Published in Image Processing On Line on 2024–07–16.
Submitted on 2023–02–06, accepted on 2024–05–03.
ISSN 2105–1232 © 2024 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2024.466

Survival Forest for Left-Truncated Right-Censored Data

Vincent Laurent[1,2], Olivier Vo Van[3]

[1]Eurobios, Cachan, France
[2]Université Paris-Saclay, ENS Paris-Saclay, Centre Borelli, Gif-sur-Yvette, France
vincent.laurent@eurobios.com
[3]SNCF Direction Technologies, Innovation & Projets Groupe, Saint-Denis, France

Communicated by Pablo Musé *Demo edited by* Vincent Laurent

Abstract

The estimation of the lifetime of an industrial equipment or a patient is often based on censored data, because the event of interest is observed only for a subsample of observations. The use of the Random Forest algorithm applied to industrial data is relevant because the algorithm presents robust performances in many applications. Coupled with survival approaches, it can produce time trajectories for each subset of the feature space and thus differentiate observed objects with respect to their lifetimes. Our work aims to generalize the existing tree-based approach CART applied to left-truncated right-censored data to obtain a Random Forest algorithm. We provide a simple API to use such algorithm as well as tools to validate a temporal score against censored data.

Source Code

The source code and documentation for this algorithm are available from the web page of this article[1]. Usage instructions are included in the `README.md` file of the archive.
The original implementation is available here[2].
This is an MLBriefs article, the source code has not been reviewed!

Keywords: survival analysis; truncated and censored data; time-dependent ROC curve; Random Forest

[1]https://doi.org/10.5201/ipol.2024.466
[2]https://github.com/eurobios-mews-labs/survival-trees

1 Introduction

In industrial or medical applications, the feared event T (death, failure, fault occurrence, rupture or cracking) is not necessarily observed, and evaluating the time to event can be complex especially when numerous non linear factors are involved. In such context, survival analysis can be used to estimate the survival function $S(t) = \mathbb{P}(T > t)$ in the presence of censored data. In the literature, the estimation of S is often carried out assuming proportional hazard or linearity as it is done in many Cox or Accelerated Failure Time (AFT) models, and these assumption can be restrictive in many applications. The Random Forest algorithm [2] is known to perform well on high dimension, noisy and non linear tabular data, but is also popular for its good performance on small or medium datasets, which makes it a good candidate to tackle real life problems. Many algorithms have been developed to adapt Random Forest [9] or boosting to survival problems (see [20] for a survey on survival machine learning methods) and focus mainly on right-censored data. Yet, left truncation is another common constraint in applications which occur especially when observations cannot be diagnosed before a certain time L. In our experience, this is often the case because the data is not immediately available or up-to-date in the information systems, and particularly for industrial asset management, where some equipment predates digitization. This implies that observations are made at a starting (random) point L and all events that come before are not in the sample. In this paper, an extension of decision trees for Left-Truncated Right-Censored (LTRC) data developed in [7] is made to obtain LTRC Forest. Our developments are similar to the recent approach presented in [21], in which the authors propose several implementations of ensemble methods applied to LTRC data. To facilitate the use of the algorithm, we have chosen to take inspiration of the Lifelines package [5], which is itself compliant with the Scikit-Learn's API [17]. Finally, we provide a simple method to assess performance based on Receiver Operating Characteristic (ROC) analysis [10].

2 Survival Analysis

Notations. The purpose of the algorithm is to estimate T, the time to event that needs to be estimated in the presence of LTRC data. Knowing the distribution of T, with f its density function and F its cumulative distribution function, is equivalent to having the survival function $S(t) = \mathbb{P}(T > t) = 1 - F(t)$, the hazard function $\lambda(t) = f(t)/S(t)$ or the cumulative hazard function $\Lambda(t) = \int_0^t \lambda(\tau)d\tau = -\log S(t)$. In the following, we can reason indifferently with one or the other of these three functions. The learning data is represented by a random descriptor X in the measurable space $\mathcal{X}$.

Censoring and truncation. Let R, L, T, C be four positive random variables. T is right censored by C if, instead of T, the couple (R, δ) is observed with $R = \min(T, C)$ and $\delta = \mathbb{I}\{T \leqslant C\}$. For an observation (R_i, δ_i), $\delta_i = 1$ when the complete duration is observed (the event occurred) while when $\delta_i = 0$ the observation is censored (not observed yet). Truncation works differently for it concerns sampling itself and the duration T is truncated by a subset A of $\mathbb{R}^+$ if T is only observed when $T \in A$. Alternatively, T is left-truncated by a positive random variable L if $T > L$. Putting altogether, dealing with LTRC data requires observing the triplet (L, δ, R) as target variable.

Neslon-Aalen estimator. An important component of the algorithm is the estimation of the survival function at each node of the tree, for a sample of observations. The Nelson-Aalen estimator computes the cumulative number of events and is a non-parametric estimator of the cumulative

hazard function

$$\hat{\Lambda}(t) = \sum_{t_i<t} d_i/n_i, \tag{1}$$

where d_i is the number of events at t_i and n_i the total number of individuals at risk at t_i.

Likelihood function. Approaches in [7, 14] are based on proportional hazard assumption. They aim to establish a link between the maximum log-likelihood and the deviance that will ultimately provide material for splitting rules. In such context, assuming proportional hazard implies that the cumulative hazard function can be written as $\Lambda(t|X) = \theta(X)\Lambda_0(t)$ and the proportional hazard function as $\lambda(t|X) = \theta(X)\lambda_0(t)$. For right-censored data problem, the log-likelihood can be expressed as follows

$$L(\theta) = \prod_i \mathbb{P}(T = R_i|\theta)^{\delta_i}\mathbb{P}(T > R_i|\theta)^{1-\delta_i} = \prod_i f(R_i|\theta)^{\delta_i}S(R_i|\theta)^{1-\delta_i} = \prod_i \lambda(R_i|\theta)^{\delta_i}S(R_i|\theta) \ ,$$

where $f(R_i|\theta)$ stands for the probability of an observed event and $S(R_i|\theta)$ the probability of a censored event at time R_i. Using the proportional hazard assumption, the likelihood becomes

$$L(\theta) = \prod_i (\theta\lambda_0(R_i))^{\delta_i}S(R_i|\theta) = \prod_i (\theta\lambda_0(R_i))^{\delta_i}\exp(-\theta\Lambda_0(R_i))$$

and the log-likelihood

$$\log L(\theta) = \sum_i \delta_i \log(\theta\lambda_0(R_i)) - \theta\Lambda_0(R_i) \ .$$

In such context, the parameter θ maximizing $\log L$ is $\sum \delta_i / \sum \Lambda_0(R_i)$. Let the deviance be $D(\theta) = 2\left[\log L^{sat} - \log L(\theta)\right]$ with L^{sat} the log-likelihood of the saturated model, which is the model with the maximum achievable likelihood by allowing one parameter θ_i to each observation namely $\delta_i\Lambda_0(R_i)$. The full log-likelihood deviance measures the adequacy of the model to the data. Define d_i as the unit contribution of observation i to the deviance ratio, $D(\theta) = \sum_i d_i(\theta)$. Thus one obtains the following contribution i to full likelihood deviance ratio

$$d_i(\theta) = 2\left[\delta_i \log\left(\frac{\delta_i}{\Lambda_0(R_i)\theta}\right) - (\delta_i - \Lambda_0(R_i)\theta)\right].$$

The authors of [14] noted that this corresponds to the unit deviance ratio of the Poisson regression of time exposure $\Lambda_0(R_i)$ with δ_i the count of observed event. Building on that statement, the authors of [7] provide an extension of the method by interpreting $\Lambda_0(R_i) - \Lambda_0(L_i)$ as a "time exposure" of observation i, they obtain

$$d_i(\theta) = 2\left[\delta_i \log\left(\frac{\delta_i}{(\Lambda_0(R_i) - \Lambda_0(L_i))\theta}\right) - (\delta_i - (\Lambda_0(R_i) - \Lambda_0(L_i))\theta)\right].$$

In the previous formula, Λ_0 is not known in practice but can be obtained in a recursive fashion using the Breslow estimator as described in [14] or estimated with Neslon-Aalen formula based on all the data available [7]. This yields,

$$\hat{d}_i(\theta) = 2\left[\delta_i \log\left(\frac{\delta_i}{(\hat{\Lambda}_0(R_i) - \hat{\Lambda}_0(L_i))\hat{\theta}}\right) - \left(\delta_i - (\hat{\Lambda}_0(R_i) - \hat{\Lambda}_0(L_i))\hat{\theta}\right)\right]$$

where $\hat{\theta} = \sum \delta_i / \sum(\hat{\Lambda}_0(R_i) - \hat{\Lambda}_0(L_i))$ is the parameter that maximizes the log-likelihood. For any subspace $\mathcal{C}$ of the feature space $\mathcal{X}$, the deviance estimated using observations $\{X_i \in \mathcal{C}\}$ is noted $D_{\mathcal{C}}$.

3 LTRC-Forest

LTRCART. The algorithm developed by [14] and adapted in [7] mainly derived from CART algorithm [3]. The original idea is based on recursive partitioning of the feature space $\mathcal{X}$: each split cuts the features space along the i^{th} feature at threshold c. The choice of c and i is originally performed using a criterion such as entropy or Gini index, the global idea being to find the split that maximizes a measure of improvement. This improvement is typically assessed using the variation of the measure between a node and its children. A natural adaptation is to use the deviation as splitting criterion, and find among all possible splits (c, i) the couple maximizing the deviance reduction

$$D_{\text{parent}} - [D_{\text{left child}} + D_{\text{right child}}]\,. \tag{2}$$

This splitting criterion is chosen by [14] for its analogy with the mean residual sum of squares in the original CART algorithm. In addition to the splitting strategy, the algorithm LTRCART [7] requires a stopping rule that terminates recursive partitioning prematurely to prevent overfitting, and a terminal node estimation. Details on the stopping rule can be found in [7, 14]. Note that because Forest algorithms aim to average several uncorrelated high-variance estimators, the question of pruning the tree will be less decisive regarding overfitting problems. Nonetheless, in the implementation details, parameters controlling the depth of trees are provided. For the estimation of the terminal node in each leaf of the tree, the Nelson-Aalen estimator (Equation (1)) is computed on the basis of all observations (L_i, δ_i, R_i) such that (X_i) are contained in the leaf. On the contrary to [9] where a leaf node must contain at least one non censored observation, LTRCART algorithm replaces zero observed events at leaf nodes by 0.5, yielding for this specific case the estimate $\hat{\theta} = 1/\sum 2(\hat{\Lambda}_0(R_i) - \hat{\Lambda}_0(L_i))$.

Survival Forest. The extension of the approach [7] to Survival Forest for LTRC data is based on:

- *Bagging (bootstrap aggregating) predictors* developed by [1], whose principle is to average trained estimators on bootstrap samples of the initial dataset. This step aims to reduce the variance of the estimator. In this context, the trees are not pruned and can be grown deep.
- *Random Forest* [2] offers an improvement over bagging predictors by constructing uncorrelated trees constraining each tree to use only m predictors. Typically, $m = d/3$ is chosen for regression task and $m = \sqrt{d}$ for classification.

The final step for Algorithm 1 is the aggregation rule that performs a combination of all the predictors. Let $\hat{s}_j$ be the survival estimate derived from Neslon-Aalen formula for one LTRC-tree. Then $\hat{s}_j(x,t) = \sum_{l \in \mathcal{L}_j} \mathbb{I}\{x \in l\} \exp(-\hat{\Lambda}_l^j(t))$, where $\mathcal{L}_j$ is a partition of the features space $\mathcal{X}$ describing the leaves of the j^{th} LTRC tree and $\hat{\Lambda}_l^j(t)$ is the Nelson-Aalen estimator using training data in the j^{th} bootstrap sample that falls in leave l. A natural aggregation rule is the averaged estimator

$$\hat{s}(x,t) = \frac{1}{n_{\text{tree}}} \sum_{j=1}^{n_{\text{tree}}} \hat{s}_j(x,t).$$

Implementation details. The algorithm was implemented in Python ; the objective is to be compliant with state-of-the-art API of Scikit-Learn [17] and Lifelines package [5], the later providing various tools for survival analysis. The implementation uses the Rpart library[3] based on the R

[3] T. Therneau, B. Atkinson, B. Ripley. Package `rpart` (accessed on April 20, 2016), https://cran.r-project.org/web/packages/rpart/

Algorithm 1: LTRC-Forest

Input tabular data: observations $(L_i, \delta_i, R_i)_{i \leqslant n}$ covariates $X_i = (X_i^1, \cdots, X_i^d)$

Param: n_{tree}, m, α

for j **from** *1* **to** n_{tree} **do**

- B_j sample of length $\lfloor \alpha n \rfloor$ from $\{1, \cdots, n\}$ with replacement (sample observations);
- F_j sample of length m from $\{1, \cdots, d\}$ (sample features);
- $X_i^{(j)} \coloneqq (X_i^k)_{i \in B_j, k \in F_j}$ and $Y_i^{(j)} \coloneqq (L_i, \delta_i, R_i)_{i \in B_j}$;
- $\hat{s}_j(t, x) = \text{LTRCART}(X_i^{(j)}, Y_i^{(j)})$

Output function: $1/n_{\text{tree}} \sum_j \hat{s}_j(t, x)$

language. The Rpart implements the CART algorithm and proceeds to the recursive segmentation of the feature space using the deviance criterion. Regarding regularization, parameters are needed to control the growth of each tree. Indeed, compared to the common regression task where having a single element per leaf can make sense, the estimation of the cumulated hazard function Λ requires more information: the estimation is degenerated when only one point is provided in Equation (1). The second reason for regularizing is the complexity: the Random Forest algorithm assumes that a large enough number of trees[4] is used to take into account the fact that trees tend to overfit data. The output here is more complex because it is a function of time; thus the prediction step is a lot more memory consuming when a large number of trees is used. Let $M_{\hat{s}}$ be the output matrix, the representation of $\hat{s}(x, t)$ where each row corresponds to the estimation of the hazard function for the observation i, and columns to the time sampling. If N is the number of training observation and $\{R_i\}_{i \leqslant N}$ the observed time to event, $M_{\hat{s}}$ has at most $N \times \#\{R_i\}_{i \leqslant N}$ elements, in the worst case N^2 when all R_i are distinct.

(i) **Regularization and pruning:** three parameters are given for this aspect; they will prevent trees to grow deep and have a single value in leaf nodes. These parameters only intervene in the LTRCART algorithm:

1. `min_samples_leaf` specifies the minimum number of sample in terminal node.
2. `min_impurity_decrease` specifies the minimum impurity decrease, in the sense of Equation (2), for a split to be attempted.
3. `max_depth` provides the maximum depth of trees.

(ii) **Bootstrap and feature selection:** three parameters are provided; they control the way the ensemble of estimators is built.

1. `max_samples` controls the percentage α of the dataset be to re-sampled
2. `max_features` is the maximum number $m \leqslant d$ of features to use when building trees independently
3. `n_estimators` the number of trees n_{tree} to be used in training.

4 Validation Methods and Datasets

The package provides material to validate the results using the review [10]. This paper presents extension of ROC analysis to temporal score validation. In this context, let (X_i, T_i) and (X_j, T_j)

[4] Scikit-Learn's implementation uses 100 trees as default value

be a pair of two independent random variables of the same law, and $s : \mathcal{X} \times \mathbb{R} \to \mathbb{R}$ be any time dependent scoring function, then the AUC(t) of the function s can be formulated as

$$\mathrm{AUC}(t) = \mathbb{P}(S(X_i, t) > S(X_j, t) | T_i \leqslant t,\, T_j > t).$$

A simple way to estimate AUC is

$$\widehat{\mathrm{AUC}}(t) = \frac{\sum_{i,j} \delta_i \mathbb{I}\{T_i \leqslant t,\, T_j > t\} \mathbb{I}\{S(X_i, t) > S(X_j, t)\}}{\sum_i \delta_i \mathbb{I}\{T_i \leqslant t\} \sum_j \mathbb{I}\{T_j > t\}}.$$

The work [10] emphasizes the redundancy of information contained in the previous formula and provides many alternatives. This formulation of the AUC is qualified as *cumulative sensitivity* and *dynamic specificity*. This concept introduced by [8] is relevant from a clinical perspective, and is used in many applications. This formula is used here for its simplicity and the fact that it does not depend on survival estimators, as opposed to the 10 alternatives presented in [10]. A simple illustration of the estimator of the AUC is provided in Figure 2 for one of the datasets presented below, as well as in the demo.[5] This formulation of AUC is an intuitive extension of the common definition where the scoring function is $s_t(x) : x \mapsto S(x, t)$ and the outcome is the binary random variable $Y = \mathbb{I}\{T \leqslant t\}$.

Datasets. In order to test the algorithm, several toy datasets are provided extracted from the python library Lifelines [5] and R library survival[6]. An extensive benchmark of performance is not intended here: the aim is to provide various datasets in size and properties to test where the algorithm is expected to perform. Note that if left truncation occurs often in practice, data are not often provided with such information. However, in [12] authors noted that many medical applications treat the age of patient as features instead of entry point in the study. Indeed the only fact that the age is known at the beginning of the medical follow-up, implies that observation of death is left-truncated by age. The first four datasets in Table 1 follow this case. Two other datasets related to societal matters are proposed as illustrative example of the variety of problems that can be addressed. For these two datasets, no truncation is provided and L is set to 0. Finally, a simulated dataset is introduced in the following paragraph, to have high dimensional data (many features) and as many samples as wanted.

Synthetic data. Following the main ideas of [16] to generate synthetic LTRC data, we start by sampling $\pi_k \sim \mathcal{U}(0.2, 0.8)$ and $\mu_k = \Phi^{-1}(\pi_k)$, where Φ is the cumulative density function of the normal distribution. Let M be a $d \times d$ matrix filled with elements sampled from $N(0, 1)$, then the matrix $S = M^T M$ is positive semi-definite. A correlation matrix Σ can be deduced by scaling S, $\Sigma = S/D^T D$ with D a vector containing the diagonal element of S. The distribution of X is a multidimensional Gaussian distribution $X \sim N(\mu, \Sigma)$. To simulate target data, namely the triplet (L, T, R), Weibull distributions $W(\alpha, s)$ are used, where α and s denote respectively the scale and shape parameters. The α parameter is fixed for L, T and R. For each observation X_i, the shape parameter is

- $s_i = e^{\alpha + g(X_i).\beta}$ for the time-to-event variable $T_i \sim W(\alpha, s_i)$,
- $\tilde{s}$ the median of s_i for right-censoring time $R_i \sim W(\alpha, \tilde{s})$,
- $\tilde{s}p$ where $p < 1$ the left-truncation time $L_i \sim W(\alpha, \tilde{s}p)$.

The idea of this setting is to make T dependent of each observation via the function g, while R and T remain independent from X. Here we use $g : x_i \in \mathbb{R}^d \mapsto (j/d \times x_{i,j}^2)_{j \in [1,d]}$ where $x_{i,j}$ denotes the j^{th}

[5] https://doi.org/10.5201/ipol.2024.466

[6] T. M. Therneau. A Package for Survival Analysis in R. R package version 3.3-1, 2022, https://CRAN.R-project.org/package=survival

coordinates of x_i. This intends to test the ability of the algorithm to carry out feature selection by introducing features with lesser impact on time-to-event. Finally we update $L_j := \min(R_j - \varepsilon, L_j)$ with $\varepsilon > 0$ to account for the fact that left truncation comes first.

Real case study: rolling contact fatigue data. Finally, results from the dataset presented in the study [13] are provided. The data collected by the French railway operator tackles rolling contact fatigue issues due to the repeated passage of trains on the rail. Input data X stands for the description of the rail – mainly geometric and operational data – and T is the date of the first rail defect due to fatigue on the rail segment. After this date, the crack is followed until the ensuing propagation requires an eventual removal of the defective segment. Preliminary data analysis shows that very few defects appeared before a certain time that advocates the use of truncated data.

Dataset name	Description	Shape	Source	Ref.
Larynx cancer	Four stages of cancer are given as predictors for the age of death.	89×4	Lifelines	[11]
Lung cancer	Patients with advanced lung cancer (North Central Cancer Treatment Group).	228×8	Lifelines	[15]
Breast cancer	German Breast Cancer Study Group (1984) the objective is to test performance of various treatment on remission.	686×3	Lifelines	[19]
FLC chain	Study the impact of serum free light chain (FLC) as a possible marker for immune disregulation.	7874×5	Survival	[6]
Democracy - Dictatorship	Classification of political regimes (democracy or dictatorship). Knowing the features of each political regime, the time of its ending is provided (if not censored).	1808×3	Lifelines	[4]
Convicts	The dataset describes convicts released from Maryland state prisons in the 1970s. The duration of interest is the duration until their re-arrest.	432×7	Lifelines	[18]
Synthetic data	X is sampled using multidimensional Gaussian distribution, T, L, R using Weibull distributions	400×30	–	[16]

Table 1: Dataset available in the demo and used to benchmark the algorithm.

5 Experimental Results

Qualitative results. Figure 1 presents results in terms of AUC for all datasets. We compare four algorithms that can tackle LTRC data, namely LTRC-Forest (Algorithm 1), LTRCART [7], a semi-parametric Cox model and AFT model from Lifelines package [5]. The experiments are repeated 20 times on benchmark data, and results are averaged over all experiments. We note that LTRC-Forest presents a certain interest compared to the LTRCART algorithm whatever the dataset considered. As expected, it seems that tree-based algorithms perform better on larger datasets like the Democracy & Dictatorship data or synthetic data. For smaller datasets, the performance of trees seems to undermine other methods. This can be due to the piecewise constant approximation that can be less effective in these particular cases. As for the application to rolling contact fatigue, results obtained are quite convincing compared to other studies of the kind (see [13] for discussion).

Temporal ROC analysis. In the demo, the results of a model are evaluated through several methods on test data, the latter being randomly sampled from the global dataset. We choose to use 60% of the data for training and 40% for testing; no cross validation is operated. Visually, the estimated survival functions for each X_i can be evaluated as shown in Figure 2. On this figure, the red curves represent the time at which the event T is observed while the blue curves are the

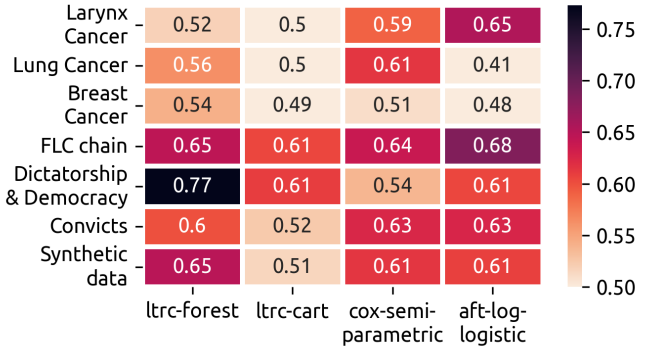

Figure 1: AUC averaged over time for different toy datasets.

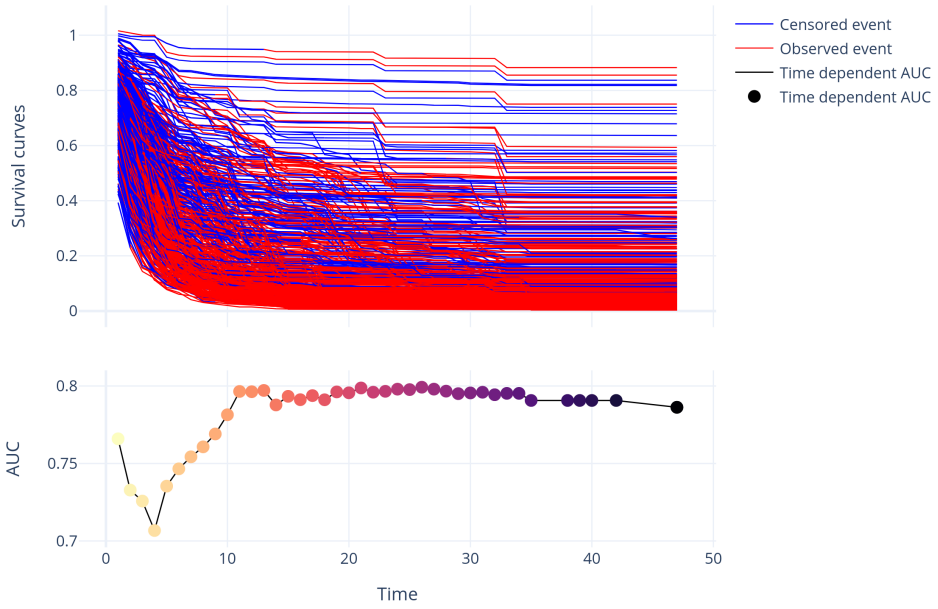

Figure 2: Survival curves and temporal AUC. On the top are represented the survival curves estimated using LTRC Forest algorithm for a testing set of data. The survival functions turn to red as soon as the T event is observed. The temporal AUC of the ROC curve on the bottom represents the AUC at time t.

individuals censored at time t. Thus, the lower the red curves are, the better the model. The AUC gives an objective measure of the model performance at time t, i.e. for all events T_i prior to t. At each time step, an AUC point is associated with a ROC curve in Figure 3, which allows to plot the performance according to each classification threshold. It is important to note that by this way of representing the performance of the model, a sample intervenes several times in the evaluation of the performance, which constitutes a limitation as noted by the authors of [10]. The results in Figures 2 and 1 represent an analysis of the outputs as presented in the demo for the dataset "Democracy - Dictatorship".

6 Conclusion

In this article, we provide a simple implementation that makes approaches proposed by [7, 14] more robust by averaging the response of several independent LTRC trees. Our goal is to provide a ready-to-use API inspired by Scikit-Learn implementations. The approach presented in this article, mostly used in the medical field, can be useful for other application domains. For example, these methods

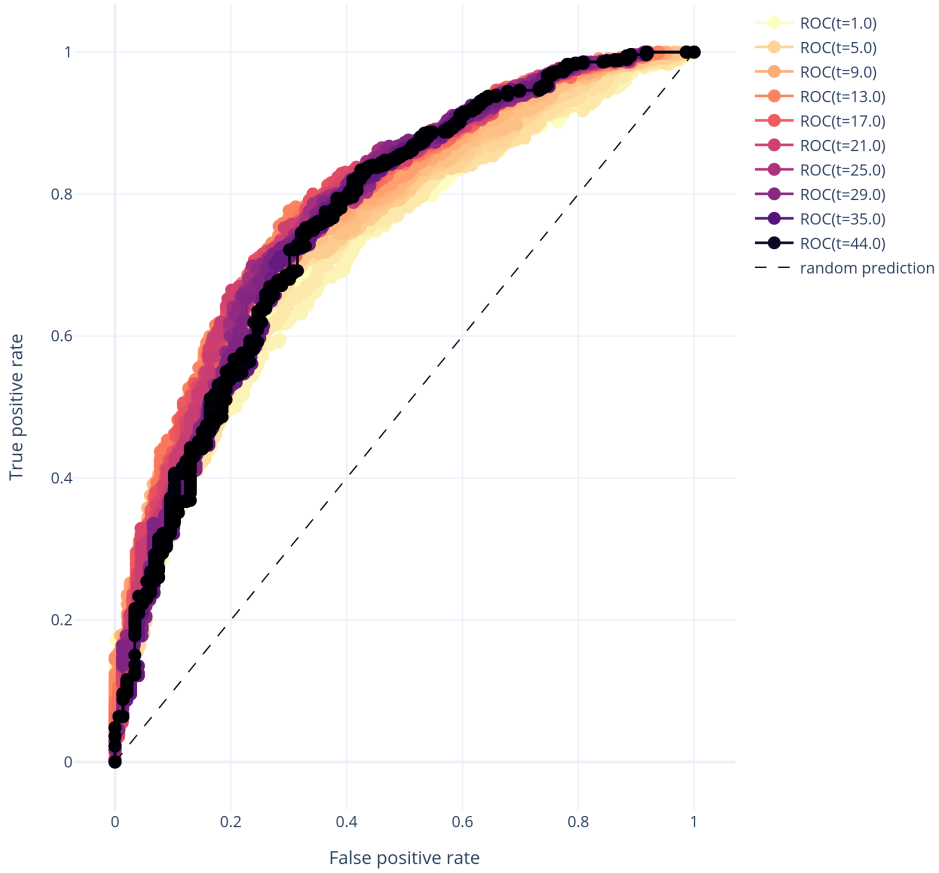

Figure 3: Temporal ROC curve. These curves are estimated on the basis of the survival function represented in Figure 2. Each curve corresponds to a time stamp; the estimated survival curve at time t can be considered as a scoring function for which all possible thresholds for the decision rule are represented here.

could be used to plan maintenance operations for the management of industrial assets. This requires the use of machine learning methods when the observed volumes are large and the interactions are complex.

Acknowledgment

This research was supported by SNCF Direction Technologies, Innovation & Projets Groupe. The authors thank them for making this research possible.

References

[1] L. Breiman, *Bagging Predictors*, Machine Learning, 24 (1996), pp. 123–140, https://doi.org/10.1007/BF00058655.

[2] ——, *Random Forests*, Machine Learning, 45 (2001), pp. 5–32.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees. Wadsworth Int*, Group, 37 (1984), pp. 237–251.

[4] J. A. Cheibub, J. Gandhi, and J. R. Vreeland, *Democracy and Dictatorship Revisited*, Public Choice, 143 (2010), pp. 67–101.

[5] C. Davidson-Pilon, *Lifelines: Survival Analysis in Python*, Journal of Open Source Software, 4 (2019), p. 1317, https://doi.org/10.21105/joss.01317.

[6] A. Dispenzieri, J. A. Katzmann, R. A. Kyle, D. R. Larson, T. M. Therneau, C. L. Colby, R. J. Clark, G. P. Mead, S. Kumar, L. J. Melton III, and S. V. Rajkumar, *Use of Nonclonal Serum Immunoglobulin Free Light Chains to Predict Overall Survival in the General Population*, in Mayo Clinic Proceedings, vol. 87, Elsevier, 2012, pp. 517–523, https://doi.org/10.1016/j.mayocp.2012.03.009.

[7] W. Fu and J. S. Simonoff, *Survival Trees for Left-Truncated and Right-Censored Data, with Application to Time-Varying Covariate Data*, Biostatistics, 18 (2017), pp. 352–369, https://doi.org/10.1093/biostatistics/kxw047. arXiv: 1606.03033.

[8] P. J. Heagerty and Y. Zheng, *Survival Model Predictive Accuracy and ROC Curves*, Biometrics, 61 (2005), pp. 92–105, https://doi.org/10.1111/j.0006-341X.2005.030814.x.

[9] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer, *Random Survival Forests*, Annals of Applied Statistics, 2 (2008), pp. 841–860, https://doi.org/10.1214/08-AOAS169.

[10] A. N. Kamarudin, T. Cox, and R. Kolamunnage-Dona, *Time-Dependent ROC Curve Analysis in Medical Research: Current Methods and Applications*, BMC Medical Research Methodology, 17 (2017), pp. 1–19, https://doi.org/10.1186/s12874-017-0332-6.

[11] O. Kardaun, *Statistical Survival Analysis of Male Larynx-Cancer Patients-A Case Study*, Statistica Neerlandica, 37 (1983), pp. 103–125.

[12] J. P. Klein and M. L. Moeschberger, *Survival Analysis: Techniques for Censored and Truncated Data*, vol. 2, Springer, 2003, https://doi.org/https://doi.org/10.1007/b97377.

[13] V. Laurent, O. V. Van, M. Mougeot, and J.-M. Ghidaglia, *Machine Learning Based Prediction of Fatigue Events in Railway Rails*, in World Congress on Engineering Asset Management, Springer, 2021, pp. 455–466, https://doi.org/10.1007/978-3-030-96794-9_42.

[14] M. LeBlanc and J. Crowley, *Relative Risk Trees for Censored Survival Data*, Biometrics, (1992), pp. 411–425, https://doi.org/10.2307/2532300.

[15] C. L. Loprinzi, J. A. Laurie, H. S. Wieand, J. E. Krook, P. J. Novotny, J. W. Kugler, J. Bartel, M. Law, M. Bateman, and N. E. Klatt, *Prospective Evaluation of Prognostic Variables from Patient-Completed Questionnaires. North Central Cancer Treatment Group.*, Journal of Clinical Oncology, 12 (1994), pp. 601–607, https://doi.org/10.1200/JCO.1994.12.3.601.

[16] S. F. McGough, D. Incerti, S. Lyalina, R. Copping, B. Narasimhan, and R. Tibshirani, *Penalized Regression for Left-Truncated and Right-Censored Survival Data*, Statistics in Medicine, 40 (2021), pp. 5487–5500, https://doi.org/https://doi.org/10.1002/sim.9136.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, and D. Cournapeau, *Scikit-Learn: Machine Learning in Python*, The Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.

[18] P. H. Rossi, R. A. Berk, and K. J. Lenihan, *Money, Work and Crime: Some Experimental Results*, 1980, https://doi.org/https://doi.org/10.1016/C2013-0-11412-2.

[19] M. Schumacher, G. Bastert, H. Bojar, K. Hübner, M. Olschewski, W. Sauerbrei, C. Schmoor, C. Beyerle, R. Neumann, and H. Rauschecker, *Randomized 2x2 Trial Evaluating Hormonal Treatment and the Duration of Chemotherapy in Node-Positive Breast Cancer Patients. German Breast Cancer Study Group.*, Journal of Clinical Oncology, 12 (1994), pp. 2086–2093, https://doi.org/10.1200/JCO.1994.12.10.2086.

[20] P. Wang, Y. Li, and C. K. Reddy, *Machine Learning for Survival ANALYSIS: A Survey*, ArXiv: 1708.04649, (2017), pp. 1–39, https://doi.org/10.48550/arXiv.1708.04649.

[21] W. Yao, H. Frydman, D. Larocque, and J. S. Simonoff, *Ensemble Methods for Survival Data with Time-Varying Covariates*, ArXiv:2006.00567, (2020), https://doi.org/10.48550/arXiv.2006.00567.