2.28(±3.66)
aids_death	2.05(±3.82)
maintenance	0.78(±1.12)
uissurv	3.41(±2.86)
veterans	0.35(±0.53)
whas500	8.76(±14.7)
gbsg2	0.20(±0.25)
insurance	0.08(±0.02)
sync	2.65(±1.93)

Table 1: Summary of OSST average running time with different configurations on various datasets.

3.5 Scalability

Figure 4 shows that OSST achieved similar scalability performance to greedy methods with datasets of fewer than 10K samples. It is slower than CTree and RPART due to their greedy nature, but interestingly, it scales better than SkSurv when the number of samples exceeds 10K. The details of this experiment can be found in Appendix I. Again, we note that greedy methods do not have any performance guarantees (unlike OSST).

3.6 Optimal Survival Trees

Figure 5 and 6 show two example optimal sparse survival trees trained on the veterans and churn dataset. More trees can be found in Appendix K.

4 LIMITATIONS

We limited our baselines to other interpretable models. In high-stakes decision making, practitioners are ethically unable to use uninterpretable models. Particularly in medical domains where data are messy and incomplete, interpretability is essential (Council, 2019; Ellis et al., 2022). At the same time, sparsity comes with a cost; it is not clear whether sparse survival techniques achieve the performance of black box models (see Appendix N for more detail).

5 CONCLUSION

We provide a practical algorithm that is able to find provably-optimal sparse survival trees within a reasonable time, despite the hardness of fully optimizing a survival tree. Our method quickly finds optimal sparse survival models that generalize well and scales nicely to large datasets. There are many possible directions for future work. One is extending the optimized objective to other metrics mentioned before and possibly new metrics defined by users. The other one is to find systematic ways to create reference models that substantially speed up the search without impacting the performance of the returned trees.

Code Availability

The implementation of **Optimal Sparse Survival Trees** can be found at `https://github.com/ruizhang1996/optimal-sparse-survival-trees-public`.

Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). Nous remercions le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) de son soutien.

We acknowledge the following grants: NIH 1R01HL166233-01 and NIH/NIDA R01DA054994.

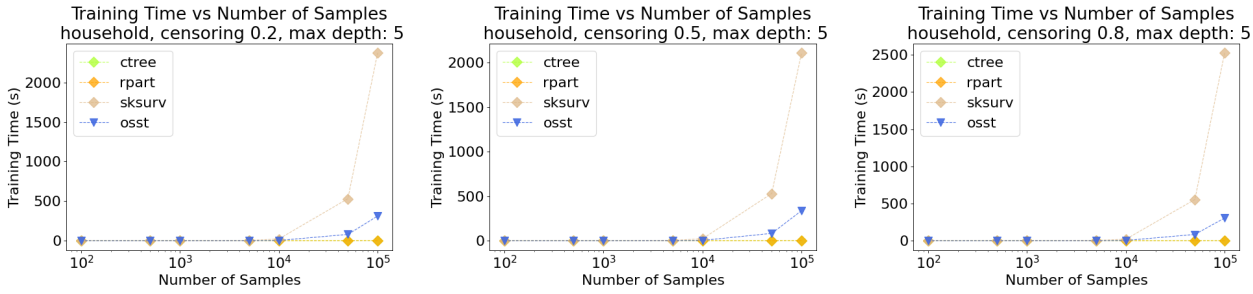

Figure 4: Training time of CTree, RPART, SkSurv and OSST as a function of sample size on household dataset, $d = 5, \lambda = 0.01$ (60-minutes time limit).

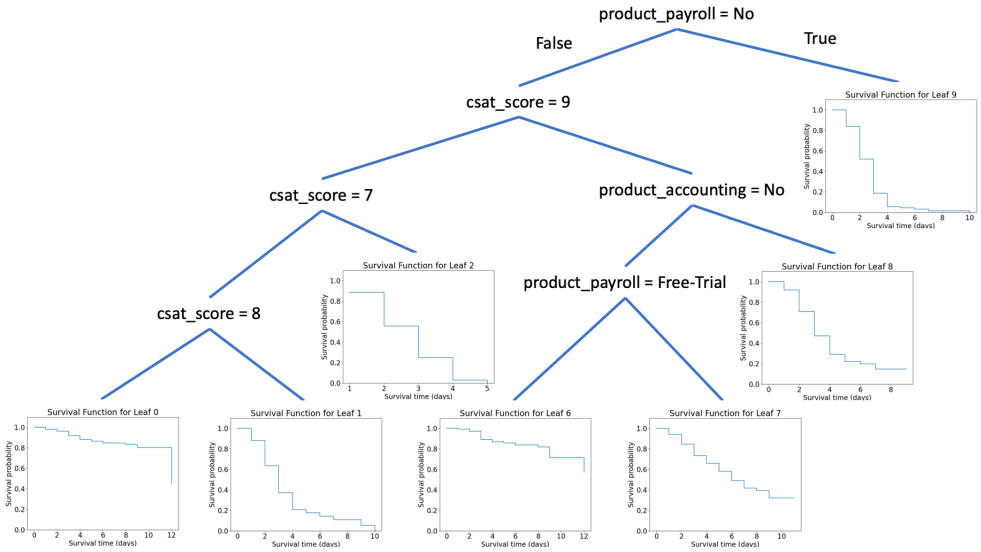

Figure 5: Optimal survival tree produced by OSST for churn dataset, 7 leaves. IBS ratio: 48.68%

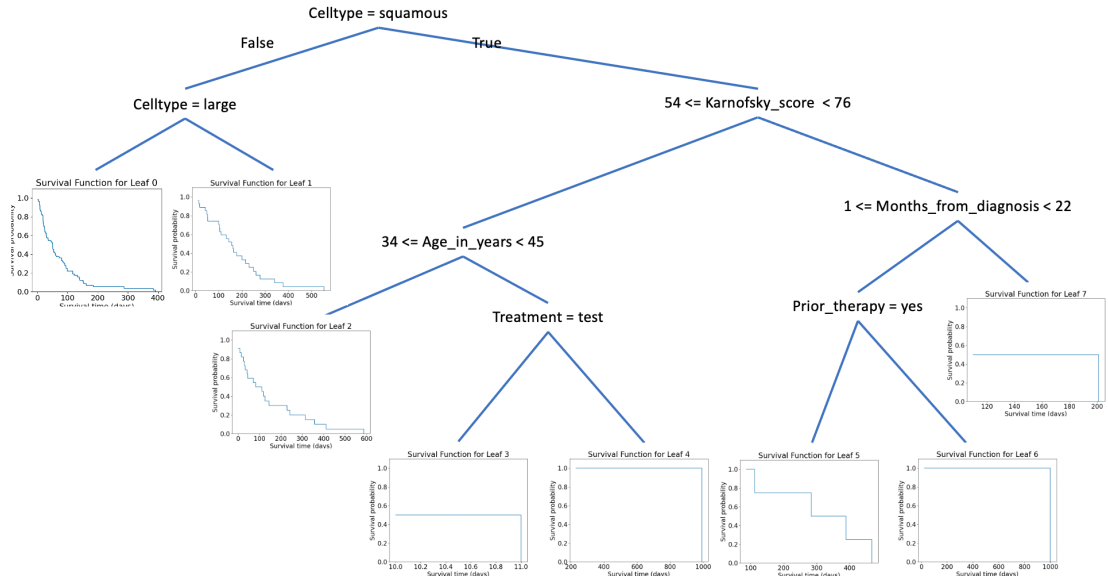

Figure 6: Optimal survival tree produced by OSST for veterans dataset, 8 leaves. IBS ratio: 32.83%

References

A. Bender, A. Groll, and F. Scheipl. A generalized additive model approach to time-to-event analysis. *Statistical Modelling*, 18(3-4):299–321, 2018.

D. Bertsimas, J. Dunn, E. Gibson, and A. Orfanoudaki. Optimal survival trees. *Machine Learning*, 111(8): 2951–3023, 2022.

Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8(1): 6085, 2018.

T. Ching, X. Zhu, and L. X. Garmire. Cox-nnet: an artificial neural network method for prognosis prediction of high-throughput omics data. *PLoS Computational Biology*, 14(4):e1006076, 2018.

M. Choi. Kaggle insurance data, 2018. URL `https://www.kaggle.com/datasets/mirichoi0218/insurance`.

A. Ciampi, S. A. Hogg, S. McKinney, and J. Thiffault. Recpam: a computer program for recursive partition and amalgamation for censored survival data and other situations frequently occurring in biostatistics. i. methods and program features. *Computer Methods and Programs in Biomedicine*, 26(3):239–256, 1988.

J. Council. Data challenges are halting AI projects, IBM executive says. *The Wall Street Journal*, 2019. URL `https://www.wsj.com/articles/data-challenges-are-halting-ai-projects-ibm-executive-says-11559035800`.

D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202, 1972.

D. R. Cox. *Analysis of binary data*. Routledge, 2018.

R. B. Davis and J. R. Anderson. Exponential survival trees. *Statistics in Medicine*, 8(8):947–961, 1989.

A. Dispenzieri, J. A. Katzmann, R. A. Kyle, D. R. Larson, T. M. Therneau, C. L. Colby, R. J. Clark, G. P. Mead, S. Kumar, L. J. Melton III, et al. Use of nonclonal serum immunoglobulin free light chains to predict overall survival in the general population. In *Mayo Clinic Proceedings*, volume 87, pages 517–523. Elsevier, 2012.

D. Dua and C. Graff. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2017. Accessed: 2022-04-01.

J. Dunn. *Optimal Trees for Prediction and Prescription*. PhD thesis, Massachusetts Institute of Technology, 2018.

R. J. Ellis, R. M. Sander, and A. Limon. Twelve key challenges in medical machine learning and solutions. *Intelligence-Based Medicine*, 6:100068, 2022. ISSN 2666-5212. doi: https://doi.org/10.1016/j.ibmed.2022.100068. URL `https://www.sciencedirect.com/science/article/pii/S2666521222000217`.

S. Fotso et al. PySurvival: Open source package for survival analysis modeling, 2019. URL `https://www.pysurvival.io/`.

E. Giunchiglia, A. Nemchenko, and M. van der Schaar. Rnn-surv: A deep recurrent model for survival analysis. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*, pages 23–32. Springer, 2018.

L. Gordon and R. A. Olshen. Tree-structured survival analysis. *Cancer Treatment Reports*, 69(10):1065–1069, 1985.

E. Graf, C. Schmoor, W. Sauerbrei, and M. Schumacher. Assessment and comparison of prognostic classification schemes for survival data. *Statistics in Medicine*, 18(17-18):2529–2545, 1999.

T. Grubinger, A. Zeileis, and K.-P. Pfeiffer. evtree: Evolutionary learning of globally optimal classification and regression trees in r. *Journal of Statistical Software*, 61:1–29, 2014.

F. E. Harrell Jr, K. L. Lee, and D. B. Mark. Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine*, 15 (4):361–387, 1996.

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

T. Hothorn, K. Hornik, and A. Zeileis. ctree: Conditional inference trees. *The Comprehensive R Archive Network*, 8, 2015.

X. Hu, C. Rudin, and M. Seltzer. Optimal sparse decision trees. In *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

H. Hung and C.-T. Chiang. Estimation methods for time-dependent auc models with survival data. *Canadian Journal of Statistics*, 38(1):8–26, 2010.

H. Ishwaran and U. B. Kogalur. Random survival forests for r. *R News*, 7(2):25–31, 2007.

H. Jin, Y. Lu, K. Stone, and D. M. Black. Alternative tree-structured survival analysis based on variance of survival time. *Medical Decision Making*, 24(6): 670–680, 2004.

J. D. Kalbfleisch and R. L. Prentice. *The statistical analysis of failure time data*. John Wiley & Sons, 2011.

E. L. Kaplan and P. Meier. Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282):457–481, 1958.

J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18(1):1–12, 2018.

S. Keles and M. R. Segal. Residual-based tree-structured survival analysis. *Statistics in Medicine*, 21(2):313–326, 2002.

D. G. Kleinbaum and M. Klein. *Survival analysis, a self-learning text*. Springer, 1996.

H. Kvamme, Ø. Borgan, and I. Scheel. Time-to-event prediction with neural networks and cox regression. *Journal of Machine Learning Research*, 20: 1–30, 2019.

J. Lambert and S. Chevret. Summary measure of discrimination in survival models based on cumulative/dynamic time-dependent roc curves. *Statistical Methods in Medical Research*, 25(5):2088–2102, 2016.

J. F. Lawless. *Statistical models and methods for lifetime data*. John Wiley & Sons, 2011.

M. LeBlanc and J. Crowley. Relative risk trees for censored survival data. *Biometrics*, pages 411–425, 1992.

M. LeBlanc and J. Crowley. Survival trees by goodness of split. *Journal of the American Statistical Association*, 88(422):457–467, 1993.

C. Lee, W. Zame, J. Yoon, and M. Van Der Schaar. Deephit: A deep learning approach to survival analysis with competing risks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

S. Lemeshow, S. May, and D. W. Hosmer Jr. *Applied survival analysis: regression modeling of time-to-event data*. John Wiley & Sons, 2011.

J. Lin, C. Zhong, D. Hu, C. Rudin, and M. Seltzer. Generalized and scalable optimal sparse decision trees. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 6150–6160, 2020.

H. McTavish, C. Zhong, R. Achermann, I. Karimalis, J. Chen, C. Rudin, and M. Seltzer. Fast sparse decision tree optimization via reference ensembles. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2022.

A. M. Molinaro, S. Dudoit, and M. J. Van der Laan. Tree-based multivariate regression and density estimation with right-censored data. *Journal of Multivariate Analysis*, 90(1):154–177, 2004.

S. Nijssen, P. Schaus, et al. Learning optimal decision trees using caching branch-and-bound search. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

L. Norton. A gompertzian model of human breast cancer growth. *Cancer Research*, 48(24_Part_1): 7067–7071, 1988.

S. Pölsterl. scikit-survival: A library for time-to-event analysis built on top of scikit-learn. *Journal of Machine Learning Research*, 21(212):1–6, 2020. URL `http://jmlr.org/papers/v21/20-729.html`.

B. D. Ripley and R. M. Ripley. Neural networks as statistical methods in survival analysis. *Clinical Applications of Artificial Neural Networks*, 237:255, 2001.

C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16(none):1 – 85, 2022. doi: 10.1214/21-SS133. URL `https://doi.org/10.1214/21-SS133`.

M. Schumacher, G. Bastert, H. Bojar, K. Hübner, M. Olschewski, W. Sauerbrei, C. Schmoor, C. Beyerle, R. Neumann, and H. Rauschecker. Randomized 2 x 2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. german breast cancer study group. *Journal of Clinical Oncology*, 12(10):2086–2093, 1994.

M. R. Segal. Regression trees for censored data. *Biometrics*, pages 35–47, 1988.

T. Therneau and B. Atkinson. *rpart: Recursive Partitioning and Regression Trees*, 2019. URL `https://CRAN.R-project.org/package=rpart`. R package version 4.1-15.

T. M. Therneau, P. M. Grambsch, and T. R. Fleming. Martingale-based residuals for survival models. *Biometrika*, 77(1):147–160, 1990.

H. Uno, T. Cai, L. Tian, and L.-J. Wei. Evaluating prediction rules for t-year survivors with censored regression models. *Journal of the American Statistical Association*, 102(478):527–537, 2007.

H. Uno, T. Cai, M. J. Pencina, R. B. D'Agostino, and L.-J. Wei. On the c-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. *Statistics in Medicine*, 30(10):1105–1117, 2011.

H. Zhang. Splitting criteria in survival trees. In *Statistical Modelling: Proceedings of the 10th International Workshop on Statistical Modelling Innsbruck, Austria, 10–14 July, 1995*, pages 305–313. Springer, 1995.

R. Zhang, R. Xin, M. Seltzer, and C. Rudin. Optimal sparse regression trees. In *Proceedings of the AAAI*

Conference on Artificial Intelligence, pages 11270–11279, 2023.

L. Zhao and D. Feng. Dnnsurv: Deep neural networks for survival analysis using pseudo values. *arXiv preprint arXiv:1908.02337*, 2019.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Extended Related Work

Early Survival Analysis: In 1958, Kaplan and Meier (1958) proposed the first survival models, Kaplan-Meier (KM) curves, which are simple non-parametric models that do not use covariates. Parametric models (Cox, 2018; Kalbfleisch and Prentice, 2011; Lawless, 2011; Norton, 1988; Kleinbaum and Klein, 1996) usually rely on strong assumptions either for the hazard rate or the underlying distribution of the survival time. They assume the hazard function follows a particular distribution, such as the exponential, Weibull, or log-normal distribution. Cox proportional hazards regression (Cox, 1972) is a practical semi-parametric approach proposed in 1972, which makes the strong assumption of a constant hazard ratio between individuals and fails to capture non-linear relationships between covariates and predicted outcomes. Even though the coefficient of a Cox model is easy to interpret as a hazard ratio, it loses interpretability if the model is not sparse in the number of features.

Decision Tree Learning for Survival Analysis: Attempts have been made to adapt traditional decision trees for censored survival data by proposing various splitting criteria to heuristically construct survival trees. Gordon and Olshen (1985) chose splits by minimizing the Wasserstein distance between the two child node's Kaplan–Meier curves. Davis and Anderson (1989) used exponential log-likelihood loss for splitting. LeBlanc and Crowley (1992) measured node deviance for splitting while Therneau et al. (1990) and Keles and Segal (2002) used martingale residuals as the splitting criteria. Zhang (1995) combined impurity measurements for observed time and portion of censored samples. Various authors proposed other techniques to maximize the distance between the two child nodes using various statistics (LeBlanc and Crowley, 1993; Ciampi et al., 1988; Segal, 1988; Hothorn et al., 2006; Jin et al., 2004). Molinaro et al. (2004) used Inverse Probability of Censoring Weight (IPCW) to reduce the bias caused by a high degree of censoring. These splitting criteria either minimize the impurity within nodes or maximize the dissimilarity between different nodes, but all of them use greedy approaches, which means if a bad split is chosen at the top, there is no way to correct it. Importantly, there is no way to determine whether the trees are optimal without a method like ours that provably optimizes the tree structure.

Black Box Survival Models: The use of decision trees for survival analysis was further extended to more sophisticated forest models such as random survival forests (Ishwaran and Kogalur, 2007) and conditional inference forests (Hothorn et al., 2006). Neural networks have been applied to survival analysis tasks as well (Katzman et al., 2018; Ching et al., 2018; Che et al., 2018; Ripley and Ripley, 2001; Giunchiglia et al., 2018). These black box models are useful for performance comparisons, but are not generally useful in practice, particularly for high-stakes decisions.

Modern Decision Tree Methods: Hu et al. (2019); Lin et al. (2020); Zhang et al. (2023); Grubinger et al. (2014); Nijssen et al. (2020); Dunn (2018) have shown the success of optimal sparse trees in both classification and regression. A single sparse optimal tree can achieve the performance of complex black box models while providing interpretability (McTavish et al., 2022). Bertsimas et al. (2022) proposed a survival tree model using local search techniques, but it holds an assumption similar to that of the Cox model; that is the ratio of the hazard functions for any two individuals is assumed to be constant over time and independent of the values of the covariates. As we show in Appendix L, in cases where we can get their code to run, OSST produces better IBS ratios than their OST algorithm.

B Theorems and Proofs

B.1 Proof of Theorem 2.1

Theorem 2.1 *The loss of a survival tree is an additive function of the observations and leaves.*

Proof. From Equation 1 and 2, we know:

$$\mathcal{L}(t, \mathbf{X}, \mathbf{c}, \mathbf{y}) = \frac{1}{y_{\max}} \int_0^{y_{\max}} BS(y) dy \tag{15}$$

$$= \frac{1}{y_{\max}} \int_0^{y_{\max}} \frac{1}{N} \sum_{i=1}^{N} \left\{ \frac{(\hat{S}_{\mathbf{x}_i}(y) - 0)^2}{\hat{G}(y_i)} \cdot \mathbf{1}_{y_i \leq y, c_i = 1} + \frac{(\hat{S}_{\mathbf{x}_i}(y) - 1)^2}{\hat{G}(y)} \cdot \mathbf{1}_{y_i > y} \right\} dy. \tag{16}$$

Extracting the constant term $\frac{1}{N}$ and putting the integral into the summation yields:

$$= \frac{1}{y_{\max}}\frac{1}{N}\sum_{i=1}^{N}\left\{\int_0^{y_{\max}}\frac{(\hat{S}_{\mathbf{x}_i}(y)-1)^2}{\hat{G}(y)}\cdot\mathbf{1}_{y_i>y}dy+\int_0^{y_{\max}}\frac{(\hat{S}_{\mathbf{x}_i}(y)-0)^2}{\hat{G}(y_i)}\cdot\mathbf{1}_{y_i\le y,c_i=1}dy\right\}$$
$$= \frac{1}{y_{\max}}\frac{1}{N}\sum_{i=1}^{N}\left\{\int_0^{y_i}\frac{(\hat{S}_{\mathbf{x}_i}(y)-1)^2}{\hat{G}(y)}dy+c_i\int_{y_i}^{y_{\max}}\frac{(\hat{S}_{\mathbf{x}_i}(y)-0)^2}{\hat{G}(y_i)}dy\right\}.$$

Now we have proved that the loss of a survival tree is additive in observations. And each sample can only fall into one leave at a time:

$$= \frac{1}{y_{\max}}\frac{1}{N}\sum_{i=1}^{N}\left\{\int_0^{y_i}\frac{(\hat{S}_{\mathbf{x}_i}(y)-1)^2}{\hat{G}(y)}dy+c_i\int_{y_i}^{y_{\max}}\frac{(\hat{S}_{\mathbf{x}_i}(y)-0)^2}{\hat{G}(y_i)}dy\right\}\cdot\mathbf{1}_{\mathrm{cap}(t_{\mathrm{fix}},\mathbf{x}_i)},$$
$$+\frac{1}{y_{\max}}\frac{1}{N}\sum_{i=1}^{N}\left\{\int_0^{y_i}\frac{(\hat{S}_{\mathbf{x}_i}(y)-1)^2}{\hat{G}(y)}dy+c_i\int_{y_i}^{y_{\max}}\frac{(\hat{S}_{\mathbf{x}_i}(y)-0)^2}{\hat{G}(y_i)}dy\right\}\cdot\mathbf{1}_{\mathrm{cap}(t_{\mathrm{split}},\mathbf{x}_i)}.$$

Therefore, the objective is additive in the observations, and observations can be grouped by leaves, so the loss is additive in the leaves. □

B.2 Proof of Theorem 2.2

Theorem 2.2 (Hierarchical Objective Lower Bound). *Any tree* $t'=(t'_{\mathrm{fix}},\delta'_{\mathrm{fix}},t'_{\mathrm{split}},\delta'_{\mathrm{split}},K',H_{t'})\in\sigma(t)$ *in the child tree set of* $t=(t_{\mathrm{fix}},\delta_{\mathrm{fix}},t_{\mathrm{split}},\delta_{\mathrm{split}},K,H_t)$ *obeys:*

$$R(t',\mathbf{X},\mathbf{c},\mathbf{y})\ge\mathcal{L}(t_{\mathrm{fix}},\mathbf{X},\mathbf{c},\mathbf{y})+\lambda H_t.$$

That is, the objective lower bound of the parent tree holds for all its child trees. This bound ensures that we do not further explore child trees if the parent tree can be pruned via the lower bound.

Proof. As we know, $K'\ge K, H_{t'}>H_t$, since t' is a child tree of t. The objective lower bound (which holds for all trees) of t' is:

$$R(t',\mathbf{X},\mathbf{c},\mathbf{y})\ge\mathcal{L}(t'_{\mathrm{fix}},\mathbf{X},\mathbf{y})+\lambda H_{t'}. \tag{17}$$

Since $\mathcal{L}(t'_{\mathrm{fix}},\mathbf{X},\mathbf{c},\mathbf{y})=\mathcal{L}(t_{\mathrm{fix}},\mathbf{X},\mathbf{c},\mathbf{y})+\mathcal{L}(t'_{\mathrm{fix}}\setminus t_{\mathrm{fix}},\mathbf{X},\mathbf{c},\mathbf{y})$ and the loss of $K'-K$ fixed leaves in t is

$$\mathcal{L}(t'_{\mathrm{fix}}\setminus t_{\mathrm{fix}},\mathbf{X},\mathbf{c},\mathbf{y})=\frac{1}{y_{\max}}\frac{1}{N}\sum_{i=1}^{N}\left\{\int_0^{y_i}\frac{(\hat{S}_{\mathbf{x}_i}(y)-1)^2}{\hat{G}(y)}dy+c_i\int_{y_i}^{y_{\max}}\frac{(\hat{S}_{\mathbf{x}_i}(y)-0)^2}{\hat{G}(y_i)}dy\right\}\cdot\mathbf{1}_{\mathrm{cap}(t'_{\mathrm{fix}}\setminus t_{\mathrm{fix}},\mathbf{x}_i)}$$

and $0\le\hat{G}(\cdot)\le 1$, so

$$\mathcal{L}(t'_{\mathrm{fix}}\setminus t_{\mathrm{fix}},\mathbf{X},\mathbf{c},\mathbf{y})\ge 0, \tag{18}$$

and thus we have:

$$\mathcal{L}(t'_{\mathrm{fix}},\mathbf{X},\mathbf{c},\mathbf{y})\ge\mathcal{L}(t_{\mathrm{fix}},\mathbf{X},\mathbf{c},\mathbf{y}) \tag{19}$$

therefore:

$$R(t',\mathbf{X},\mathbf{c},\mathbf{y})\ge\mathcal{L}(t_{\mathrm{fix}},\mathbf{X},\mathbf{c},\mathbf{y})+\lambda H_t. \tag{20}$$

□

B.3 Proof of Theorem 2.3

Theorem 2.3 *(Objective Lower Bound with One-step Lookahead). Let* $t=(t_{\mathrm{fix}},\delta_{\mathrm{fix}},t_{\mathrm{split}},\delta_{\mathrm{split}},K,H_t)$ *be a tree with* H_t *leaves. If* $\mathcal{L}(t_{\mathrm{fix}},\mathbf{X},\mathbf{c},\mathbf{y})+\lambda H_t+\lambda>R^c$*, even if its objective lower bound* $\mathcal{L}(t_{\mathrm{fix}},\mathbf{X},\mathbf{c},\mathbf{y})+\lambda H_t\le R^c$*, then for any child tree* $t'\in\sigma(t)$*,* $R(t',\mathbf{X},\mathbf{c},\mathbf{y})>R^c$*. That is, even if a parent tree cannot be pruned via its objective lower bound, if* $\mathcal{L}(t_{\mathrm{fix}},\mathbf{X},\mathbf{y})+\lambda H_t+\lambda>R^c$*, all of its child trees are sub-optimal and can be pruned (and never explored).*

Proof. This bound adapts directly from OSRT (Zhang et al., 2023), where the proof can be found. □

B.4 Proof of Lemma 2.4

Lemma 2.4 *(Equivalent Loss). Let u be a set of equivalent points defined as Equation 8. We denote ${}^*S$ as the optimal step function that minimizes the IBS loss only for set u (leaf contains set u only), such that:*

$$ {}^*S = \underset{S}{\operatorname{argmin}} \sum_{k=1}^{|u|} \mathcal{L}(S, \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}). $$

*We define **Equivalent Loss** for set u as*

$$ \mathcal{E}_u = \sum_{k=1}^{|u|} \mathcal{L}({}^*S, \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}) $$

then any leaf that captures set u in a survival tree trained on dataset $\{\mathbf{X}, \mathbf{c}, \mathbf{y}\}$ has loss $\geq \mathcal{E}_u$.

Proof. In any survival tree, the points in set u always get assigned to the same leaf. Let l_u be a leaf node that contains set u only and let $\hat{S}$ be its KM estimator. Since the KM estimator is not the minimizer of the IBS loss, the loss of l_u obeys:

$$ \mathcal{L}(l_u, \mathbf{X}, \mathbf{c}, \mathbf{y}) = \sum_{k=1}^{|u|} \mathcal{L}(\hat{S}, \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}) \geq \sum_{k=1}^{|u|} \mathcal{L}({}^*S, \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}) = \mathcal{E}_u. \tag{21} $$

Therefore the loss of l_u is at least $\mathcal{E}_u$.

Let l'_u be a leaf node that contains both set u and N' other sample points ($1 \leq N' \leq N - |u|$), and denote $\hat{S}'$ as its KM estimator. We need to prove that for leaf node l'_u, its loss is at least $\mathcal{E}_u$. The loss of l_u is given by:

$$ \mathcal{L}(l'_u, \mathbf{X}, \mathbf{c}, \mathbf{y}) = \sum_{k=1}^{|u|} \mathcal{L}(\hat{S}', \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}) + \sum_{k=1}^{N'} \mathcal{L}(\hat{S}', \mathbf{x}_{i_k}, \mathbf{c}_{i_k}, \mathbf{y}_{i_k}) $$

Let ${}^*S'$ be the optimal step function that minimizes the IBS loss for leaf node l'_u, such that:

$$ {}^*S' = \underset{S}{\operatorname{argmin}} \sum_{k=1}^{|u|} \mathcal{L}(S, \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}) + \sum_{k=1}^{N'} \mathcal{L}(S, \mathbf{x}_{i_k}, \mathbf{c}_{i_k}, \mathbf{y}_{i_k}). $$

Then, the loss of l'_u obeys:

$$ \sum_{k=1}^{|u|} \mathcal{L}(\hat{S}', \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}) + \sum_{k=1}^{N'} \mathcal{L}(\hat{S}', \mathbf{x}_{i_k}, \mathbf{c}_{i_k}, \mathbf{y}_{i_k}) \geq \sum_{k=1}^{|u|} \mathcal{L}({}^*S', \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}) + \sum_{k=1}^{N'} \mathcal{L}({}^*S', \mathbf{x}_{i_k}, \mathbf{c}_{i_k}, \mathbf{y}_{i_k}). \tag{22} $$

${}^*S$ and ${}^*S'$ obey:

$$ \sum_{k=1}^{|u|} \mathcal{L}({}^*S', \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}) + \sum_{k=1}^{N'} \mathcal{L}({}^*S', \mathbf{x}_{i_k}, \mathbf{c}_{i_k}, \mathbf{y}_{i_k}) \geq \sum_{k=1}^{|u|} \mathcal{L}({}^*S', \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}) \geq \sum_{k=1}^{|u|} \mathcal{L}({}^*S, \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}). \tag{23} $$

Substituting Equation 23 into Equations 22 and 21, we have:

$$ \sum_{k=1}^{|u|} \mathcal{L}(\hat{S}', \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}) + \sum_{k=1}^{N'} \mathcal{L}(\hat{S}', \mathbf{x}_{i_k}, \mathbf{c}_{i_k}, \mathbf{y}_{i_k}) \geq \mathcal{E}_u. $$

Therefore, for any leaf that captures the equivalent set u, its loss is greater than or equal to the equivalent loss of the set $\mathcal{E}_u$. □

Lemma 2.5 *Let l be a leaf node that captures n equivalent sets $\{u_i\}_{i=1}^n$ with corresponding $\mathcal{E}_{u_i}$. The loss of l obeys: $\mathcal{L}(l, \mathbf{X}, \mathbf{c}, \mathbf{y}) \geq \sum_{i=1}^n \mathcal{E}_{u_i}$. That is, the lower bound of a leaf is the sum of equivalent losses of the equivalent sets it captures.*

Proof.

$$\begin{aligned}
\mathcal{L}(l, \mathbf{X}, \mathbf{c}, \mathbf{y}) &\geq \sum_{i=1}^{n} \sum_{k=1}^{|u_i|} \mathcal{L}(\hat{S}, \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}) \\
&\geq \sum_{i=1}^{n} \sum_{k=1}^{|u_i|} \mathcal{L}({}^{*}S, \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}), \text{where } {}^{*}S = \operatorname*{argmin}_{S} \sum_{i=1}^{n} \sum_{k=1}^{|u_i|} \mathcal{L}(S, \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}), \\
&\geq \sum_{i=1}^{n} \sum_{k=1}^{|u_i|} \mathcal{L}({}^{*}S_i, \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}), \text{where } {}^{*}S_i = \operatorname*{argmin}_{S} \sum_{k=1}^{|u_i|} \mathcal{L}(S, \mathbf{x}_{j_k}, \mathbf{c}_{j_k}, \mathbf{y}_{j_k}) \\
&\geq \sum_{i=1}^{n} \mathcal{E}_{u_i}.
\end{aligned}$$

□

B.5 Proof of Theorem 2.6

Theorem 2.6 *(Equivalent Points Lower Bound). Let $t = (t_{\text{fix}}, \delta_{\text{fix}}, t_{\text{split}}, \delta_{\text{split}}, K, H_t)$ be a tree with K fixed leaves and $H_t - K$ splitting leaves. For any child tree $t' = (t'_{\text{fix}}, \delta'_{\text{fix}}, t'_{\text{split}}, \delta'_{\text{split}}, K', H_{t'}) \in \sigma(t)$:*

$$R(t', \mathbf{X}, \mathbf{c}, \mathbf{y}) \geq \mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{y}) + \lambda H_t + \sum_{u=1}^{U} \mathcal{E}_u \cdot \mathbf{1}_{cap(t_{\text{split}}, u)}, \tag{24}$$

where $\mathbf{1}_{cap(t_{\text{split}}, u)}$ is 1 when t_{split} captures set u, 0 otherwise. Combining with the idea of Theorem 2.3, we have:

$$R(t', \mathbf{X}, \mathbf{y}) \geq \mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_t + \lambda + \sum_{u=1}^{U} \mathcal{E}_u \cdot \mathbf{1}_{cap(t_{\text{split}}, u)}. \tag{25}$$

Proof.

$$\begin{aligned}
R(t', \mathbf{X}, \mathbf{c}, \mathbf{y}) &= \mathcal{L}(t', \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_{t'} \\
&= \mathcal{L}(t'_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \mathcal{L}(t'_{\text{split}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_{t'} \\
&= \mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \mathcal{L}(t'_{\text{fix}} \setminus t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \mathcal{L}(t'_{\text{split}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_{t'}.
\end{aligned} \tag{26}$$

Since samples captured by t_{split} are captured either by $t'_{\text{fix}} \setminus t_{\text{fix}}$ or t'_{split}, and equivalence loss cannot be eliminated in any tree, from Lemma 2.5, we have:

$$\mathcal{L}(t'_{\text{fix}} \setminus t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \mathcal{L}(t'_{\text{split}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) \geq \sum_{u=1}^{U} \mathcal{E}_u \cdot \mathbf{1}_{\text{cap}(t_{\text{split}}, u)}. \tag{27}$$

Substituting Equation 27 into Equation 26, we have:

$$R(t', \mathbf{X}, \mathbf{c}, \mathbf{y}) \geq \mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \sum_{u=1}^{U} \mathcal{E}_u \cdot \mathbf{1}_{\text{cap}(t_{\text{split}}, u)} + \lambda H_{t'}.$$

Because $H_{t'} \geq H_t + 1$:

$$R(t', \mathbf{X}, \mathbf{c}, \mathbf{y}) \geq \mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) + \lambda H_t + \lambda + \sum_{u=1}^{U} \mathcal{E}_u \cdot \mathbf{1}_{\text{cap}(t_{\text{split}}, u)}.$$

□

B.6 Proof of Theorem 2.7

Theorem 2.7 *(Guarantee on guessed survival tree performance). Given dataset $\{\mathbf{X}, \mathbf{c}, \mathbf{y}\}$, depth constraint d, leaf penalty λ and reference model T, let t_{guess} be the tree returned using lb_{guess} defined in Equation (12) and $R(t_{guess}, \mathbf{X}, \mathbf{c}, \mathbf{y})$ be its objective. Let t^* be the true optimal tree. We have:*

$$R(t_{guess}, \mathbf{X}, \mathbf{c}, \mathbf{y}) \leq \sum_{i=1}^{N} \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i), \mathcal{L}(\hat{S}^{t^*}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i)\right\} + \lambda H_{t^*} \tag{28}$$

where $\hat{S}^T_{\mathbf{x}_i}(\cdot)$ is the predicted survival function of reference model T on sample i, and $\hat{S}^{t^*}_{\mathbf{x}_i}(\cdot)$ is the predicted survival function of optimal tree t^*. That is, the objective of the guessed tree is no worse than the union of errors made by the reference model and the optimal tree.

Proof. Define $R_{\text{guess}}(s_a, d, \lambda)$ as the objective of the solution using the guessed lower bound for subproblem s_a. Then, $R(t_{\text{guess}}, \mathbf{X}, \mathbf{c}, \mathbf{y})$ can be rewritten as $R_{\text{guess}}(s_a, d, \lambda)$ where $s_a = \{1, 2, 3, \cdots, N\}$. We want to prove that for any s_a,

$$R_{\text{guess}}(s_a, d, \lambda) \leq \sum_{i \in s_a} \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i), \mathcal{L}(\hat{S}^{t^*_a}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i)\right\} + \lambda H_{t^*(s_a, d, \lambda)} \tag{29}$$

where t^*_a is the optimal solution to subproblem s_a. Since the lower bound of s_a, which we denote by lb_a, is non-decreasing during the optimization process, we denote the highest value lb_a gets updated to as $lb_{\max}(s_a, d, \lambda)$. The subproblem is solved when the upper bound of s_a, $ub_a \leq lb_a$, which means

$$R_{\text{guess}}(s_a, d, \lambda) \leq lb_{\max}(s_a, d, \lambda). \tag{30}$$

Now we will prove the following inequality holds using induction:

$$lb_{\max}(s_a, d, \lambda) \leq \sum_{i \in s_a} \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i), \mathcal{L}(\hat{S}^{t^*_a}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i)\right\} + \lambda H_{t^*_a}. \tag{31}$$

Either the initial lower bound $lb_{\text{guess}}(s_a)$ is greater than the initial upper bound ub, or $lb_{\text{guess}}(s_a) \leq ub$. If $lb_{\text{guess}}(s_a) > ub$, the subproblem s_a is solved without further exploration, and if $lb_{\text{guess}}(s_a) \leq ub$, then the lower bound lb_a will get updated until it reaches ub. Since t^*_a is a leaf node, the upper bound ub_a is fixed at ub. Also, $lb_{\max}(s_a, d, \lambda)$ will never be higher than the initial upper bound ub.

Base case: The optimal solution t^*_a is a leaf node. Combining these two possibilities, we have

$$lb_{\max}(s_a, d, \lambda) \leq \max\{lb_{\text{guess}}(s_a), ub\}. \tag{32}$$

Substituting the definition of $lb_{\text{guess}}(s_a)$ and ub, we have:

$$\leq \max\left\{\sum_{i \in s_a} \mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{X}(s_a), \mathbf{c}(s_a), \mathbf{y}(s_a)) + \lambda, \sum_{i \in s_a} \mathcal{L}(\hat{S}^{t^*}(\cdot), \mathbf{X}(s_a), \mathbf{c}(s_a), \mathbf{y}(s_a)) + \lambda\right\}$$

and taking the sum out:

$$\leq \sum_{i \in s_a} \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{X}(s_a), \mathbf{c}(s_a), \mathbf{y}(s_a)), \mathcal{L}(\hat{S}^{t^*}, \mathbf{X}(s_a), \mathbf{c}(s_a), \mathbf{y}(s_a))\right\} + \lambda. \tag{33}$$

Because $t^*(s_a, d, \lambda)$ is a leaf node, $H_{t^*} = 1$. Equation 31 holds.

Inductive case: The optimal solution t^*_a is a tree. In this case $d > 0$. Assume Equation 31 holds for any left and right children pair of $t^*(s_a, d, \lambda)$, s_{j_l}, s_{j_r}, created by splitting feature j, such that:

$$lb_{\max}(s_{j_l}, d-1, \lambda) \leq \sum_{i \in s_{j_l}} \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i), \mathcal{L}(\hat{S}^{t^*_{j_l}}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i)\right\} + \lambda H_{t^*_{j_l}} \tag{34}$$

and

$$lb_{\max}(s_{j_r}, d-1, \lambda) \leq \sum_{i \in s_{j_r}} \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i), \mathcal{L}(\hat{S}^{t^*_{j_r}}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i)\right\} + \lambda H_{t^*_{j_r}} \tag{35}$$

where $s_{j_l} \cap s_{j_r} = \emptyset, s_{j_l} \cup s_{j_r} = s_a$, $t^*_{j_l}, t^*_{j_r}$ are the optimal solutions to s_{j_l} and s_{j_r} respectively.

If initial upper bound $ub \leq lb_{\text{guess}}(s_a) + \lambda$, then we consider s_a solved without further exploration and its solution is a leaf node in t_{guess}.

From Equation 32 we know:

$$\begin{aligned} lb_{\max}(s_a, d, \lambda) &\leq \max\{lb_{\text{guess}}(s_a), ub\} \\ &\leq \max\{lb_{\text{guess}}(s_a), lb_{\text{guess}}(s_a) + \lambda\} \\ &\leq lb_{\text{guess}}(s_a) + \lambda \\ &\leq \sum_{i \in s_a} \mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{X}(s_a), \mathbf{c}(s_a), \mathbf{y}(s_a)) + 2\lambda \end{aligned}$$

$$\leq \sum_{i \in s_a} \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{X}(s_a), \mathbf{c}(s_a), \mathbf{y}(s_a)), \hat{S}^{t^*_a}_{\mathbf{x}_i}(\cdot), \mathbf{X}(s_a), \mathbf{c}(s_a), \mathbf{y}(s_a))\right\} + 2\lambda. \tag{36}$$

Since t^*_a is a tree, $H_{t^*_a} \geq 2$, therefore:

$$lb_{\max}(s_a, d, \lambda) \leq \sum_{i \in s_a} \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{X}(s_a), \mathbf{c}(s_a), \mathbf{y}(s_a)), \hat{S}^{t^*_a}_{\mathbf{x}_i}(\cdot), \mathbf{X}(s_a), \mathbf{c}(s_a), \mathbf{y}(s_a))\right\} + \lambda H_{t^*_a}. \tag{37}$$

If initial upper bound $ub > lb_{\text{guess}}(s_a) + \lambda$, we explore this subproblem s_a by splitting features and creating pairs of children subproblems s_{j_l}, s_{j_r} split on feature j. The lower bound of s_a will get updated until converged with the upper bound, and lb_a is updated as: $lb_a \leftarrow \max(lb_a, \min(ub_a, lb_{\text{split}}))$, where $lb_{\text{split}} \leftarrow \min_{j \in \text{features}} lb_{j_l} + lb_{j_r}$. From the way of updating lb_a, we know for any feature j:

$$lb_{\max}(s_a, d, \lambda) \leq lb_{\max}(s_{j_l}, d-1, \lambda) + lb_{\max}(s_{j_r}, d-1, \lambda). \tag{38}$$

Using the hypothesis assumption in Equation 34 and 35:

$$\leq \sum_{i \in s_{j_l}} \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i), \mathcal{L}(\hat{S}^{t^*_{j_l}}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i)\right\} + \lambda H_{t^*_{j_l}} \tag{39}$$

$$+ \sum_{i \in s_{j_r}} \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i), \mathcal{L}(\hat{S}^{t^*_{j_r}}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i)\right\} + \lambda H_{t^*_{j_r}}.$$

Equation 39 holds for $t^*_{j_l}, t^*_{j_r}$ that are subtrees of t^*_a,

$$\leq \sum_{i \in s_a} \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i), \mathcal{L}(\hat{S}^{t^*}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i)\right\} + \lambda H_{t^*}. \tag{40}$$

Here we proved that Inequality 31 holds for any subproblem s_a, including the subproblem that is the whole dataset, $s_a = \{1, 2, \cdots, N\}$. □

B.7 Proof of Corollary 2.7.1

Let T, t_{guess}, t^* be defined as in Theorem 2.7. If the reference model T performs no worse than t^* on each sample, the tree returned using lb_{guess} is still optimal. In other words, if the reference model has good performance, the tree returned after using guessing is still optimal or very close to optimal.

Proof. Subtracting $R(t^*, \mathbf{X}, \mathbf{c}, \mathbf{y}) = \sum_{i=1}^N \mathcal{L}(\hat{S}^{t^*}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i) + \lambda H_{t^*}$ from the two sides of Inequality 13, we have:

$$R(t_{\text{guess}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) - R(t^*, \mathbf{X}, \mathbf{c}, \mathbf{y}) \leq \sum_{i=1}^N \max\left\{\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i) - \hat{S}^{t^*}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i), 0\right\}.$$

If $\mathcal{L}(\hat{S}^T_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i) \leq \hat{S}^{t^*}_{\mathbf{x}_i}(\cdot), \mathbf{x}_i, c_i, y_i)$ then $R(t_{\text{guess}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) - R(t^*, \mathbf{X}, \mathbf{c}, \mathbf{y}) \leq 0$, which means the tree returned is still optimal. □

B.8 Splitting Bounds

When constructing a new child tree $t' = (t'_{\text{fix}}, \delta'_{\text{fix}}, t'_{\text{split}}, \delta'_{\text{split}}, K', H_{t'})$, t'_{split} needs to be determined. Splitting bounds help determine which leaves in t' *cannot be further split* and which leaves *must be further split.*

Theorem B.1. *(Incremental Progress Bound to Determine Splitting). For any optimal tree t^*, any parent node of its leaves must have loss at least $\geq \lambda$ when considered as a leaf.*

Proof. This bound adapts directly from OSRT (Zhang et al., 2023), where the proof can be found. □

Theorem B.2. *(Lower Bound on Incremental Progress). Consider any optimal tree $t^* = \{l_1, l_2, \ldots, l_i, l_{i+1}, \ldots, l_{H_{t^*}}\}$ with H_{t^*} leaves. Let $t' = \{l_1, l_2, \ldots, l_{i-1}, l_{i+2}, \ldots, l_{H_{t^*}, l_j}\}$ be a tree created by deleting a pair of leaves l_i and l_{i+1} in t^* and adding their parent node l_j. The reduction in loss obeys:*

$$\mathcal{L}(l_j, \mathbf{X}, \mathbf{c}, \mathbf{y}) - \mathcal{L}(l_i, \mathbf{X}, \mathbf{c}, \mathbf{y}) - \mathcal{L}(l_{i+1}, \mathbf{X}, \mathbf{c}, \mathbf{y}) \geq \lambda.$$

Proof. This bound adapts directly from OSRT (Zhang et al., 2023), where the proof can be found. □

When constructing new trees, if a leaf has loss less than λ (it fails to meet Theorem B.1), then it cannot be further split. If a pair of leaves in that tree reduce loss from their parent node by less than λ (the leaves fail to meet Theorem B.2), then at least one of this pair of leaves must be further split to search for optimal trees.

B.9 Leaf Bounds

The following bounds on the number of leaves allow us to prune trees whose number of leaves exceed these upper bounds.

Theorem B.3. *(Upper Bound on the Number of Leaves). Let H_t be the number of leaves of tree t and let R^c be the current best objective. For any optimal tree t^* with H_{t^*} leaves, it is true that:*

$$H_{t^*} \leq \min\{\lfloor R_c/\lambda \rfloor, 2^M\}, \tag{41}$$

where M is the number of features.

Proof. This bound adapts directly from OSDT (Hu et al., 2019), where the proof can be found. □

Theorem B.4. *(Parent-specific upper bound on the number of leaves). Let $t = (t_{fix}, \delta_{fix}, t_{split}, \delta_{split}, K, H_t)$ be a tree with child tree $t' = (t'_{fix}, \delta'_{fix}, t'_{split}, \delta'_{split}, K', H_{t'}) \in \sigma(t)$ with $H_{t'}$ leaves. Then:*

$$H_{t'} \leq \min\left\{H_t + \left\lfloor \frac{R_c - \mathcal{L}(t_{\text{fix}}, \mathbf{X}, \mathbf{c}, \mathbf{y}) - \lambda H_t}{\lambda} \right\rfloor, 2^M\right\}.$$

Proof. This bound adapts directly from OSDT (Hu et al., 2019), where the proof can be found. □

B.10 Permutation Bound

Theorem B.5. *(Leaf Permutation Bound). Let π be any permutation of $\{1 \ldots H_t\}$. Let $t = \{l_1, l_2, \ldots, l_{H_t}\}$, $T = \{l_{\pi(1)}, l_{\pi(2)}, \ldots, l_{\pi(H_t)}\}$, that is, the leaves in T are a permutation of the leaves in t. The objective lower bounds of t and T are the same and their child trees correspond to permutations of each other.*

Proof. This bound adapts directly from OSDT (Hu et al., 2019), where the proof can be found. □

This bound avoids duplicate computation of trees with leaf permutation.

B.11 Hierarchical Objective Lower Bound for Sub-trees

Theorem B.6. *(Hierarchical Objective Lower Bound for Sub-trees). Let R^c be the current best objective so far. Let t be a tree such that the root node is split by a feature, where two sub-trees t_{left}, t_{right} are generated with H_{left} leaves for t_{left} and H_{right} leaves for t_{right}. The data captured by the left tree is $(\mathbf{X}_{left}, \mathbf{y}_{left})$ and the data captured by the right tree is $(\mathbf{X}_{right}, \mathbf{y}_{right})$. Then, the objective lower bounds of the left sub-tree and right sub-tree are $b(t_{left}, \mathbf{X}_{left}, \mathbf{c}_{left}, \mathbf{y}_{left})$ and $b(t_{right}, \mathbf{X}_{right}, \mathbf{c}_{right}, \mathbf{y}_{right})$, which obey $R(t_{left}, \mathbf{X}_{left}, \mathbf{c}_{left}, \mathbf{y}_{left}) \geq b(t_{left}, \mathbf{X}_{left}, \mathbf{c}_{left}, \mathbf{y}_{left})$, and $R(t_{right}, \mathbf{X}_{right}, \mathbf{c}_{right}, \mathbf{y}_{right}) \geq b(t_{right}, \mathbf{X}_{right}, \mathbf{c}_{right}, \mathbf{y}_{right})$. If $b(t_{left}, \mathbf{X}_{left}, \mathbf{c}_{left}, \mathbf{y}_{left}) > R^c$ or $b(t_{right}, \mathbf{X}_{right}, \mathbf{c}_{right}, \mathbf{y}_{right}) > R^c$ or $b(t_{left}, \mathbf{X}_{left}, \mathbf{c}_{left}, \mathbf{y}_{left}) + b(t_{right}, \mathbf{X}_{right}, \mathbf{c}_{right}, \mathbf{y}_{right}) > R^c$, then t is not an optimal tree, and none of its child trees are optimal.*

Proof. This bound adapts directly from GOSDT (Lin et al., 2020), where the proof can be found. □

This bound can be applied to any tree, even if the tree is partially constructed. In a partially constructed tree t, if one of its subtrees has objective worse than current best objective R^c, we can prune tree t and all of its child trees without constructing the other subtree.

B.12 Subset Bound

Theorem B.7. *Let t and T to be two trees with the same root node, where t uses feature f_1 to split the root node and T uses feature f_2 to split the root node. Let t_1, t_2 be subtrees of t under its root node, and $(\mathbf{X}_{t_1}, \mathbf{c}_{t_1}, \mathbf{y}_{t_1}), (\mathbf{X}_{t_2}, \mathbf{c}_{t_2}, \mathbf{y}_{t_2})$ be samples captured by t_1 and t_2. Similarly, let T_1, T_2 be subtrees of T under its root node, and $(\mathbf{X}_{T_1}, \mathbf{c}_{T_1}, \mathbf{y}_{T_1}), (\mathbf{X}_{T_2}, \mathbf{c}_{T_2}, \mathbf{y}_{T_2})$ be samples captured by T_1 and T_2. Suppose t_1, t_2 are optimal trees for $(\mathbf{X}_{t_1}, \mathbf{c}_{t_1}, \mathbf{y}_{t_1}), (\mathbf{X}_{t_2}, \mathbf{c}_{t_2}, \mathbf{y}_{t_2})$ respectively, and T_1, T_2 are optimal trees for $(\mathbf{X}_{T_1}, \mathbf{c}_{T_1}, \mathbf{y}_{T_1}), (\mathbf{X}_{T_2}, \mathbf{c}_{T_2}, \mathbf{y}_{T_2})$ respectively. If $R(t_1, \mathbf{X}_{t_1}, \mathbf{c}_{t_1}, \mathbf{y}_{t_1}) \leq R(T_1, \mathbf{X}_{T_1}, \mathbf{c}_{T_1}, \mathbf{y}_{T_1})$ and $(\mathbf{X}_{t_2}, \mathbf{c}_{t_2}, \mathbf{y}_{t_2}) \subset (\mathbf{X}_{T_2}, \mathbf{c}_{T_2}, \mathbf{y}_{T_2})$, then $R(t, \mathbf{X}, \mathbf{c}, \mathbf{y}) \leq R(T, \mathbf{X}, \mathbf{c}, \mathbf{y})$.*

Proof. This bound adapts directly from GOSDT (Lin et al., 2020), where the proof can be found. □

Similar to Theorem B.6, this bound ensures that we can safely prune a partially constructed tree without harming optimality. It checks whether subtree t_1 has a better objective than T_1, despite handling more data.

C Metrics

We use several metrics to evaluate the quality of our survival trees. Note that some of the following metrics are not additive, which means that if we want to optimize them then we must construct the entire survival tree. Therefore we use these metrics only in the evaluation stage. We found that even though we optimize the IBS, the optimal survival trees often perform better on other metrics as well, which we show in Section 3.3 and Appendix G.

C.1 Integrated Brier Score Ratio

To be consistent with other metrics (the higher the better), we used Integrated Brier Score Ratio (IBS Ratio) to report IBS loss in all experimental sections. The IBS Ratio of tree t is defined as

$$\text{IBS Ratio}(t) = 1 - \frac{\mathcal{L}(t, \mathbf{X}, \mathbf{c}, \mathbf{y})}{\mathcal{L}(t_0, \mathbf{X}, \mathbf{c}, \mathbf{y})} \tag{42}$$

where $\mathcal{L}$ is the IBS loss defined in Equation 1 and t_0 is a single node containing all samples (equivalently a KM estimator ignoring all features). This ratio is very similar to the R^2 in regression.

C.2 Concordance Indices

Harrell Jr et al. (1996) adapted the Concordance Statistic from logistic regression into survival analysis. Given a pair of samples, i and j, a good survival model should predict a lower survival probability for the sample that is closer to the actual death time. A *concordant* pair is defined as a pair of samples that satisfy this expectation

(e.g., $\hat{S}_{\mathbf{x}_i}(y_i) < \hat{S}_{\mathbf{x}_j}(y_i), y_i < y_j$). When the predicted survival probabilities are tied, we consider them have 0.5 expectation to be concordant due to random arrangement. A pair is *comparable* either when both samples are observed (not censored) or when one sample died before the other was censored. The concordance index is defined as the number of concordant pairs divided by the number of comparable pairs. Formally, Harrell's C-index is given by:

$$C_H = \frac{\sum_i \sum_j c_i \cdot \mathbf{1}_{y_i<y_j} \cdot \left(\mathbf{1}_{\hat{S}_{\mathbf{x}_i}(y_i)<\hat{S}_{\mathbf{x}_j}(y_i)} + 0.5 \cdot \mathbf{1}_{\hat{S}_{\mathbf{x}_i}(y_i)=\hat{S}_{\mathbf{x}_j}(y_i)}\right)}{\sum_i \sum_j c_i \cdot \mathbf{1}_{y_i<y_j}}. \tag{43}$$

Since the predicted survival function is time-dependent, we use the earlier observation time ($\min\{y_i, y_j\}$) when evaluating whether a comparable pair is concordant.
However, Harrell's C-index is biased when there are a large number of censored samples. Uno et al. (2011) applied the inverse probability of censoring weights to Harrell's C-index to make it robust to a high degree of censoring. Uno's C-index is given by:

$$C_U = \frac{\sum_i \sum_j c_i \cdot \hat{G}^{-2}(y_i) \cdot \mathbf{1}_{y_i<y_j} \cdot \left(\mathbf{1}_{\hat{S}_{\mathbf{x}_i}(y_i)<\hat{S}_{\mathbf{x}_j}(y_i)} + 0.5 \cdot \mathbf{1}_{\hat{S}_{\mathbf{x}_i}(y_i)=\hat{S}_{\mathbf{x}_j}(y_i)}\right)}{\sum_i \sum_j c_i \cdot \hat{G}^{-2}(y_i) \cdot \mathbf{1}_{y_i<y_j}}. \tag{44}$$

Remember, $\hat{G}(\cdot)$ is the Kaplan–Meier estimate of the censoring distribution $\mathbf{c}$ under the assumption that it is independent of the covariates. Uno et al. (2011) claimed that Uno's C-index is 'quite robust even when the censoring is dependent on the covariates.'

C.3 Cumulative-Dynamic-AUC

The receiver operating characteristic curve (ROC) plots the the true positive rate versus the false positive rate in binary classification tasks. The area under the ROC curve (AUC) can be extended to survival analysis: given a time y, the true positives are samples that died before or at time y, e.g., $c_i = 1, y_i < y$ (cumulative cases), and true negatives are samples that are still alive at time y, e.g., $y_j > y$ (dynamic controls).

As we change the threshold of predicted survival function, τ, to classify samples into cumulative cases and dynamic control cases, the true positive and false positive rates also change, which gives a time-dependent ROC curve. The time-dependent ROC curve measures how well a survival model separates cumulative cases from dynamic control cases at time y.

The specificity at time t is given by the proportion of the samples with predicted survival function greater than or equal to τ:

$$\widehat{Sp}(\tau, y) = \frac{\sum_i \mathbf{1}_{y_i>y} \cdot \mathbf{1}_{\hat{S}(y|\mathbf{x}_i)\geq\tau}}{\sum_i \mathbf{1}_{y_i>y}}. \tag{45}$$

Uno et al. (2007) and Hung and Chiang (2010) proposed to perform IPCW when calculating the sensitivity:

$$\widehat{Se}(\tau, y) = \frac{\sum_i c_i \cdot \hat{G}^{-1}(y_i) \cdot \mathbf{1}_{y_i\leq y} \cdot \mathbf{1}_{\hat{S}(y|\mathbf{x}_i)<\tau}}{\sum_i c_i \cdot \hat{G}^{-1}(y_i) \cdot \mathbf{1}_{y_i\leq y}} \tag{46}$$

which gives the AUC at time y as:

$$\widehat{AUC}(y) = \frac{\sum_i \sum_j c_i \cdot \hat{G}^{-1}(y_i) \cdot \mathbf{1}_{y_i\leq y} \cdot \mathbf{1}_{y_j>y} \cdot \mathbf{1}_{\hat{S}(y|\mathbf{x}_i)<\hat{S}(y|\mathbf{x}_j)}}{\left(\sum_i c_i \cdot \hat{G}^{-1}(y_i) \cdot \mathbf{1}_{y_i\leq y}\right)\left(\sum_j \mathbf{1}_{y_j>y}\right)}. \tag{47}$$

Lambert and Chevret (2016) proposed to restrict a time-dependent weighted AUC estimator to a fixed time interval, which summarizes the mean AUC over the time range. Here we compute the time range from $y_{\min} = \min\{y_i\}_{i=1}^N$ to $y_{\max} = \max\{y_i\}_{i=1}^N$:

$$\overline{AUC} = \frac{1}{\hat{S}(y_{\min}) - \hat{S}(y_{\max})} \int_{y_{\min}}^{y_{\max}} \widehat{AUC}(y) d\hat{S}(y). \tag{48}$$

D Experiment Setup Details

D.1 Datasets

We used 17 datasets of which 11 are real-world survival data and 6 are synthetic data (manually censoring on real-world regression datasets).

Real-world survival data:

- **Aids**:AIDS Clinical Trial (Lemeshow et al., 2011). The event is AIDS-defining event.
- **Aids_death**: AIDS Clinical Trial (Lemeshow et al., 2011). The event is death.
- **Churn**: The task of predicting when a software as a service (SaaS) company's customers are likely to stop their monthly subscription. The event is end of subscription. This dataset is from pysurvival (Fotso et al., 2019).
- **Credit**: Lenders need to predict when a borrower will repay a loan. The event is loan repayment. This dataset is from pysurvival (Fotso et al., 2019), adapted from UCI Machine Learning Repository (Dua and Graff, 2017).
- **Employee**: The task of predicting when an employee will quit. The event is an employee's leaving. This dataset is from pysurvival (Fotso et al., 2019).
- **Maintenance**: The task of predicting when equipment failure will occur. The event is machine failure. This dataset is from pysurvival (Fotso et al., 2019).
- **Veterans**: Veterans' Administration Lung Cancer Trial (Kalbfleisch and Prentice, 2011). The event is death.
- **Whas500**: Worcester Heart Attack Study (Lemeshow et al., 2011). The event is death.
- **Flchain**: Use of nonclonal serum immunoglobulin free light chains to predict overall survival in the general population (Dispenzieri et al., 2012). The event is death.
- **Gbsg2**: German Breast Cancer Study Group 2 (Schumacher et al., 1994). The event is recurrence free survival.
- **Uissurv**: UMASS Aids Research Unit IMPACT Study (Lemeshow et al., 2011). The event is returning to drugs.

Synthetic data: We randomly censored samples with a censoring rate of 20%.

- **Airfoil**: Airfoil self-noise. This dataset is from UCI Machine Learning Repository (Dua and Graff, 2017).
- **Insurance**: Medical cost personal. This dataset is from a Kaggle competition (Choi, 2018).
- **Real-estate**: Real estate valuation. This dataset is from the UCI Machine Learning Repository (Dua and Graff, 2017).
- **Sync**: Synchronous machine. This dataset is from the UCI Machine Learning Repository (Dua and Graff, 2017).
- **Servo**: The rise time of a servomechanism. This dataset is from the UCI Machine Learning Repository (Dua and Graff, 2017).
- **Household**: Individual household electric power consumption. This dataset is from the UCI Machine Learning Repository (Dua and Graff, 2017).

D.2 Data Preprocessing

First, we removed all observations with missing values. Because we are performing hundreds of experiments consuming months of compute time, for each dataset, we categorized some continuous features into 4 equal-width partitions and used one-hot encoding to convert all the features into binary features:

Aids: each of *age, cd4, priorzdv* were discretized into 4 categories.
Aids_death: each of *age, cd4, priorzdv* were discretized into 4 categories.
Churn: each of *articles_viewed, minutes_customer_support, smartphone_notifications_viewed, social_media_ads_viewed, and marketing_emails_clicked* were discretized into 4 categories.
Credit: each of *amount, installment_rate, present_residence, age, number_of_credits, andpeople_liable* were discretized into 4 categories.
Employee: feature *average_montly_hours* was discretized into 4 categories.
Maintenance: each of *pressureInd, moistureInd, and temperatureInd* were discretized into 4 categories.
Veterans: each of *Age_in_years, Karnofsky_score, Months_from_Diagnosis* were discretized into 4 categories.
Whas500: each of *age, bmi, diasbp, hr, los, sysbp* were discretized into 4 categories.
Flchain: each of *age, creatinine, kappa, lambda* were discretized into 4 categories.
Gbsg2: each of *age, estrec, pnodes, progrec, tsize* were discretized into 4 categories.

Uissurv: each of *age, beck, ndrugtx, los* were discretized into 4 categories.
Airfoil: We discretized each of the features *frequency, angle of attack, suction side displacement thickness* into 4 categories.
Insurance: We discretized each of *age, bmi* into 4 categories.
Real-estate: We discretized each continuous feature into 4 categories.
Servo: We directly use this dataset that only contains categorical features.
Sync: We discretized each feature into 4 categories.
Household: We transformed the *Date* feature into *Month*, *Time* into *Hour*. Then we discretized each of *Month, Hour, Global_reactive_power, Voltage, Global_intensity* into 4 categories.
Table 2 summarizes all the datasets after preprocessing.

Dataset	Samples	Orig. Features	Encoded Binary Features	Event (Censoring Rate)	Time to Event
churn	2000	12	42	churned (53.4%)	months active
credit	1000	19	56	fully_repaid (30%)	duration
employee	15000	7	27	left (77.2%)	time spend company
maintenance	1000	5	14	broken (60.3%)	lifetime
aids	1151	11	25	aids (91.7%)	time
aids_death	1151	11	25	death (97.7%)	time
flchain	7478	9	47	death (72.5%)	time
gbsg2	686	8	19	recurrence free survival (56.4%)	time
whas500	500	14	26	death (57%)	time
veterans	137	6	14	death (6.6%)	time
uissurv	628	12	26	censoring (19.1%)	time
airfoil	1503	5	17	censoring (20%)	scaled sound pressure level
insurance	1338	6	16	censoring (20%)	charges
real-estate	414	6	18	censoring (20%)	house price of unit area
servo	167	4	15	censoring (20%)	class
sync	557	4	12	censoring (20%)	"If"
household	2,049,280	5	15	censoring (20%, 50%, 80%)	global active power

Table 2: Datasets Summary.

D.3 Experiment Platform

We ran all experiments on a 48-core TensorEX TS2-673917-DPN Intel Xeon Gold 6226 Processor, 2.7Ghz with 768GB RAM. We set a time limit of 60 minutes and a memory limit of 200GB. All algorithms ran single-threaded.

D.4 Software Packages

Recursive Partitioning and Regression Trees (RPART): CRAN package of Rpart, version 4.1.19 (https://cran.r-project.org/web/packages/rpart/index.html).
Conditional Inference Trees (CTree): CRAN package of partykit, version 1.2-16 (https://cran.r-project.org/web/packages/partykit/index.html).
Scikit-survival(SkSurv): scikit-survival version-0.20.0 (https://scikit-survival.readthedocs.io/en/stable/index.html).

E Experiments: Optimality

Collection and Setup: We ran this experiment on 8 datasets: *aids, aids_ death, uissurv, gbsg2, churn, maintenance, veterans, whas500*. We trained models on the entire dataset to measure time to convergence/optimality. For each dataset, we ran algorithms with different configurations:

- **CTree**: We ran this algorithm with 8 different configurations: depth limit, d, ranging from 2 to 9, and a corresponding maximum leaf limit 2^d. All other parameters were set to the default.
- **SkSurv**: We ran this algorithm with 8 different configurations: depth limit, d, ranging from 2 to 9, and a corresponding maximum leaf limit 2^d. The random state was set to 2023 and all other parameters were set to the default.
- **RPART**: We ran this algorithm with 8×12 different configurations: depth limits ranging from 2 to 9, and 12 different regularization coefficients (0.1, 0.05, 0.025, 0.01, 0.005, 0.0025, 0.001, 0.0005, 0.00025, 0.0001, 0.00005, 0.00001). All other parameters were set to the default.
- **OSST** (our method): We ran this algorithm with 8×10 different configurations: depth limits ranging from 2 to 9, and 12 different regularization coefficients (0.1, 0.05, 0.01, 0.005, 0.0025, 0.001, 0.0005, 0.00025, 0.0001, 0.00005, 0.000025, 0.00001). The minimum sample required in each leaf was set to 7, which is consistent with other three methods' default value.

Calculations: For each combination of dataset and depth limit, we drew one plot (training loss against number of leaves). Under the same depth limit, runs of one algorithm with different regularization coefficients may result in trees with same number of leaves. In this case, we plotted the median loss of those trees and showed the best and worst loss among these trees as lower and upper error values respectively. These plots do not display trees where the number of leaves exceeds 30, as these trees are more likely to be overfitted and hard to interpret. A single root node is excluded from the plots as well, as it is not meaningful.

Results: Figures 7 to 14 show that OSST consistently produces optimal trees that achieve the best IBS score, which means they minimize the objective defined in Equation 4. OSST, which produces provably optimal trees, defines a frontier between training loss and the number of leaves. These plots also show how far away other methods' objectives are from the optimal solution. *OSST make it possible to quantify how far other methods are from the optimal solution.* We also noticed that the plots from depth 7 to 9 look similar or identical; when the regularization coefficient is too small or the depth limit is too large, OSST produces complex and overfitted trees with more than 30 leaves, which we excluded from the plots.

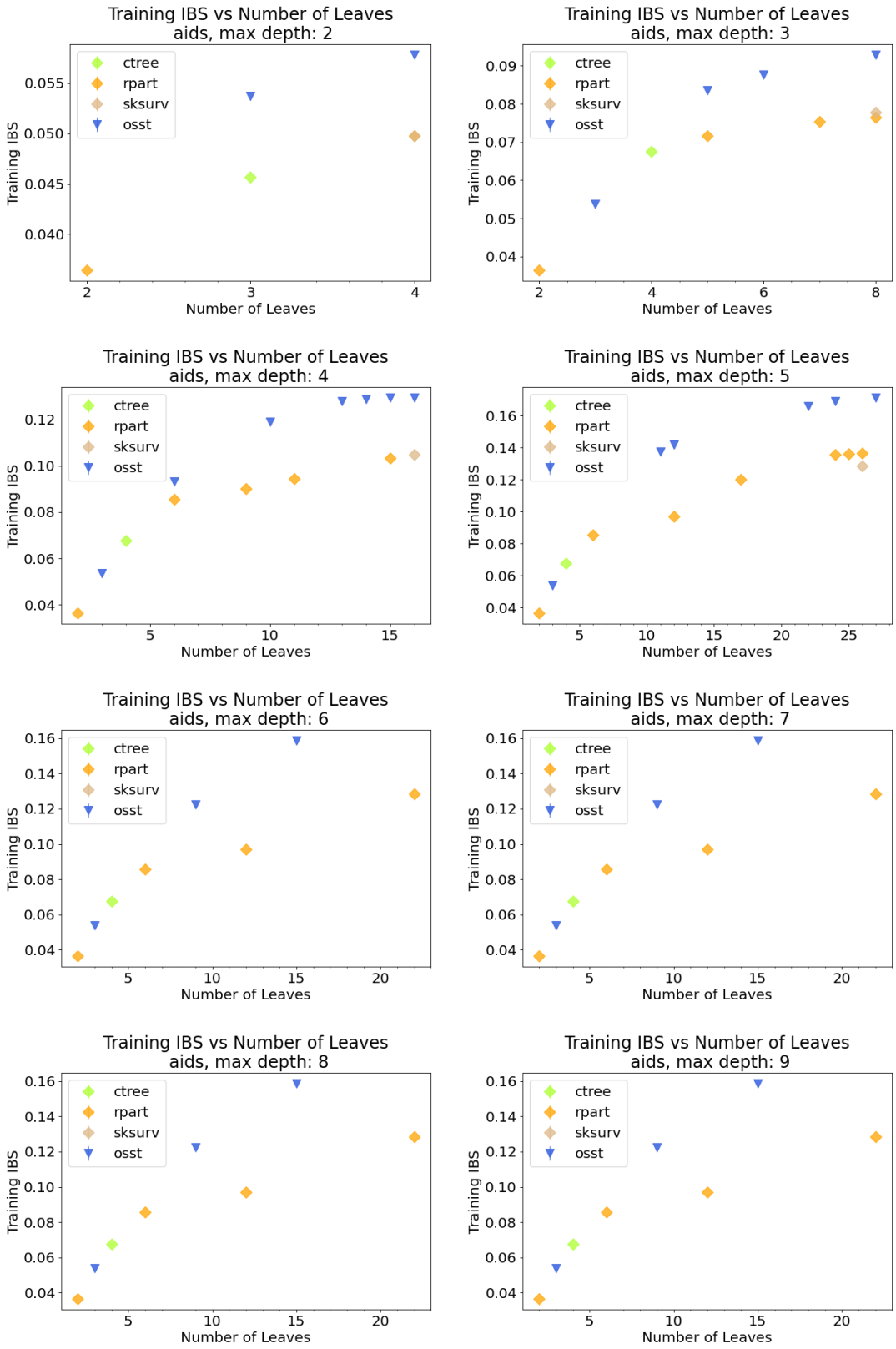

Figure 7: Training IBS achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: aids.

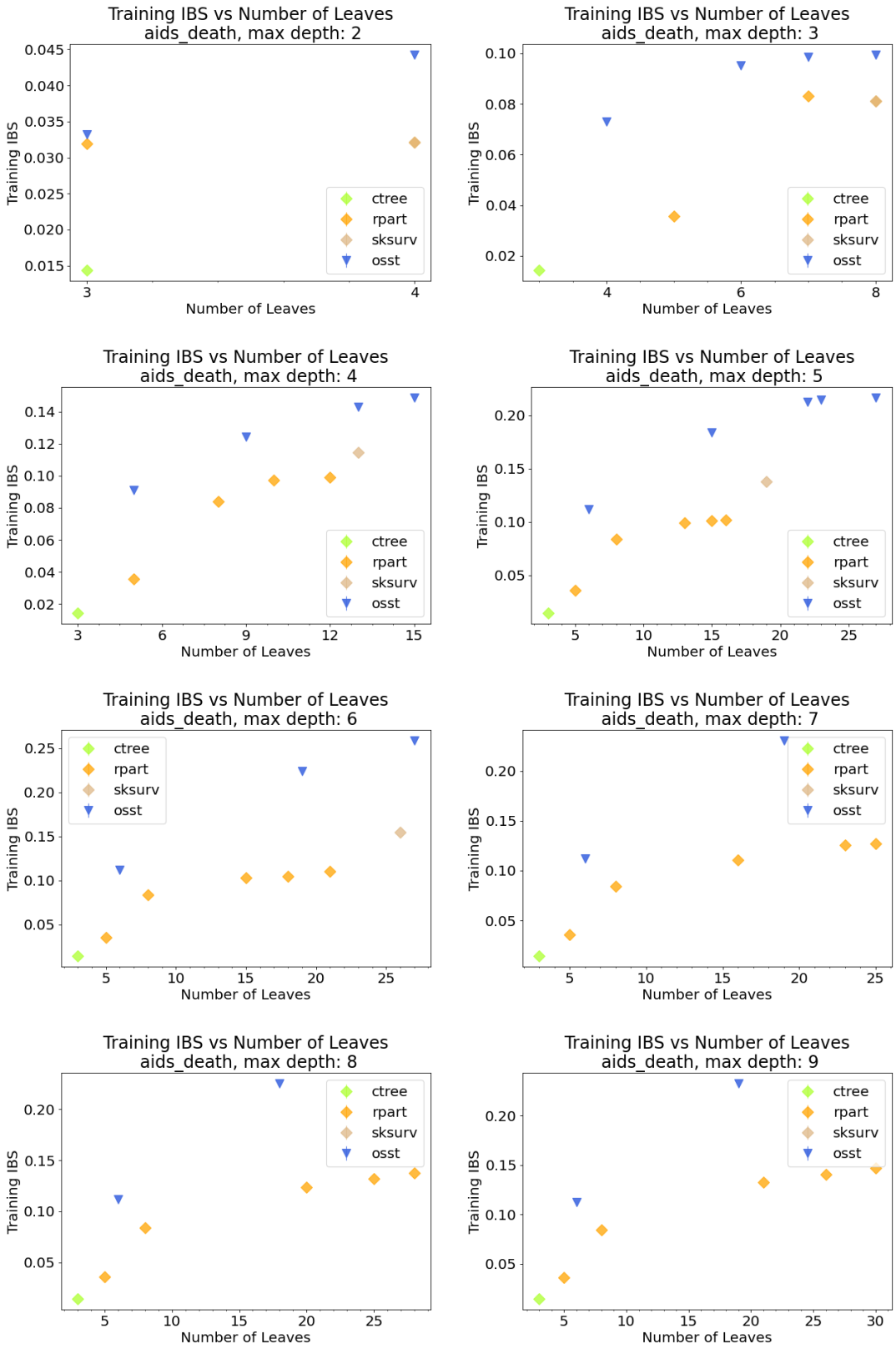

Figure 8: Training IBS achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: aids_death.

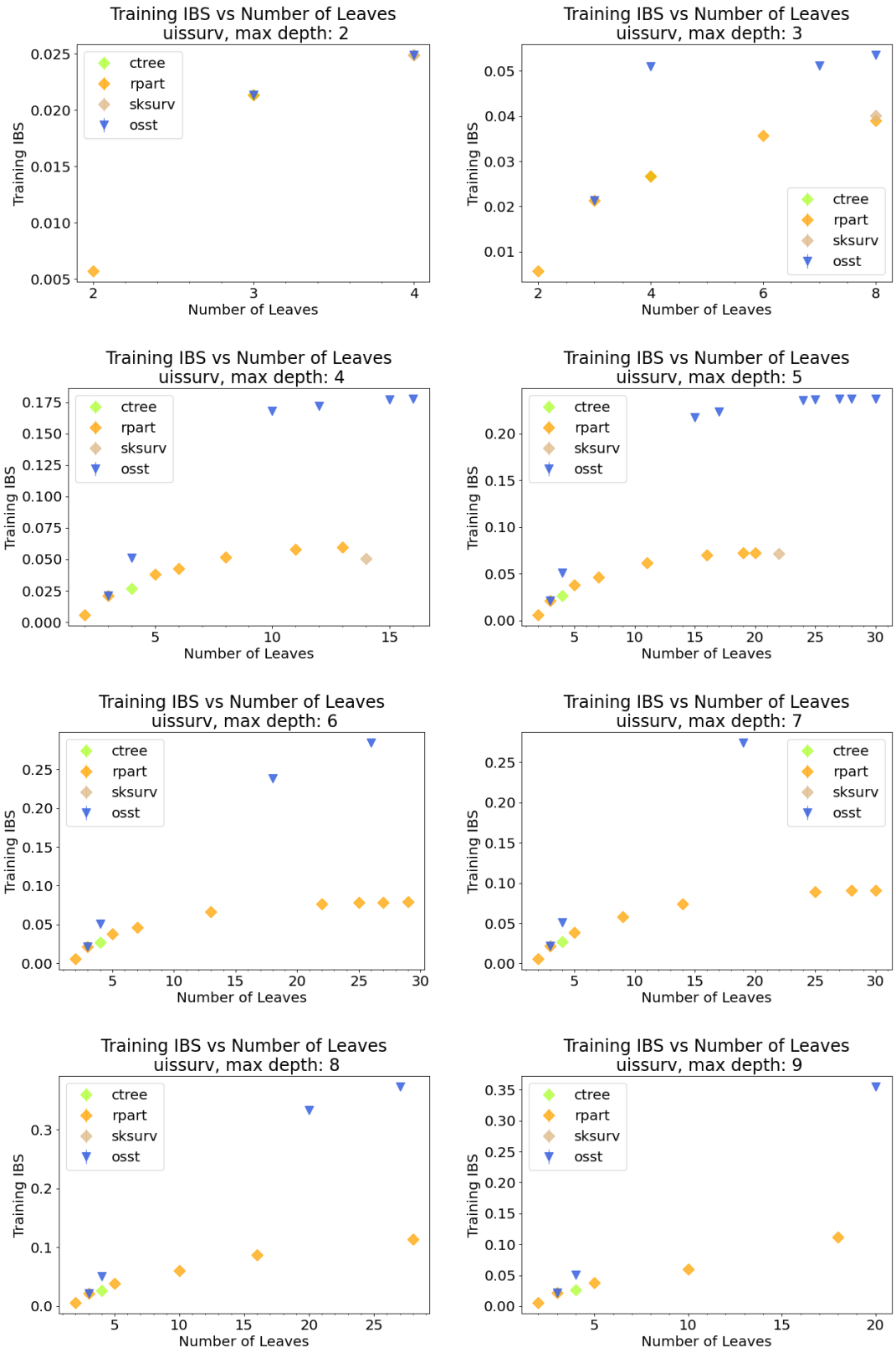

Figure 9: Training IBS achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: uissurv.

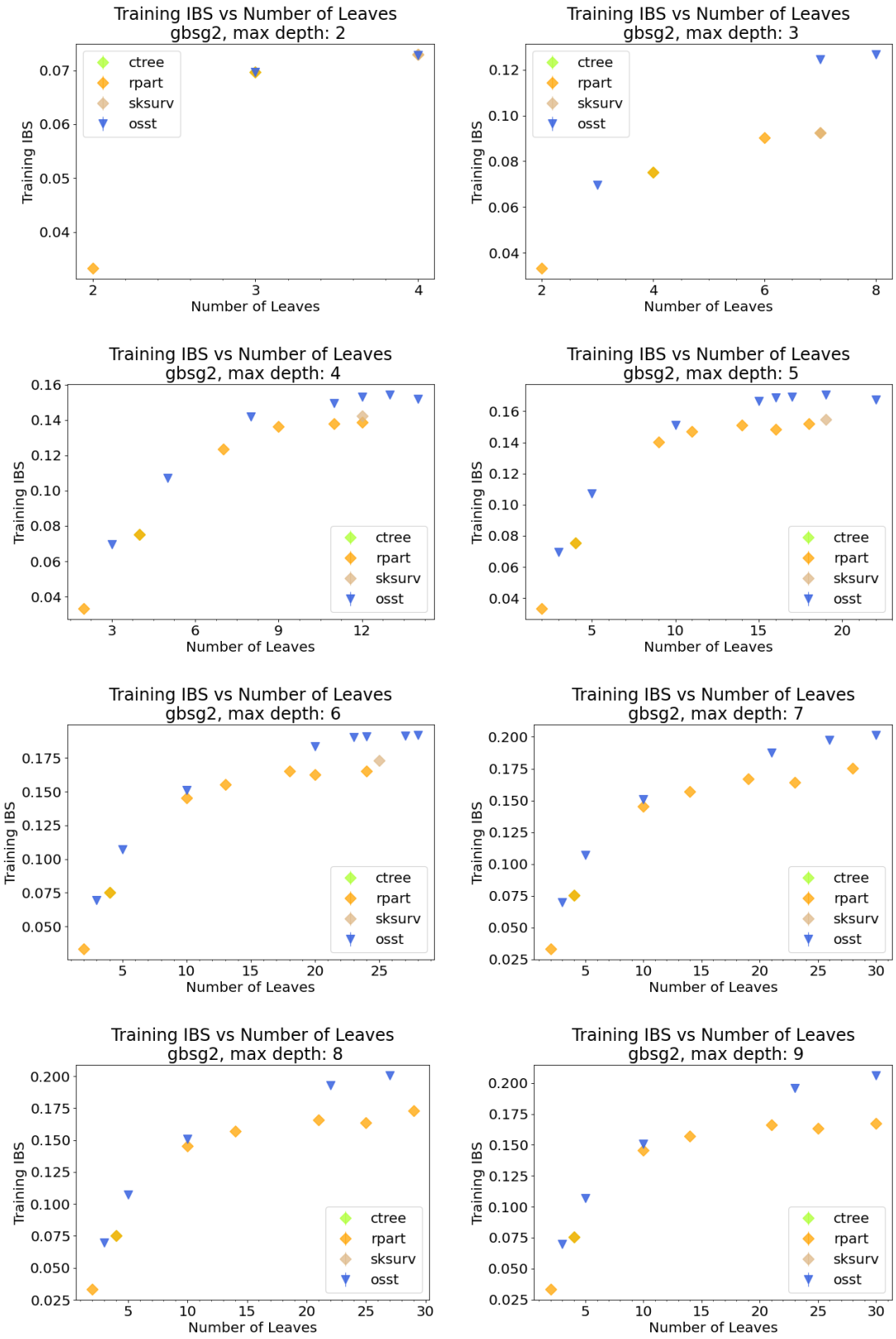

Figure 10: Training IBS achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: gbsg2.

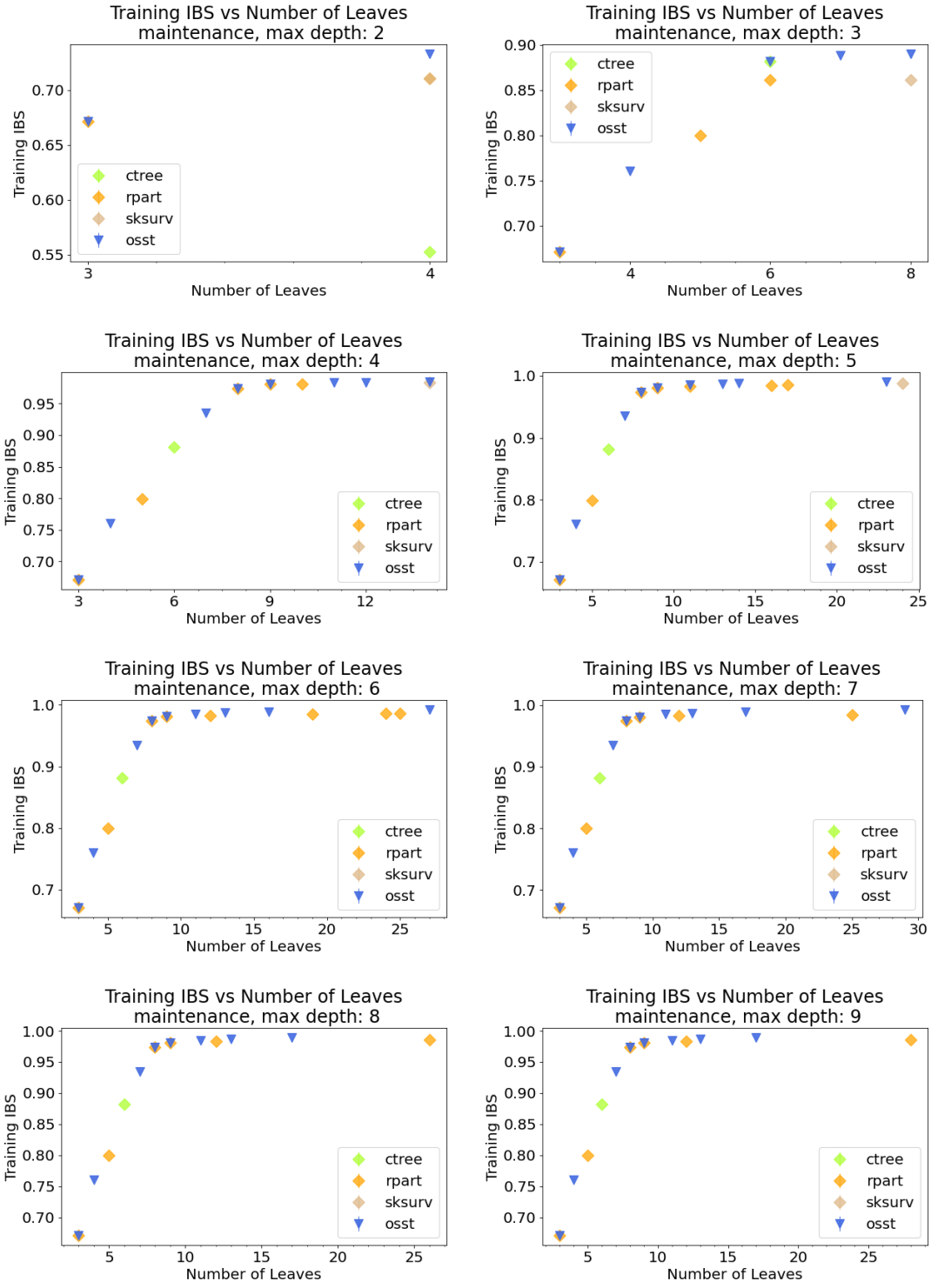

Figure 11: Training IBS achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: maintenance.

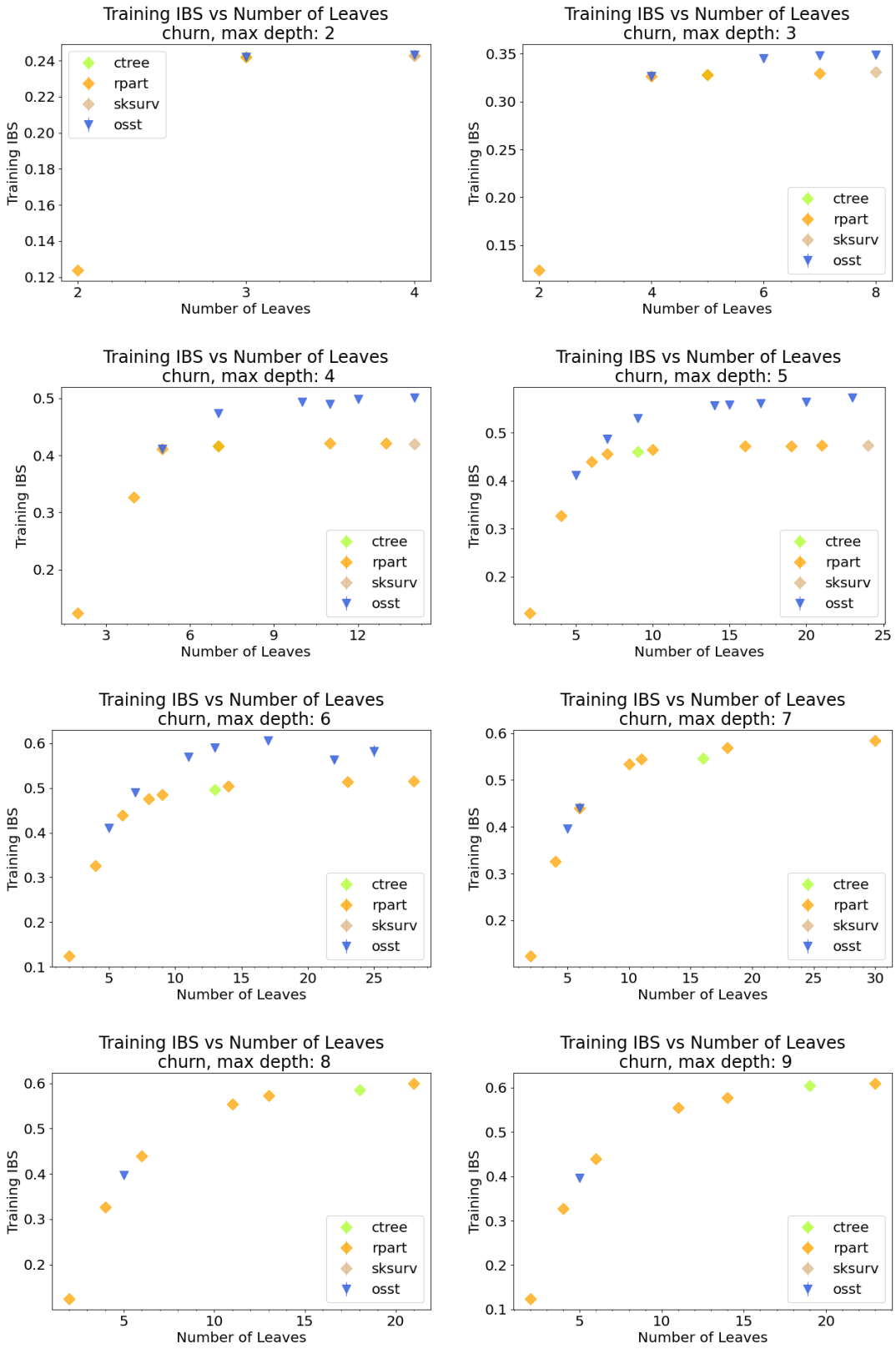

Figure 12: Training IBS achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: churn.

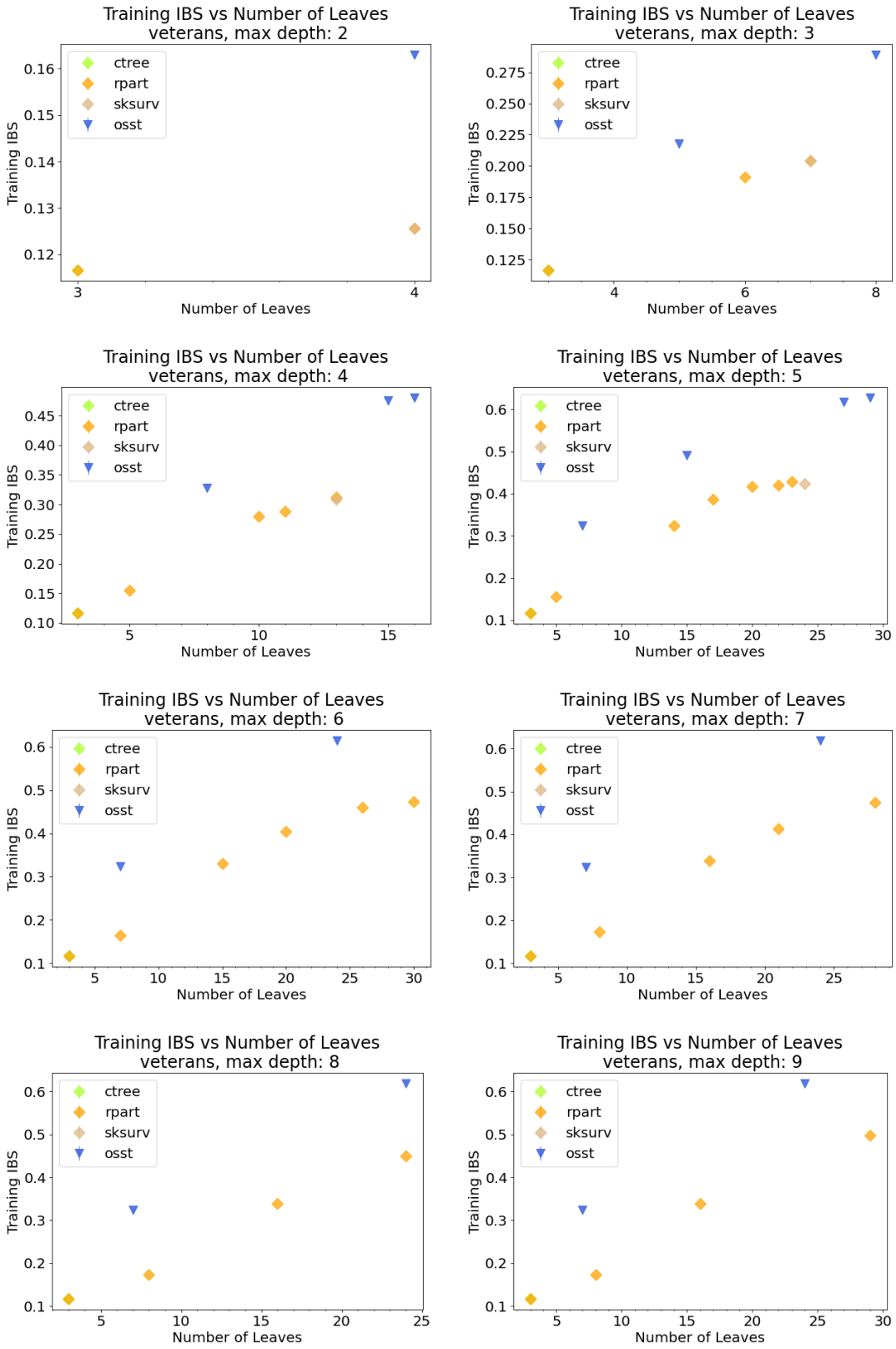

Figure 13: Training IBS achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: veterans.

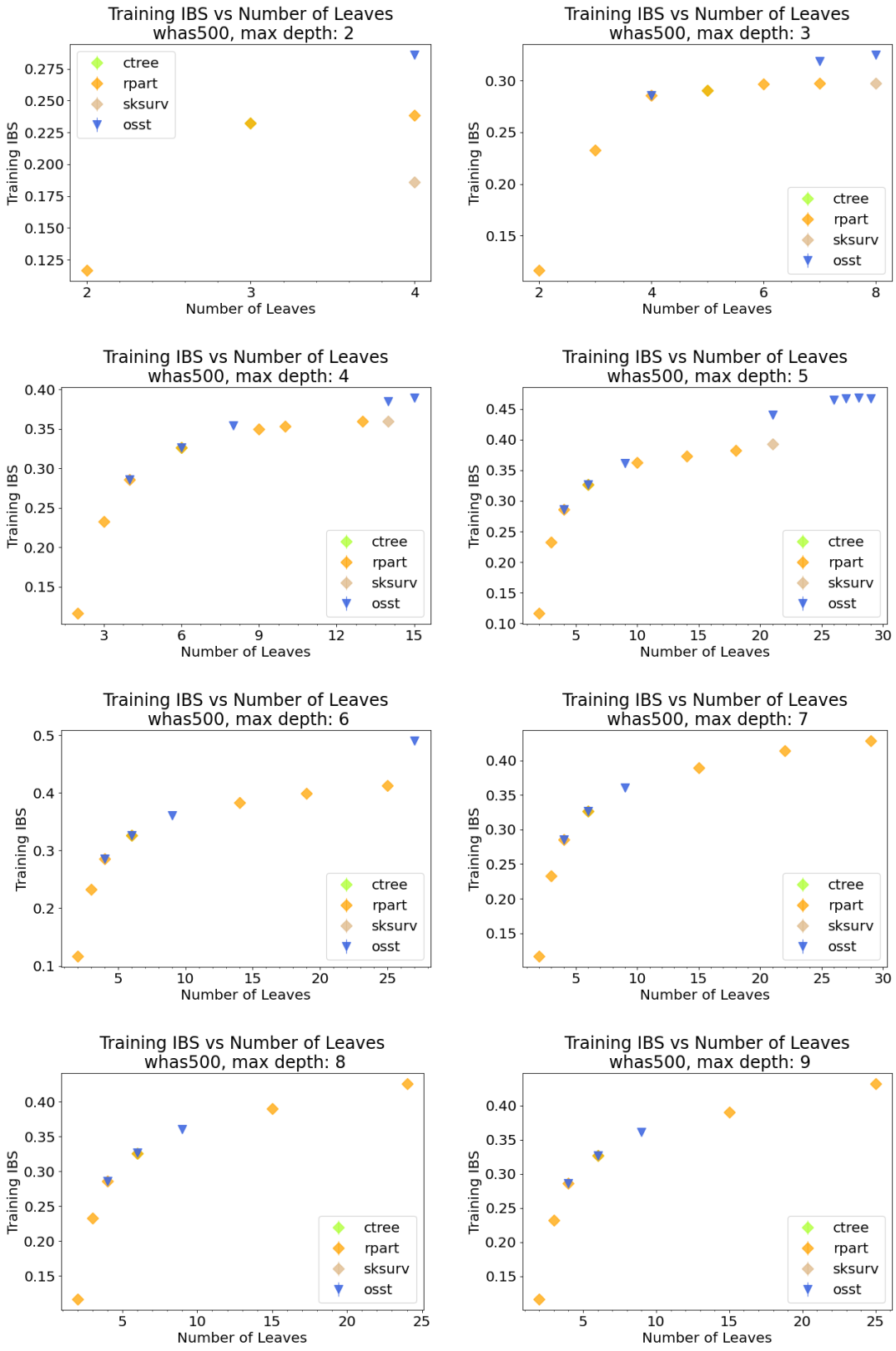

Figure 14: Training IBS achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: whas500.

F Experiments: Generalization

Collection and Setup: We ran 5-fold cross-validation on the 5 datasets: *churn, credit, employee, maintenance, and servo*. The time limit was set to 60 minutes. For each dataset, we ran algorithms with different configurations:

- **CTree**: We ran this algorithm with 4 different configurations: depth limit, d, ranging from 2 to 5, and a corresponding maximum leaf limit 2^d. All other parameters were set to the default.
- **SkSurv**: We ran this algorithm with 4 different configurations: depth limit, d, ranging from 2 to 5, and a corresponding maximum leaf limit 2^d. The random state was set to 2023 and all other parameters were set to the default.
- **RPART**: We ran this algorithm with 4×12 different configurations: depth limits ranging from 2 to 5, and 12 different regularization coefficients (0.1, 0.05, 0.025, 0.01, 0.005, 0.0025, 0.001, 0.0005, 0.00025, 0.0001, 0.00005, 0.00001). All other parameters were set to the default.
- **OSST** (our method): We ran this algorithm with 4×12 different configurations: depth limits ranging from 2 to 5, and 12 different regularization coefficients (0.1, 0.05, 0.01, 0.005, 0.0025, 0.001, 0.0005, 0.00025, 0.0001, 0.00005, 0.000025, 0.00001).

Calculations: We drew one plot per dataset and depth. For each combination of regularization coefficient and algorithm in the same plot, we produced a set of up to 5 trees, depending on if the runs exceeded the time limit. We summarized the measurements of training loss, testing loss and number of leaves across the set of up to 5 trees by plotting the median, showing the 25th and 75th quantiles for number of leaves, training/testing error in the set as the lower and upper error values respectively. As in Section E, trees with more than 30 leaves and single root node were excluded.

Results: Figures 15 to 19 show that OSST trees produce the best testing score among all the survival trees. If an optimal tree significantly outperforms other sub-optimal trees in terms of training performance, it is also outperformed in testing (e.g., *churn* depth 4 and 5; *credit* depth 4), otherwise the difference in testing score becomes insignificant due to generalization error. Note that larger trees start overfitting when depth is greater than 4 or 5 (e.g., *servo* depth 5), and sparse trees tend to have better generalization.

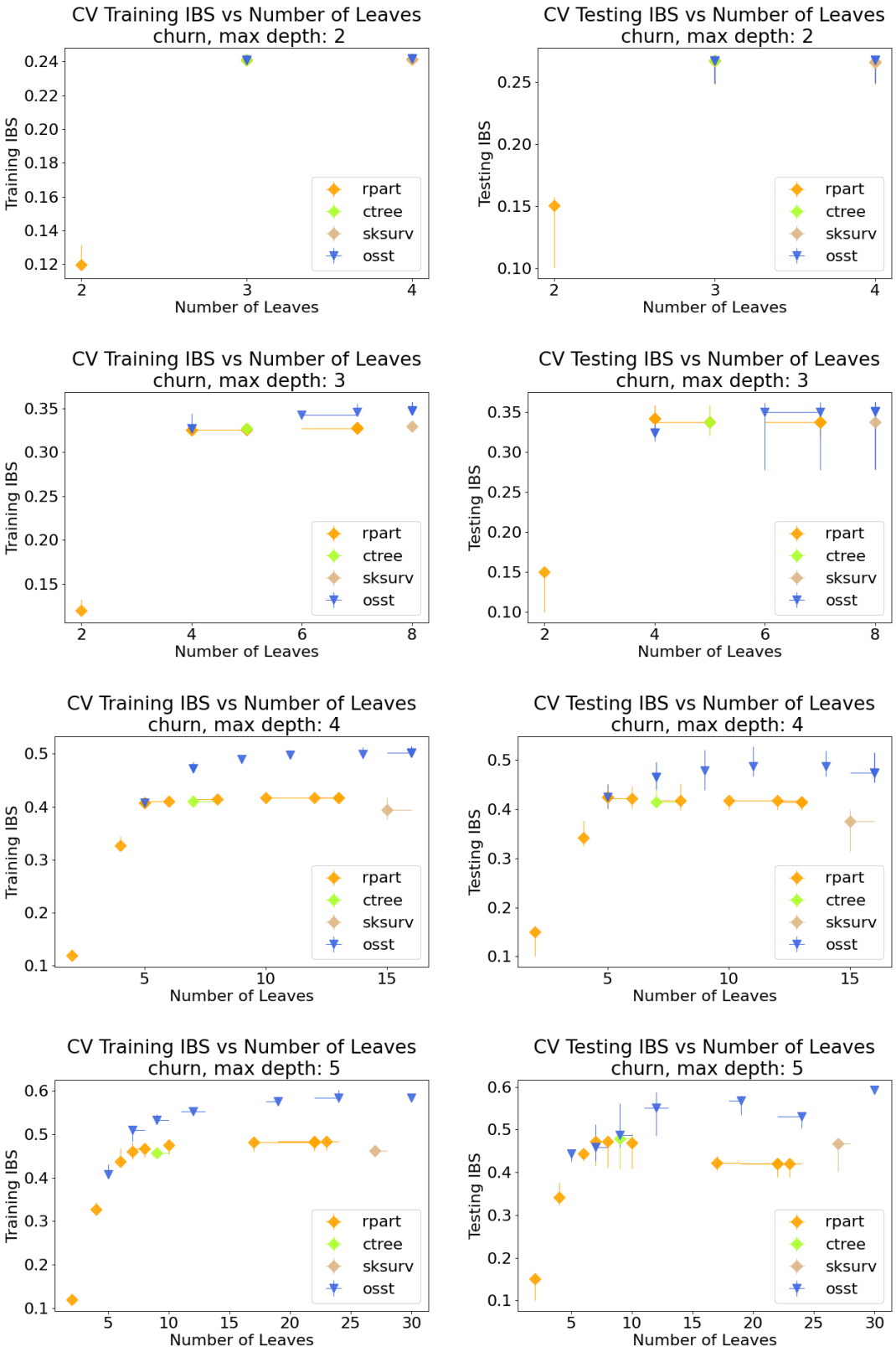

Figure 15: 5-fold CV of CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: churn.

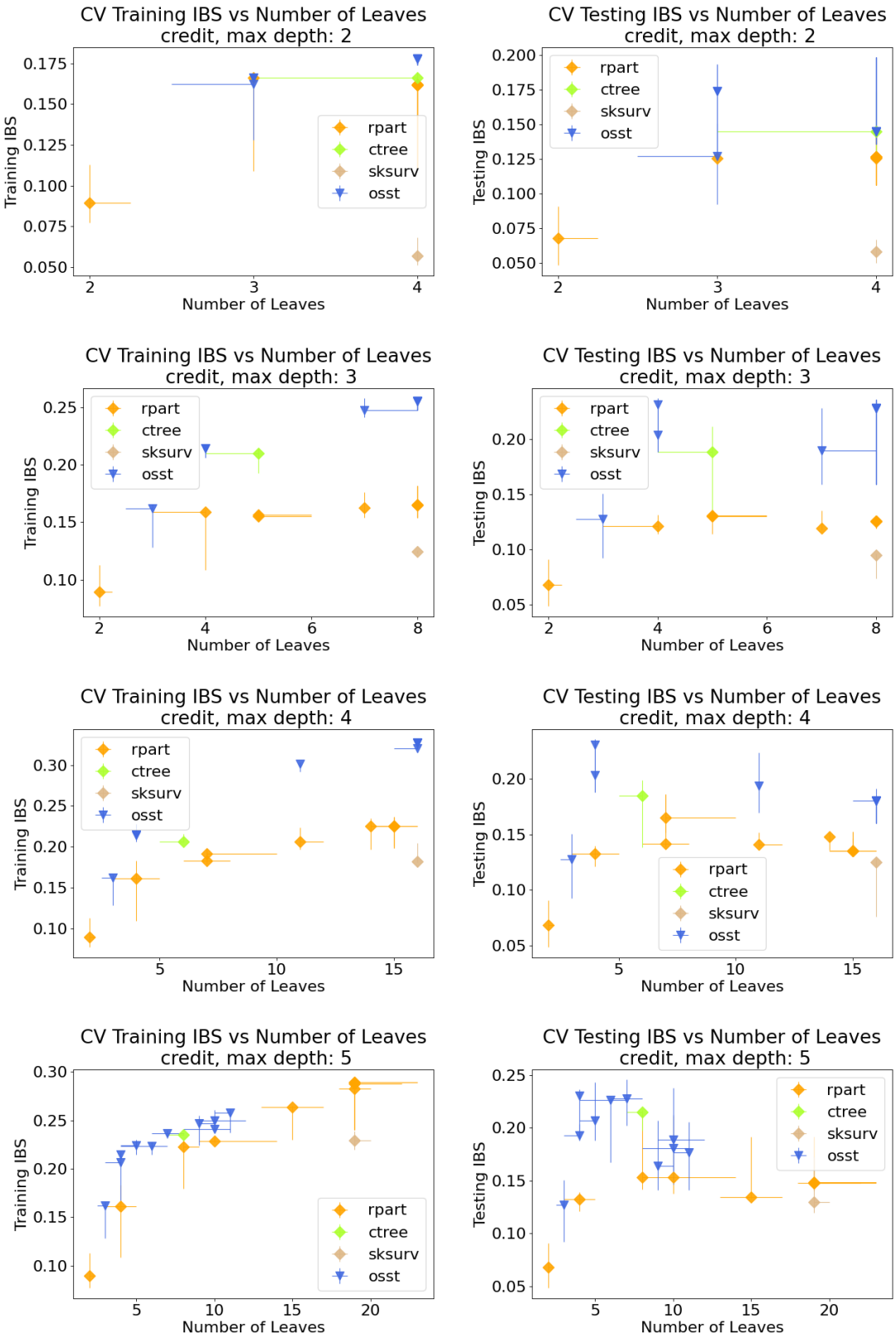

Figure 16: 5-fold CV of CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: credit.

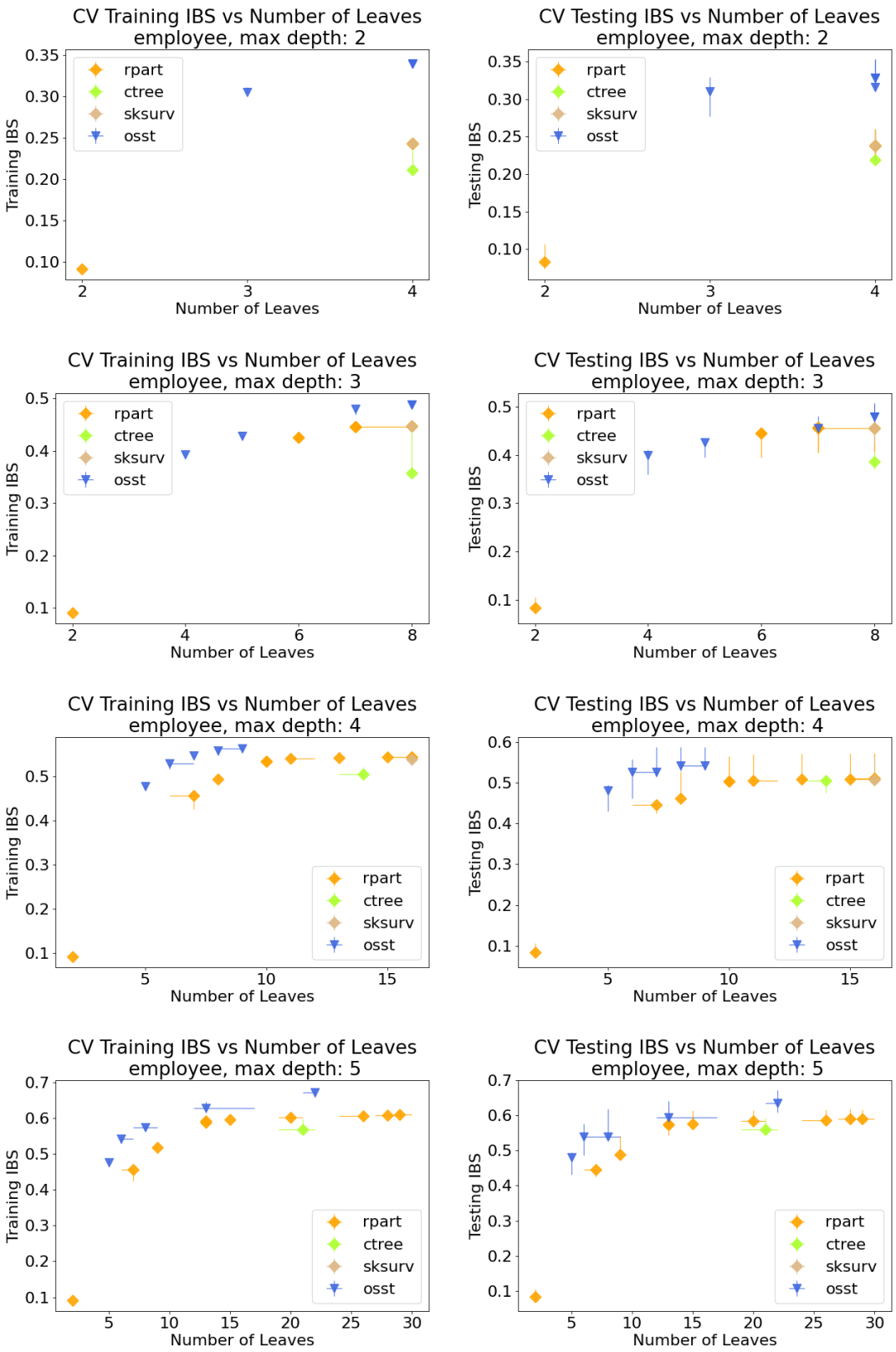

Figure 17: 5-fold CV of CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: employee.

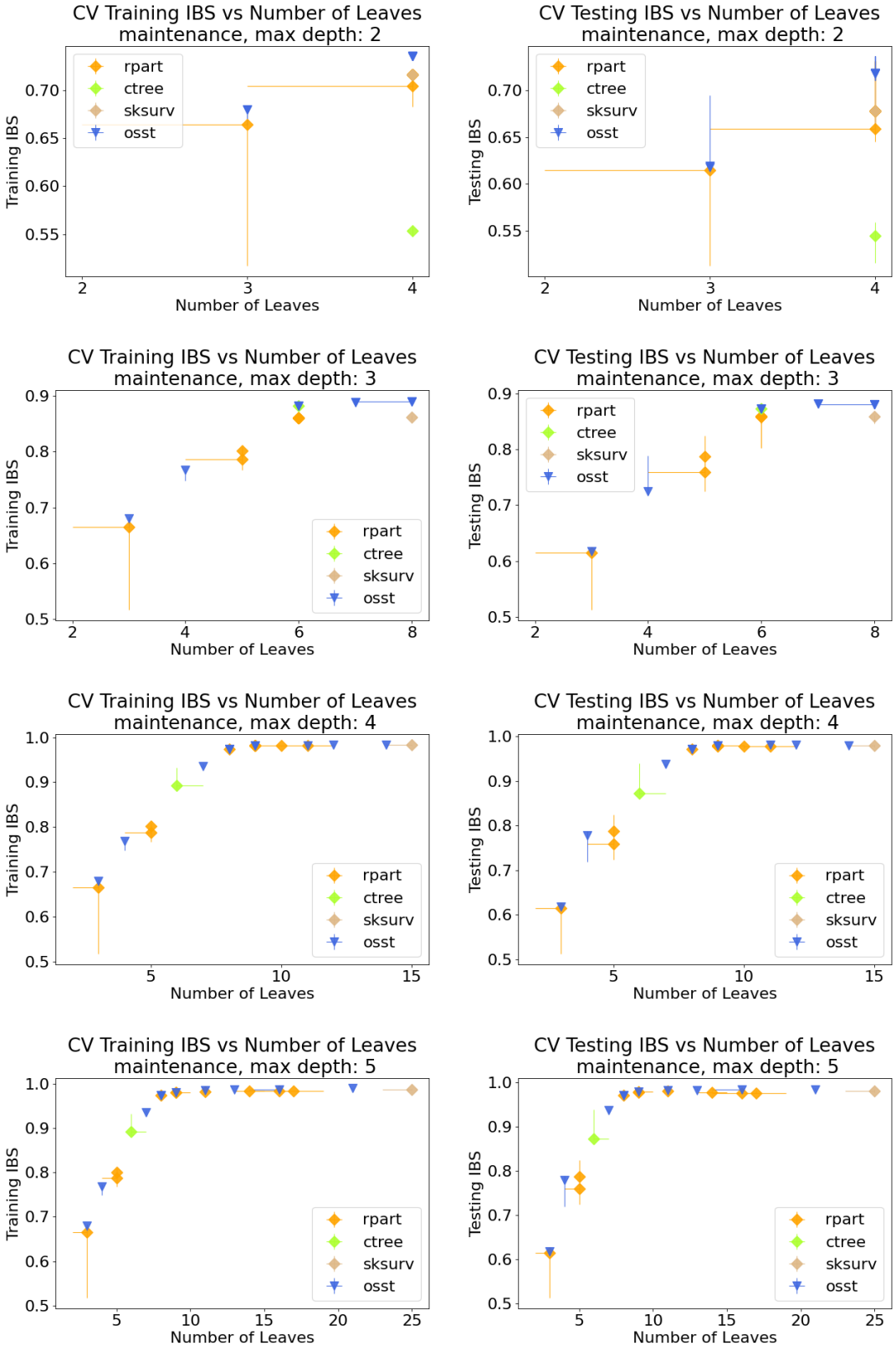

Figure 18: 5-fold CV of CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: maintenance.

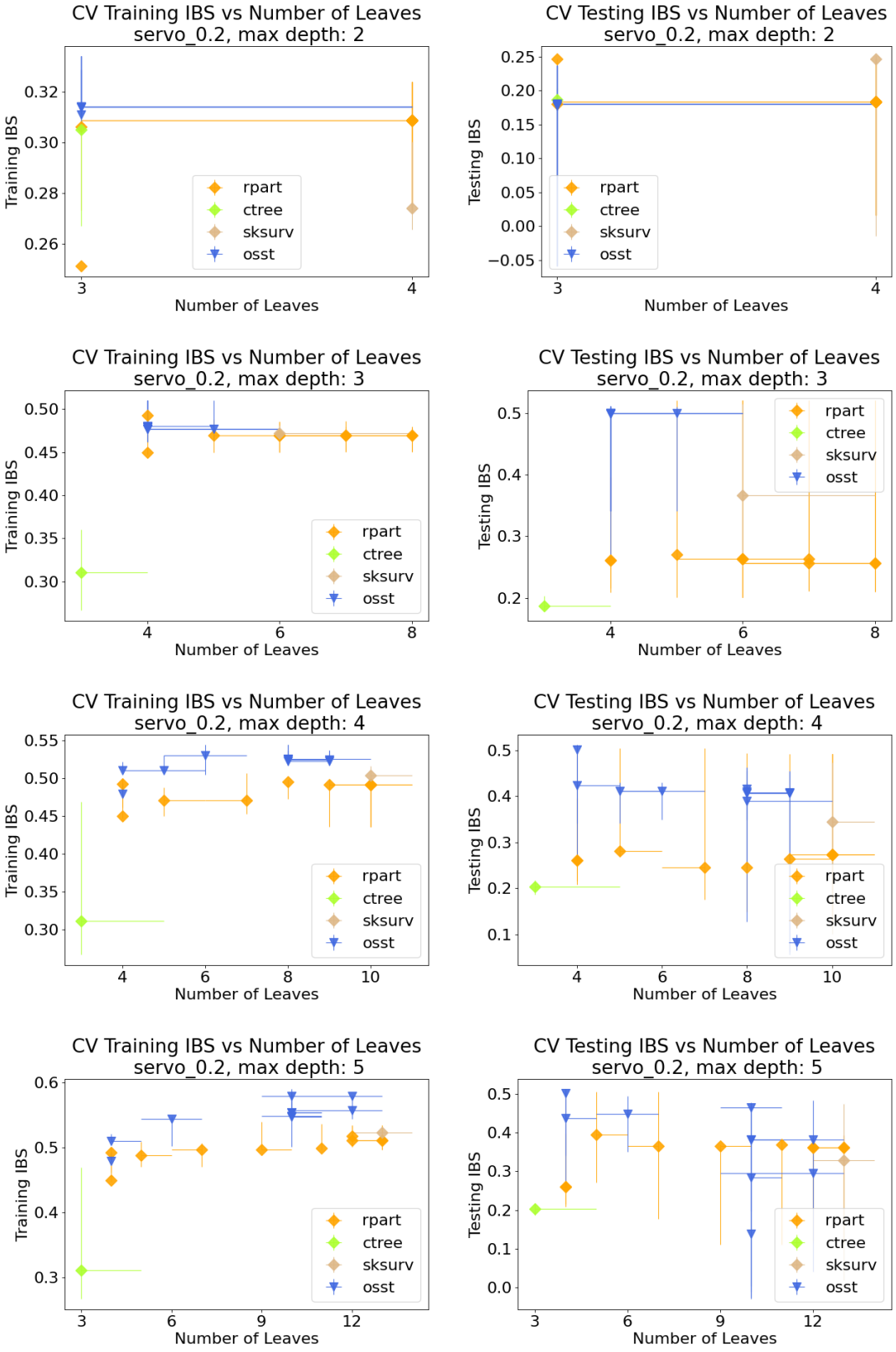

Figure 19: 5-fold CV of CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: servo. (Trees with more than 9 leaves overfitted at depth 5)

G Experiment: Tree Quality

Collection and Setup: One limitation of the IBS metric is that it heavily penalizes models for being inaccurate in the early stages of follow-up, so we would like to see if the comprehensive quality of optimal sparse trees found by OSST is actually better than other sub-optimal trees. We evaluated the experiments again in Section E and F but using **Uno's C-index, Harrell's C-index** and **AUC** (note that OSST optimizes the IBS loss). We evaluate the optimization experiments on *gbsg2, whas500* in Section E and the cross-validation experiments on *churn, employee* in Section F.

Calculations: Again, we drew one plot per dataset and depth. We replaced the performance metrics to Uno's C-index, Harrell's C-index and AUC.

Results: Figure 20 to 31 show that OSST generally obtains better training and testing performance of other metrics as well, even though it is optimized on the IBS Loss. This indicates that the sparse optimal trees found by OSST *have better overall quality* over trees returned by other methods. In other word, OSST is better in capturing the patterns in survival data than other methods.

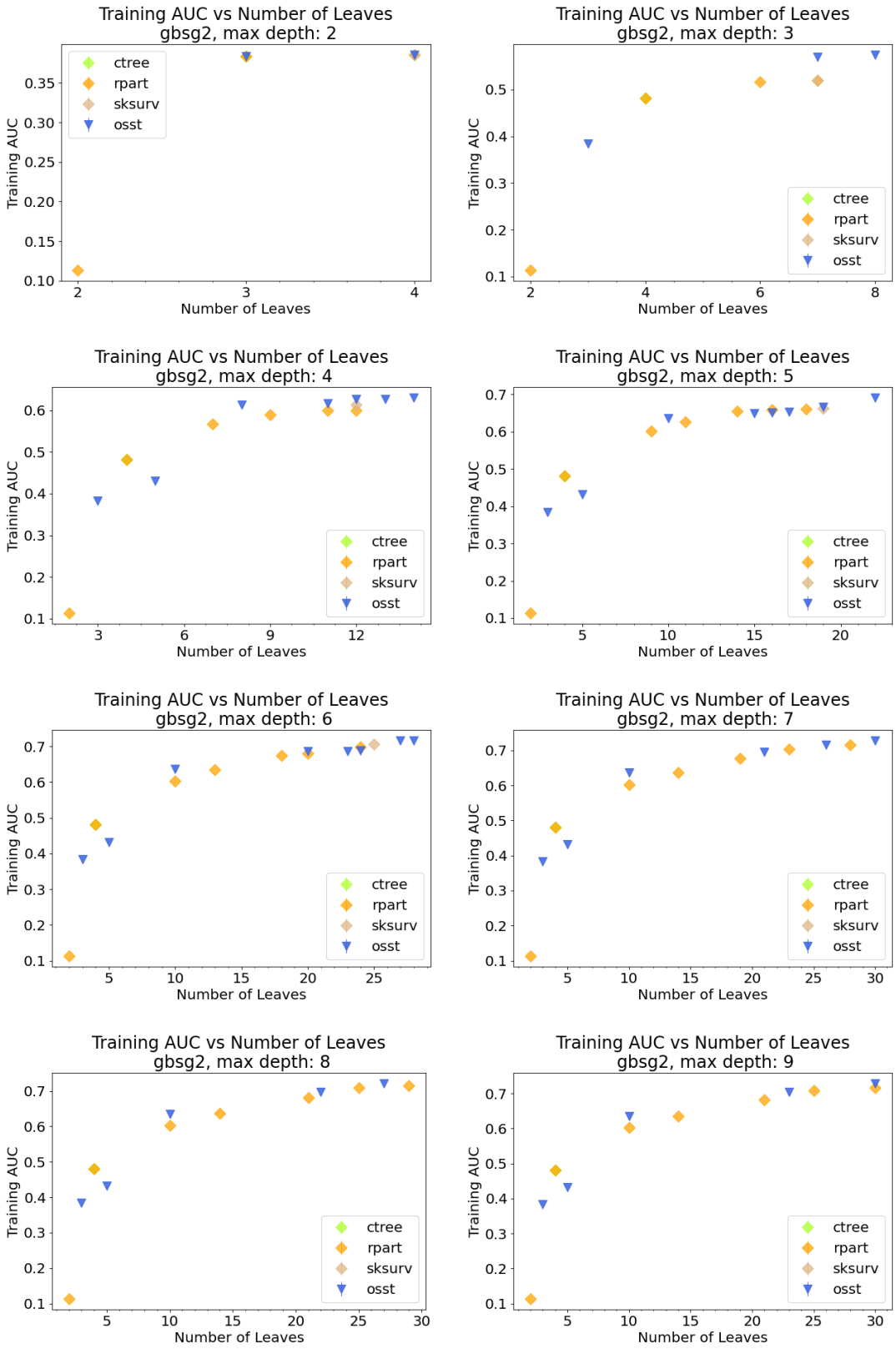

Figure 20: Training auc achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: gbsg2.

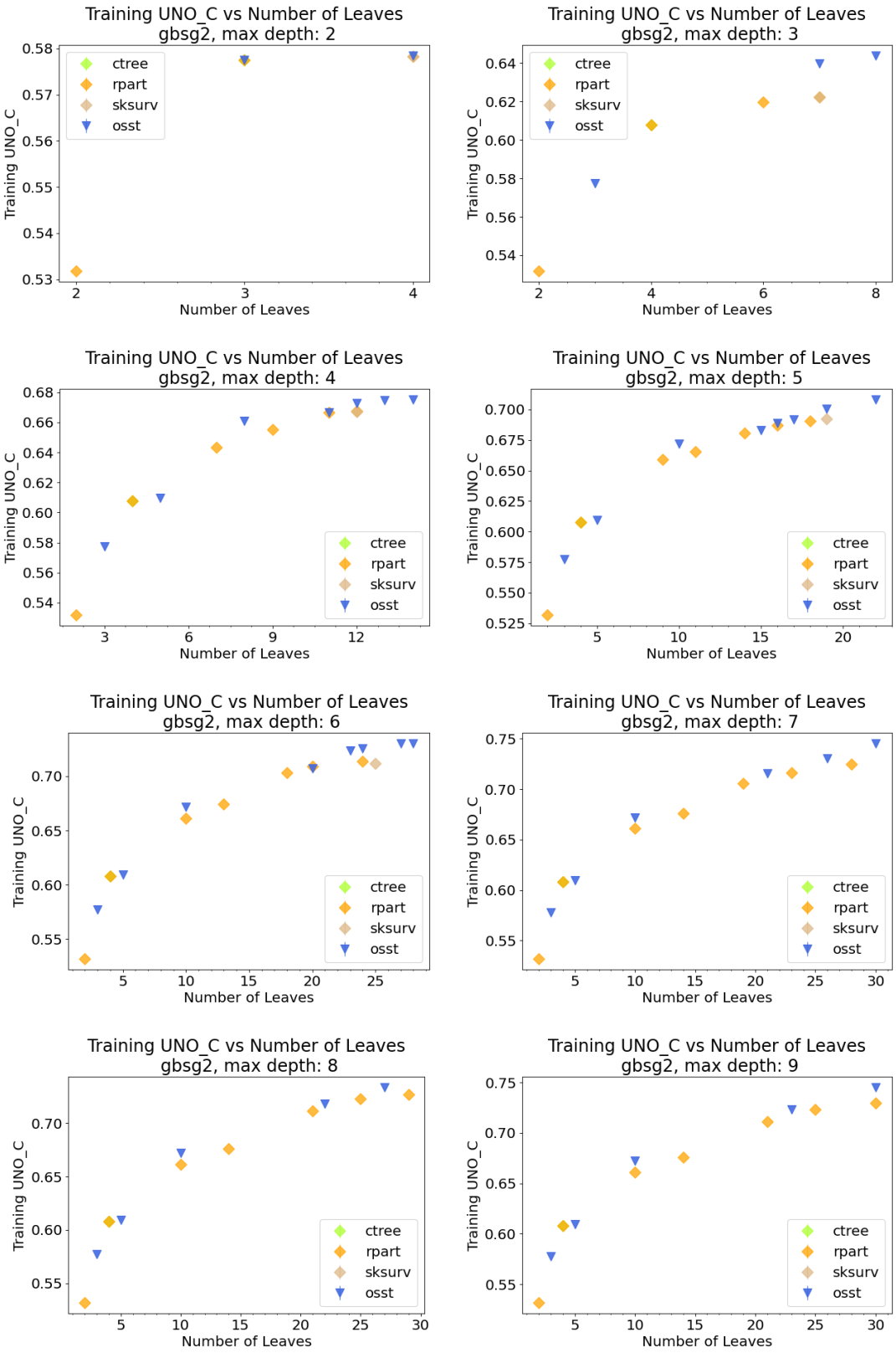

Figure 21: Training uno_c achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: gbsg2.

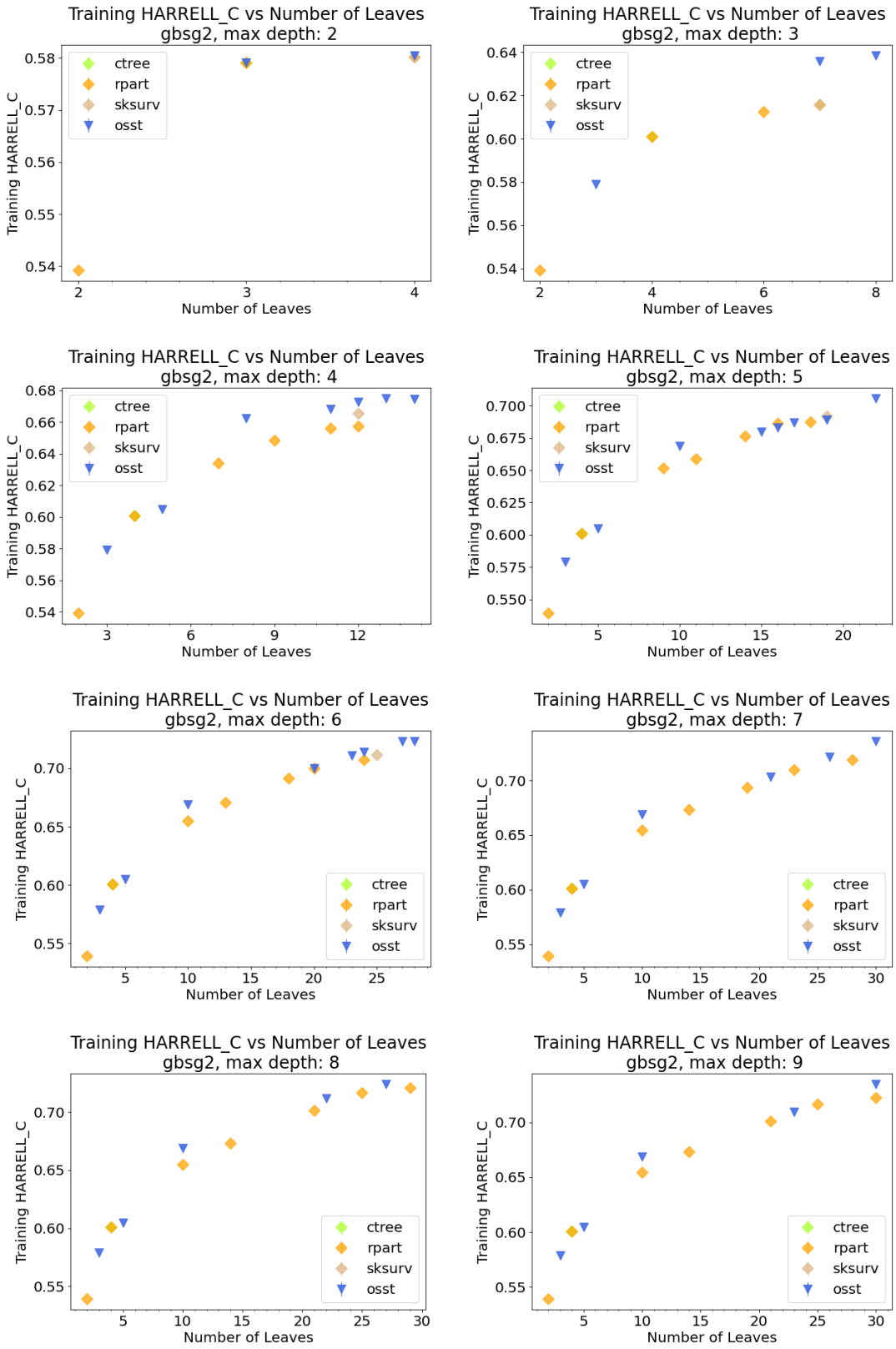

Figure 22: Training harrell_c achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: gbsg2.

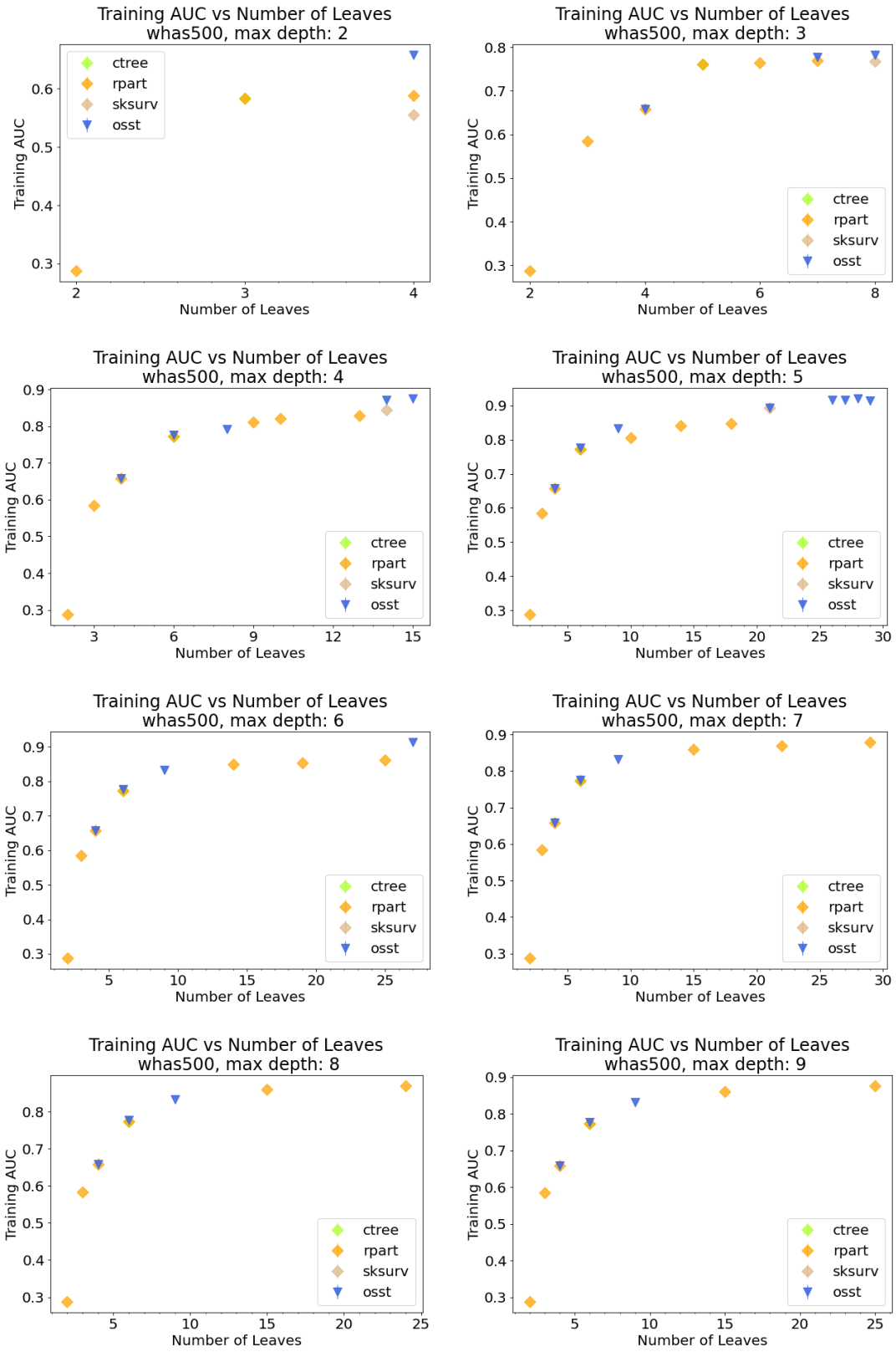

Figure 23: Training auc achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: whas500.

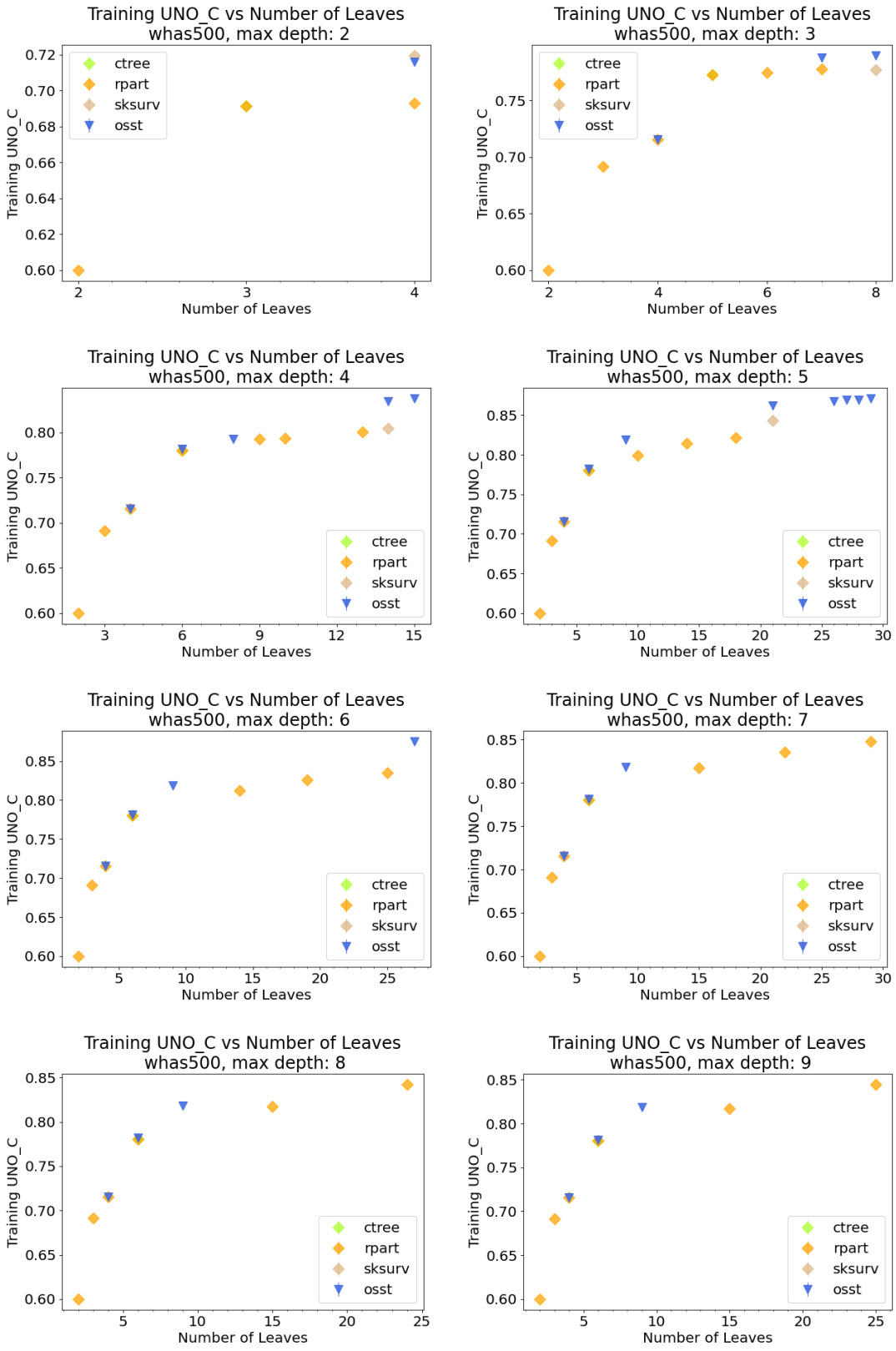

Figure 24: Training uno_c achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: whas500.

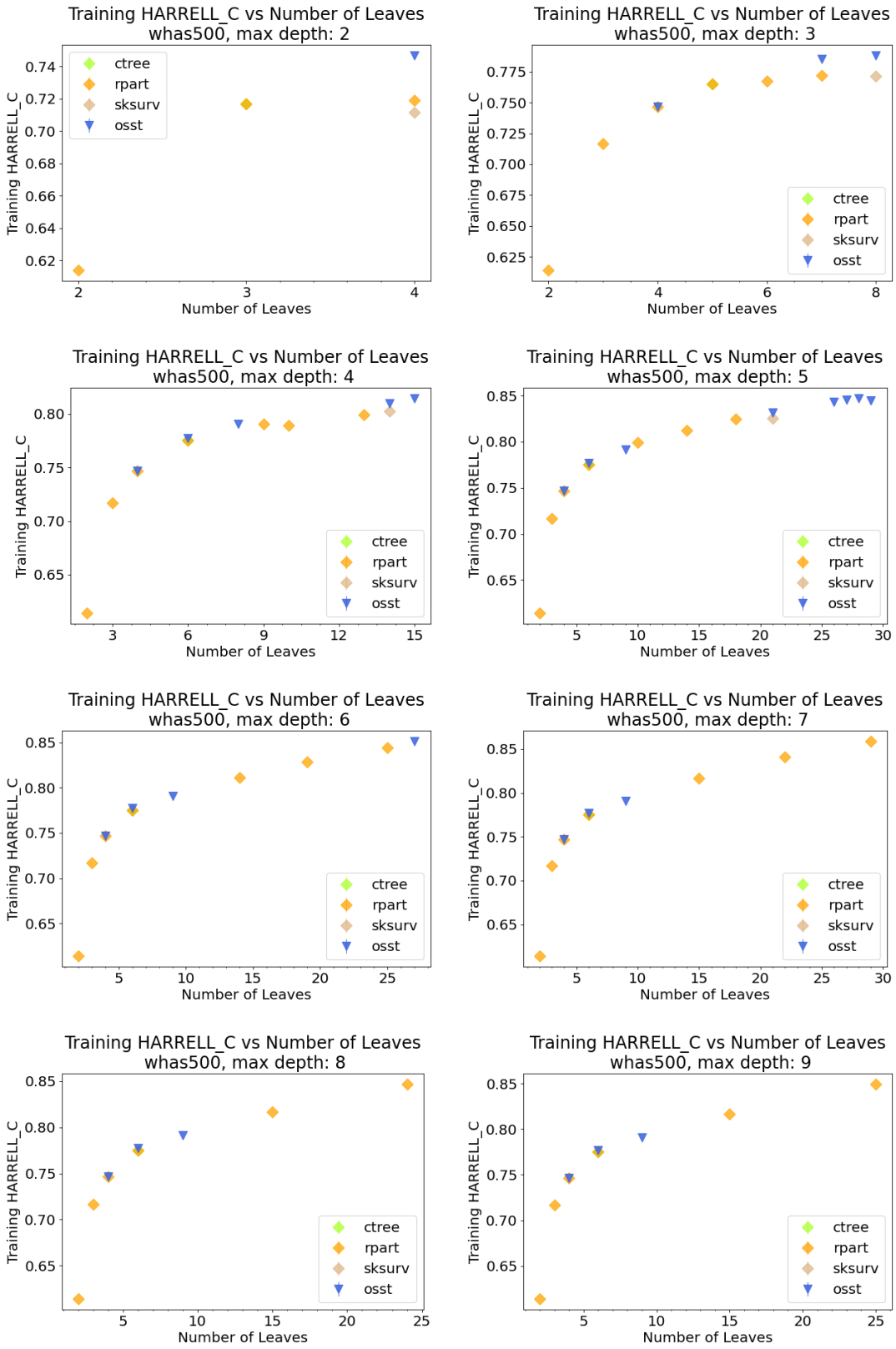

Figure 25: Training harrell_c achieved by CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: whas500.

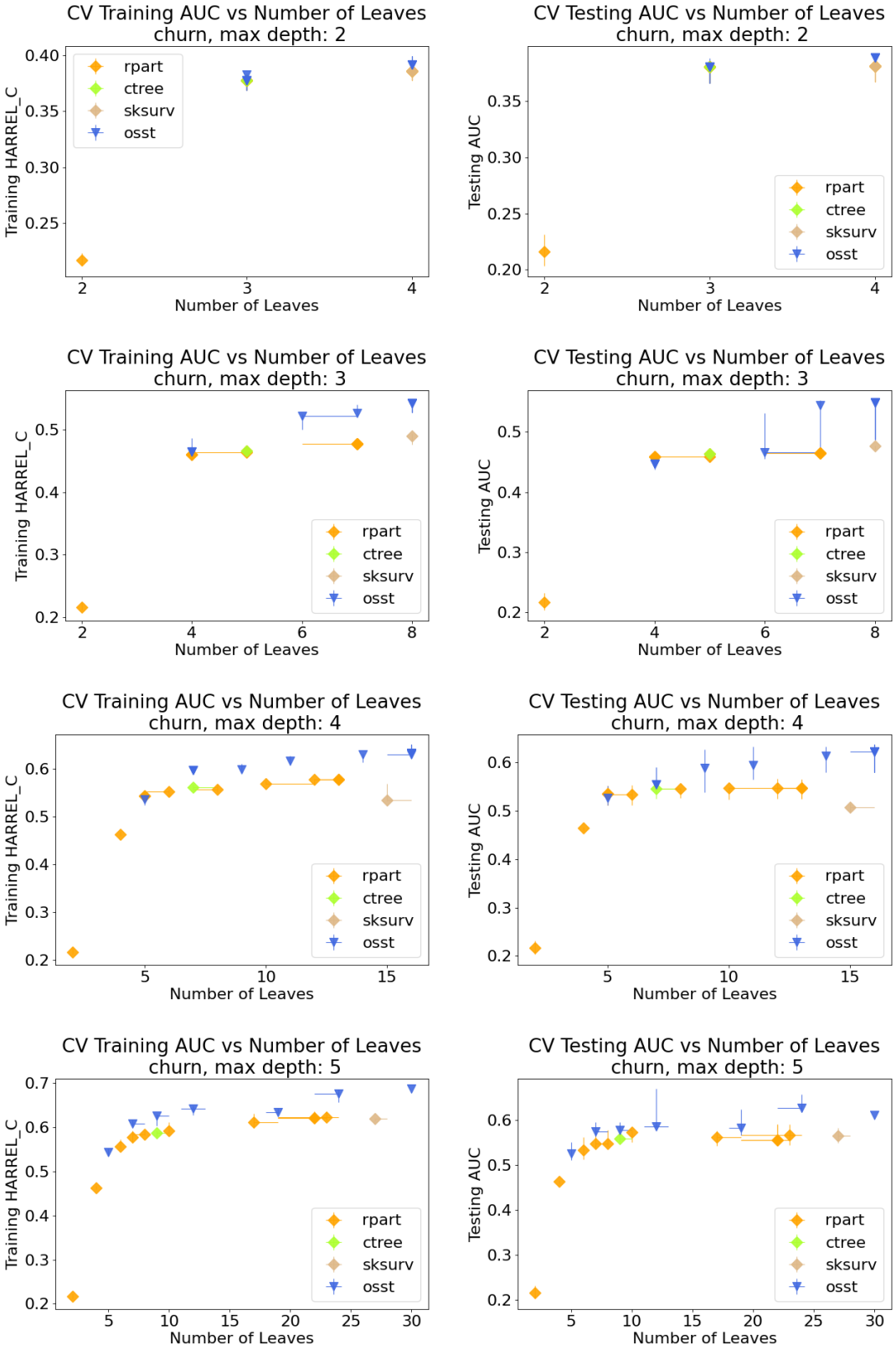

Figure 26: 5-fold CV of CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: churn, metric: auc.

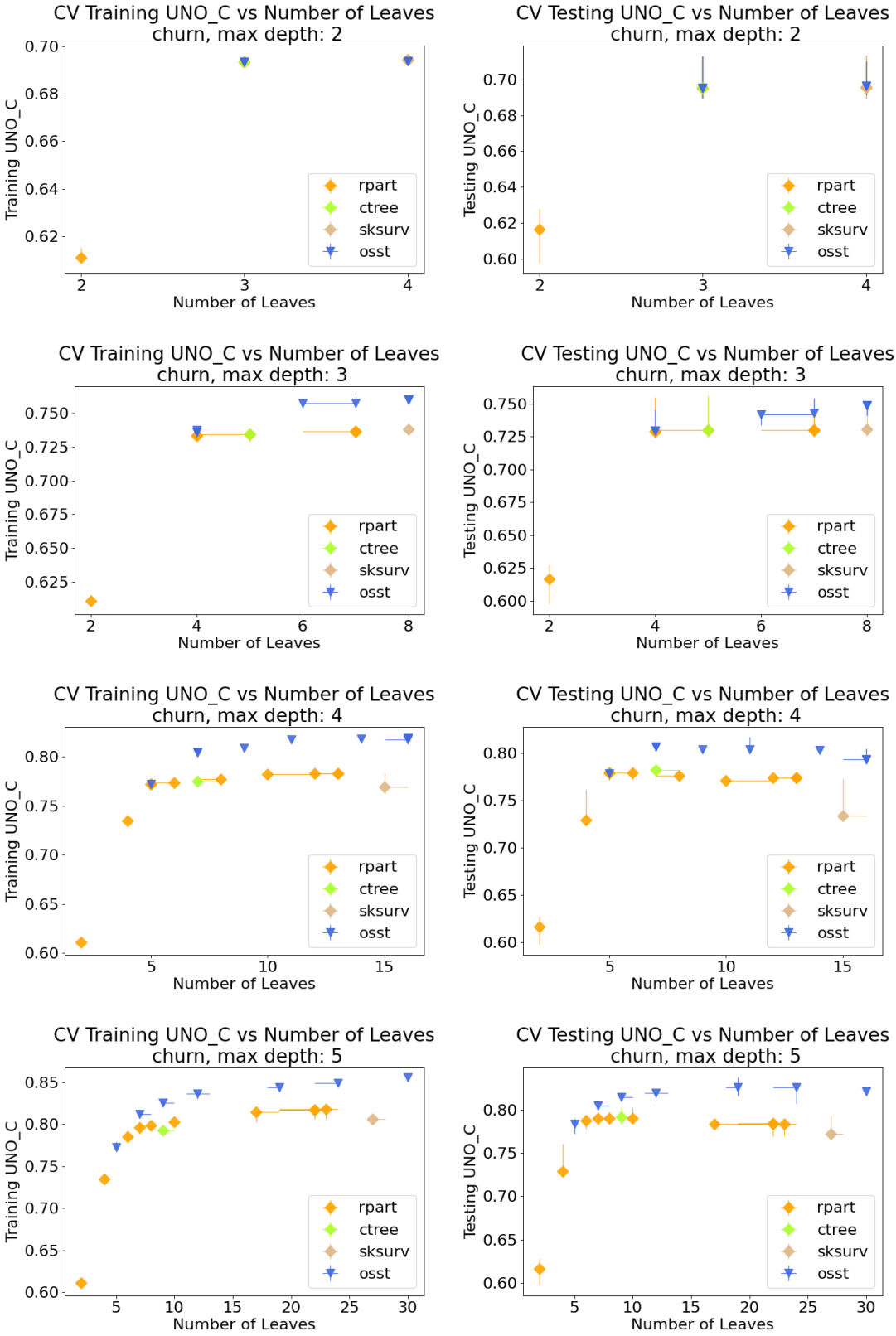

Figure 27: 5-fold CV of CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: churn, metric:uno_c.

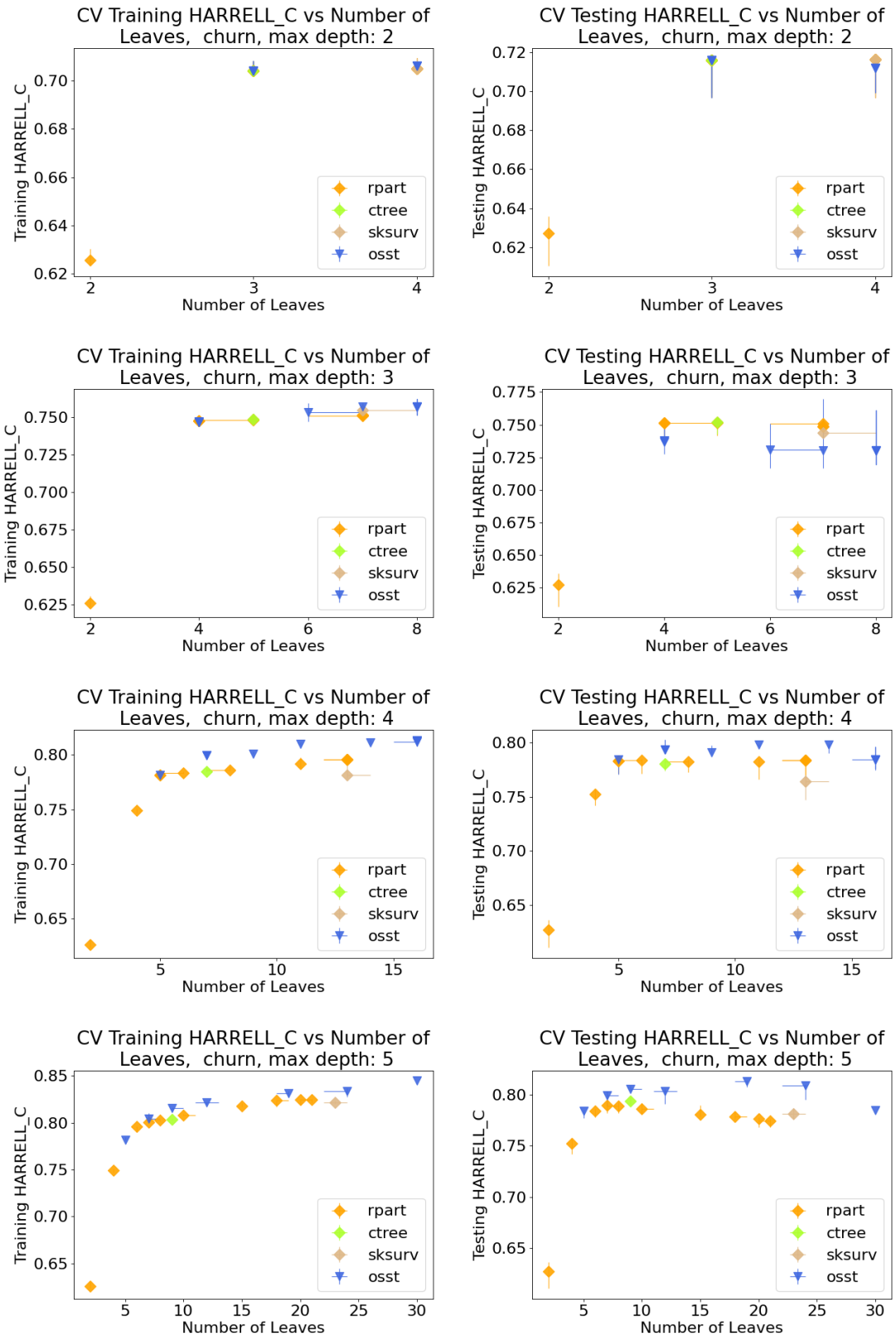

Figure 28: 5-fold CV of CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: churn, metric: harrell_c.

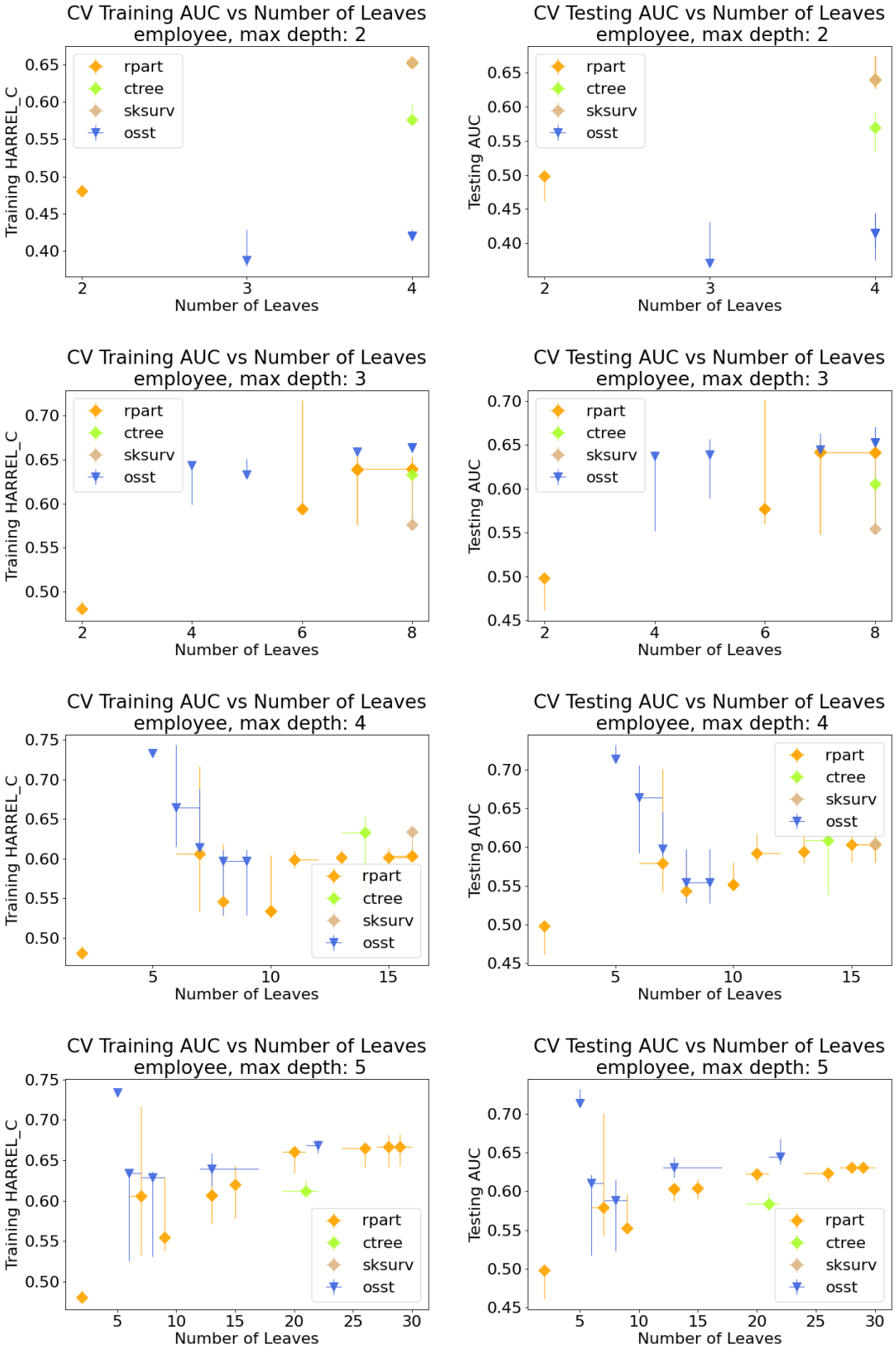

Figure 29: 5-fold CV of CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: employee, metric: auc.

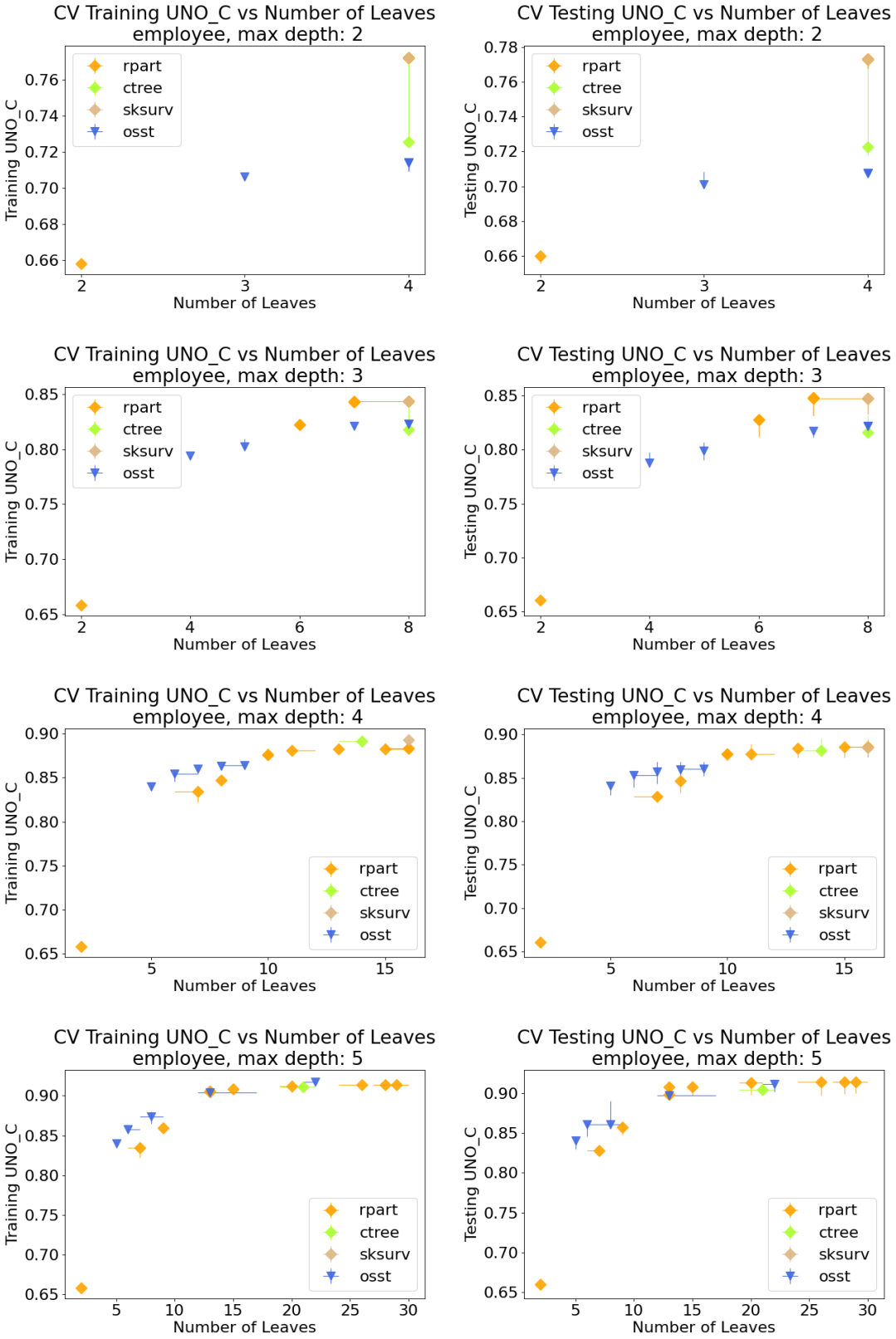

Figure 30: 5-fold CV of CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: employee, metric: uno_c.

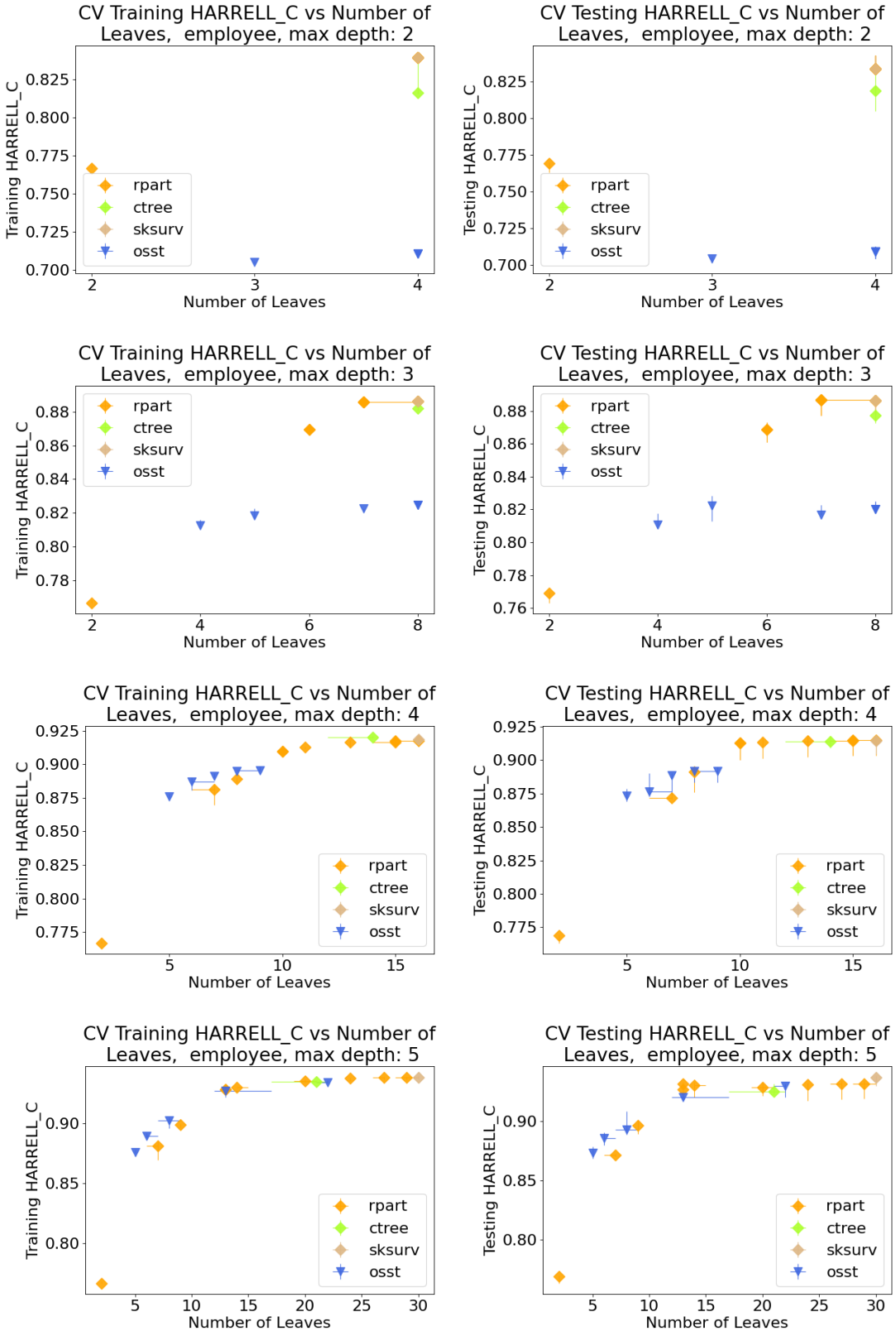

Figure 31: 5-fold CV of CTree, RPART, SkSurv and OSST as a function of number of leaves on dataset: employee, metric: harrell_c.

H Experiment: Running Time

Collection and Setup: The experiments in Section E and F suggest that configurations with large depth limit (e.g., $d = 7, 8, 9$) and small regularization coefficient often result in very large trees (with more than 40 leaves), which lose generalization and interpretability. We compared the running time of different methods on each dataset from depth 2 to 6, with the time limit of 60 minutes, using following configurations:

- **CTree**: We ran this algorithm with 5 different configurations: depth limit, d, ranging from 2 to 6, and a corresponding maximum leaf limit 2^d. All other parameters were set to the default.
- **SkSurv**: We ran this algorithm with 5 different configurations: depth limit, d, ranging from 2 to 6, and a corresponding maximum leaf limit 2^d. The random state was set to 2023 and all other parameters were set to the default.
- **RPART**: We ran this algorithm with 5×12 different configurations: depth limits ranging from 2 to 6, and 12 different regularization coefficients (0.1, 0.05, 0.025, 0.01, 0.005, 0.0025, 0.001, 0.0005, 0.00025, 0.0001, 0.00005, 0.00001). All other parameters were set to the default.
- **OSST** (our method): We ran this algorithm with 5×12 different configurations: depth limits ranging from 2 to 6, and 12 different regularization coefficients (0.1, 0.05, 0.01, 0.005, 0.0025, 0.001, 0.0005, 0.00025, 0.0001, 0.00005, 0.000025, 0.00001). We used a random survival forest that consists of 100 trees with maximum depth 9 (in order to preserve high quality) as a reference model on datasets: *airfoil, insurance, real-estate, sync, servo, flchain* for all depths, *churn, credit, employee* for depths 5-6. No reference model was used on other datasets.

Results: Tables 3 to 5 summarize the average running time and its 95% confidence interval for different methods. *OSST often finds the optimal survival tree in seconds.* Greedy methods are generally faster, but fail to produce an optimal tree. Interestingly, we found that the running speed of OSST on real-world survival data is generally faster than on synthetic data if no guessing technique was applied.

Dataset	Depth	CTree	RPART	SkSurv	OSST
aids	2	0.192(±0)	0.033(±0.003)	0.006(±0)	0.011(±0.000)
	3	0.201(±0)	0.039(±0.005)	0.007(±0)	0.083(±0.003)
	4	0.217(±0)	0.045(±0.006)	0.008(±0)	0.457(±0.056)
	5	0.214(±0)	0.048(±0.007)	0.009(±0)	2.215(±0.533)
	6	0.206(±0)	0.052(±0.008)	0.010(±0)	8.650(±2.530)
aids_death	2	0.192(±0)	0.033(±0.003)	0.006(±0)	0.011(±0.000)
	3	0.201(±0)	0.039(±0.005)	0.007(±0)	0.083(±0.003)
	4	0.217(±0)	0.045(±0.006)	0.008(±0)	0.457(±0.056)
	5	0.214(±0)	0.048(±0.007)	0.009(±0)	2.215(±0.533)
	6	0.206(±0)	0.052(±0.008)	0.010(±0)	8.650(±2.530)
gbsg2	2	0.201(±0)	0.023(±0.002)	0.003(±0)	0.003(±0.000)
	3	0.209(±0)	0.025(±0.003)	0.004(±0)	0.019(±0.001)
	4	0.213(±0)	0.027(±0.003)	0.004(±0)	0.078(±0.006)
	5	0.225(±0)	0.030(±0.004)	0.005(±0)	0.264(±0.026)
	6	0.214(±0)	0.031(±0.500)	0.005(±0)	0.667(±0.077)
maintenance	2	0.188(±0)	0.027(±0.004)	0.003(±0)	0.033(±0.000)
	3	0.205(±0)	0.031(±0.005)	0.004(±0)	0.141(±0.022)
	4	0.218(±0)	0.034(±0.001)	0.005(±0)	0.390(±0.090)
	5	0.220(±0)	0.037(±0.003)	0.005(±0)	0.950(±0.320)
	6	0.229(±0)	0.039(±0.004)	0.006(±0)	2.400(±1.110)

Table 3: Running time in seconds of CTree, RPART, SkSurv and OSST.

Dataset	Depth	CTree	RPART	SkSurv	OSST
veterans	2	0.203(±0)	0.013(±0.000)	0.001(±0)	0.007(±0.001)
	3	0.200(±0)	0.014(±0.001)	0.002(±0)	0.024(±0.001)
	4	0.202(±0)	0.014(±0.001)	0.001(±0)	0.095(±0.009)
	5	0.208(±0)	0.015(±0.001)	0.001(±0)	0.380(±0.080)
	6	0.204(±0)	0.015(±0.001)	0.001(±0)	1.270(±0.380)
whas500	2	0.217(±0)	0.026(±0.001)	0.003(±0)	0.006(±0.000)
	3	0.203(±0)	0.029(±0.005)	0.003(±0)	0.007(±0.001)
	4	0.229(±0)	0.033(±0.006)	0.004(±0)	0.690(±0.020)
	5	0.224(±0)	0.036(±0.001)	0.005(±0)	6.040(±0.470)
	6	0.234(±0)	0.038(±0.002)	0.005(±0)	37.01(±4.530)
uissurv	2	0.198(±0)	0.046(±0.001)	0.004(±0)	0.006(±0.000)
	3	0.206(±0)	0.051(±0.001)	0.005(±0)	0.058(±0.001)
	4	0.212(±0)	0.056(±0.006)	0.006(±0)	0.410(±0.010)
	5	0.225(±0)	0.060(±0.006)	0.007(±0)	2.680(±0.140)
	6	0.214(±0)	0.065(±0.008)	0.008(±0)	13.88(±1.200)
airfoil	2	0.230(±0)	0.032(±0.013)	0.190(±0)	38.12(±1.540)
	3	0.286(±0)	0.139(±0.025)	0.240(±0)	32.00(±1.250)
	4	0.448(±0)	0.289(±0.066)	0.288(±0)	31.90(±2.560)
	5	0.272(±0)	0.348(±0.047)	0.320(±0)	31.97(±1.533)
	6	0.335(±0)	0.453(±0.088)	0.360(±0)	31.98(±2.510)
real-estate	2	0.273(±0)	0.017(±0.003)	0.018(±0)	0.824(±0.005)
	3	0.211(±0)	0.018(±0.004)	0.020(±0)	100.3(±13.92)
	4	0.215(±0)	0.02(±0.006)	0.030(±0)	285.3(±18.61)
	5	0.222(±0)	0.022(±0.001)	0.030(±0)	290.4(±16.42)
	6	0.221(±0)	0.022(±0.001)	0.030(±0)	289.4(±36.42)
sync	2	0.388(±0)	0.017(±0.003)	0.026(±0)	1.920(±0.020)
	3	0.201(±0)	0.019(±0.000)	0.030(±0)	2.120(±0.030)
	4	0.269(±0)	0.019(±0.006)	0.053(±0)	3.788(±0.056)
	5	0.204(±0)	0.021(±0.007)	0.036(±0)	3.804(±0.033)
	6	0.261(±0)	0.021(±0.008)	0.040(±0)	3.785(±0.053)
credit	2	0.221(±0)	0.043(±0.007)	0.011(±0)	0.056(±0.010)
	3	0.246(±0)	0.055(±0.010)	0.014(±0)	1.473(±0.482)
	4	0.273(±0)	0.065(±0.010)	0.020(±0)	33.06(±12.90)
	5	0.283(±0)	0.074(±0.017)	0.020(±0)	610.9(±272.3)
	6	0.317(±0)	0.082(±0.020)	0.022(±0)	930.8(±398.4)
employee	2	0.260(±0)	0.412(±0.001)	0.087(±0)	0.226(±0.018)
	3	0.272(±0)	0.492(±0.005)	0.122(±0)	2.090(±0.110)
	4	0.335(±0)	0.552(±0.012)	0.154(±0)	16.20(±1.370)
	5	0.370(±0)	0.610(±0.030)	0.183(±0)	101.8(±15.36)
	6	0.443(±0)	0.6594(±0.04)	0.210(±0)	538.3(±125.7)
churn	2	0.211(±0)	0.060(±0.006)	0.016(±0)	0.138(±0.000)
	3	0.225(±0)	0.074(±0.001)	0.022(±0)	0.880(±0.070)
	4	0.250(±0)	0.090(±0.006)	0.030(±0)	11.67(±2.360)
	5	0.281(±0)	0.048(±0.007)	0.009(±0)	125.2(±35.53)
	6	0.306(±0)	0.117(±0.008)	0.038(±0)	1197(±410.5)
insurance	2	0.220(±0)	0.043(±0.003)	0.180(±0)	0.083(±0.010)
	3	0.234(±0)	0.049(±0.001)	0.230(±0)	0.083(±0.010)
	4	0.258(±0)	0.058(±0.002)	0.261(±0)	0.084(±0.010)
	5	0.259(±0)	0.064(±0.003)	0.277(±0)	0.084(±0.010)
	6	0.250(±0)	0.064(±0.005)	0.290(±0)	0.084(±0.010)

Table 4: Running time in seconds of CTree, RPART, SkSurv and OSST (continued).

Dataset	Depth	CTree	RPART	SkSurv	OSST
flchain	2	0.418(±0)	0.523(±0.013)	0.109(±0)	0.481(±0.022)
	3	0.460(±0)	0.521(±0.001)	0.110(±0)	0.336(±0.240)
	4	0.523(±0)	0.652(±0.042)	0.209(±0)	0.419(±0.110)
	5	0.559(±0)	0.824(±0.103)	0.244(±0)	1.264(±0.510)
	6	0.601(±0)	0.064(±0.005)	0.285(±0)	7.024(±3.230)
servo	2	0.198(±0)	0.014(±0.000)	0.004(±0)	0.075(±0.001)
	3	0.260(±0)	0.015(±0.002)	0.004(±0)	6.100(±0.360)
	4	0.199(±0)	0.015(±0.006)	0.004(±0)	227.4(±45.10)
	5	0.206(±0)	0.016(±0.007)	0.004(±0)	428.4(±85.30)
	6	0.206(±0)	0.016(±0.006)	0.004(±0)	2192(±586.0)

Table 5: Running time in seconds of CTree, RPART, SkSurv and OSST (continued).

I Experiment: Scalability

Collection and Setup: We ran this experiment on the dataset *household* for CTree, RPART, SkSurv and OSST. We subsampled 100, 500, 1000, 5000, 10,000, 50,000, 100,000, 500,000, 1000,000 samples from it to form 10 datasets (including the original dataset). We randomly censored samples in each dataset with censoring rates of 20%, 50% and 80%. The depth limit was set to 5 (for all methods) and regularization coefficient to 0.01 for OSST and 0.05 for RPART, in order to generate trees with similar size. OSST used an extreme random survival forest that consists of 20 trees with maximum depth 3 as the reference model. We set the time limit of each run to 60 minutes in this experiment.

Results: Figure 32 shows that OSST achieves similar performance with other greedy methods when the number of samples is no more than ten thousand. It scales better than SkSurv on datasets with more than ten thousand samples (8-10 times faster than SkSurv). OSST and SkSurv timed out when there are more than 0.1 million samples in the dataset. There is no obvious difference in training time when the rate of censoring varies.

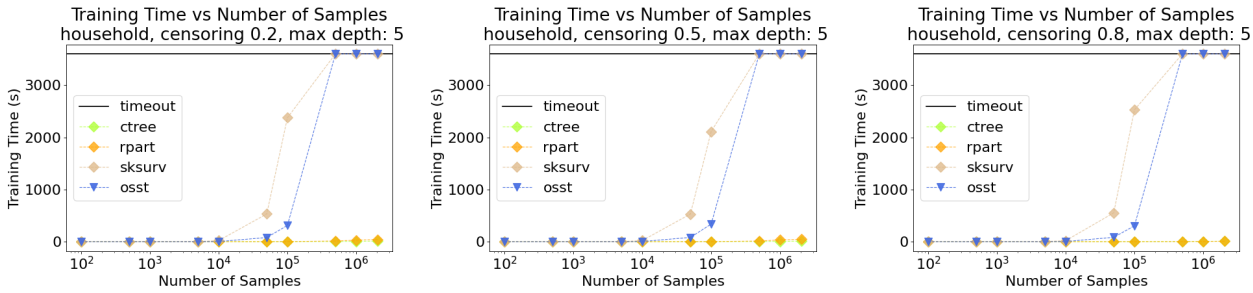

Figure 32: Training time of CTree, RPART, SkSurv and OSST as a function of sample size on dataset *household*, $d = 5, \lambda = 0.01$. (60-minutes time limit)

J Ablation Study: Guessing

We explored if OSST is still able to find optimal trees when our guessing technique is applied and how much time it saves.

Collection and Setup: We ran this experiment on 4 datasets *(churn, employee, servo, sync)* for variations of OSST.

- *churn*: We set depth limit to 6 and regularization coefficient to 0.0025.
- *employee*: We set depth limit to 6 and regularization coefficient to 0.01.
- *servo*: We set depth limit to 5 and regularization coefficient to 0.01.
- *sync*: We set depth limit to 5 and regularization coefficient to 0.01.

For each dataset, we ran OSST twice, once using the *reference model lower bound* and once using the *equivalent*

points lower bound. We used random survival forests with 100 trees of depth 9 as reference models.

Calculations: We recorded the elapsed time (includes the training time of reference models), IBS scores, iterations, and size of dependency graph for each run of OSST. We plot the training IBS against training time for each dataset.

Results: Figure 33 shows that our reference model lower bound substantially reduces the training time (reduction ranges from 40% to 95%), without losing the optimality of generated trees.

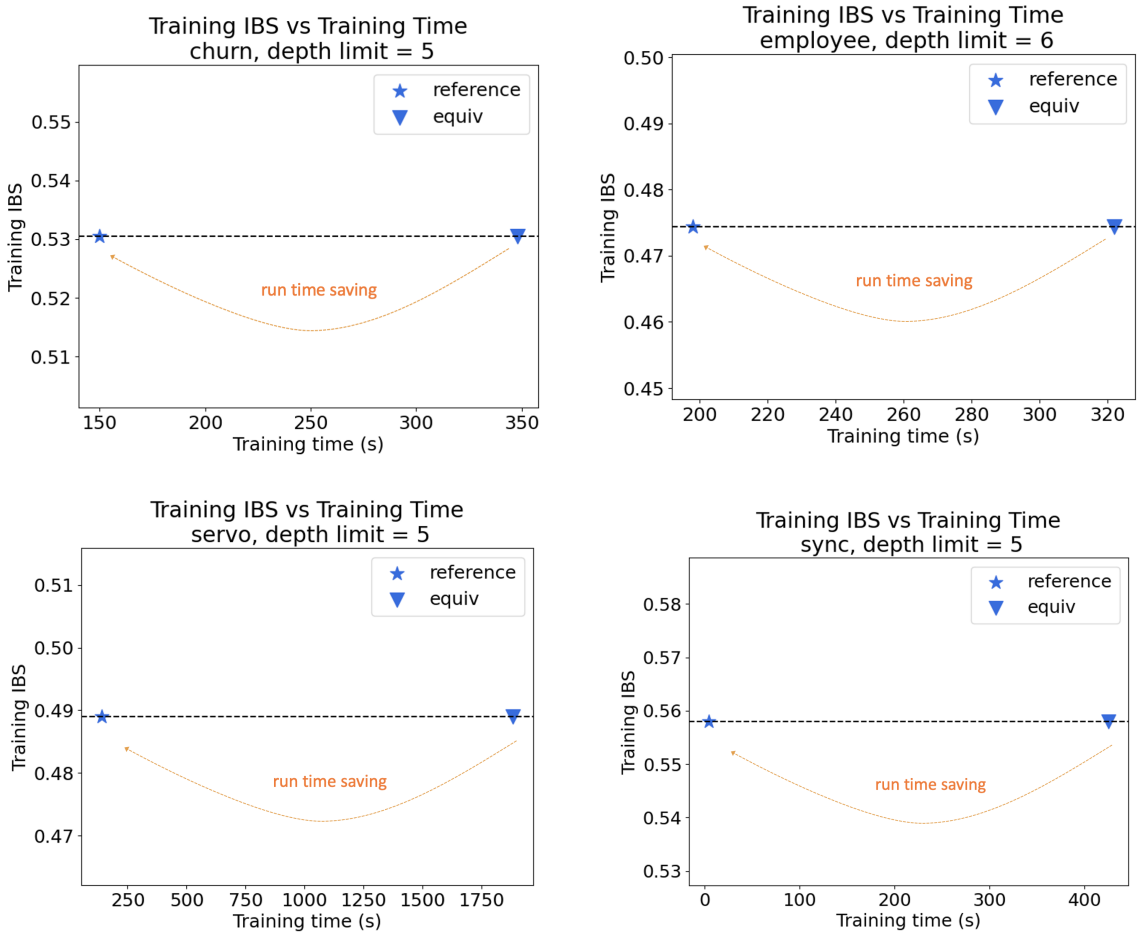

Figure 33: Training time savings of OSST using reference model lower bound on dataset *churn, employee, servo, sync*.

K Optimal Survival Trees

We visually compare the optimal survival trees returned by OSST and sub-optimal trees returned by other methods. Figure 34 shows an optimal tree found by OSST with 8 leaves (IBS ratio of 32.83%) and Figure 35 shows a sub-optimal tree found by RPART with 10 leaves (IBS = 27.95%). Here, OSST found a tree with fewer leaves yet better performance. Figures 36 to 38 show three 4-leaf survival trees returned by OSST, CTree and SkSurv respectively, among which OSST achieves the best performance.

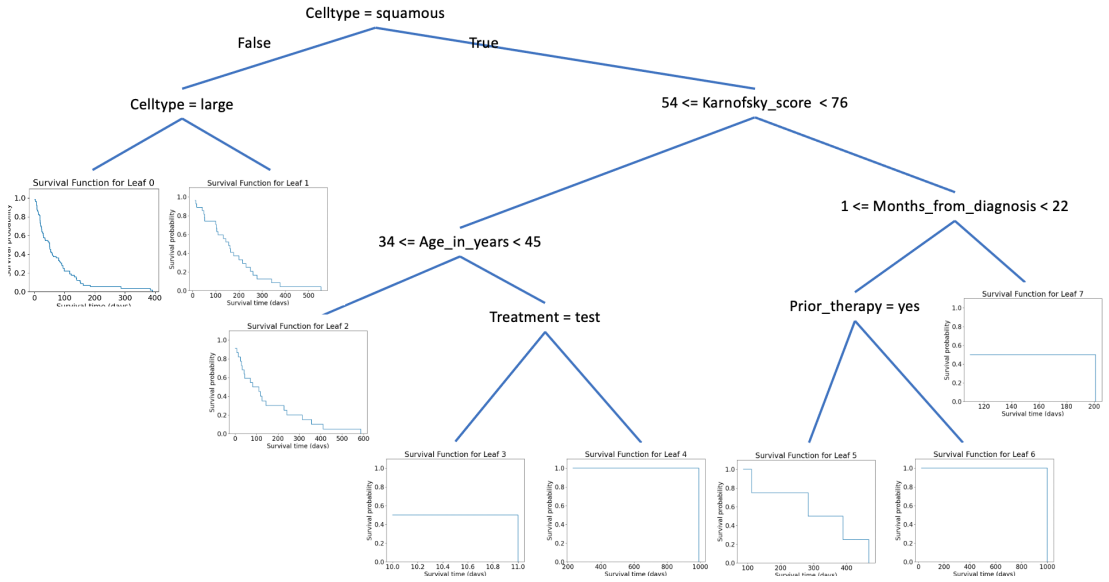

Figure 34: Optimal survival tree produced by OSST for **veterans** dataset with 8 leaves, IBS: 32.83%.

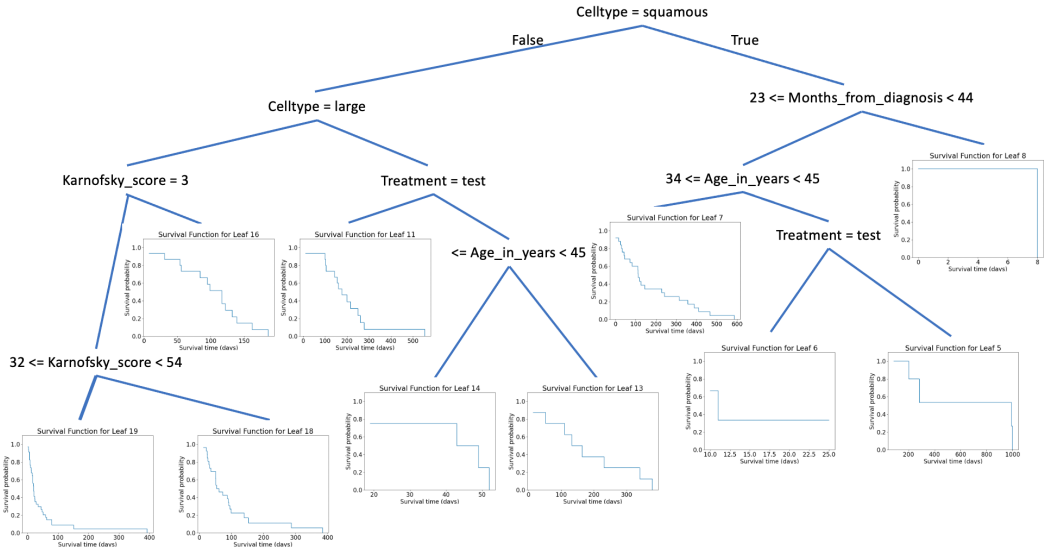

Figure 35: Sub-optimal survival tree produced by RPART for **veterans** dataset with 10 leaves, IBS: 27.95%.

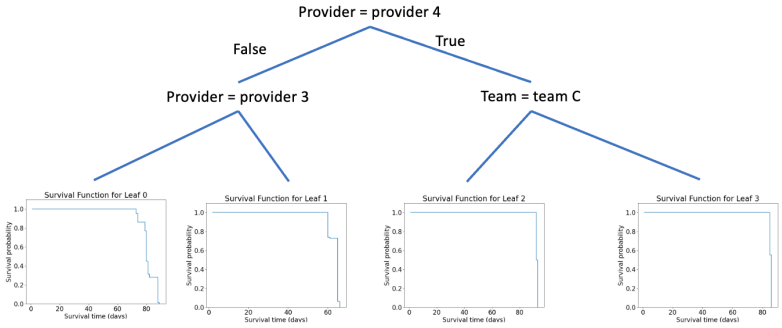

Figure 36: Optimal survival tree produced by OSST for **maintenance** dataset with 4 leaves, IBS: 73.25%.

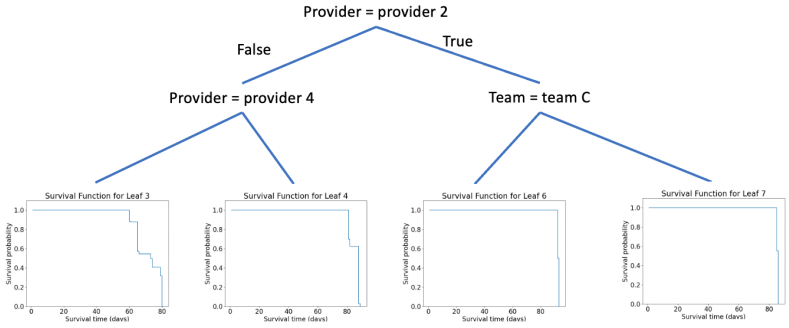

Figure 37: Sub-optimal survival tree produced by CTree for **maintenance** dataset with 4 leaves, IBS: 52.26%.

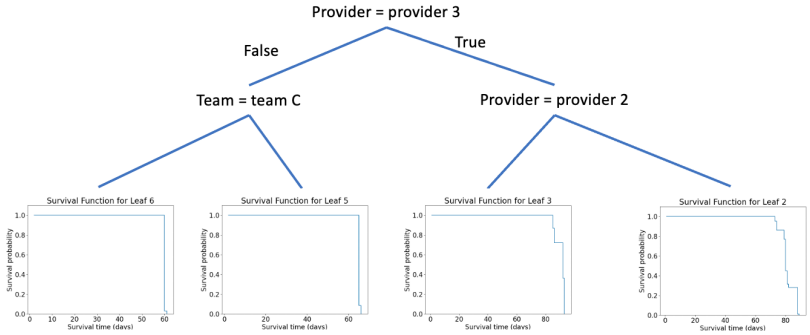

Figure 38: Sub-optimal survival tree produced by SkSurv for **maintenance** dataset with 4 leaves, IBS: 71.05%.

L Comparison with Interpretable AI

Interpretable AI implements the Optimal Survival Tree (OST) algorithm of Bertsimas et al. (2022) which optimizes an objective that is totally different from OSST. It does not use any of the survival metrics defined in Section C. OST assumes the cumulative hazard function of each sample is proportional to the baseline cumulative hazard function, like Cox models do, and samples in the same leaf have the same adjusted coefficient to the baseline. It considers the 'ground truth' of each sample's adjusted coefficient as the coefficient when each sample stays in its own leaf, which is not always possible due to the existence of equivalent points as we discussed before in Theorem 2.6. OST uses the log-likelihood difference between the fitted leaf node coefficients and the 'true' coefficients as the prediction error and penalizes it with the number of leaves.

OST frequently crashed in the experiments in Section E to I (reporting segment faults, see Listing 1). With limited results returned by IAI, we found that trees generated by IAI are *not optimal* in terms of IBS (ratio). Figure 39 shows that the trees of IAI sometimes are worse than those of the greedy methods.

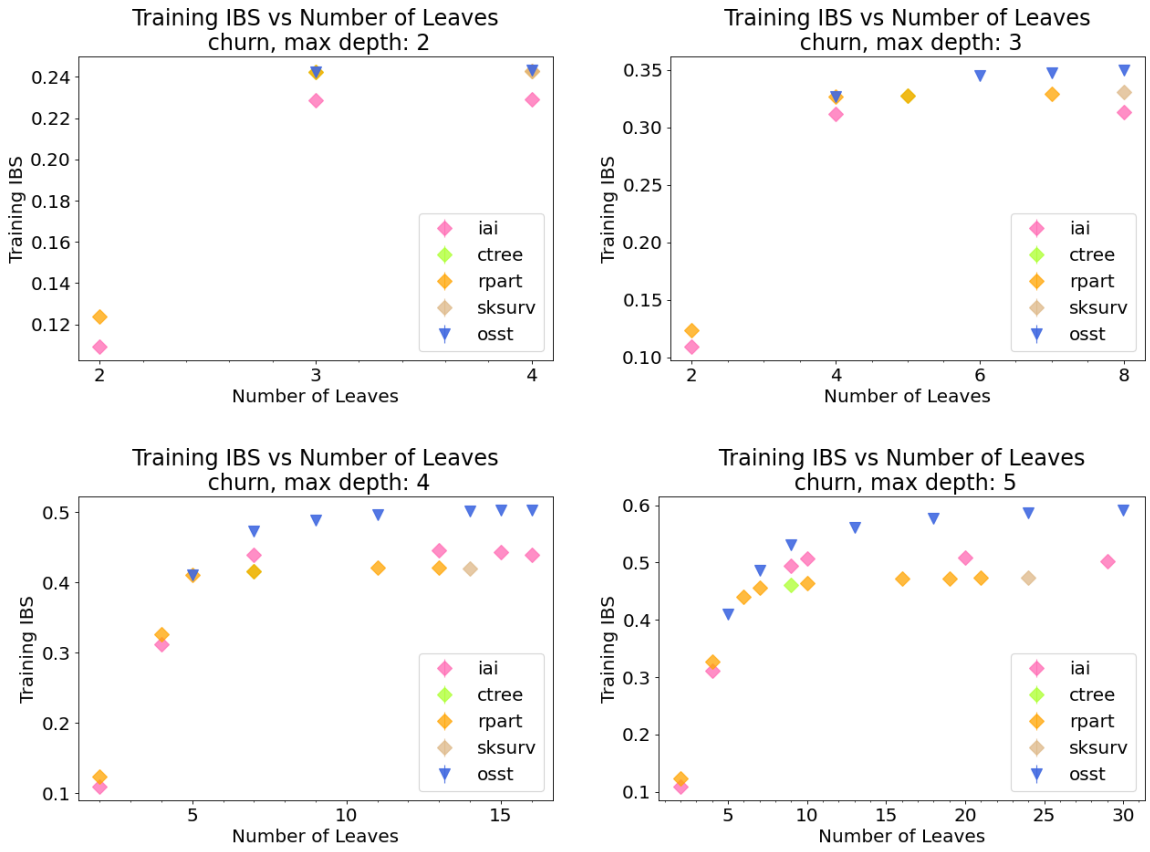

Figure 39: Training IBS achieved by CTree, RPART, SkSurv, IAI and OSST as a function of number of leaves on dataset: churn.

Listing 1: IAI Error Code

```
[ Warning: This copy of Interpretable AI software is for academic purposes
only and not for commercial use.

signal (11): Segmentation fault
in expression starting at none:0
_Py_DECREF at /usr/local/src/conda/python-3.9.12/Include/object.h:422 [inlined]
GetResult at /usr/local/src/conda/python-3.9.12/Modules/_ctypes/callproc.c:989 [inlined]
_ctypes_callproc at /usr/local/src/conda/python-3.9.12/Modules/_ctypes/callproc.c:1301
PyCFuncPtr_call at /usr/local/src/conda/python-3.9.12/Modules/_ctypes/_ctypes.c:4201
_PyObject_MakeTpCall at python3 (unknown line)
_PyEval_EvalFrameDefault at python3 (unknown line)
_PyFunction_Vectorcall at python3 (unknown line)
unknown function (ip: 0x55cf104ac72e)
_PyFunction_Vectorcall at python3 (unknown line)
unknown function (ip: 0x55cf104ac72e)
_PyFunction_Vectorcall at python3 (unknown line)
unknown function (ip: 0x55cf104ac72e)
unknown function (ip: 0x55cf10544bf8)
unknown function (ip: 0x55cf104ac754)
unknown function (ip: 0x55cf10544bf8)
unknown function (ip: 0x55cf104f5286)
unknown function (ip: 0x55cf104f54dc)
unknown function (ip: 0x55cf1053a102)
PyObject_GetAttr at python3 (unknown line)
_PyObject_GetMethod at python3 (unknown line)
_PyEval_EvalFrameDefault at python3 (unknown line)
_PyFunction_Vectorcall at python3 (unknown line)
unknown function (ip: 0x55cf1056ef4e)
PyObject_GetItem at python3 (unknown line)
_PyEval_EvalFrameDefault at python3 (unknown line)
_PyFunction_Vectorcall at python3 (unknown line)
unknown function (ip: 0x55cf104aaae5)
unknown function (ip: 0x55cf10543662)
_PyFunction_Vectorcall at python3 (unknown line)
unknown function (ip: 0x55cf104aaae5)
unknown function (ip: 0x55cf10543662)
PyEval_EvalCodeEx at python3 (unknown line)
PyEval_EvalCode at python3 (unknown line)
unknown function (ip: 0x55cf105f050a)
unknown function (ip: 0x55cf10620f74)
unknown function (ip: 0x55cf104c1986)
PyRun_SimpleFileExFlags at python3 (unknown line)
Py_RunMain at python3 (unknown line)
Py_BytesMain at python3 (unknown line)
__libc_start_main at /lib/x86_64-linux-gnu/libc.so.6 (unknown line)
unknown function (ip: 0x55cf105ae09f)
Allocations: 57922925 (Pool: 57903382; Big: 19543); GC: 62
Segmentation fault (core dumped)
```

M Comparison with Other Interpretable Survival Models

We also compared OSST with other interpretable models. The Cox proportional hazards model provides more flexible predictions of survival functions but fails to capture non-linear relationships among the features. More importantly, standard Cox models generally do not have sparsity regularization and they are not sparse even with ridge (ℓ_2) regularization. We consider a single Cox model with many non-zero coefficients less interpretable than a single sparse tree model. Generalized Additive Models (GAMs) is another interpretable survival model form that is not as popular as Cox models. Bender et al. (2018) provide one possible way to apply GAMs to survival analysis: the survival data should first be transformed into a format that is suitable to fit Piecewise Exponential Additive Mixed Models (PAMMs); fitted PAMMs can be represented as GAMs and be estimated using generic GAM software.

Collection and Setup: We ran this experiment on 5 datasets *(churn, credit, employee, maintenance, servo)* for Cox, PAMM and OSST. We split each dataset into halves (training and testing sets). For each method, we performed 5-fold cross-validation on the training set to choose hyper-parameters, trained models with the chosen hyper-parameters on the training set and tested the performance (IBS ratio) on the testing set. There are no available hyper-parameters to tune in *pammtools*.

Cox: We used *CoxPHSurvivalAnalysis* from scikit-survival.
PAMM: We used *pam* from *pammtools* package, `https://cran.r-project.org/web/packages/pammtools/index.html`.

Results: We found that OSST approaches the performance of Cox while PAMM is dominated by OSST and a single Kaplan-Meier curve. Table 6 shows the best hyper-parameters, training accuracies and testing accuracies of each method on various datasets.

Dataset	Method	Tuned Hyper-parameters		Training IBS Ratio	Testing IBS Ratio
churn	OSST	regularization	max_depth	0.602	0.470
		0.001	5		
	Cox	alpha		0.571	0.495
		1			
	PAMM	N/A		-0.500	0.490
		N/A			
credit	OSST	regularization	max_depth	0.199	0.170
		0.0025	5		
	Cox	alpha		0.323	0.135
		1			
	PAMM	N/A		-0.443	error
		N/A			
employee	OSST	regularization	max_depth	0.691	0.634
		0.0005	5		
	Cox	alpha		0.441	0.376
		5			
	PAMM	N/A		error	error
		N/A			
maintenance	OSST	regularization	max_depth	0.985	0.983
		0.0001	5		
	Cox	alpha		0.901	0.898
		1			
	PAMM	N/A		-6.54	-5.89
		N/A			
servo	OSST	regularization	max_depth	0.544	0.360
		0.01	4		
	Cox	alpha		0.624	0.487
		1			
	PAMM	N/A		error	-10.963
		N/A			

Table 6: The best hyper-parameters, training accuracies and testing accuracies of OSST, Cox and PAMM on datasets churn, credit, employee, maintenance, servo.

N Black Box Survival Models

We further explored whether optimal sparse survival trees achieve the performance of black box models. Random Survival Forest (RSF) improves predictive performance and reduces overfitting by aggregating the predicted survival functions from individual survival trees. Gradient-boosted Cox proportional hazard loss with regression trees as the base learner can capture complex non-linear relationships and interactions between predictors, producing more accurate predictions of survival outcomes. However, it significantly increases model complexity and makes the model extremely hard to interpret. We tried to compare to several deep learning survival models implemented in package *survivalmodels* (`https://cran.r-project.org/web/packages/survivalmodels/index.html`), including Deepsurv (Katzman et al., 2018), Deephit (Lee et al., 2018), Coxtime (Kvamme et al., 2019), DNNSurv (Zhao and Feng, 2019), but found that they were no longer able to install or run even their example codes.

Collection and Setup: We ran this experiment on 5 datasets *(churn, credit, employee, maintenance, servo)* for RSF, GB-Cox. We split each dataset into halves (training and testing sets). For each method, we performed 5-fold cross-validation on the training set to choose hyper-parameters, trained models with the chosen hyper-parameters on the training set and tested the performance (IBS ratio) on the testing set.

RSF: We used *RandomSurvivalForest* from scikit-survival.
GB-Cox: We used *GradientBoostingSurvivalAnalysis* from scikit-survival.

Results: We found a single optimal sparse survival tree generally outperforms a random survival forest and approaches the performance of GB-Cox. Remember that these methods sacrifice intepretability. Table 7 shows the best hyper-parameters, training accuracies and testing accuracies of each method on various datasets.

Dataset	Method	Tuned Hyper-parameters			Training IBS Ratio	Testing IBS Ratio
churn	OSST	regularization		max_depth	0.602	0.470
		0.001		5		
	RSF	max_features	n_estimators	max_depth	0.533	0.398
		sqrt	200	9		
	GB-Cox	max_features	n_estimators	max_depth	0.677	0.561
		log2	500	3		
credit	OSST	regularization		max_depth	0.199	0.170
		0.0025		5		
	RSF	max_features	n_estimators	max_depth	0.221	0.154
		sqrt	100	5		
	GB-Cox	max_features	n_estimators	max_depth	0.309	0.209
		sqrt	100	3		
employee	OSST	regularization		max_depth	0.691	0.634
		0.0005		5		
	RSF	max_features	n_estimators	max_depth	0.676	0.616
		log2	50	8		
	GB-Cox	max_features	n_estimators	max_depth	0.784	0.625
		log2	500	12		
maintenance	OSST	regularization		max_depth	0.985	0.983
		0.0001		5		
	RSF	max_features	n_estimators	max_depth	0.565	0.545
		log2	50	3		
	GB-Cox	max_features	n_estimators	max_depth	0.880	0.866
		log2	500	3		
servo	OSST	regularization		max_depth	0.544	0.360
		0.01		4		
	RSF	max_features	n_estimators	max_depth	0.360	0.260
		log2	500	3		
	GB-Cox	max_features	n_estimators	max_depth	0.803	0.499
		log2	200	6		

Table 7: The best hyper-parameters, training accuracies and testing accuracies of OSST, RSF and GB-Cox on datasets churn, credit, employee, maintenance, servo.