Optimal Cox Survival Trees

Matei Mihai Mirica[1]

Supervisor(s): Emir Demirović [1], Jacobus G. M. van der Linden[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Matei Mihai Mirica
Final project course: CSE3000 Research Project
Thesis committee: Emir Demirović, Jacobus G. M. van der Linden, David Tax

Abstract

Survival analysis is a branch of statistics concerned with studying and estimating the expected time duration until some event, such as biological death, occurs. Survival distributions are fitted based on historical data, where some instances are censored, meaning that the actual time of the event is not known precisely. Survival trees extend on the classical statistical methods developed and can capture complex non-linear relations between the variables by recursively splitting the instances by generated rules and fitting a different survival distribution in each leaf. Moreover, decision trees are desirable models due to their interpretable nature. We extend existing optimal survival tree methods by considering Cox Proportional Hazard models in each leaf node, which allows us to find more complex yet interpretable relationships than existing methods. The experiments show that our model outperforms state-of-the-art methods for creating survival trees, SurTree, OST, and CTree, especially in determining the relative risks between out-of-sample observations while generating significantly smaller trees.

1 Introduction

Survival analysis is a statistical methodology that analyses the expected time duration until one event occurs. It is traditionally employed in medicine and clinical research, especially in studies developed to assess the effectiveness of different treatments or procedures over time. Still, it extends to other fields such as engineering, economics, or sociology. It aims to predict the time until an event occurs based on historical data, for which the proper time to event (death) for some instances is known. However, the difficulty of survival analysis arises from the fact that, for some other cases, the proper time to event is unknown. These instances are named censored, and only some lower or upper bounds on the time to event are known.

Several statistical techniques for approximating the survival function (probability that a subject survives longer than some time) have been developed throughout the years, including non-parametric approaches such as the Kaplan-Meier method [1] and the Nelson-Aalen estimator [2], [3].

While these methods can encapsulate censored data, they have some limitations, especially in their capacity to adjust for covariates - characteristics of individuals that may affect the risk of experiencing the event. These could include age, sex, or pre-existing conditions.

When adjustments for covariates are needed, other types of models, such as semi-parametric and parametric models, are better suited. Parametric survival models assume specific statistical distributions. They can provide efficient estimates if the assumed distribution closely resembles the underlying distribution.

Cox proportional hazards regression is a semi-parametric model that specifies the effect of covariates without requiring assumptions about the overall shape of the underlying distributions [4]. However, this technique alone assumes a specific form of the relationship between the covariates and the survival probability, which can be overly simplistic sometimes.

Models that are capable of detecting non-linear relations between covariates have been developed, including decision trees. They are most commonly trained using heuristics, such as CART [5] and CTree [6], generating trees that greedily select features to branch on.

While this usually leads to good results, optimal decision trees may yield better results. Optimal decision trees are methods that globally optimise the objective function over the training data for a given tree size limit. Typically, they also generalise to better out-of-sample performance than greedy heuristics [7], [8].

While finding optimal decision trees (for a given size limit) is an NP-hard problem [9], many mixed-integer programming (MIP) approaches [7], [10]–[12], MaxSAT approaches [13], or constraint programming (CP) approaches [14] have been developed to generate (near-)optimal trees. However, these techniques do not scale well to large datasets. In contrast, dynamic programming (DP) approaches outperform the MIP methods in runtime by several orders of magnitude [8], [15].

Decision trees offer a significant advantage due to their interpretability. Unlike complex black-box models that produce difficult-to-understand results, (small) decision trees provide unambiguous and interpretable outcomes [16], [17]. This clarity is particularly valuable today, as there is increasing demand for transparency and explainability in models, frequently driven by regulatory requirements, such as the European Union's General Data Protection Regulation (GDPR) and proposals like the Artificial Intelligence Act.

These attributes of decision trees make them an appropriate option for exploration within the context of survival analysis. In fields like medicine, where survival analysis is extensively applied, the interpretability of machine learning models is particularly valued.

Survival trees are constructed by recursively splitting instances by some features, then fitting a survival distribution in their leaves (as shown, for example, in Figure 1). Several techniques using decision trees for survival analysis have been developed [18]–[20]. They generate decision trees (either with dynamic programming or a local search approach) and fit some survival distribution in their leaves. These techniques, in particular, [18] and [19], fit a constant proportional hazard parameter model, which assumes a joint baseline survival distribution adjusted in each leaf by some parameter. Here, neither the baseline survival distribution nor the adjusted parameter are influenced by covariates, as they are relevant only for the splits. This motivates the goal of this research, which aims to generate optimal survival trees using the dynamic programming approach and fit a Cox Proportional Hazards Model in each leaf node. A Cox Proportional Hazards Model still assumes a common baseline survival distribution (per leaf), but the particular realization of an instance's variables adjusts the final distribution. Thus, each instance has its own individual survival distribution assigned.

Main Contributions: In this work, we show how fitting a Cox Proportional Hazards Model in the leaves of an opti-

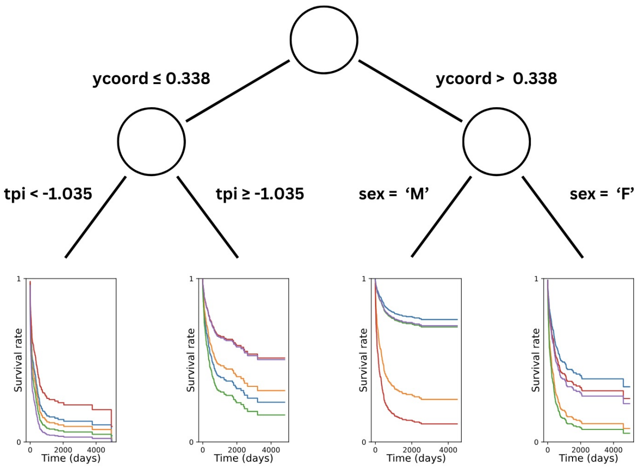

Figure 1: An example of a survival tree. Each leaf showcases five distributions, each corresponding to one instance in the respective leaf.

mal survival tree influences the prediction quality. This research extends on **SurTree** [18] by fitting a semi-parametric distribution in the tree leaves. Moreover, a detailed experimental setup and its result are presented to compare the performance of our new model to the state-of-the-art models of SurTree [18], Optimal Survival Trees (OST) [19], and Conditional Inference Trees (CTree) [6].

The following sections introduce the preliminaries for this work, including a discussion of SurTree [18]. We then present the method used to produce the optimal decision tree and its performance on both real and synthetic data.

2 Preliminaries

This section presents an overview of the preliminaries required for a thorough and clear understanding of the research. Section 2.1 lists definitions and notations used throughout the paper. A more detailed background on survival analysis is explained in Section 2.2. Lastly, we present commonly used survival metrics, which we also use to assess the performance of the models.

2.1 Terminology

In order to train the model, we are using a set of historical data $\mathcal{D}$. In the case of survival analysis, the instances that compose the data set are of two kinds:

- *uncensored* instances, for which we know they experienced the event of interest. For these, we also know the exact time when the event occurred.
- *censored* instances, for which we cannot tell whether they experienced the event of interest. For these, we cannot precisely tell when the event occurred, but we have some information about it.

For this study, we are only concerned with dealing with *right-censored-data*. That is, for the censored instances, we cannot tell exactly the time of the event, but we know some lower bounds of the actual value. This may happen, for example, when a patient left a clinical trial before the event happened. In this case, we know that the event **was not** observed before they left.

Each instance i belonging to the data set $\mathcal{D}$ is represented by a tuple $(t_i, \delta_i, fv_i, bv_i)$. The instance is characterised by its feature vector fv_i, which is composed of zero or more continuous features and zero or more categorical features and is a realisation of the feature space $\mathcal{F}$. We also need to consider (create) bv_i, the binarised adaption of fv_i. This means that for each feature j and each instance i it holds that $bv_{ij} \in \{0, 1\}$. The binarisation step is required as the framework of optimal decision trees is designed to work on binary predicates. Moreover, each feature is described by a censoring indicator $\delta_i \in \{0, 1\}$, representing whether the event of interest was observed ($\delta_i = 1$ if the event occurred, otherwise $\delta_i = 0$). Additionally, the observed time $t_i > 0$ denotes the exact time to event for uncensored data ($\delta_i = 1$) and a lower bound for the time to event for censored data ($\delta_i = 0$).

We define the following notations to describe the splits made in each decision node of a tree. A data set $\mathcal{D}$ can be split on one feature f_j. To represent this, we write $\mathcal{D}(f_j)$ to describe the subset of instances i for which $bv_{ij} = 1$ and analogously we write $\mathcal{D}(\overline{f_j})$ to denote the subset of instances i for which $bv_{ij} = 0$. Additionally, a dataset can be split by more than one feature. For instance, $\mathcal{D}(f_a, \overline{f_b}, f_c)$ represent the subset consisting of the instances i for which $bv_{ia} = 1 \wedge bv_{ib} = 0 \wedge bv_{ic} = 1$.

2.2 Survival analysis

Survival analysis aims to fit a survival function $S(t) = P(T \geq t)$, showing the probability of survival after a time t, with time T being the true time to event. Examples of survival functions are the distributions fit in the leaves of Figure 1. Complementary, the lifetime distribution function, usually denoted by $F(t) = P(T \leq t) = 1 - S(T)$, is defined as the probability of experiencing the event before time t. If F is differentiable, the event density function is defined as $f(t) = F'(t) = \frac{d}{dt}F(t)$.

The hazard function quantifies the rate of the event happening at time t, given that it did not before and is denoted by $\lambda(t) = \lim_{dt \to 0} \frac{\Pr(t \leq T < t+dt)}{dt \cdot S(t)} = \frac{f(t)}{S(t)} = \frac{\frac{d}{dt}F(t)}{S(t)} = \frac{\frac{d}{dt}(1-S(t))}{S(t)} = -\frac{S'(t)}{S(t)} = -\frac{d}{dt}logS(t)$. The cumulative hazard function, defined as the accumulation of the hazard function, can be written as $\Lambda(t) = \int_0^t \lambda(x)\, dx$. Integrating over the expression $\lambda(t) = -\frac{d}{dt}logS(t)$ presented above yields the following relation between the survival function and the cumulative hazard function:

$$S(t) = e^{-\Lambda(t)} \tag{1}$$

A non-parametric method devised for computing the cumulative hazard function is the Nelson-Aalen estimator [2], [3] and is defined as:

$$\hat{\Lambda}(t) = \sum_{t' \leq t} \frac{d(t')}{n(t')} \tag{2}$$

Where $d(t)$ represents the number of deaths at time t and $n(t)$ represents the number of individuals still at risk at time t.

$$d(t) = \sum_{i=1}^{|\mathcal{D}|} \mathbf{1}_{\{t_i = t \wedge \delta_i = 1\}} \tag{3}$$

$$n(t) = \sum_{i=1}^{|\mathcal{D}|} \mathbf{1}_{\{t_i \geq t\}} \tag{4}$$

Equations (1) and (2) can be used together to provide an estimation for the survival function $\hat{S}(t)$.

2.3 Survival metrics

Two commonly used metrics for assessing the performance of survival analysis models are *Harrell's C-Index* [21] and the *integrated Brier score* [22].

Harrell's C-Index, also commonly known as the concordance statistic, measures the proportion of correctly ordered comparable pairs of observations. We say that two observations i and j are comparable if one instance experienced the event of interest before the other ($t_i < t_j \wedge \delta_i = 1$). The relative order is determined by the risk score η associated with each instance. The two instances are *concordant* if the one experiencing the event first has a higher risk, *disconcordant* if the one experiencing the event first has a lower risk, and *tied-risk* if they have the same risk. The number of such pairs can be computed as follows:

$$CC = \sum_{i,j} \mathbf{1}(t_i < t_j \wedge \delta_i = 1)\mathbf{1}(\eta_i > \eta_j) \tag{5}$$

$$DC = \sum_{i,j} \mathbf{1}(t_i < t_j \wedge \delta_i = 1)\mathbf{1}(\eta_i < \eta_j) \tag{6}$$

$$TR = \sum_{i,j} \mathbf{1}(t_i < t_j \wedge \delta_i = 1)\mathbf{1}(\eta_i = \eta_j) \tag{7}$$

The C-Index can then be obtained as:

$$H_C = \frac{CC + 0.5 \cdot TR}{CC + TR + DC} \tag{8}$$

The advantage of this metric is that it does not make any parametric assumptions about the data. Note that a random predictor has an expected H_C of 0.5.

The **integrated Brier score** is another metric used for evaluating survival models. The Brier score [23] measures the accuracy of the probabilistic forecasts by evaluating the survival distribution at specific points. The integrated Brier score [22] was developed to measure the whole distribution and can be computed as:

$$IB = \frac{\sum_i \int_{t_{min}}^{t_i} \frac{(1-\hat{S}_i(t))^2}{\hat{G}(t)} dt + \delta_i \int_{t_i}^{t_{max}} \frac{(\hat{S}_i(t))^2}{\hat{G}(t_i)} dt}{|\mathcal{D}|(t_{max} - t_{min})} \tag{9}$$

The integrated Brier score computes the Brier score over a time interval $[t_{min}, t_{max}]$ and is weighted by the Kaplan-Meier estimator of the censoring distribution $\hat{G}(t)$. Importantly, this metric does not rely on any parametric assumptions about the underlying data.

3 Related Work

This section discusses previously done work related to the research presented. SurTree [18] is presented in Section 3.1. Section 3.2 details the Cox Proportional Hazards Model approach to estimate the survival distribution. Lastly, Section 3.3 presents the approach proposed in [24] that fits a Cox regression model.

3.1 SurTree

SurTree [18], an adaptation of MurTree [8] to survival analysis, fits optimal survival trees using a dynamic programming approach. This technique is guaranteed to produce the optimal tree for a given data set and tree size. It starts with the whole data set $\mathcal{D}$ in the root, splits it into two disjoint subsets $\mathcal{D}_1$ and $\mathcal{D}_2$, and recursively proceeds to split the subsets. A survival function is fitted in the leaves, and a loss metric is computed. Each split is done by selecting a feature j and then computing $\mathcal{D}(f_j)$ and $\mathcal{D}(\overline{f_j})$. The feature is chosen to minimise the sum of losses over the two resulting sub-trees.

The main idea at the basis of SurTree is to fit a baseline hazard function $\hat{\Lambda}(t)$, described in equation (2), which is shared by all the leaves, then fit different distribution in each leaf i by adjusting the common hazard function by some parameter θ_i:

$$\hat{S}_i(t) = e^{-\theta_i \hat{\Lambda}(t)} \tag{10}$$

The effect of θ on the survival distribution is illustrated in Figure 2.

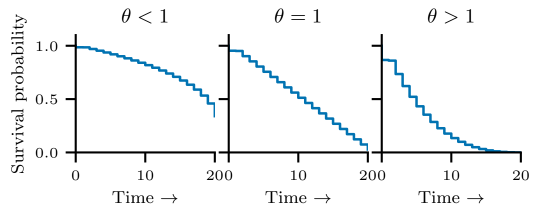

Figure 2: A visualization of how θ affects a survival distribution $\hat{S}(t)$. Every plot share the same $\hat{\Lambda}(t)$, but use θ = 0.3, θ = 1 and θ = 3 respectively.

Then, as similarly done in Optimal Survival Trees [19], the optimal value for a given data set $\mathcal{D}$ is proved in [25] to be:

$$\hat{\theta} = \frac{\sum_{(t_i,\delta_i,fv_i)\in\mathcal{D}} \delta_i}{\sum_{(t_i,\delta_i,fv_i)\in\mathcal{D}} \hat{\Lambda}(t_i)} \tag{11}$$

The optimal saturated value for a single instance i is then given by:

$$\hat{\theta}_i^{sat} = \frac{\delta_i}{\hat{\Lambda}(t_i)} \tag{12}$$

The loss for a single instance is defined as the difference between the log-likelihood of the instance when setting θ to be the one of the leaf and the log-likelihood of the instance when setting θ to be the saturated one.

This allows to formulate the loss function for a given data set $\mathcal{D}$ and a fixed value θ as:

$$\mathcal{L}(\mathcal{D},\hat{\theta}) = \sum_{(t_i,\delta_i,fv_i)\in\mathcal{D}} \left(\hat{\Lambda}(t_i)\hat{\theta} - \delta_i \log \hat{\Lambda}(t_i) - \delta_i \log \hat{\theta} - \delta_i\right) \tag{13}$$

Having all of the information presented, it is now possible to formulate the dynamic programming approach [8] that allows for the construction of the SurTree [18]:

$$T(\mathcal{D},d,n) = \begin{cases} \min_{\hat{\theta}} \mathcal{L}(\mathcal{D},\hat{\theta}) & \text{if } n = 0 \\ T(\mathcal{D},d,2^d-1) & \text{if } n > 2^d - 1 \\ T(\mathcal{D},n,n) & \text{if } d > n \\ \min\Big\{T(\mathcal{D}(\overline{f}),d-1,n-i-1) + \\ \quad T(\mathcal{D}(f),d-1,i) : \\ \quad f \in F, i \in [0,n-1]\Big\} & \text{otherwise} \end{cases} \tag{14}$$

In the presented formula, $\mathcal{D}$ accounts for the data set of instances to fit, d is the maximum depth of the tree, and n denotes the *node budget*, the number of decision nodes to include in the tree. The first case, $n = 0$, is concerned with fitting the model in a leaf where no more splits occur. It finds an optimal estimate for $\hat{\theta}$ that minimises the loss function described in equation (13). The following two cases, $n > 2^d - 1$ and $d > n$, re-adjust the depth or node limits accordingly, as the two impose bounds on each other. The last case tries to find the optimal split by trying all features and all combinations of node budgets and selecting the one that minimises the sum of the two sub-components. The solutions $T(\mathcal{D},d,n)$ are also cached to improve the scalability of the algorithm.

3.2 Cox Proportional Hazards Model

In contrast to the constant proportional model used in SurTree [18], a Cox model is adjusted to account for the covariates when fitting the hazard function. In a proportional hazards model, each one-unit increase in a covariate multiplies the hazard rate by a constant factor. For example, accounting for patients' age or blood pressure levels may be relevant when estimating a more accurate survival distribution.

The Cox model consists of two parts: an underlying baseline hazard function, denoted by $\lambda_0(t)$, shared by all the instances, and the effect parameters, denoted by β, describing the influence of the covariates on the hazard function. Thus, each instance is going to have distinct hazard and survival functions, depending on the realisations of the covariates. The hazard function of an instance i, introduced in the Cox model, is given by:

$$\lambda_i(t) = \lambda_0(t) * e^{\sum_{j=1}^{p} \beta_j * X_{i,j}} \tag{15}$$

Here, $X_i = (X_{i,1}, X_{i,2}, ..., X_{i,p})$ represent the realisation for the features of instance i. Thus, the hazard function for each individual is the baseline hazard function scaled by the exponential of the linear combination of the fitted coefficients β and the covariates. Integrating equation (15) results in the cumulative hazard function:

$$\begin{aligned} \Lambda_i(t) &= \int_0^t \lambda_i(x)\,dx = \int_0^t \lambda_0(x) * e^{\sum_{j=1}^{p} \beta_j * X_{i,j}}\,dx \\ &= e^{\sum_{j=1}^{p} \beta_j * X_{i,j}} * \int_0^t \lambda_0(x)\,dx \\ &= \Lambda_0(t) * e^{\sum_{j=1}^{p} \beta_j * X_{i,j}} \end{aligned} \tag{16}$$

For a given dataset of instances $\mathcal{D} = \{X_1, X_2, ..., X_n\}$, $n = |\mathcal{D}|$, Breslow [26] suggested an estimator of the baseline cumulative hazard function in a discussion on Cox's paper as:

$$\hat{\Lambda}_0(t) = \sum_{t' \leq t} \frac{d(t')}{\sum_{i=1}^{n} \mathbf{1}_{\{t_i \geq t'\}} * e^{\beta^T X_i}} \tag{17}$$

Combining the results (16) and (17), by applying a rationale similar to the one in equation (1), we can obtain the following estimation for the survival function of instance i:

$$\hat{S}_i(t) = e^{-\hat{\Lambda}_0(t) * e^{\beta^T X_i}} \tag{18}$$

The Cox partial likelihood is obtained by plugging the Breslow estimator in the full likelihood function and is given by:

$$L(\beta) = \prod_{i\in[1,n]:\delta_i=1} \frac{e^{\beta^T X_i}}{\sum_{j=1}^{n} \mathbf{1}_{\{t_j \geq t_i\}} * e^{\beta^T X_j}} \tag{19}$$

The goal of the Cox Proportional Hazards model is to fit the coefficients $\hat{\beta}$ that maximise the partial likelihood, or equivalently, that minimise the negative partial log-likelihood:

$$\hat{\beta} = \arg\min_{\beta}(-\sum_{i\in[1,n]:\delta_i=1} (\beta^T X_i - log \sum_{j=1}^{n} \mathbf{1}_{\{t_j \geq t_i\}} * e^{\beta^T X_j})) \tag{20}$$

3.3 Regularisation Paths for Cox's Proportional Hazards Model via Coordinate Descent

The goal for the Cox model is to fit the set of coefficients β that maximise (minimise) the (negative log-)likelihood, shown in (19) and (20). In order to do so, for this research, we have opted to base the implementation on the technique presented in [24], which generates a path of regularised Cox's Proportional Hazards Model solutions. An elastic net penalisation is applied to the objective function, and the coefficients are optimised via a coordinate descent approach. Thus, the new goal is to find the coefficients $\beta = (\beta_1, \beta_2, ..., \beta_p)$ minimising the adjusted negative log-likelihood:

$$\hat{\beta} = \arg\min_{\beta}(-\ell(\beta) + \lambda * P_\alpha(\beta)) \tag{21}$$

where:

$$\ell(\beta) = \sum_{i \in [1,n]:\delta_i = 1} (\beta^T * X_i - log \sum_{j=1}^{n} \mathbf{1}_{\{t_j \geq t_i\}} * e^{\beta^T * X_j}) \tag{22}$$

$$\lambda * P_\alpha(\beta) = \lambda * (\alpha * \sum_{i=1}^{p} |\beta_i| + (1-\alpha) * \sum_{i=1}^{p} \beta_i^2) \tag{23}$$

In [24], the equation is also scaled for convenience. The coefficient λ scales the regularization term, while α and $1-\alpha$ account for the Lasso (l_1) and Ridge (l_2) penalties. In contrast to the standard Cox model, with no regularisation, this method can find good solutions for cases where the number of features is comparable or larger than the number of instances.

This algorithm fits a *path* of solutions: it does not generate only one solution, but several of them for different values of λ. Precisely, it produces a path of solutions: $(\lambda_1, \hat{\beta_1}), (\lambda_2, \hat{\beta_2}), ..., (\lambda_k, \hat{\beta_k})$, where $\lambda_1 > \lambda_2 > ... > \lambda_k$. A warm starts technique is employed, where an initial solution $\hat{\beta}_1$ for $\lambda_1 = \lambda_{max}$ is generated. Then, the previously found solution is optimised sequentially for the new value of λ.

In [24], a two-term Taylor series expansion is used to approximate $l(\beta)$. For $X = (X_1, X_2, ..., X_n)$ the design matrix and $\eta = X\beta$, the following approximation can be computed around $\tilde{\beta}$:

$$\ell(\beta) \approx \ell(\tilde{\beta}) + (\beta - \tilde{\beta})^T \ell'(\tilde{\beta}) + \frac{1}{2}(\beta - \tilde{\beta})^T \ell''(\tilde{\beta})(\beta - \tilde{\beta}) \tag{24}$$

where ℓ' and ℓ'' denote the gradient and the Hessian matrix (here, with respect to $\tilde{\beta}$). Moreover, in [24] is shown that by letting $\tilde{\eta} = X\tilde{\beta}$, $\ell(\beta)$ can be approximated as:

$$\ell(\beta) \approx \frac{1}{2}(z(\tilde{\eta}) - X\beta)^T \ell''(\tilde{\eta})(z(\tilde{\eta}) - X\beta) + C(\tilde{\eta}, \tilde{\beta}) \tag{25}$$

where $z(\tilde{\eta}) = \tilde{\eta} - \ell''(\tilde{\eta})^{-1}\ell'(\tilde{\eta})$ and $C(\tilde{\eta}, \tilde{\beta})$ does not depend on β. To speed up the computation of $\ell''(\tilde{\eta})$, a matrix with $\mathcal{O}(n^2)$ entries, only its diagonal values are being kept, while zeros replace the rest of the matrix. Further, $\ell''(\tilde{\eta})_{i,i}$ will be noted as $w(\tilde{\eta})_i$. Having all the notations and equations into consideration, the algorithm for finding a solution for a fixed λ can be summarised as:

We initialise $\tilde{\beta}$ by the solution computed in the previous iteration of λ. A coordinate descent approach is adopted to find the value β that minimises the objective function. That is, we cyclically iterate over each coordinate $\beta_1, \beta_2, ..., \beta_p, \beta_1, \beta_2, ...$ and at each step, all of the other coefficients are kept constants while we only change the current coordinate. The value to assign to the coefficient is determined by taking the derivative of the objective function with respect to it, and by setting it to zero, we obtain the following optimal value:

$$\beta_k = \frac{S\left(\frac{1}{n}\sum_{i=1}^{n} w(\tilde{\eta}_i) x_{i,k} \left[z(\tilde{\eta}_i) - \sum_{j \neq k} x_{ij}\beta_j\right], \lambda\alpha\right)}{\frac{1}{n}\sum_{i=1}^{p} w(\tilde{\eta}_i) x_{ik}^2 + \lambda(1-\alpha)} \tag{26}$$

Algorithm 1 Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent

```
for $\lambda \leftarrow \lambda_1, ..., \lambda_k$ do
    initialize $\tilde{\beta}$
    while $\tilde{\beta}$ has not converged do
        $\tilde{\eta} \leftarrow X\tilde{\beta}$
        compute $z(\tilde{\eta})$
        compute $w(\tilde{\eta})$
        $\tilde{\beta} \leftarrow \arg\min_\beta(\frac{1}{n}\sum_{i=1}^{n} w(\tilde{\eta})_i (z(\tilde{\eta})_i - X_i^T\beta)^2 + \lambda P_\alpha(\beta))$
    end while
end for
```

where:

$$S(x, \lambda) = \text{sgn}(x)(|x| - \lambda)_+ \tag{27}$$

We set $\tilde{\beta}$ to $(0, 0, ..., 0)$ for the first iteration, where no previous solutions were computed. To be able to do this, we need to set λ_1 sufficiently large such that $(0, 0, ..., 0)$ is indeed the solution for the fixed value of λ_1. It is shown in [24] that the value that the value that matches the condition is:

$$\lambda_1 = \max_j \frac{1}{n\alpha} \sum_{i=1}^{n} w_i(0) x_{ij} z(0)_i \tag{28}$$

In our implementation, the other values λ_i are generated as the geometric sequence $\lambda_i = \lambda_1 * r^{i-1}$, where the ratio r is smaller than 1.

This algorithm is implemented in the leaves of an optimal decision tree, and the approach is discussed in the next section.

4 Methodology

We present *Cox SurTree*, a method that implements the Cox proportional hazards model proposed in [24] in the leaves of an optimal survival tree to create a model that can detect non-linear relations between covariates while also accounting for the effect of the covariates on the time-to-event. The same dynamic programming approach (14) is maintained, with only the loss function (first case, $n = 0$) being adapted. In Section 4.1, we present the methods used to select one of the models generated by the algorithm discussed in Section 3.3. Then, in Section 4.2, we showcase the details of incorporating the Cox model into the dynamic programming approach of SurTree [18] and how predictions are made.

4.1 Model selection

In this subsection, we illustrate how we use the approach in Section 3.3 to select a set of coefficients given a dataset $\mathcal{D}$, which we use as a terminal solver in the leaves of an optimal survival tree.

Because the algorithm presented in Section 3.3 generates several models, we must define a method to select only one. For this, we randomly split the dataset $\mathcal{D}$ into two disjoint subsets, $\mathcal{D}_{train}$ and $\mathcal{D}_{validation}$. We allocate 80% of the original instances to the former to ensure that the model has access to a large enough volume of data to effectively learn while reserving the rest 20% to the latter to provide a significant sample of unseen data to evaluate the performance of the models generated during training. We use $\mathcal{D}_{train}$ for creating solutions and $\mathcal{D}_{validation}$ for selecting one of the generated models.

Moreover, the algorithm requires tuning a set of parameters. While the size of generated solutions is set to at most 15, and for each solution, we allow for at most 100 optimisation iterations, the value of the l_1-ratio (α) is fixed per leaf using an exhaustive search strategy due to its thoroughness, simplicity and reliability. For every value of l_1 in $\{0.2, 0.4, 0.6, 0.8\}$, we fit a path of solutions for the dataset $\mathcal{D}_{train}$. Then, we use the validation set to select one of the models. For this, two approaches were implemented:

- **Likelihood** optimisation: the chosen model is the one minimising the negative log-likelihood described in equation (20) (on the validation set).
- **C-Index** [21] optimisation: the chosen model is the one maximising Harrell's C-index described in equation (8) (on the validation set).

Given the increased complexity of the model, we limited our validation functions to the two previously mentioned. We chose not to validate for the integrated Brier score [22], as it would have significantly increased the complexity of the model.

4.2 Cox SurTree

This subsection presents how the previously described method for selecting a Cox model is integrated to generate optimal survival trees using a dynamic programming approach.

First, one of the two validation techniques must be selected and used. In order to adapt equation (14), let us denote by $Cox(\mathcal{D}) = (\beta_1, \beta_2, ..., \beta_p)$ the solution selected by the algorithm described in Section 4.1 for a given dataset of instances $\mathcal{D}$ and the chosen validation scheme. We use the negative log-likelihood function as the loss function for the dynamic programming setup. Thus, we can formulate the following recurrence:

$$T(\mathcal{D},d,n) = \begin{cases} \infty & \text{if } n = 0 \wedge |U(\mathcal{D})| < 10 * |\mathcal{F}| \\ -\ell(Cox(\mathcal{D})) & \text{if } n = 0 \\ T(\mathcal{D}, d, 2^d - 1) & \text{if } n > 2^d - 1 \\ T(\mathcal{D}, n, n) & \text{if } d > n \\ \min\Big\{ T(\mathcal{D}(\overline{f}), d-1, n-i-1) + T(\mathcal{D}(f), d-1, i) : f \in F, i \in [0, n-1] \Big\} & \text{otherwise} \end{cases} \tag{29}$$

In the first case, $U(\mathcal{D})$ denotes the number of uncensored instances in dataset $\mathcal{D}$ and the condition was added to avoid overfitting and biases. We adhere to the *one in ten rule*, which states that the number of uncensored instances we fit should be at least ten times bigger than the number of features (original features, continuous and categorical, not binarised) [27].

Having the tree generated with the coefficients fitted in each leaf, we generate a distinct baseline hazard function for each leaf. The baseline cumulative hazard function is computed using the Breslow estimator (17). The survival function is generated based on the baseline cumulative hazard function, as shown in equation (18).

For the **C-Index validation** approach, we improve the time complexity by using a **binary indexed tree** [28]. Denoting by m the number of instances for which we compute the metric, a naive approach which pair-wise measures the observations has a time complexity of $\mathcal{O}(m^2)$. The binary indexed tree can be used to compute prefix or suffix sums, and we use it to store the risk scores η and calculate the number of concordant, discordant, and tied-risk pairs. We sort the observations based on their time to event t_i, then iterate over the instances in ascending order. For each instance j, we perform two actions. First, we query the binary indexed tree to count how many previously (uncensored) traversed instances i are concordant, discordant, and tied-risk with instance j. Second, if the instance is uncensored, we update its risk score η_j into the binary indexed tree. This approach has a time complexity of $\mathcal{O}(m \times log(m))$, which arises from sorting and performing $\mathcal{O}(m)$ updates and queries to the binary indexed tree, each one taking $\mathcal{O}(log(m))$ time.

An approach to **predict** the time-to-event for an instance given its survival distribution $\hat{S}_i(t)$, as also mentioned in [22], is to give the median of the estimated survival curve. Thus, the prediction is the time that splits the survival function into two equal areas regions and is computed by finding the value $\hat{T}$ that satisfies the relation:

$$\int_0^{\hat{T}} \hat{S}_i(t)\, dt = \int_{\hat{T}}^{\infty} \hat{S}_i(t)\, dt \tag{30}$$

The estimated survival distribution is a step function, and its area can be computed by adding up the individual areas of the rectangles that compose it. The estimate for the time $\hat{T}$ is the value that splits this area into half.

5 Experimental Setup and Results

This section describes the experimental setup implemented as part of this research and discusses the results of the experiments.

5.1 Experiment Setup

The algorithm presented in this research was integrated into the STreeD framework [15].[1] The core is implemented in C++, with a Python interface encapsulating the code. The experiment setup[2] extends the one used for SurTree, which implements a comparison between the STreeD interface of SurTree [18], the Julia implementation of OST [19], and the R implementation of CTree [6].

The experiments compare the method implemented in this research against SurTree, OST, and CTree on the metrics introduced in Section 2.3, namely Harrell's C-Index and the integrated Brier score. For these methods, we maintain the

[1]https://github.com/MateiMirica/STreeDCoxSA

[2]https://github.com/MateiMirica/STreeDCoxSA_Pipeline

	Data set characteristics				**Harrell's C-Index**					**Integrated Brier Score**				
Dataset	$\lvert\mathcal{D}\rvert$	**Censoring (%)**	$\lvert\mathcal{F}_{num}\rvert$	$\lvert\mathcal{F}\rvert$	**CTree**	**OST**	**SurTree**	**Cox SurTree LL**	**Cox SurTree CI**	**CTree**	**OST**	**SurTree**	**Cox SurTree LL**	**Cox SurTree CI**
Aids2	2839	38.0%	4	22	**0.54**	0.52	0.53	**0.54**	**0.54**	**0.01**	0.00	**0.01**	**0.01**	0.00
Dialysis	6805	76.4%	4	35	0.64	**0.67**	0.66	0.66	0.66	0.07	**0.12**	0.10	-0.03	-0.05
Framingham	4658	68.5%	7	60	0.68	0.67	0.67	**0.71**	**0.71**	0.11	0.10	0.10	**0.14**	**0.14**
Unempdur	3241	38.7%	6	45	0.69	0.69	0.69	0.69	**0.70**	**0.08**	**0.08**	0.07	0.03	0.03
Acath	2258	34.0%	3	21	0.59	0.59	0.59	0.59	**0.60**	**0.03**	**0.03**	**0.03**	**0.03**	0.01
Csl	2481	89.1%	6	42	**0.78**	0.76	0.77	**0.78**	**0.78**	0.12	0.14	0.13	**0.15**	0.14
Datadivat1	5943	83.6%	5	21	0.63	0.63	0.63	**0.64**	**0.64**	0.05	0.05	0.06	**0.09**	0.07
Datadivat3	4267	94.4%	7	30	0.66	0.65	0.66	**0.71**	**0.71**	-0.00	**0.03**	**0.03**	**0.03**	**0.03**
Divorce	3371	69.4%	3	5	**0.53**	**0.53**	**0.53**	**0.53**	**0.53**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**
Flchain	6524	69.9%	10	60	0.92	0.92	0.92	**0.93**	**0.93**	**0.64**	**0.64**	**0.64**	0.56	0.56
Hdfail	52422	94.5%	6	27	0.82	**0.86**	0.84	**0.86**	0.80	0.33	0.42	0.38	0.47	**0.50**
Nwtco	4028	85.8%	7	17	0.70	0.69	0.70	**0.72**	0.71	**0.13**	**0.13**	**0.13**	0.07	0.07
Oldmort	6495	69.7%	7	33	0.64	0.64	0.64	**0.66**	**0.66**	**0.07**	0.05	0.05	0.05	0.05
Prostatesurvival	14294	94.4%	3	8	**0.76**	**0.76**	**0.76**	**0.76**	**0.76**	**0.11**	**0.11**	**0.11**	-0.06	-0.05
Rott2	2982	57.3%	11	50	0.68	0.68	0.68	**0.71**	0.70	0.12	0.15	0.14	**0.16**	0.15
Wins per metric					4	4	2	**12**	11	**8**	**8**	7	**8**	4
Average rank					3.47	3.73	3.63	**2.07**	2.10	3.10	**2.77**	2.93	2.83	3.37

Table 1: Out-of-sample Harrell's C-Index and integrated Brier score for data sets from SurvSet [29]. CTree, OST, and SurTree were tested on a maximum depth $d = 4$. CoxSurTree LL and Cox SurTree CI were tested on a fixed depth of $d = 2$. $|\mathcal{F}_{num}|$ is the number of original features. $|\mathcal{F}|$ is the resulting number of binarised features.

same hyper-tuning strategies as in [18]: for SurTree, the depth and node budget are tuned; for OST, the depth and cost-complexity parameter are tuned; and for CTree, the confidence criterion is tuned. All the methods are tuned with a ten-fold cross-validation. For Cox SurTree, we generate two trees: *Cox SurTree LL*, which uses the log-likelihood validation scheme and the *Cox SurTree CI*, which uses the C-Index validation scheme. Due to the increased complexity and time overhead required by the leaf solvers, we did not use any cross-validation and kept the depth of the trees fixed to two and the node budget set to either two or three.

The experiments were run on both real and synthetic data, which are discussed in the following sections. The real datasets were split into five train-test split sub-datasets, while for the synthetic data, five datasets per setting were generated. The metrics, presented in Section 2.3, were computed as the average of the five obtained results per dataset.

When computing Harrell's C-Index, note that for the Cox SurTree models, a lower estimation $\hat{T}$, computed as in (30), means a higher risk, while for the other models, a lower estimation $\hat{\theta}$ means a lower risk.

The integrated Brier score is assessed on the test data and the interval of time accounting for the 10% and 90% quantiles of the times in the test data, denoted by t_{min} and t_{max} in (9). We report a normalised score $1 - IB/IB_0$, where IB_0 represents the score derived from the Kaplan-Meier estimator for the entire dataset.

5.2 Data

The real datasets, which are publicly available, are imported from the SurvSet [29] repository. We only keep those datasets with at least 2000 instances, as the methods' differences are clearer for large datasets [18]. The datasets are split into five train-test subsets, and the out-of-sample performance for a dataset is computed as the mean of the metric of the five generated test subsets.

As mentioned, since the model operates on binary predicates, we had to **preprocess** the data. First, the non-numeric data, such as strings or lists, were converted into numbers. One-hot encoding was used to binarise the categorical variables. If more than ten categories are present, we group the least frequent ones into a separate category. The continuous variables are also split into at most ten categories. Moreover, redundant features are removed. These include identical features and binary features that are present in less than 1% of the data.

We evaluate CTree and OST on the numeric data, SurTree on the binarised data, and Cox SurTrees on both. The binary data is used to split the instances and generate the tree, while the numeric data fits the Cox model in the leaves. Additionally, for Cox SurTree, we take a further step and normalise the data before fitting the Cox model.

5.3 Results

We ran the real datasets on all the mentioned algorithms and computed Harrell's C-Index and the integrated Brier score. No time limits were imposed on any of the algorithms. For *CTree*, *OST*, and *SurTree*, we have fixed a maximum depth of four and a maximum number of nodes of 15. For *Cox SurTree LL* and *Cox SurTree CI*, we fixed the depth to **half**, namely to two. The number of nodes was chosen between two or three.

We can see in Table 1 that our implemented models perform significantly better on the C-Index. They can capture the relative risks better than the state-of-the-art models for trees that are **four times less** in size. Note that the log-likelihood validation approach performs slightly better than the C-Index one. Even though the latter is specifically designed to choose the model that optimises Harrell's C-Index, this only happens at a leaf level, and there are no guarantees that aggregating the results for several leaves still performs better.

The increased performance can be attributed to the fact that all the instances are assigned different distributions, which leads to different predictions for the instances. In contrast, the other models assign a unique common distribution to all the instances within a leaf. The number of distinct predictions is, at most, equal to the number of leaves of the tree, which is relatively small. Moreover, within a leaf, the C-Index is always 0.5 as all the observations are tied-risk.

However, our models do not perform as well on the integrated Brier score compared to the other models. The log-likelihood validation method produces results comparable to the other methods, while the C-Index validation approach performs the worst. Moreover, our models perform even worse than the basic Kaplan-Meier estimator on some datasets.

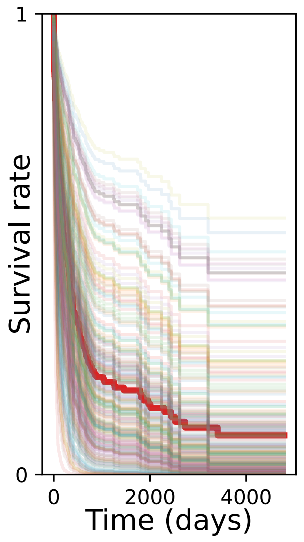

Figure 3: Example of distributions within a leaf. The bold red line represents the Kaplan-Meier estimator of the instances. The transparent lines represent the estimated survival functions of all the instances within the leaf.

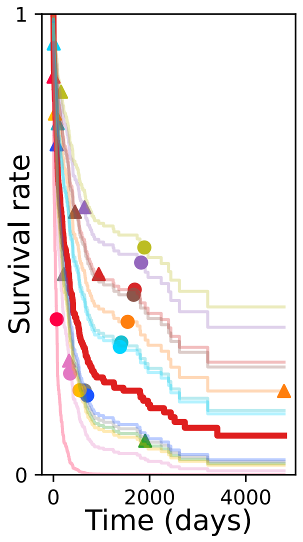

Figure 4: Example of distributions within a leaf. The bold red line represents the Kaplan-Meier estimator of the instances. The transparent lines represent the estimated survival functions for some of the instances within the leaf. Circles denote the predicted value; triangles denote the event time.

We investigated the results and came up with some conclusions. The integrated Brier score measures the deviation from the actual distributions. More precisely, good survival functions should have high values before the time-to-event, reflecting a significant probability of surviving before experiencing the event, and low values after the actual time-to-event, reflecting a small probability of surviving after experiencing the event. In Figure 3, we can see that the distributions follow the Kaplan-Meier estimate, with a high variance and deviance from it. Since the distributions have the same shape, it is not always possible to demonstrate the properties previously specified, especially when the time-to-event highly varies, as this would require the survival functions to have different shapes, and only scaling them does not suffice.

Moreover, in Figure 4, we can see that the mean (prediction) of the fitted distributions is quite far from the actual true time-to-event. For many of the instances illustrated, the survival probabilities are still high even after experiencing the event. These all indicate that the fitted distributions do not always resemble the actual ones.

We also compared all of the models on the same depth limit $d = 2$ to ensure that the other methods do not somehow perform better on lower depth limits. In Table 2 (Appendix A.1), we see that the Cox SurTrees perform even better than the rest when we set the same depth limit for all the methods.

We have also tested the implemented models on synthetic data, similarly generated as described in [19] and [18]. This data does not follow the Cox model's assumptions, namely the proportional scaling of the hazard by the change in covariates. We can see in Figure 5 (Appendix A.2) that our models perform worse on the C-Index in this case, even when we decrease the depth limit of the other models.

6 Responsible Research

In this section, we reflect on the ethical aspects of the research, address its possible societal impact, and discuss the reproducibility of our methods. We emphasise the need to be transparent about our processes and findings to build trust and support further research.

6.1 Ethical Considerations

The research meets ethical standards in both data usage and experimental setup. The real data used is publicly available and can be directly downloaded, as we did within our experiment pipeline. We provide the code that downloads the data, but we did not publish the datasets to ensure no personal data is exposed.

We note, however, that the algorithm may contain particular biases towards some of the features. The algorithm aims to generate optimal survival trees optimised on specific objective functions that do not consider the meaning of the features they use. It is at anyone's discretion to investigate the generated trees and assess any harms or biases induced by the model. This is also possible due to the interpretable nature of the model.

6.2 Societal Impact

This research is designed with the potential to improve survival predictions in different contexts, such as clinical or mechanical settings, by trying to provide a more accurate model. However, the results generated by the algorithm should not be taken by any means as a ground truth, especially when dealing with humans and, more specifically, patients. The presented model has limitations, and overgeneralisation of the results should not apply. The model was tested only on some datasets with specific characteristics that do not apply to every data.

6.3 Transparency and Reproducibility

We commit to transparency by making both the source code and the experimental setup available and ready to use. The synthetically generated datasets used are explicitly reported. For the real data, we provide the code used to download the datasets. To completely ensure the reproducibility of the results, we provide, for each train-test split, the indices from the original data that were used. Doing so allows fully reproducible results while ensuring no data ownership violation.

7 Conclusions and Future Work

We present Cox SurTree and its two possible adaptations, a model that fits survival trees with Cox Proportional Hazards models in its leaves. The dynamic programming approach of the model ensures global optimality on the train data for fixed-size trees for the implemented objective and validation schemes. The technique manages to capture the relative risk better than the state-of-the-art methods that fit a constant proportional hazard parameter model, including the greedy heuristic approach of CTree [6], the local search approach of OST [19], and the dynamic programming approach of SurTree [18]. The improved performance is highlighted by the C-Index metric. Moreover, Cox SurTree manages to do so for significantly smaller trees, thus improving the interpretability and explainability of the model.

While the findings of this research are promising, they are subject to several limitations. First, sometimes, the model performs worse in fitting distributions as a whole, as remarked by the integrated Brier score tests. Moreover, we showed on the synthetic data that if the data does not follow the proportional hazards condition, it performs worse than the other models, even in terms of the concordance index. Also, the leaf solvers are more complex as they fit several models and require additional validation and normalisation. Fitting a path of solutions (per leaf) as described in Section 3.3 on a dataset $\mathcal{D}$ for fixed values for the l_1-ratio, for n_λ (the number of solutions to fit), and for n_{iter} (the maximum number of iterations per solution), has a time complexity of $\mathcal{O}(|\mathcal{D}| \times n_\lambda \times n_{iter})$. Moreover, we run the algorithm for four values of the l_1-ratio. In comparison, the leaf solver in SurTree has a time complexity of $\mathcal{O}(|\mathcal{D}|)$. The overhead can be reflected by the increased run times of the approach.

Future work could explore in more depth the root cause for the high deviance of the fitted distributions from the absolute distributions. Additionally, changing the leaf solvers to fit a non-regularised Cox model could be analysed. This approach can be further explored with the possibility of adapting the depth two scalability improvement, as was similarly done for regression in [30].

References

[1] E. L. Kaplan and P. Meier, "Nonparametric Estimation from Incomplete Observations," *Journal of the American Statistical Association*, vol. 53, no. 282, pp. 457–481, 1958.

[2] W. Nelson, "Theory and Applications of Hazard Plotting for Censored Failure Data," *Technometrics*, vol. 14, no. 4, pp. 945–966, 1972.

[3] O. Aalen, "Nonparametric Inference for a Family of Counting Processes," *The Annals of Statistics*, vol. 6, no. 4, pp. 701–726, 1978.

[4] D. R. Cox, "Regression Models and Life-Tables," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 34, no. 2, pp. 187–202, 1972.

[5] L. Breiman, J. Friedman, R. Olshen, and C. J. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[6] T. Hothorn, K. Hornik, and A. Zeileis, "Unbiased Recursive Partitioning: A Conditional Inference Framework," *Journal of Computational and Graphical Statistics*, vol. 15, pp. 651–674, 2006.

[7] D. Bertsimas and J. Dunn, "Optimal Classification Trees," *Machine Learning*, vol. 106, no. 7, pp. 1039–1082, 2017.

[8] E. Demirović, A. Lukina, E. Hebrard, J. Chan, J. Bailey, C. Leckie, K. Ramamohanarao, and P. J. Stuckey, "MurTree: Optimal Classification Trees via Dynamic Programming and Search," *Journal of Machine Learning Research*, vol. 23, no. 26, pp. 1–47, 2022.

[9] L. Hyafil and R. L. Rivest, "Constructing Optimal Binary Decision Trees is NP-Complete," *Information processing letters*, vol. 5, no. 1, pp. 15–17, 1976.

[10] S. Aghaei, A. Gómez, and P. Vayanos, "Strong Optimal Classification Trees," *arXiv preprint arXiv:2103.15965*, 2021.

[11] S. Verwer and Y. Zhang, "Learning Decision Trees with Flexible Constraints and Objectives Using Integer Optimization," in *Proceedings of CPAIOR-17*, 2017, pp. 94–103.

[12] S. Verwer and Y. Zhang, "Learning Optimal Classification Trees Using a Binary Linear Program Formulation," in *Proceedings of AAAI-19*, 2019, pp. 1625–1632.

[13] H. Hu, M. Siala, E. Hebrard, and M.-J. Huguet, "Learning Optimal Decision Trees with MaxSAT and its Integration in AdaBoost," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 2020, pp. 1170–1176.

[14] H. Verhaeghe, S. Nijssen, G. Pesant, C.-G. Quimper, and P. Schaus, "Learning Optimal Decision Trees Using Constraint Programming," *Constraints*, vol. 25, no. 3, pp. 226–250, 2020.

[15] J. G. M. van der Linden, M. M. D. Weerdt, and E. Demirović, "Necessary and Sufficient Conditions for Optimal Decision Trees using Dynamic Programming," in *Advances in NeurIPS-23*, 2023.

[16] C. Molnar, G. Casalicchio, and B. Bischl, "Interpretable Machine Learning – A Brief History, State-of-the-Art and Challenges," in *Communications in Computer and Information Science*. Springer International Publishing, 2020, pp. 417–431.

[17] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens, "An Empirical Evaluation of the Comprehensibility of Decision Table, Tree and Rule Based Predictive Models," *Decision Support Systems*, vol. 51, no. 1, pp. 141–154, 2011.

[18] T. Huisman, J. G. M. van der Linden, and E. Demirović, "Optimal Survival Trees: A Dynamic Programming Approach," in *Proceedings of AAAI-24*, 2024.

[19] D. Bertsimas, J. Dunn, E. Gibson, and A. Orfanoudaki, "Optimal Survival Trees," *Machine Learning*, vol. 111, no. 8, pp. 2951–3023, 2022.

[20] R. Zhang, R. Xin, M. Seltzer, and C. Rudin. "Optimal Sparse Survival Trees." arXiv: 2401.15330. (2024).

[21] F. E. Harrell, R. M. Califf, D. B. Pryor, K. L. Lee, and R. A. Rosati, "Evaluating the Yield of Medical Tests," *Jama*, vol. 247, no. 18, pp. 2543–2546, 1982.

[22] E. Graf, C. Schmoor, W. Sauerbrei, and M. Schumacher, “Assessment and Comparison of Prognostic Classification Schemes for Survival Data,” *Statistics in Medicine*, vol. 18, no. 17-18, pp. 2529–2545, 1999.

[23] G. W. Brier, “Verification of Forecasts Expressed in Terms of Probability,” *Monthly Weather Review*, vol. 78, no. 1, pp. 1–3, 1950.

[24] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, “Regularization Paths for Cox’s Proportional Hazards Model via Coordinate Descent,” *Journal of statistical software*, vol. 39, no. 5, p. 1, 2011.

[25] M. LeBlanc and J. Crowley, “Relative Risk Trees for Censored Survival Data,” *Biometrics*, vol. 48, no. 2, pp. 411–425, 1992.

[26] N. E. Breslow, “Discussion of Professor Cox’s Paper,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 34, no. 2, pp. 216–217, 1972.

[27] P. Peduzzi, J. Concato, E. Kemper, T. R. Holford, and A. R. Feinstein, “A Simulation Study of the Number of Events per Variable in Logistic Regression Analysis,” *Journal of clinical epidemiology*, vol. 49, no. 12, pp. 1373–1379, 1996.

[28] P. M. Fenwick, “A New Data Structure for Cumulative Frequency Tables,” *Software: Practice and experience*, vol. 24, no. 3, pp. 327–336, 1994.

[29] E. Drysdale, *Survset: An open-source time-to-event dataset repository*, 2022. arXiv: 2203.03094.

[30] M. van den Bos, J. G. M. van der Linden, and E. Demirović, “Piecewise Constant and Linear Regression Trees: An Optimal Dynamic Programming Approach,” in *Proceedings of ICML-24*, 2024.

A Appendix

A.1 Real data

Table 2 shows the results obtained from running all the models on the real data for a depth $d = 2$. Both Cox SurTrees perform better on the C-Index than the other models. Cox SurTree LL produces comparable, even slightly better results than the existing models for the integrated Brier score. Cox SurTree CI performs the worst out of all the tested models on the integrated Brier score. As discussed in Section 5.3, both Cox SurTrees produce results worse than the Kaplan-Meier estimator over the whole instances for some datasets.

	Data set characteristics				**Harrell's C-Index**					**Integrated Brier Score**				
Dataset	$\lvert\mathcal{D}\rvert$	**Censoring (%)**	$\lvert\mathcal{F}_{num}\rvert$	$\lvert\mathcal{F}\rvert$	**CTree**	**OST**	**SurTree**	**Cox SurTree LL**	**Cox SurTree CI**	**CTree**	**OST**	**SurTree**	**Cox SurTree LL**	**Cox SurTree CI**
Aids2	2839	38.0%	4	22	**0.54**	0.52	0.52	**0.54**	**0.54**	**0.01**	0.00	**0.01**	**0.01**	0.00
Dialysis	6805	76.4%	4	35	0.63	0.63	0.63	**0.66**	**0.66**	0.06	**0.07**	**0.07**	-0.03	-0.05
Framingham	4658	68.5%	7	60	0.64	0.64	0.64	**0.71**	**0.71**	0.07	0.07	0.08	**0.14**	**0.14**
Unempdur	3241	38.7%	6	45	**0.70**	0.69	0.69	0.69	**0.70**	**0.08**	**0.08**	**0.08**	0.03	0.03
Acath	2258	34.0%	3	21	0.59	0.59	0.59	0.59	**0.60**	**0.03**	0.02	**0.03**	**0.03**	0.01
Csl	2481	89.1%	6	42	0.73	0.74	0.76	**0.78**	**0.78**	0.10	0.12	0.11	**0.15**	0.14
Datadivat1	5943	83.6%	5	21	0.62	0.63	0.63	**0.64**	**0.64**	0.05	0.04	0.04	**0.09**	0.07
Datadivat3	4267	94.4%	7	30	0.67	0.65	0.66	**0.71**	**0.71**	0.02	**0.03**	**0.03**	**0.03**	**0.03**
Divorce	3371	69.4%	3	5	**0.53**	**0.53**	**0.53**	**0.53**	**0.53**	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**
Flchain	6524	69.9%	10	60	0.92	0.91	0.91	**0.93**	**0.93**	**0.63**	**0.63**	**0.63**	0.56	0.56
Hdfail	52422	94.5%	6	27	0.79	0.81	0.79	**0.86**	0.80	0.22	0.30	0.23	0.47	**0.50**
Nwtco	4028	85.8%	7	17	0.67	0.69	0.69	**0.72**	0.71	**0.12**	**0.12**	0.11	0.07	0.07
Oldmort	6495	69.7%	7	33	0.63	0.63	0.63	**0.66**	**0.66**	**0.06**	0.04	0.05	0.05	0.05
Prostatesurvival	14294	94.4%	3	8	0.74	0.74	0.74	**0.76**	**0.76**	**0.09**	**0.09**	**0.09**	-0.06	-0.05
Rott2	2982	57.3%	11	50	0.67	0.68	0.68	**0.71**	0.70	0.12	0.14	0.14	**0.16**	0.15
Wins per metric					3	1	1	**13**	12	**8**	7	**8**	**8**	4
Average rank					3.77	3.80	3.83	1.83	**1.77**	3.07	3.10	2.80	**2.77**	3.27

Table 2: Out-of-sample Harrell's C-Index and integrated Brier score for data sets from SurvSet [29]. CTree, OST, and SurTree were tested on a maximum depth $d = 2$. CoxSurTree LL and Cox SurTree CI were tested on a fixed depth of $d = 2$. $|\mathcal{F}_{num}|$ is the number of original features. $|\mathcal{F}|$ is the resulting number of binarised features.

A.2 Synthetic data

Figure 5 compares Harrell's C-Index for SurTree and both Cox SurTrees on synthetic data similarly generated as described in [19] and [18] on an increasing number of instances and different censoring levels: 10%, 50%, 80%. When the data does not follow the Cox model's assumptions, namely the proportional scaling of the hazard by the change in covariates, our models perform worse than SurTree.

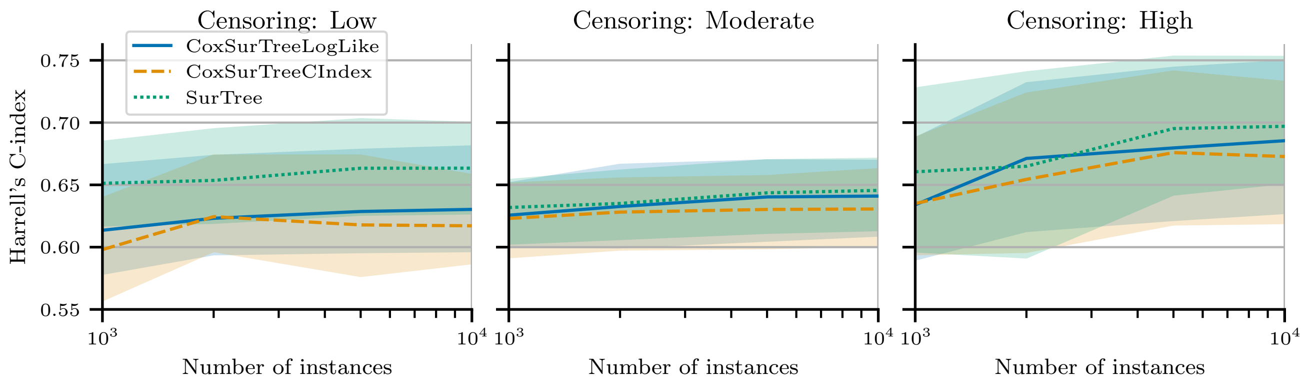

Figure 5: Harrell's C-Index on the synthetic data sets. Both CoxSurTrees were tested on depth $d = 2$. SurTree was tested on a maximum depth $d = 3$.