Optimal Survival Trees

Dimitris Bertsimas* Jack Dunn† Emma Gibson‡ Agni Orfanoudaki§

September 16, 2018

Abstract

Tree-based models are increasingly popular due to their ability to identify complex relationships that are beyond the scope of parametric models. Survival tree methods adapt these models to allow for the analysis of censored outcomes, which often appear in medical data. We present a new Optimal Survival Trees algorithm that leverages mixed-integer optimization (MIO) and local search techniques to generate globally optimal survival tree models. We demonstrate that the OST algorithm improves on the accuracy of existing survival tree methods, particularly in large datasets.

1 Introduction

Survival analysis is a cornerstone of healthcare research and is widely used in the analysis of clinical trials as well as large-scale medical datasets such as Electronic Health Records and insurance claims. Survival analysis methods are required for censored data in which the outcome of interest is generally the time until an event (onset of disease, death, etc.), but the exact time of the event is unknown (censored) for some individuals. When a lower bound for these missing values is known (for example, a patient is known to be alive until at least time t) the data is said to to right-censored.

A common survival analysis technique is Cox proportional hazards regression[11] which models the hazard rate for an event as a linear combination of covariate effects. Although this model is widely used and easily interpreted, its parametric nature makes it unable to identify non-linear effects or interactions between covariates[5].

Recursive partitioning techniques (also referred to as *trees*) are a popular alternative to parametric models. When applied to survival data, survival tree algorithms partition the covariate space into smaller and smaller regions (*nodes*)

*Operations Research Center and Sloan School of Management, Massachusetts Institute of Technology, dbertsim@mit.edu

†Operations Research Center, Massachusetts Institute of Technology, jackdunn@mit.edu

‡Operations Research Center, Massachusetts Institute of Technology, emgibson@mit.edu

§Operations Research Center, Massachusetts Institute of Technology, agniorf@mit.edu

containing observations with homogeneous survival outcomes. The survival distribution in the final partitions (*leaves*) can be analyzed using a variety of statistical techniques such as Kaplan-Meier curve estimates [23].

Most recursive partitioning algorithms generate trees in a top-down, greedy manner, which means that each split is selected in isolation without considering its effect on subsequent splits in the tree. However, Bertsimas and Dunn [3,4] have proposed a new algorithm which uses modern mixed-integer optimization (MIO) techniques to form the entire decision tree in a single step, allowing each split to be determined with full knowledge of all other splits. This *Optimal Trees* algorithm allows the construction of single decision trees for classification and regression that have performance comparable with state-of-the-art methods such as random forests and gradient boosted trees without sacrificing the interpretability offered by a single tree.

The key contributions of this paper are:

1. We present *Optimal Survival Trees* (OST), a new survival trees algorithm that utilizes the *Optimal Trees* framework to generate interpretable trees for censored data.

2. We propose a new accuracy metric that evaluates the fit of Kaplan-Meier curve estimates relative to known survival distributions in synthetic datasets. We also demonstrate that this metric is consistent with the Integrated Brier Score [17], which can be used to evaluate the fit of Kaplan-Meier curves when the true distributions are unknown.

3. We evaluate the performance of our method on synthetic datasets and demonstrate improved accuracy relative to two existing algorithms, particularly in large datasets.

4. Finally, we provide an example of how the algorithm can be used to predict the risk of adverse events associated with cardiovascular health in the Framingham Heart Study (FHS) dataset.

The structure of this paper is as follows. We review existing survival tree algorithms in Section 2 and discuss some of the technical challenges associated with building trees for censored data. In Section 3, we give an overview of the Optimal Trees trees algorithm proposed by Bertsimas and Dunn [3] and we adapt this algorithm for Optimal Survival Trees in Section 4. Section 5 begins with a discussion of our survival tree accuracy metrics, followed by the results for the OST algorithm in synthetic and real datasets. We conclude in Section 6 with a brief summary of our contributions.

2 Review of Survival Trees

Recursive partitioning methods have received a great deal of attention in the literature, the most prominent method being the Classification and Regression Tree algorithm (CART) [7]. Tree-based models are appealing due to their logical,

interpretable structure as well as their ability to detect complex interactions between covariates. However, traditional tree algorithms require complete observations of the dependent variable in training data, making them unsuitable for censored data.

Tree algorithms incorporate a splitting rule which selects partitions to add to the tree, and a pruning rule determines when to stop adding further partitions. Since the 1980s, many authors have proposed splitting and pruning rules for censored data. Splitting rules in survival trees are generally based on either (a) node distance measures that seek to maximize the difference between observations in separate nodes or (b) node purity measures that seek to group similar observation in a single node[26,33].

Algorithms based on node distance measures compare the two adjacent child nodes that are generated when a parent node is split, retaining the split that produces the greatest difference in the child nodes. Proposed measures of node distance include the two-sample logrank test[10], the likelihood ratio statistic[9] and conditional inference permutation tests[20].

Dissimilarity-based splitting rules are unsuitable for certain applications (such as the Optimal Trees algorithm) because they do not allow for the assessment of a single node in isolation. We will therefore focus on node purity splitting rules for developing the OST algorithm.

Gorden and Olshen[16] published the first survival tree algorithm with a node purity splitting rule based on Kaplan–Meier estimates. Davis and Anderson[12] used a splitting rule based on the negative log-likelihood of an exponential model, while Therneau et al.[32] proposed using martingale residuals as an estimate of node error. LeBlanc and Crowley[25] suggested comparing the log–likelihood of a saturated model to the first step of a full likelihood estimation procedure for the proportional hazards model and showed that both the full likelihood and martingale residuals can be calculated efficiently from the Nelson-Aalen cumulative hazard estimator[1,27]. More recently, Molinaro et al.[26] proposed a new approach to adjust loss functions for uncensored data based on inverse probability of censoring weights (IPCW).

Most survival tree algorithms make use of cost-complexity pruning to determine the correct tree size, particularly when node purity splitting is used. Cost-complexity pruning selects a tree that minimizes a weighted combination of the total tree error (i.e., the sum of each leaf node error) and tree complexity (the number of leaf nodes), with relative weights determined by cross-validation. A similar split-complexity pruning method was suggested by LeBlanc and Crowley[24] for node distance measures, using the sum of the split test statistics and the number of splits in the tree. Other proposals include using the Akaike Information Criterion (AIC)[10] or using a p-value stopping criterion to stop growing the tree when no further significant splits are found[20].

Survival tree methods have been extended to include "survival forest" algorithms which aggregate the results of multiple trees. Breiman[6] adapted the CART-based random forest algorithm to survival data, while both Hothorn et al.[21] and Ishwaran et al.[22] proposed more general methods that generate survival forests from any survival tree algorithm. The aim of survival forest models

is to produce more accurate predictions by avoiding the instability of single-tree models. However, this approach leads to "black-box" models which are not interpretable and therefore lack one of the primary advantages of single-tree models.

Relatively few survival tree algorithms have been implemented in publicly available, well-documented software. Two user-friendly options are available in **R**[28] packages: Therneau's algorithm based on martingale residuals is implemented in the **rpart** package[31] and Hothorn's conditional inference (**ctree**) algorithm in the **party** package[19].

3 Review of Optimal Predictive Trees

In this section, we briefly review approaches to constructing decision trees, and in particular, we outline the Optimal Trees algorithm. The purpose of this section is to provide a high-level overview of the Optimal Trees framework; interested readers are encouraged to refer to Bertsimas and Dunn[4,13] for more detailed technical information.

Traditionally, decision trees are trained using a greedy heuristic that recursively partitions the feature space using a sequence of locally-optimal splits to construct a tree. This approach is used by methods like CART[7] to find classification and regression trees. The greediness of this approach is also its main drawback—each split in the tree is determined independently without considering the possible impact of future splits in the tree on the quality of the here-and-now decision. This can create difficulties in learning the true underlying patterns in the data and lead to trees that generalize poorly. The most natural way to address this limitation is to consider forming the decision tree in a single step, where each split in the tree is decided with full knowledge of all other splits.

Optimal Trees is a novel approach for decision tree construction that significantly outperforms existing decision tree methods[4]. It formulates the decision tree construction problem from the perspective of global optimality using mixed-integer optimization (MIO), and solves this problem with coordinate descent to find optimal or near-optimal solutions in practical run times. These Optimal Trees are often as powerful as state-of-the-art methods like random forests or boosted trees, yet they are just a single decision tree and hence are readily interpretable. This obviates the need to trade off between interpretability and state-of-the-art accuracy when choosing a predictive method.

The Optimal Trees framework is a generic approach that tractably and efficiently trains decision trees according to a loss function of the form

$$\min_{T} \quad \texttt{error}(T, D) + \alpha \cdot \texttt{complexity}(T), \tag{1}$$

where T is the decision tree being optimized, D is the training data, $\texttt{error}(T, D)$ is a function measuring how well the tree T fits the training data D, $\texttt{complexity}(T)$ is a function penalizing the complexity of the tree (for a tree with splits

parallel to the axis, this is simply the number of splits in the tree), and α is the *complexity parameter* that controls the tradeoff between the quality of the fit and the size of the tree.

There have been many attempts in the literature to construct globally optimal predictive trees[2,18,30]. However, these methods could not scale to datasets of the sizes required by practical applications, and therefore did not displace greedy heuristics as the approach used in practice. Unlike the others, Optimal Trees is able scale to large datasets (n in the millions, p in the thousands) by using coordinate descent to train the decision trees towards global optimality. When training a tree, the splits in the tree are repeatedly optimized one-at-a-time, finding changes that improve the global objective value in Problem (1). To give a high-level overview, the nodes of the tree are visited in a random order and at each node we consider the following modifications:

- If the node is not a leaf, delete the split at that node;
- If the node is not a leaf, find the optimal split to use at that node and update the current split;
- If the node is a leaf, create a new split at that node.

For each of the changes, we calculate the objective value of the modified tree with respect to Problem (1). If any of these changes result in an improved objective value, then the modification is accepted. When a modification is accepted or all potential modifications have been dismissed, the algorithm proceeds to visit the nodes of the tree in a random order until no further improvements are found, meaning that this tree is a locally optimal for Problem (1). The problem is non-convex, so we repeat the coordinate descent process from various randomly-generated starting decision trees, before selecting the final locally-optimal tree with the lowest overall objective value as the best solution. For a more comprehensive guide to the coordinate descent process, we refer the reader to Bertsimas and Dunn[4].

Although only one tree model is ultimately selected, information from multiple trees generated during the training process is also used to improve the performance of the algorithm. For example, the Optimal Trees algorithm combines the result of multiple trees to automatically calibrate the complexity parameter (α) and to calculate variable importance scores in the same way as random forests or boosted trees. More detailed explanations of these procedures can be found in Dunn[13].

The coordinate descent approach used by Optimal Trees is generic and can be applied to optimize a decision tree under any objective function. For example, the Optimal Trees framework can train Optimal Classification Trees (OCT) by setting $\texttt{error}(T, D)$ to be the misclassification error associated with the tree predictions made on the training data. We provide a comparison of performance between various classification methods from Bertsimas and Dunn[4] in Figure 1. This comparison shows the performance of two versions of Optimal Classification Trees: OCT with parallel splits (using one variable in each split); and OCT with

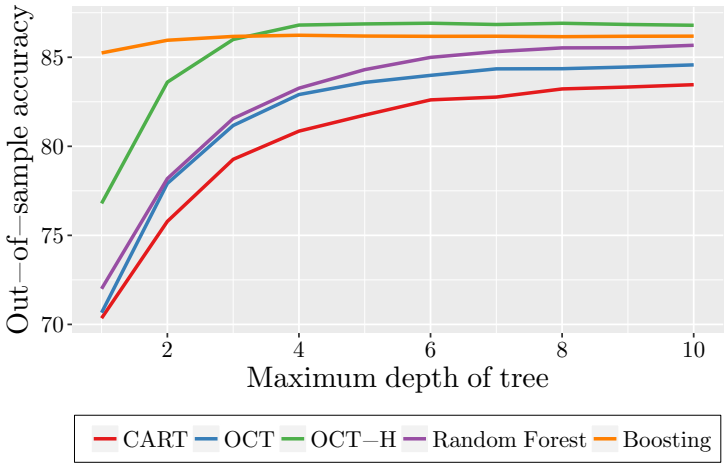

Figure 1: Performance of classification methods averaged across 60 real-world datasets. OCT and OCT-H refer to Optimal Classification Trees without and with hyperplane splits, respectively.

hyperplane splits (using a linear combination of variables in each split). These results demonstrate that not only do the Optimal Tree methods significantly outperform CART in producing a single predictive tree, but also that these trees have performance comparable with some of the best classification methods.

In Section 4, we will extend the Optimal Trees framework to work with censored data and generate Optimal Survival Trees.

4 Survival tree algorithm

In this section, we adapt the Optimal Trees algorithm described in Section 3 for the analysis of censored data. For simplicity, we will use terminology from survival analysis and assume that the outcome of interest is the time until death. We begin with a set of observations $(t_i, \delta_i)_{i=1}^n$ where t_i indicates the time of last observation and δ_i indicates whether the observation was a death ($\delta_i = 1$) or a censoring ($\delta_i = 0$).

Like other tree algorithms, the OST model requires a target function that determines which splits should be added to the tree. Computational efficiency is an important factor in the choice of target function, since it must be re-evaluated for every potential change to the tree during the optimization procedures. A key requirement for the target function is that the "fit" or error of each node should be evaluated independently of the rest of the tree. In this case, changing a particular split in the tree will only require re-evaluation of the subtree directly below that split, rather than the entire tree. This requirement restricts the choice of target function to the node purity approaches described in Section 2.

The splitting rule implemented in the OST algorithm is based on the likelihood method proposed by LeBlanc and Crowley [25]. This splitting rule is derived from a proportional hazards model which assumes that the underlying survival distribution for each observation is given by

$$\mathrm{P}(S_i \leq t) = 1 - e^{-\theta_i \Lambda(t)}, \tag{2}$$

where $\Lambda(t)$ is the baseline cumulative hazard function and the coefficients θ_i are the adjustments to the baseline cumulative hazard for each observation.

In a survival tree model we replace $\Lambda(t)$ with an empirical estimate for the cumulative probability of death at each of the observation times. This is known as the Nelson-Aalen estimator [1,27],

$$\hat{\Lambda}(t) = \sum_{i:t_i \leq t} \frac{\delta_i}{\sum_{j:t_j \geq t_i} 1}. \tag{3}$$

Assuming this baseline hazard, the objective of the survival tree model is to optimize the hazard coefficients θ_i. We impose that the tree model uses the same coefficient for all observations contained in a given leaf node in the tree, i.e. $\theta_i = \hat{\theta}_{T(i)}$. These coefficients are determined by maximizing the within-leaf sample likelihood

$$L = \prod_{i=1}^{n} \left(\theta_i \frac{d}{dt} \Lambda(t_i) \right)^{\delta_i} e^{-\theta_i \Lambda(t_i)}, \tag{4}$$

to obtain the node coefficients

$$\hat{\theta}_k = \frac{\sum_i \delta_i I_{\{T(i)=k\}}}{\sum_i \hat{\Lambda}(t_i) I_{\{T(i)=k\}}}. \tag{5}$$

To evaluate how well different splits fit the available data we compare the current tree model to a tree with a single coefficient for each observation. We will refer to this as a fully saturated tree, since it has a unique parameter for every observation. The maximum likelihood estimates for these saturated model coefficients are

$$\hat{\theta}_i^{sat} = \frac{\delta_i}{\hat{\Lambda}(t_i)}, \quad i = 1, \ldots, n. \tag{6}$$

We calculate the prediction error at each node as the difference between the log-likelihood for the fitted node coefficient and the saturated model coefficients at that node:

$$\texttt{error}_k = \sum_{i:T(i)=k} \left(\delta_i \log \left(\frac{\delta_i}{\hat{\Lambda}(t_i)} \right) - \delta_i \log(\hat{\theta}_k) - \delta_i + \hat{\Lambda}(t_i) \hat{\theta}_k \right). \tag{7}$$

The overall error function used to optimize the tree is simply the sum of the errors across the leaf nodes of the tree T given the training data D:

$$\texttt{error}(T, D) = \sum_{k \in \text{leaves}(T)} \texttt{error}_k(D). \tag{8}$$

We can then apply the Optimal Trees approach to train a tree according to this error function by substituting this expression into the overall loss function (1). At each step of the coordinate descent process, we determine new estimates for $\hat{\theta}_k$ for each leaf node k in the tree using (5). We then calculate and sum the errors at each node using (7) to obtain the total error of the current solution, which is used to guide the coordinate descent and generate trees that minimize the error (8).

5 Results

In this section we evaluate the performance of the Optimal Survival Trees (OST) algorithm and compare it to two existing survival tree models available in the R packages **rpart** and **ctree**. At the end of the section we provide an example of a real-world application of the OST algorithm to a Coronary Heart Disease dataset.

5.1 Synthetic datasets

Censored data poses a challenge when evaluating the fit of survival models, since it is difficult to measure predictive accuracy when the outcome of interest is only partially observed. For this reason, we test our algorithm on synthetic datasets where exact survival times are known for all observations. We generate artificial censoring times in order to create censored training datasets, but we evaluate the accuracy of the models with respect to the actual distributions used to generate the data.

The procedure for generating synthetic datasets is as follows:

1. Randomly generate 20000 observations of three uniform continuous covariates and three uniform discrete random variables with 2, 3 and 5 levels.

2. Generate a random "ground truth" tree model to partition the dataset based on these six covariates. Assign a survival distribution to each leaf node in the tree (see Appendix for additional details).

3. Classify observations into "true" node classes $C(i)$ according to the ground truth model. Generate a survival time, s_i, for each observation based the survival distribution of its node: $S_i \sim F_{C(i)}(t)$.

4. Generate censoring times $c_i = \kappa(1-u_i^2)$, where u_i follows a uniform distribution and κ is a non-negative parameter used to control the proportion of censored individuals.

5. Assign observation times $t_i = \min(s_i, c_i)$. Individuals are marked as censored ($\delta_i = 0$) if $t_i = c_i$.

The synthetic datasets described above contain comprehensive survival information: exact survival times for all observations and the "true" tree structure

with node classes $C(i)$ and survival distributions $F_{C(i)}(t)$. Note that the survival distribution of each observation is entirely determined by its place in the ground truth tree model.

The structure of these synthetic datasets has been deliberately selected to facilitate a clear evaluation of the performance of survival tree algorithms. The existence of a "true" tree provides an unambiguous target against which we can measure the accuracy of various models. In this context, a perfect survival tree model should be able to achieve the following objectives:

1. Recover the true tree structure.
2. Recover the corresponding survival distributions.

The next section provides a brief discussion of the relevance of these objectives in survival tree applications, followed by a description of the performance metrics that we will use to measure how well these objectives are met.

5.2 Survival tree accuracy metrics

There is no general consensus in the literature on the best accuracy metric for survival trees[17], and many of the proposed metrics are only suitable under additional assumptions (such as a parametric model). A key difficulty in selecting performance metrics for survival tree models is that the definition of "accuracy" can depend on the context in which the model will be used.

For example, consider a survival tree that models the relationship between lifestyle factors and age of death. A medical researcher may use such a model to identify risk factors associated with early death, while an insurance firm may use this model to estimate the volume of life insurance policy pay-outs in the coming years. The medical researcher is primarily interested in objective 1 (whether the model has identified important splits), while the insurer is more focused on objective 2 (whether the model can accurately estimate survival distributions).

In subsequent sections we refer to these two performance criteria as *classification accuracy* and *prediction accuracy*. It is important to recognize that these two objectives are not necessarily consistent. In small datasets, trees with high classification accuracy may contain many splits and have a small number of observations in each leaf node, but the survival estimates obtained from these node populations will be noisy and have low prediction accuracy. Our synthetic tests measure both the classification and prediction accuracy in order to provide a comprehensive overview of the performance of the OST algorithm.

5.2.1 Classification accuracy metrics

For classification accuracy, we consider the following two metrics:

1. **Node homogeneity**

   Consider a "ground truth" tree C in which each observation i is assigned to a class $C(i)$ that represents a node in the true tree. We measure node

homogeneity in a new tree model T by counting the proportion of the observations in each node $k \in T$ that have the same true class in C. Let $p_{k,l}$ be the proportion of observations in node $k \in T$ that came from class $\ell \in C$ and let $n_{k,l}$ be the total number of observations at node $k \in T$ from class $\ell \in C$. Then,

$$NH = \frac{100}{n} \sum_{k \in T} \sum_{l \in C} n_{k,l} p_{k,l}. \tag{9}$$

A score of $NH = 100$ indicates that each node in the new tree model contains observations from a single class in the dataset. This does not necessarily mean that the new tree matches the true classifications. For example, a saturated tree with a single observation in each node would have a perfect node homogeneity score (see Figure 2). The node homogeneity metric is therefore biased towards larger trees with few observations in each node.

2. **Class recovery**

   Class recovery is a measure of how well a new tree model is able to keep similar observations together in the same node, thereby avoiding unnecessary splits. Class recovery is calculated by counting the proportion of observations from a true class $\ell \in C$ that are placed in the same node in T. Let $q_{k,l}$ be the proportion of observations from class $\ell \in C$ that are classified in node $k \in T$ and let $n_{k,l}$ be the total number of observations at node $k \in T$ from class $\ell \in C$. Then,

$$CR = \frac{100}{n} \sum_{\ell \in C} \sum_{k \in T} n_{k,l} q_{k,l}. \tag{10}$$

   This metric is biased towards smaller trees, since a null tree with a single node would have a perfect class recovery score. It is therefore useful to consider both the class recovery and node homogeneity scores simultaneously in order to assess the performance of a tree model (see Figure 2 for examples).

The objective of these classification metrics is to answer the following question: given survival data with a known "true" tree structure, how well can the algorithm recover that structure? These metrics are intended to provide a theoretical benchmark for the classification accuracy of survival tree algorithms in synthetic datasets.

The node homogeneity and class recovery scores can also be used in real-world datasets to compare any two tree models, T_1 and T_2. In this case, these metrics should be interpreted as a measure of structural similarity between the two tree models. Note that when T_1 and T_2 are applied to the same dataset, the node homogeneity for model T_1 relative to T_2 is equivalent to the class recovery for T_2 relative to T_1, and vice versa. The average node homogeneity score for T_1 and T_2 is therefore equal to the average class recovery score for T_1 and T_2. We will refer to this average as the *similarity score* for models T_1 and T_2.

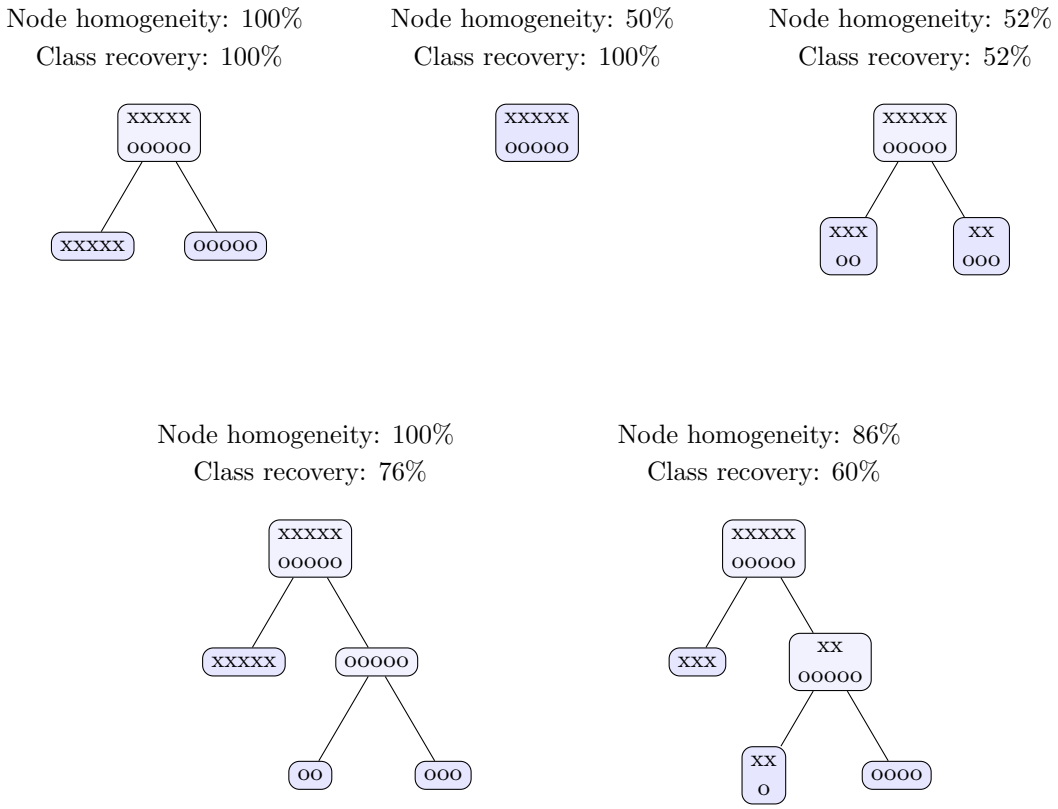

Figure 2: Classification accuracy metrics for survival trees.

5.2.2 Prediction accuracy metrics

Our prediction accuracy metrics focus on the non-parametric Kaplan-Meier curves produced at each leaf of the survival tree models. We use these curves to estimate the survival distribution of a node population and evaluate how well these predictions match the observations in an unseen testing dataset. We use the following two metrics to measure prediction accuracy:

1. **Area between curves (ABC)**

   The area between curves is a measure of how well the Kaplan-Meier estimates at each leaf capture the true survival distributions of synthetic data (see Figure 3 for illustration). For an observation i with true survival distribution $F_{C(i)}(t)$, suppose that $\hat{S}_{T(i)}(t)$ is the Kaplan-Meier estimate at the corresponding node in tree T. The area between the true survival curve and the tree estimate is given by

   $$ABC_i^T = \frac{1}{t_{max}} \int_0^{t_{max}} |1 - F_{C(i)}(t) - \hat{S}_{T(i)}(t)| dt. \tag{11}$$

   To make this metric easier to interpret, we compare the area between curves in a given tree to the score of a null tree with a single node (T_0). The area ratio (AR) is given by

   $$AR = 1 - \frac{\sum_i ABC_i^T}{\sum_i ABC_i^{T_0}}. \tag{12}$$

   Similar to the popular R^2 metric for regression models, the AR indicates how much accuracy is gained by using the Kaplan-Meier estimates generated by the tree relative to the baseline accuracy obtained by using a single estimate for the whole population. To the best of our knowledge, this particular accuracy metric has not appeared in previous publications.

2. **Integrated Brier score**

   The Brier[8] score metric is commonly used to evaluate classification trees and was modified by Graf et al.[17] for trees with censored data. The integrated Brier score (IBS) uses Kaplan-Meier estimates for both the survival distribution, $1 - \hat{S}(t)$, and the censoring distribution, $1 - \hat{G}(t)$. The IBS for a given observation is

   $$IBS_i^T = \frac{1}{t_{max}} \int_0^{t_i} \frac{(1 - \hat{S}_{T(i)}(t))^2}{\hat{G}_{T(i)}(t)} dt + \frac{\delta_i}{t_{max}} \int_{t_i}^{t_{max}} \frac{(\hat{S}_{T(i)}(t))^2}{\hat{G}_{T(i)}(t_i)} dt, \tag{13}$$

   and the IBS for a tree is the average score across all observations. We report the Brier score ratio (BR), which compares the sum of the Brier scores in a given tree to the corresponding Brier scores in a null tree*:

   $$BR = 1 - \frac{\sum_i IBS_i^T}{\sum_i IBS_i^{T_0}}. \tag{14}$$

*Radespiel-Tröger et al.[29] calls this *explained residual variation*

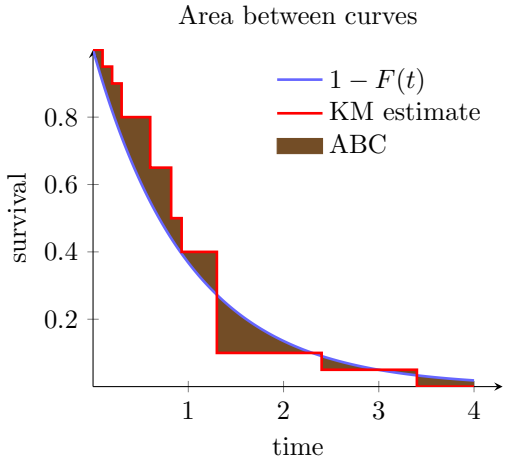

Figure 3: An illustration of the area between the true survival distribution and the Kaplan-Meier curve.

The IBS is suitable for both synthetic and real-world data, since these calculations do not require knowledge of the underlying survival distributions.

5.3 Synthetic results

We evaluated the performance of the OST algorithm relative to two existing algorithms available in the **R** packages **rpart**[31] and **ctree**[19]. Since neither **rpart** nor **ctree** have built-in methods for selecting tree parameters, we used a similar 5-fold cross-validation procedure for tuning all three algorithms. We considered tree depths up to three levels greater than the true tree depth and complexity parameter/significance values between 0.001 and 0.1 for the **rpart** and **ctree** algorithms (the OST complexity parameter is automatically selected during training). Equation (7) was used as the scoring metric used to evaluate out-of-sample performance during cross-validation, and the minimum node size for all algorithms was fixed at 5 observations.

The synthetic tests were run on 1000 datasets based on ground truth trees with a minimum depth of 3 and a maximum depth of 4 (i.e., $2^4 = 16$ leaf nodes). The median number of leaf nodes in the true trees was 6. Censoring was applied at nine different levels to generate examples with low censoring (0%, 10%, 20%), moderate censoring (30%, 40%, 50%) and high censoring (60%, 70%, 80%).

In each instance, 10000 observations were set aside for testing. Training datasets ranging from 100 to 10000 observations were drawn from the remaining data and used to train models with the OST, **rpart** and **ctree** algorithms.

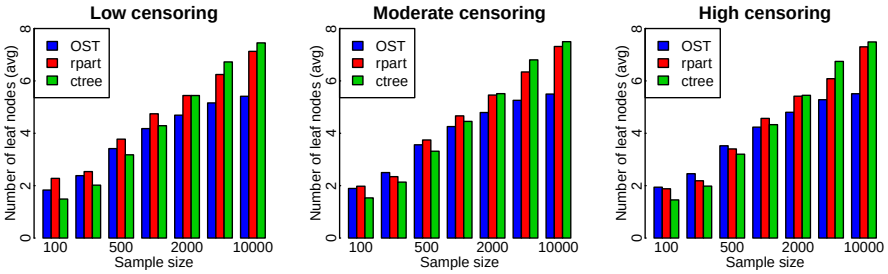

Figure 4: The average tree size for models trained on various sample sizes.

	Low censoring			Moderate censoring			High censoring		
n	rpart	ctree	OST	rpart	ctree	OST	rpart	ctree	OST
100	38/87	**40**/77	37/**93**	38/90	**40**/78	37/**92**	37/89	**40**/78	37/**90**
200	42/89	**45**/76	43/**91**	42/**90**	**46**/77	45/90	42/**91**	45/78	**45**/90
500	53/84	56/71	**57**/**88**	55/84	57/70	**59**/**88**	53/85	56/72	**59**/**88**
1000	63/82	66/63	**68**/**89**	65/82	67/63	**70**/**89**	64/82	66/64	**70**/**89**
2000	70/81	73/57	**76**/**89**	72/81	75/57	**78**/**90**	72/81	74/58	**78**/**90**
5000	76/80	82/53	**84**/**91**	77/80	83/53	**85**/**92**	77/80	82/53	**85**/**91**
10000	82/79	85/50	**87**/**91**	84/79	86/51	**89**/**92**	84/78	86/51	**88**/**91**

Table 1: A summary of the average node homogeneity/class recovery scores for synthetic experiments.

A summary of the average model size for each algorithm is provided in Figure 4. The amount of censoring did not have a significant impact on the size of the tree models, but there is a clear link between tree size and the number of training observations. All three algorithms produced relatively small trees when trained on small sample sizes. In larger datasets, the OST models grew to approximately the same size as the true tree models (6 nodes, on average), while the **rpart** and **ctree** models models were slightly larger.

5.3.1 Classification accuracy

The out-of-sample classification accuracy metrics for all three algorithms are summarized in Table 1 and Figure 5. The average node homogeneity/class recovery scores are given side-by-side to allow for a comprehensive assessment of each algorithm's performance. These results demonstrate that the OST models perform significantly better than the other two models across all censoring levels.

Node homogeneity scores for all three algorithms increase with larger sample size, indicating that the availability of additional data leads to better detection of relevant splits. In large populations, the OST algorithm selects more efficient splits than the other models and is able to achieve better node homogeneity with fewer splits.

The relationship between tree size and class recovery rates is somewhat more

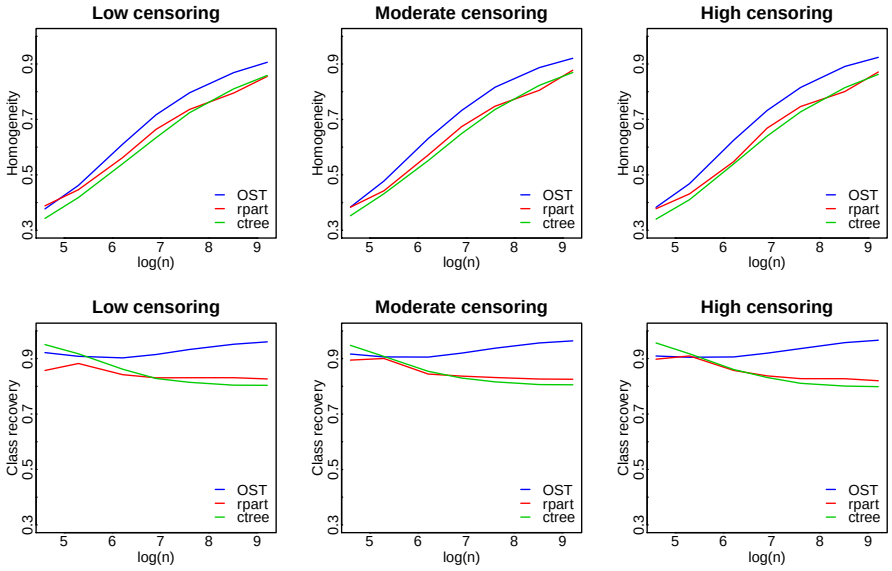

Figure 5: A summary of the classification accuracy metrics for survival tree algorithms.

complicated. In datasets smaller than 500 observations the class recovery rates seem to be closely linked to the tree size: the **ctree** models have the highest average class recovery for models trained on 100 and 200 observations, and also the smallest number of nodes (see Figure 4). However, this trend does not hold in datasets with 500 observations, where OST models are larger than the **ctree** models on average, but also have slightly better class recovery. This suggests that tree size is no longer a dominant factor in larger datasets ($n \geq 500$).

In these larger datasets we observe distinct trends in class recovery scores. The OST class recovery rate increases consistently despite the increases in model size, which means that the OST models are able to produce more complex trees without overfitting in the training data. By contrast, both of the other algorithms have consistently worse class recovery rates as sample size increases and their models become larger. Based on this trend, neither of these algorithms will reliably converge to the true tree.

5.3.2 Prediction accuracy

The out-of-sample prediction accuracy metrics for each of the three algorithms are summarized in Tables 2–3 and Figure 6. Overall, the results indicate that sample size plays the most significant role in out-of-sample accuracy across all three algorithms. There is also a small increase in accuracy when censoring is increased, which is due to the reduction in the maximum observed time, t_{max}. The OST results are generally better than the other algorithms across all sample

	Low censoring			Moderate censoring			High censoring		
n	rpart	ctree	OST	rpart	ctree	OST	rpart	ctree	OST
100	6.87	4.79	**9.30**	10.61	7.74	**11.01**	**10.79**	7.76	9.99
200	18.69	16.82	**20.99**	21.93	21.09	**25.25**	24.20	21.24	**26.13**
500	35.03	32.56	**41.17**	40.14	37.12	**47.16**	40.84	38.34	**48.21**
1000	51.27	44.29	**56.44**	57.28	49.68	**61.99**	58.86	51.30	**63.95**
2000	62.76	55.04	**67.97**	68.71	60.30	**73.53**	70.35	61.67	**75.31**
5000	72.62	66.94	**79.45**	77.26	71.63	**83.50**	79.22	72.38	**84.68**
10000	80.06	73.57	**84.41**	84.84	77.44	**87.77**	85.80	77.94	**88.72**

Table 2: A summary of the average Kaplan-Meier area ratio (AR) scores for synthetic experiments.

	Low censoring			Moderate censoring			High censoring		
n	rpart	ctree	OST	rpart	ctree	OST	rpart	ctree	OST
100	2.42	2.65	**3.06**	4.12	3.60	**4.22**	**3.84**	2.74	3.58
200	5.07	5.10	**5.68**	7.03	7.03	**8.03**	7.72	7.05	**8.38**
500	8.35	8.05	**9.46**	11.11	10.53	**12.61**	11.69	10.97	**13.34**
1000	10.74	10.00	**11.44**	14.12	12.80	**14.91**	15.03	13.46	**15.90**
2000	12.06	11.24	**12.59**	15.64	14.36	**16.20**	16.67	15.28	**17.29**
5000	12.88	12.41	**13.42**	16.53	15.70	**17.09**	17.55	16.56	**18.04**
10000	13.43	12.91	**13.71**	17.19	16.40	**17.40**	18.10	17.19	**18.29**

Table 3: A summary of the average Brier ratio (BR) scores for synthetic experiments.

sizes, although the performance gap is relatively small in smaller datasets.

To illustrate the effect of sample size on the accuracy of the Kaplan-Meier estimates, Figure 6 also shows the curve accuracy metrics for the "ground truth" tree structures that were used to generate the data. It is immediately apparent that even the true tree models produce poor survival curve estimates in small datasets. Based on these results, it may be necessary to increase the minimum node size to at least 50 observations in applications where Kaplan-Meier curves will be used to summarize survival tree nodes.

Since the Kaplan-Meier score cannot be used in practical examples, we also investigated how well this information is captured by the Brier score metric. We used both metrics to rank the tree models produced by each algorithm and found that they selected the same "best" model in approximately 83% of our test instances. This suggests that the Brier score can be used as a reasonable substitute for the Kaplan-Meier score when the underlying survival distributions are unknown.

5.3.3 Stability

A frequent criticism of single-tree models is their sensitivity to small changes in the training data. This may be apparent when a tree algorithm produces very different models for different training datasets sampled from the same popu-

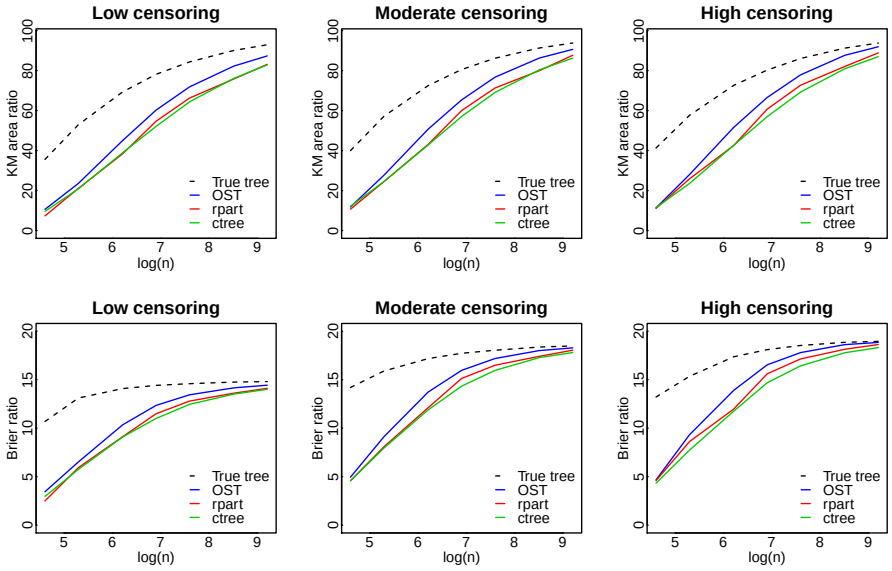

Figure 6: A summary of the prediction accuracy metrics for survival tree algorithms. Dashed and dotted lines correspond to instances with noise added to training data.

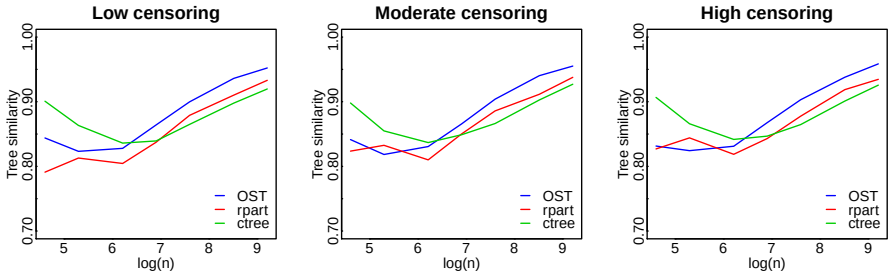

Figure 7: A summary of the average similarity scores between pairs of trees trained on mutually exclusive sets of observations.

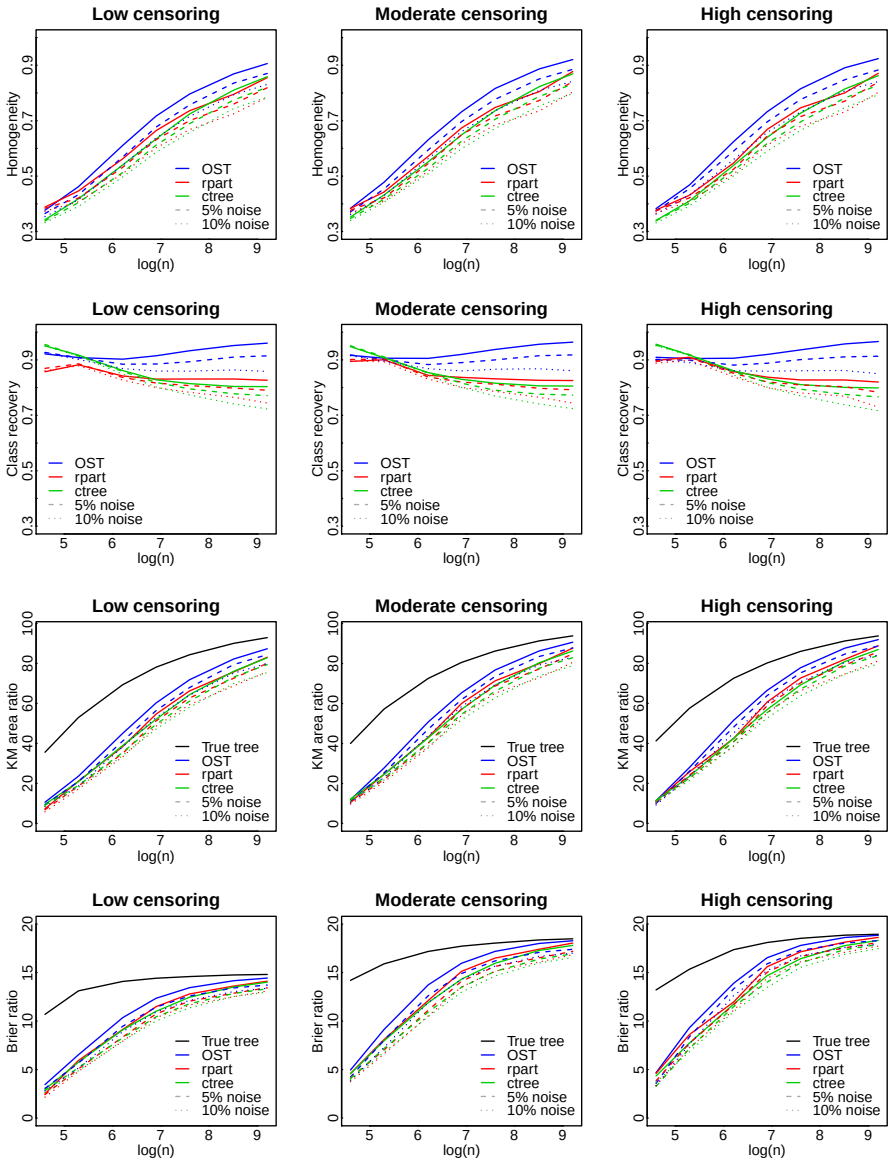

Figure 8: A summary of the prediction accuracy metrics for survival tree algorithms relative to the "ground truth" tree model.

lation. This type of instability is often an indication that the model will not perform well on unseen data.

Given the challenges associated with measuring out-of-sample accuracy for survival tree algorithms, it may be tempting to use stability as a performance metric for these models. Stability is a necessary condition for accuracy in tree models (provided that a tree structure is suitable for the data) but stable models are not necessarily accurate. For example, greedy tree models with depth 1 may select the same split for all permutations of the training data, but these models will not be accurate if the data requires a tree of depth 3.

Although stability is not necessarily a good indicator of the quality of a model, it is nevertheless interesting to consider how the stability of globally optimized trees may differ to the stability of greedy trees. Globally optimized trees are theoretically capable of greater stability because they may include splits that are not necessarily locally optimal for a particular training dataset. However, globally optimized trees also consider a significantly larger number of possible tree configurations and therefore have many more opportunities for overfitting on features of a particular training dataset.

We ran two sets of experiments to investigate the stability of the survival tree models in our synthetic datasets. In the first set of experiments we used each algorithm to train two models, T_1 and T_2, on completely separate training datasets of equal size. We then applied each model to the entire dataset (20000 observations) and used the tree similarity score described in Section 5.2.1 to assess the structural similarity between the two models. The average similarity scores for each algorithm are illustrated in Figure 7.

These results demonstrate that stability across different training datasets is not a sufficient condition for accuracy: models trained on 100 and 200 observations are both more stable and less accurate than models trained on 500 observations. The **ctree** algorithm produced the most stable results in smaller datasets due to the smaller model sizes selected during cross-validation. For example, 33.1% of **ctree** models trained on 100 observations had fewer than 2 splits, compared to 29.5% of the **rpart** models and 26.5% of the OST models.

The stability results for larger training datasets ($n > 1000$) are reasonably consistent with the accuracy results in Figure 6, and both stability and accuracy increase with sample size across all three algorithms. The OST models have the highest average similarity scores in large datasets and the **rpart** models are slightly more stable than the **ctree** models.

In the second set of stability experiments we investigated how small perturbations to the covariate values in the training dataset affect the out-of-sample accuracy of each model. We added noise to the training data by replacing the original continuous covariate values, x_{ij}, with "noisy" values $\tilde{x}_{ij} = x_{ij} + \epsilon_{ij}$. The initial covariates were uniformly distributed between 0 and 1 and the added noise terms were generated from the following two distributions:

$$\epsilon_{ij} \sim U(-0.05, 0.05) \qquad \text{(5\% noise), and}$$
$$\epsilon_{ij} \sim U(-0.1, 0.1) \qquad \text{(10\% noise).}$$

A similar approach was applied to the categorical variables, which were generated by rounding off continuous values (x_{ij} or $\tilde{x}_{ij}$) to the appropriate thresholds. Note that noise was only added to the observations used for training data; the testing data was unchanged.

The results of these experiments are contrasted with the initial outcomes (without added noise) in Figure 8. The effects of additional noise in the training data are visible in the results of all three algorithms and the drop in accuracy appears to be fairly consistent. Overall, the OST models maintain the best performance on each dataset.

These results indicate that perturbations in the training data affect the OST and greedy tree algorithms in similar ways. The OST algorithm's performance is diminished by adding noise to the training data, but its ability to consider a wider range of split configurations does not make it more sensitive to these perturbations. In fact, the OST algorithm is generally slightly more stable than the greedy algorithms across permutations of the training data because it tends to produce models that are consistently closer to the true tree.

5.4 Real data

In this section, we provide an example of a practical application of the OST algorithm to a real-world dataset from the FHS. The FHS successfully identified first the common factors or characteristics that contribute to CHD using the Cox regression model[11]. In our survival tree model, we include all participants diagnosed with Coronary Heart Disease (CHD) from the original cohort (1948-2014) and the offspring cohort (1971-2014). The event of interest in this model is the occurrence of a myocardial infarction or stroke. All patients were followed for a period of at least 10 years after their first diagnosis of CHD; observations are marked as censored if no event was observed while the patient was under observation.

We applied our algorithm to the primary variables that have been used in the established 10-year Hard Coronary Heart Disease (HCHD) Risk Calculator and the Cardiovascular Risk Calculator[14,15]. For each participant who was diagnosed with CHD, we include the following covariates in our training dataset: gender, smoking status (smoke), Systolic Blood Pressure (SBP), Diastolic Blood Pressure (DBP), use of anti-hypertensive medication (AHT), Body Mass Index (BMI), diabetic status (diabetes). We did not include cholesterol levels in our analysis because these variables are highly correlated with the use of lipid lowering treatment and a high proportion of the sample population did not have sufficient data to account for this interaction.

In Figure 9 we illustrate the output of our algorithm on the FHS dataset. Every node of the tree provides the following information:

- The node number.
- Number of observations classified into the node.
- Proportion of the node population which has been censored.

- A plot of survival probability vs. time. In this example, the x-axis represents age and the y-axis gives the Kaplan-Meier estimate for the probability of experiencing no adverse events.
- Color-coded survival curves to describe the different sub-populations. In each node, the blue curves describe the individuals classified into that node.
- In internal (parent) nodes, the orange/green curves describe the sub-populations that are split into the left/right child node. After each split, the sub-population with higher likelihood of survival goes into the left node.
- In leaf nodes, the red curve shows the average survival curve for the entire tree. This facilitates easy comparisons between the survival of a specific node and the rest of the population.

The splits illustrated in Figure 9 include known risk factors for heart disease and make sense from a medical perspective. The algorithm identified a BMI threshold of 25 as the first split (node 1), which is in accordance with the NIH BMI ranges that classify an individual as overweight if his/her BMI is greater than or equal to 25. Multiple splits indicated a higher risk of heart attack or stroke in patients who smoke (nodes 2, 6). The group with the highest risk of an adverse event was overweight patients with diabetes (node 5).

Figures 10 and 11 illustrate the output of the **ctree** and **rpart** algorithms applied to the same FHS population. The **rpart** model has a single split (BMI), while the **ctree** model contains the same variables as the OST output. The Brier scores for each model are 0.0486 (OST), 0.0249 (**rpart**) and 0.0467 (**ctree**).

The discrepancy in the Brier scores for the OST and **ctree** models is due to slight differences in the threshold and position of certain splits. For example, both methods identify that BMI is the most appropriate variable for the first split, but the BMI threshold differs. The **ctree** model sets the splitting threshold to 24.117, which is the locally optimal value for the split when building the tree greedily (the same threshold is used in the **rpart** model). By contrast, the OST algorithm selects a threshold of 25.031. This example demonstrates how the OST algorithm's efforts to find a globally optimal solution differ from the results of locally optimal splits.

A second difference between the tree models is the order of the smoking and diabetes splits within the overweight population. The **ctree** model splits on smoking first, since this split has the most significant p-value of the variables at node 5 in the **ctree** tree. The algorithm also recognizes that diabetes is a risk factor and incorporates this in the subsequent split. Since greedy approaches like **ctree** do not reevaluate the spits once they have been decided, the algorithm does not recognize that the overall quality of the tree can be improved by reversing the order of these splits. This discrepancy in two otherwise similar trees highlights the advantages of the more sophisticated optimization conducted by OST.

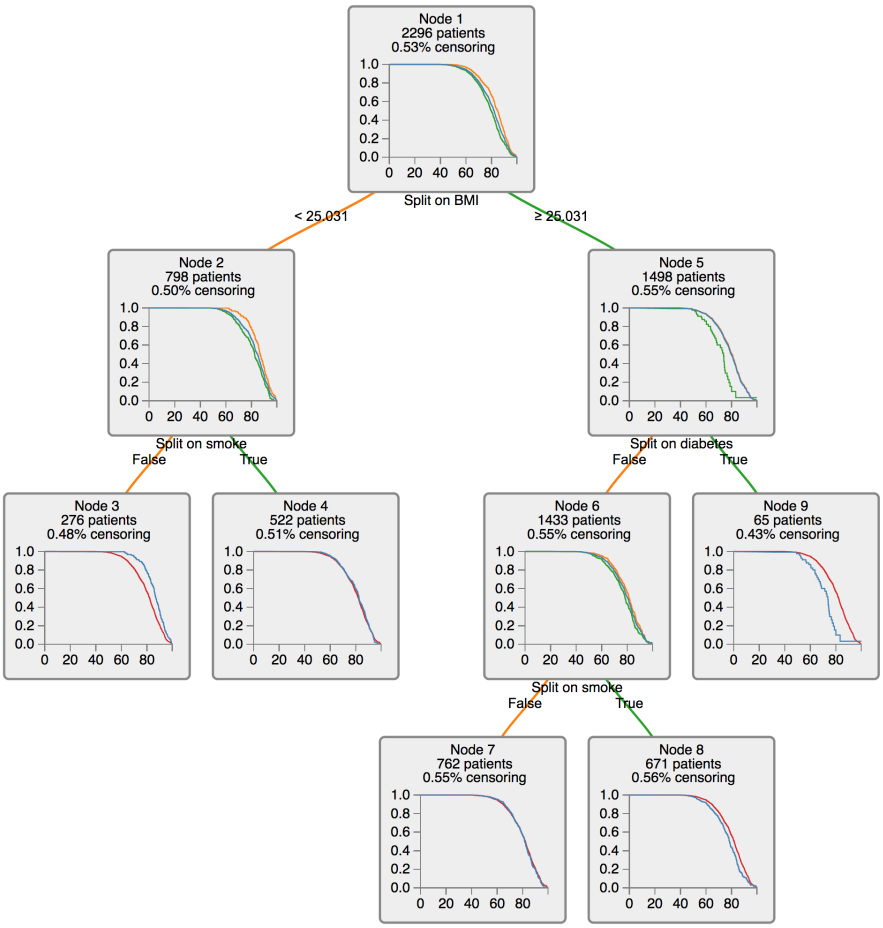

Figure 9: An illustration of Optimal Survival Trees for chd patients in the FHS.

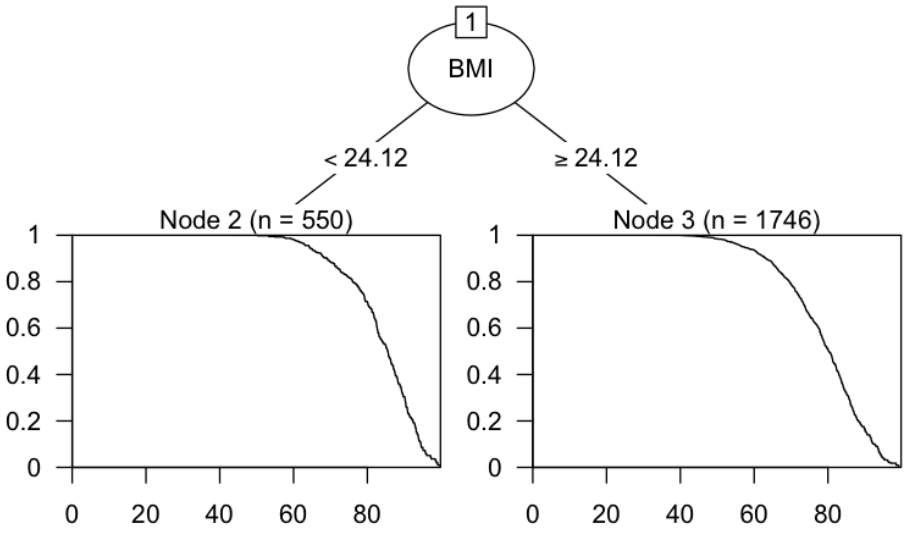

Figure 10: Illustration of the **rpart** output for chd patients in the FHS.

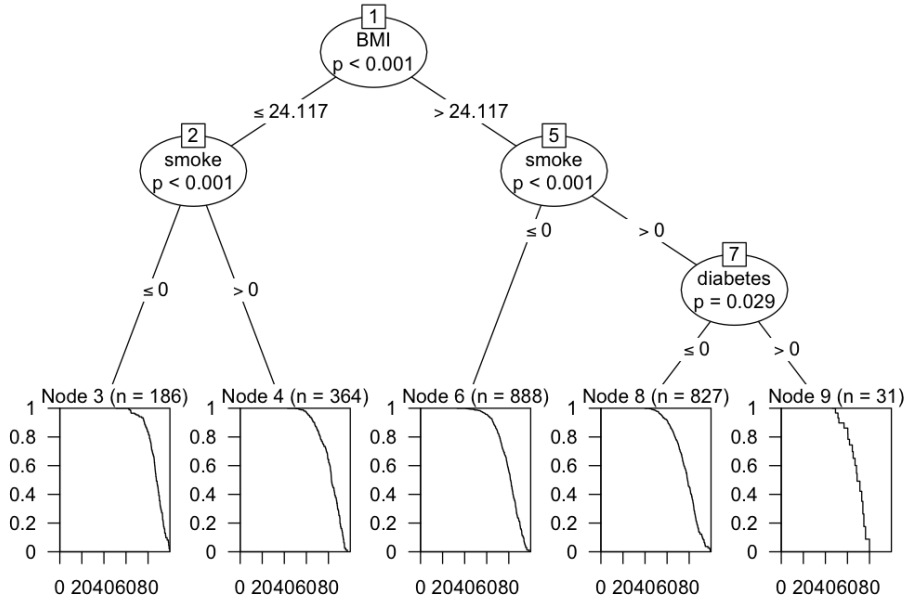

Figure 11: Illustration of the **ctree** output for chd patients in the FHS.

6 Conclusion

In this paper, we have extended the state-of-the-art Optimal Trees framework to generate interpretable models for censored data. We have also introduced a new accuracy metric, the Kaplan-Meier Area Ratio, which provides effective way to measure the predictive power of survival tree models in synthetic datasets.

The Optimal Survival Trees algorithm improves on the performance of existing algorithms in terms of both classification and predictive accuracy. Our results in synthetic datasets indicate that the OST models improve consistently with increasing sample size, whereas existing algorithms are prone to overfitting in larger datasets. This is particularly important, given that the volume of medical data available for research is likely to increase significantly over the coming years.

Acknowledgements This authors wish to thank the anonymous reviewers for their helpful comments on an earlier draft of this manuscript.

References

[1] Aalen, O. (1978) Nonparametric inference for a family of counting processes. *The Annals of Statistics*, 701–726.

[2] Bennett, K. and Blue, J. (1996) Optimal decision trees. *Rensselaer Polytechnic Institute Math Report*, **214**.

[3] Bertsimas, D. and Dunn, J. (2017) Optimal classification trees. *Machine Learning*, 1–44.

[4] Bertsimas, D. and Dunn, J. (2018) *Machine Learning under a Modern Optimization Lens*. Belmont: Dynamic Ideas. To appear.

[5] Bou-Hamad, I., Larocque, D. and Ben-Ameur, H. (2011) A review of survival trees. *Statistics Surveys*, **5**, 44–71.

[6] Breiman, L. (2002) Software for the masses. URL: `www.stat.berkeley.edu/ breiman/wald2002-3.pdf`.

[7] Breiman, L., Friedman, J., Stone, C. and Olshen, R. (1984) *Classification and regression trees*. CRC press.

[8] Brier, G. (1950) Verification of forecasts expressed in terms of probability. *Monthey Weather Review*, **78**, 1–3.

[9] Ciampi, A., Chang, C.-H., Hogg, S. and McKinney, S. (1987) Recursive partition: A versatile method for exploratory-data analysis in biostatistics. 23–50. Springer.

[10] Ciampi, A., Thiffault, J., Nakache, J. and Asselain, B. (1986) Stratification by stepwise regression, correspondence analysis and recursive partition: a comparison of three methods of analysis for survival data with covariates. *Computational statistics & data analysis*, **4**, 185–204.

[11] Cox, D. (1972) Regression models and life tables. *Journal of the Royal Statistical Society*, **34**, 187–220.

[12] Davis, R. and Anderson, J. (1989) Exponential survival trees. *Statistics in medicine*, **8**, 947–961.

[13] Dunn, J. (2018) *Optimal Trees for Prediction and Prescription*. Ph.D. thesis, Massachusetts Institute of Technology.

[14] D'Agostino, R., Vasan, R., Pencina, M., Wolf, P., Cobain, M., Massaro, J. and Kannel, W. (2008) General cardiovascular risk profile for use in primary care. *Circulation*, **117**.

[15] Expert Panel on Detection and Evaluation and Treatment of High Blood Cholesterol in Adults (2001) Executive summary of the third report of the national cholesterol education program (ncep) expert panel on detection, evaluation, and treatment of high blood cholesterol in adults (adult treatment panel III). *JAMA*, **285**.

[16] Gordon, L. and Olshen, R. (1985) Tree-structured survival analysis. *Cancer treatment reports*, **69**, 1065–1069.

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

[18] Grubinger, T., Zeileis, A. and Pfeiffer, K.-P. (2014) evtree: Evolutionary learning of globally optimal classification and regression trees in R. *Journal of statistical software*, **61**, 1–29. URL: `https://www.jstatsoft.org/v061/i01`.

[19] Hothorn, T., Hornik, K., Strobl, C. and Zeileis, A. (2010) Party: A laboratory for recursive partitioning.

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

[21] Hothorn, T., Lausen, B., Benner, A. and Radespiel-Tröger, M. (2004) Bagging survival trees. *Statistics in medicine*, **23**, 77–91.

[22] Ishwaran, H., Kogalur, U., Blackstone, E. and Lauer, M. (2008) Random survival forests. *The annals of applied statistics*, 841–860.

[23] Kaplan, E. and Meier, P. (1958) Nonparametric estimation from incomplete observations. *Journal of the American statistical association*, **53**, 457–481.

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

[25] LeBlanc, M. and Crowley, J. (1992) Relative risk trees for censored survival data. *Biometrics*, 411–425.

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

[27] Nelson, W. (1972) Theory and applications of hazard plotting for censored failure data. *Technometrics*, **14**, 945–966.

[28] R Core Team (2017) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL: `https://www.R-project.org/`.

[29] Radespiel-Tröger, M., Rabenstein, T., Schneider, H. and Lausen, B. (2003) Comparison of tree-based methods for prognostic stratification of survival data. *Artificial Intelligence in Medicine*, **28**, 323–341.

[30] Son, N. (1998) From optimal hyperplanes to optimal decision trees. *Fundamenta Informaticae*, **34**, 145–174.

[31] Therneau, T., Atkinson, B. and Ripley, B. (2010) The rpart package.

[32] Therneau, T., Grambsch, P. and Fleming, T. (1990) Martingale-based residuals for survival models. *Biometrika*, **77**, 147–160.

[33] Zhou, Y. and McArdle, J. (2015) Rationale and applications of survival tree and survival ensemble methods. *Psychometrika*, **80**, 811–833.

A Appendix

Distributions used in tree simulations

- Exponential(θ):

  Parameters: 0.3, 0.4, 0.6, 0.8, 0.9, 1.15, 1.5, 1.8

- Weibull(k,λ):

  Parameters: (0.8,0.4), (0.9,0.5), (0.9,0.7), (0.9,1.1), (0.9,1.5), (1,1.1),(1,1.9), (1.3,0.5)

- Lognormal(μ,σ^2):

  Parameters: (0.1,1.0), (0.2,.75), (0.3,0.3), (0.3,0.5), (0.3,0.8), (0.4,0.32), (0.5,0.3), (0.5,0.7)

- Gamma(k,θ):

  Parameters: (0.2,.75), (0.3,1.3), (0.3,2), (0.5,1.5), (0.8,1.0), (0.9,1.3), (1.4,0.9), (1.5,0.7)

Real dataset from FHS inclusion criteria

- Participation in the Original and Offspring cohort of the FHS.
- Formal diagnosis with chd (as indicated by the records of FHS).
- Participants outcomes were followed for 10 consecutive years after diagnosis.