*mathematics*

MDPI

*Article*

# A Fuzzy Random Survival Forest for Predicting Lapses in Insurance Portfolios Containing Imprecise Data

**Jorge Luis Andrade * and José Luis Valencia**

Faculty of Statistics, Complutense University, Puerta de Hierro, 28040 Madrid, Spain
* Correspondence: jorandra@ucm.es; Tel.: +346-5320-7234

**Abstract:** We propose a fuzzy random survival forest (FRSF) to model lapse rates in a life insurance portfolio containing imprecise or incomplete data such as missing, outlier, or noisy values. Following the random forest methodology, the FRSF is proposed as a new machine learning technique for solving time-to-event data using an ensemble of multiple fuzzy survival trees. In the learning process, the combination of methods such as the c-index, fuzzy sets theory, and the ensemble of multiple trees enable the automatic handling of imprecise data. We analyse the results of several experiments and test them statistically; they show the FRSF's robustness, verifying that its generalisation capacity is not reduced when modelling imprecise data. Furthermore, the results obtained using a real portfolio of a life insurance company demonstrate that the FRSF has a better performance in comparison with other state-of-the-art algorithms such as the traditional Cox model and other tree-based machine learning techniques such as the random survival forest.

**Keywords:** survival analysis; fuzzy logic; lapse rates; imprecise data

**MSC:** 62N86



**Citation:** Andrade, J.L.; Valencia, J.L. A Fuzzy Random Survival Forest for Predicting Lapses in Insurance Portfolios Containing Imprecise Data. *Mathematics* **2023**, *11*, 198. https://doi.org/10.3390/math11010198

Academic Editor: David Zapletal

Received: 24 November 2022
Revised: 25 December 2022
Accepted: 28 December 2022
Published: 30 December 2022



## 1. Introduction

The interest of survival analysis is modelling time-to-event datasets in which the response variable is composed of two vectors: the time, which is an interval variable, and the binary variable of the censoring status to indicate the occurrence of the event.

In the insurance industry, lapse rates information occurs as time-to-event datasets, so they are predicted using survival analysis.

Lapse rates modelling is required for commercial product design and regulatory reasons, as well as to measure the underlying lapse risk. Insurance companies under the regulatory framework of IFRS 17 [1] and the European Solvency II standard are required to estimate future lapse rates to calculate their best estimate liabilities and the solvency capital required. Moreover, according to [2], the lapse risk in a life insurance portfolio is the most important risk, accounting for almost 40% of the capital requirement within the life underwriting risk module, which includes risk factors such as longevity, mortality, disability, catastrophe, and expenses.

Time-to-event data have been commonly analysed using the Cox regression model [3] under the proportional hazards assumption. The random survival forest (RSF) algorithm has been proposed as an extension of the random forest algorithm [4] for the survival problem. In the learning process at each tree, the maximisation of the log-rank statistical test [5] is used for node splitting, [6] demonstrates that the RSF's performance is higher than that of the Cox model, and its use is recommended when the assumptions of proportional risks are not met [7] or when the effect of the explanatory variables is nonlinear.

The RSF algorithm has been used in numerous medical, financial, and economic studies [8,9]. It demonstrates a lower generalisation error than other right-censored data techniques; the log-rank test and the c-index [10] are used as splitting rules at each node.

Recently, fuzzy set theory has been included in learning decision trees [11–15] because datasets contain imprecise data such as missing values and noise in the class or outlier examples. In insurance datasets, incomplete and imprecise data are present for many reasons, such as the expense of obtaining the information, differences in the underwriting process among insureds, changes in data storage, or errors in obtaining and processing the insureds' data. Fuzzy logic and fuzzy sets provide the flexibility to deal with these types of datasets without affecting the performance of the algorithm.

Previously, in [16], we proposed the fuzzy survival tree (FST) algorithm as a fuzzy tree-based algorithm for survival analysis in clinical datasets. It presented the inclusion of fuzzy logic in combination with the c-index in the learning process as a rule for splitting a node. The obtained results validated FST as a robust algorithm for solving time-to-event data.

In the present article, we introduce the fuzzy random survival forest (FRSF), which uses multiple trees for the survival estimation, as an extension of the RSF and FST. FRSF uses the robustness of the randomness of RSF and the flexibility for handling imprecise data of FST. The algorithm steps and ensemble strategy are presented, as well as a large variety of experiments for modelling imprecise data.

This article presents a new FRSF algorithm that improves the results obtained with only one tree-based learner. Moreover, FRSF has the capacity to handle incomplete and imprecise data such as missing values, outliers, and noisy values properly. Additionally, the results show the application of FRSF and FST to an insurance portfolio and, finally, FRSF's robustness in comparison with other tree-based algorithms and Cox's model.

This paper is presented as follows: Section 2 is dedicated to explaining the FRSF algorithm as an ensemble of multiple FSTs, and Section 3 presents the protocol, the dataset description, and the results of the experiments. In all experiments, the dataset used comes from a life insurance company.

The proposed algorithm was developed in Python code, and this implementation takes as its starting point the RSF algorithm developed by [17].

## 2. FRSF Algorithm for Survival Analysis

### *2.1. FRSF: The Survival Analysis and Fuzzy Logic Basis*

The survival datasets are organised by rows and columns $i \times j$; $i$ represents a policyholder in the insurance portfolio, and $j$ is one explanatory or response variable. Survival datasets are represented by $([T_i, \alpha_i], X_i)$, where, for a policyholder $i$, the response variable is composed by time variable $T_i$ and binary censoring status variable $\alpha_i$ indicates whether the event has occurred or not; and $X_i$ is the explanatory variables vector.

In this case, for lapse rates prediction, the time variable is the permanence of the policy in the portfolio from the effective date, and the binary variable indicates lapse or no lapse. The policy can be censored because it reaches maturity, because at the end of the study, it has not experienced an event, or due to premature termination for reasons other than lapses.

In survival analysis, the cumulative hazard function (CHF) is a risk measure for the portfolio in the study; it is computed as $\Lambda_t = \sum_m \frac{\alpha_{t_m}}{|E_{t_m}|}$, where $\alpha_{t_m}$ is the lapses produced and $|E_{t_m}|$ is the policyholders in force at time $t_m$. The times are discretised and can only take the values $t \epsilon t_1, t_2, t_3, \ldots, t_e$.

The fuzzy system implementation in this study comprises the fuzzification, inference system, and defuzzification processes.

- Fuzzification translates crisp inputs to fuzzy values. In the fuzzy sets representation, let $|\mathrm{E}|$ denote the universe of all policyholders in a dataset. A fuzzy set $M \subset |E|$ is characterised by a membership function $\mu_M : |E| \rightarrow [0,1]$ that associates each policyholder $i$ of $|\mathrm{E}|$ with a number $\mu_M(i)$ in the interval $[0,1]$ representing the degree of membership of policyholder $i$ to data subset $M$. The fuzzy sets and partitions are defined at each test node in a tree. A linear membership function is used.
- The inference system process comprises the fuzzy rule base (FRB) and the inference mechanism:

- FRB is the set of fuzzy rules and the fuzzy database with the fuzzy sets information. The fuzzy rule can be expressed by [18]:

$$If\ (x_1\ b\ is\ A_1\ b)\ and\ (x_2\ b\ is\ A_2\ b)\ and \ldots and\ \left(x_j\ b\ is\ A_j\ b\right)\ \ then\ (CHF\ b\ is\ \Lambda\ b)$$

  where $A$ is the explanatory variable value and $b$ represents the number of rules.
- The inference mechanism generates the output from the system using fuzzy reasoning, using the FRB to map the input to the output. It is implemented through the node daughters' definition and in the estimation process before defuzzification. This point deploys that one policyholder activates several terminal nodes at the same tree.

- Defuzzification is the final step: the trees outputs are combined to obtain a crisp output. This step is detailed in Section 2.2.2.

### *2.2. FRSF Is an Ensemble Based on Fuzzy Survival Trees*

We propose an FRSF algorithm to generate a fuzzy random forest whose trees are fuzzy survival trees. FRSF follows the Breiman's methodology, and randomisation is introduced in two forms. (1) For growing a tree, bagging is used, so only 67% of the data are used for training; (2) for the node splitting, only a random subset of the explanatory variables is used, meaning that if an important variable is excluded in a node split, it might be considered in other nodes splits in the same tree. The fuzzy logic is introduced in the learning process. At each test node, fuzzy sets are defined, and its flexibility allows handling incomplete and imprecise data in training and test stages. The combination of random forest and fuzzy logic increases the diversity of the trees and improves the performance when they are ensembled; this algorithm is presented in Table 1.

**Table 1.** FRSF Algorithm.

| | |
|---|---|
| 1. | Generate $s$ bootstrap samples with replacement. |
| 2. | Apply FST algorithm to each sample obtained in the previous step to build a fuzzy survival tree. |
| 3. | Repeat steps 1 and 2 until multiple FSTs are constructed. |
| 4. | Estimate the $\Lambda_t$ for unknown policyholders applying the FRSF ensemble algorithm detailed in Section 2.2.2. |

Each tree in the FRSF is constructed as a fuzzy survival tree; we proposed this tree by the FST algorithm presented in [16].

#### 2.2.1. In FRSF the Base Learner Is a Fuzzy Survival Tree

FST is a tree-based survival algorithm; the splitting criterion is a combination of c-index and fuzzy sets. In the learning process, at each new node, the policyholder's membership degree is considered for c-index maximisation, then a genetic algorithm works to create the fuzzy nodes. Table 2 shows the main steps of the algorithm.

**Table 2.** FST Algorithm.

| | |
|---|---|
| 1. | At root node, start with all policyholders with $\mu_{nroot}(i) = 1$. |
| 2. | Define the discriminator function to do the split |
| | a. Select a temporary crisp split at each node, $\theta_n$. |
| | b. Determine the overlap region by $\beta_n$. |
| 3. | Repeat steps a and b until the scoring functions are minimised. |
| 4. | Grow the FST until the stopping criteria are met. |

A tree is grown starting at the root node, which comprises all the data. At the root node are all the training policyholders $|E|$ with a membership degree equal to 1; $\mu_{nroot}(i) = 1$.

The root node is divided into two daughter nodes. Left and right $\{n_L, n_R\}$ nodes are created thanks to an iterative search for the most discriminant explanatory variable among a subset of them; this is a variant of the original FST algorithm that we proposed in [16], where all explanatory variables were candidates at each node split. The selection from a random subset of explanatory variables is important under a forest vision instead of a single tree to avoid correlation among trees in the forest.

The process is repeated in a binary and recursive fashion at each node until the stopping criteria are met. Finally, a label is attached to every terminal node.

A discriminator function is attached to each node; we use a linear membership as a discriminator. It determines the split of the test node and its fuzziness by $(X_{ij}, \theta_n, \beta_n) \rightarrow [0, 1]$; the parameters defining it are as follows: $X_{ij}$ is the value for a policyholder $i$ for the selected explanatory variable $j$, $\theta_n$ is the value of the split, and $\beta_n$ is the parameter that determines the overlap region.

In crisp trees, a policyholder goes only to one of the daughter nodes created in the split, so a unique path is followed, and then a unique terminal node is reached. In fuzzy or soft trees, if the policyholder's value is within the overlap region determined by the discriminator function, it goes to both daughter nodes following multiple paths in parallel; as a consequence, a policyholder can activate multiple terminal nodes.

In Figure 1, the FST has three test nodes and five terminal nodes or leaves. Each test node is marked with the parameters of its discriminator function $f(X_j, \theta_n, \beta_n)$. In the FST learning, some policyholders go only to the right daughter node, others go only to the left, and those in the overlap region go to both successors. As an example, for survival prediction of a policyholder $i$ and explanatory variables $X_j$= [$X_0$ = 0.08, $X_1$= 0.31], the path starts at the root node *Root*, and passes through both test nodes [$l$ and $r$] because the $X_1$ value is within the overlap region; then, in nodes $l$ and $r$, it is evaluated on variables $X_0$ and $X_1$, respectively. Finally, it activates two leaves [*1.l.r* and *r.r*].

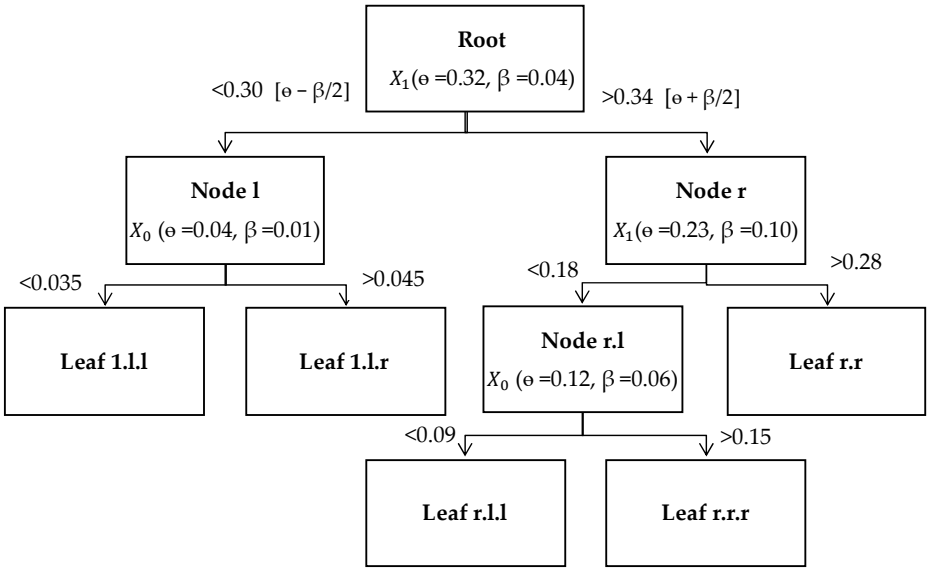



**Figure 1.** Example of a fuzzy survival tree (FST).

A piecewise linear membership function is used to calculate the degree of membership $\mu$ of a policyholder $i$ to the daughter node $\{n_L, n_R\}$:

$$\mu_{nL}(i) = \begin{cases} 1 & X_i \leq \theta_n - \beta_n/2 \\ \frac{(\theta_n + \beta_n/2) - X_i}{\beta_n} & \theta_n - \frac{\beta_n}{2} < X_i \leq \theta_n + \frac{\beta_n}{2} \\ 0 & X_i > \theta_n + \beta_n/2 \end{cases} \qquad \mu_{nR} = 1 - \mu_{nL}$$

Following the formula above, the outputs for the membership degree are not singleton.

If the policyholder's value $X_i$ is equal to the split $\theta_n$, its membership degree is 50%.

To handle a missing value in the explanatory variable selected for the split $X_j$, policyholder $i$ continues to both daughter nodes, and the degree of membership in both daughter nodes is:

$$\mu_{n_L} = \mu_{n_R}(i) = \frac{1}{2}$$

This feature of the algorithm makes it possible to automatically handle missing data in the learning process of the algorithm without making a priori imputations of the value to be taken by an explanatory variable in the case of missing data.

At each node, a class label is represented by $\Lambda$, the CHF value. The degree of membership of a policyholder $i$ in a defined class label $\Lambda$ is $\mu_\Lambda(i)$ and is computed taking into account the CHF value in both daughter nodes:

$$\mu_\Lambda(i) = \sum_{k \in \{n_L, n_R\}} \mu_k(i)\Lambda_k$$

The calculated CHF is common for all policyholders of node $n$; then, the following applies to the policyholders that conform to a specific node: $\Lambda_n(t \mid X_i) = \Lambda_n(t)$, if $i \, \epsilon \, n$.

For the FST learning process, we describe the two steps presented in the study [16]. The objective is to compute the discriminator function at each node. The first step searches the most discriminant $X_j$ explanatory variable and its value $\theta_n$ among a subset of explanatory variables. In a second step, the overlap region is determined by $\beta_n$, and the tree grows until one of the stop criteria is met.

Searching for the Explanatory Variable and Splitting (Crisp)

As in the RSF algorithm, the cut-off value $\theta_n$ is chosen among a random subset of explanatory variables $X$ with the particularity that its values are modified to consider the node membership degree; in this step, the target is to maximise the c-index statistic.

The RSF iterative algorithm [6] for trees construction is adapted to consider the fuzzy sets dynamics. The degree of membership of every policyholder's value $X_i$ in the node is set as follows: at each node, the $X_i$ value is adapted to consider the cumulative membership degree and is renamed $X_{i(n)}$. The cumulative membership degree is calculated recursively from the root to the parent node $\mu_p(i)$. The policyholder's renamed value $X_{i(n)}$ is computed as $X_{i(nL)} = X_i\mu_{nL}(i)\mu_p(i)$ and $X_{i(nR)} = X_i\mu_{nR}(i)\mu_p(i)$ for the left and right daughter nodes, respectively.

To determine $\theta_n$, the iterative search uses the Harrell's c-index statistics. It is designed to estimate the concordance probability $P\big(\Lambda_i\langle\Lambda_j \mid T_i > T_j\big)$; its inclusion as a splitting criterion is to maximise the $P\big(\, i \in n_l \,, j \in n_r \mid T_i < T_j\big)$ in the partition of policyholders into daughter nodes by $\theta_n$. In addition, in this way, the discrepancy between splitting and evaluation criteria is overcome in survival trees. Ref. [10] justifies and recommends c-index as the splitting criterion because it improves RSF in small-scale clinical studies.

The c-index is calculated as follows [19]:

Let $\Lambda_n(t \mid X_i)$ be the CHF of a policyholder $i$ that belongs to node $n$; policyholder $i$ has a worse prediction than the individual $j$ if $\Lambda_n(t \mid i) > \Lambda_n(t \mid j)$.

1. Form all possible pairs of observations from the dataset.
2. Omit those pairs where the shorter event time is censored. Pairs $(i, j)$ are also omitted if $T_i = T_j$ unless $\big(\delta_i = 1, \delta_j = 0\big)$ *or* $\big(\delta_i = 0, \delta_j = 1\big)$ *or* $\big(\delta_i = 1, \delta_j = 1\big)$. This last restriction means that the pair is computed when its times are equal if at least one of the observations is an event.
3. Let the resulting pairs be denoted by Y. *Permissible* = $|\mathrm{Y}|$.
4. Then, evaluate the computation of all the permissible pairs. The sum is called *Concordance*.
   (a) If $T_i \neq T_j$, count 1 for each $y \in \mathrm{Y}$ in which the individual with the shorter time has a worse predicted result.

(b) If $T_i \neq T_j$, count 0.5 for each $y \in \mathrm{Y}$ in which $\Lambda_n(t\,|j) = \Lambda_n(t\,|i)$.
(c) If $T_i = T_j$, count 1 for each $y \in \mathrm{Y}$ in which $\Lambda(t\,|j) = \Lambda_n(t\,|i)$.
(d) If $T_i = T_j$, count 0.5 for each $y \in \mathrm{Y}$ in which $\Lambda_n(t\,|j) \neq \Lambda_n(t\,|i)$.

5. The c-index is defined by: $c - index = Concordance/Permissible$
6. The error rate is $Error = 1 - c - index$. If $c - index = 0.5 = Error$, this means that the algorithm is doing no better than random guessing.

The daughter nodes defined by the cut-off value $\theta_n$ are temporary because the parameter for defining the overlap region $\beta_n$ is equal to zero (crisp split). In the following step, $\beta_n$ and the fuzzy sets configuration are computed to determine the sets of policyholders for each daughter node.

Fuzzification and Labelling

The fuzzification is determined at each node by the algorithm learning process and is represented by the overlap region and the fuzzy sets dynamic. The labelling is represented by the cumulative hazard attached to each node $\Lambda_n$.

In this step, the cut-off point determined by the most discriminant explanatory variable $X_j$ and its value $\theta_n$ obtained in the previous step are kept frozen.

There are several methods for defining the overlap region, such as genetic algorithms, artificial neural networks, and fuzzy clustering algorithms [20–22]; our proposition is to use a genetic algorithm.

The best overlap region is determined by the genetic algorithm proposed in [23]; the $\beta_n$ width defines the overlap region, and it is found by minimising the fitness function:

$$\sum_{i \in n} \mu_p(i) \left[\mu_\Lambda(i) - \mu'_\Lambda(i)\right]^2$$

where $\mu'_\Lambda(i) = \mu'_{nL}(i)\Lambda_{nL} + \mu'_{nR}(i)\Lambda_{nR}$ is the membership function to the left and right daughter nodes defined in the previous subsection, but in this case, it is updated with every candidate value of $\beta_n$ that comes from every iteration of the genetic algorithm. A new CHF is proposed in each genetic algorithm's iteration; this is the labelling.

As the overlap region is necessary for each node, the genetic algorithm runs at each node splitting in every tree.

The range of possible values for the $\beta_n$ parameter is delimited by the range of variation of the explanatory variable $X_j$ of the policyholders belonging to the node $n$, and it is defined by the interval:

$$max\{0, min\left[\left(max\left(X_j(\cdot)_n\right) - \theta_n\right), \left(\theta_n - min\left(X_j(\cdot)_n\right)\right)\right]\}$$

The parameters for genetic algorithm are implemented through empirical testing due to the high complexity of the existing methods and the flexibility of the fuzzy logic in adjusting the parameters [24]. The genetic algorithm was run with the following recommended parameters:

max_num_iteration:10, population_size:10, mutation_probability:0.01,
elit_ratio: 0.20, crossover_probability: 0.7, parents_portion: 0.3,
crossover_type:'uniform'.

This configuration of the genetic algorithm provided satisfactory solutions in a feasible amount of time.

At the end of this step a discriminator function is fitted and represented by $\left(X_j, \theta_n, \beta_n\right)$ for each node, and the fuzzy sets dynamic works through a piecewise linear membership function using the discriminator function. Finally, the new daughter nodes are configured.

2.2.2. FRSF Defuzzification and Ensemble

The FRSF is an ensemble of multiple FSTs; in this section, we describe how the defuzzification and ensemble process is carried out to estimate a unique $\Lambda(t \mid X_i)$ of an

unknown policyholder in the forest. Table 3 shows the part of the FRSF algorithm for the ensemble.

**Table 3.** Algorithm FRSF—Ensemble.

| | |
|---|---|
| 1. | Estimation of $\Lambda(t \mid X_i)$ at each tree |
| 2. | Apply the Aggregation 1 function over $\Lambda(t \mid X_i)$ |
| 3. | The FRSF computes the $\Lambda(t \mid X_i)$ as an average over the aggregator obtained in the previous step (Aggregation 2). |

The FRSF stops when the number of trees is reached, and at each tree, the growing process stops when one of the following criteria are met:

- The node contains the minimum number of policyholders with true events;
- The set of explanatory variables for the split is empty;
- Or the minimum number of policyholders allowed in a node has been reached.

The defuzzification and ensemble strategy is represented in Figure 2; first, the CHF is estimated at each tree, the results of the terminal nodes for an unknown policyholder are aggregated with the Aggregation 1 function, then a $\Lambda(t \mid X_i)$ for each tree is computed. To combine the information of every tree, the Aggregation 2 function is used, and, finally, a unique CHF for a policyholder is estimated. In [25], multiple strategies are presented as aggregation functions.

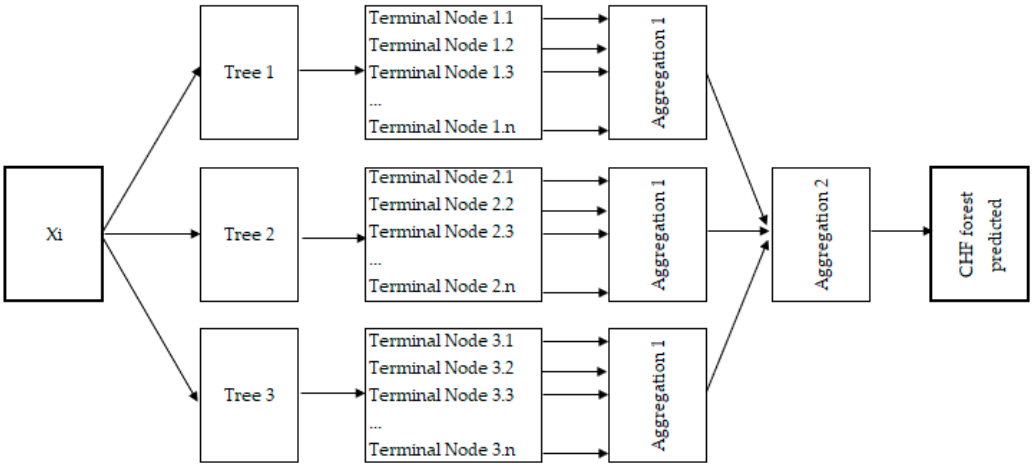



**Figure 2.** FRSF defuzzification and ensemble strategy.

Once the growing forest learning process has finished, the defuzzification process starts at each tree; the objective is to compute a unique $\Lambda$ for every unknown policyholder in the test dataset. In every tree, the new policyholder is dropped from the root to the terminal nodes. At each test node, policyholder $i$ is evaluated according to the discriminatory function $(X_j, \theta_n, \beta_n)$ and follows one or multiple paths simultaneously. If there are missing data, as in the training process, the policyholder continues through the two daughter nodes, creating at least two simultaneous paths in the tree.

In every terminal node $n_t$, a different $\Lambda_{n_t}$ is calculated, which represents the node and is common for all policyholders that belong to $n_t$. As mentioned before, it is possible for a policyholder to reach multiple terminal nodes in the same tree, so for the tree prediction of a unique CHF of an unknown policyholder, one can follow some basic aggregation operators as the maximum, minimum, or average; this is represented as the Aggregation 1 function in Figure 2.

The FRSF ensemble for the forest consists of aggregating the different tree results; it follows the basic strategy represented in Table 3. The information from the different trees is combined to obtain a unique $\Lambda$ of each policyholder of the test dataset.

In the following sections, the average is used as an aggregator operator for the Aggregation 1 and 2 functions.

## 3. Results

### *3.1. Protocol*

This section presents the set of experiments carried out; the comparative results show the better performance of the FRSF. All experiments are conducted on a life insurance portfolio, which is described below. We form two groups of experiments:

- FRSF behaviour in processing imprecise data.
  - Missing values.
  - Noisy data and outliers examples.
- Application of fuzzy survival models on a real insurance portfolio and comparison with other survival models.
  - FST in comparison with RST_log, RST_cind, and the Cox model.
  - FRSF in comparison with RSF_log, RSF_cind, and the Cox model.

Each tree of the forest is built to its maximum size; therefore, until one of the parameters indicated below is met, pruning is not working.

In these experiments, the FRSF is compared with other state-of-the-art methods such as RSF, when the rule to split is the log-rank test (RSF_log) or the c-index (RSF_cind), and the Cox model. The Python libraries used are randomsurvivalforest [17], Skranger [26], and scikit-survival [27].

For comparison of the tree-based survival algorithms, the following parameters are set. The number of trees is 10, the maximum number of policyholders in a terminal node is 100, and the number of lapse events in a terminal node is 50. The number of explanatory variables for node splitting is sqrt($a$), where $a$ is the number of explanatory variables available at the node.

Prior to algorithm training, the numerical explanatory variables are normalised [0, 1]. Other variables are not transformed or entered into the fuzzification process.

To replicate the results shown below, the seed is fixed in all the randomness components of the algorithm, such as bagging, bootstrapping, and the selection of explanatory variables for the split, and in the genetic algorithm, the seed is chosen randomly but then is fixed to be able to replicate its outputs.

### *3.2. Validating the Experiments by Statistical Tests*

In these experiments, a fivefold cross-validation non-stratified technique was performed 10 or 5 times independently. This is a statistical method that uses different partitions of the dataset. Five random subsets of equal size are created; four subsets are used for training, and the fifth is used for testing.

To compare the algorithms' performance in the experiments below, the nonparametric Friedman aligned test is used. Under the null hypothesis, it states that the algorithms are equivalent, a rejection of which implies significant differences in the performance of all algorithms. Finally, a post hoc Holm test [28] is performed to find statistically significant differences between the control algorithm FRSF and the other algorithms.

The Wilcoxon signed-rank test [29] is used to compare the results of two algorithms. The objective of this test is to detect differences in the performance of the two methods when the null hypothesis is rejected.

### *3.3. Datasets Description*

All experiments are applied on a real portfolio of a life insurance company; the dataset is provided by one of the largest insurers in the Ecuadorian market. This dataset is available at the following link: https://github.com/Jorandra/FRSF_survival_fuzzy_prediction (accessed on 23 November 2022). A previous study used a similar but older dataset from the same insurance company, which was presented and described in [30].

The number of policyholders is 7914, and the study period is from 2007 to 2021; Table 4 describes the 11 explanatory variables.

**Table 4.** Description of Explanatory Variables.

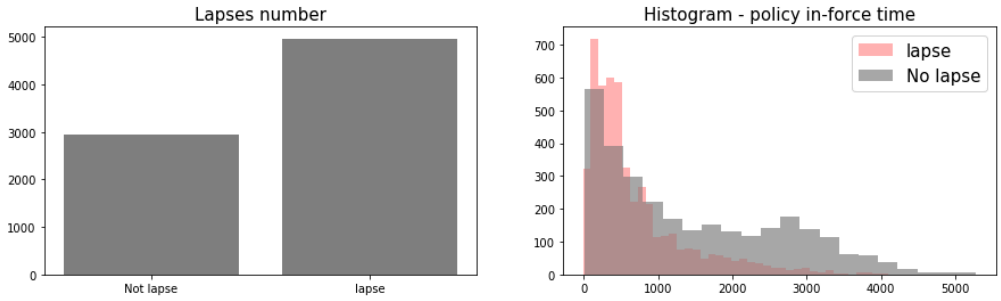
| Variable | Data | Description |
|---|---|---|
| Age | Min 18, Mean 38.12, Max 73 | |
| Premium-Product | Mean 209.12, std 618.25 | main coverage |
| Premium–Complementaries | Mean 39.05, std 582.95 | complementaries coverages |
| Smoker-condition | Yes: 6.42%, No: 93.58% | |
| Sex | F: 43.20%, M: 56.8% | |
| Point of sales | Q 36.25%, G 47.18%, C 8.45%, M 7.28%, A 0.83% | The letters represent different cities |
| Product type | Term life 54.12%, Universal life 45.88% | There are 11 sub-categories of products |
| Distribution channel | Tied Agents 80.19%, Corporate Agents 11.57%, Individual Agents 8.21%, Others 0.03% | |
| Payment frequency | Monthly 77.69%, Annual 18.81%, Others 3.50% | Others include four-monthly, quarterly, and biannual |
| Payment method | Bank debit 83.03%, Credit card 16.94%, Payroll 0.03% | |
| Profession | A 78.09%, B 18.09%, C 3.82% | Variable not detailed |

The response variable is composed of two variables; the time variable measures the policyholder's time in the portfolio since the effective date, and the binary variable censoring status indicates whether or not a lapse occurs in the study window. Table 5 describes the response variable.

**Table 5.** Description of the Response Variable.

| Variable | Data | Description |
|---|---|---|
| Time | Min 0, Mean 964, Max 5280 | Data in days |
| Censoring Status | Yes: 63%, No: 37% | Yes: Lapse, No: others |

Figure 3a shows the percentage of policyholders for which an event did or did not occur, and Figure 3b shows the number of policyholders at each time differentiated according to their status of lapse or no lapse.




(**a**) (**b**)

**Figure 3.** Response variable (**a**) event status and (**b**) policies survival time by event type.

For the following experiments, we also create several sub-datasets to demonstrate and validate the FRSF results.

The new sub-datasets are random samples from the original dataset (n_100). Their sizes are 50%, 30%, and 10% (n_50, n_30, n_10) of the original. Sub-datasets have 11 explanatory

variables, as with the original one. To obtain the algorithm parameters for the sub-datasets training, the indicated percentages are multiplied by the parameters of the original dataset.

### *3.4. FRSF Results Processing Imperfect Data*

The following experiments have material importance for datasets with imprecise, vague, imperfect, or incomplete data. This situation is frequent in insurance business portfolios. This experiment measures the behaviour and stability of the FRSF ensemble in datasets with missing values, noise in the time, and outlier examples in the explanatory variables.

As we mentioned in Section 2, in FRSF, one advantage is that missing values are not excluded in the learning process, and they are handled automatically. No a priori assumption on imputation to a missing value is necessary.

This experiment is divided into two, one for processing missing data and another for noisy data.

The results of this subsection are shown as a comparison between FRSF performance before and after introducing imperfect data. It measures how much the FRSF c-index decreases in the presence of imperfect data.

The decrease in the percentage of the c-index is computed, in accordance with [25], as:

$$\%decrease\ c-index = 100 \cdot \frac{c-index(original) - c-index(imperfect)}{c-index(original)}$$

where $c-index(original)$ is the $c-index$ average for the original data and c-index(imperfect) is the average for the dataset with imprecise data.

#### 3.4.1. Results of the FST Processing Missing Data

In this experiment, missing data are introduced in the training and test datasets, both in categorical and numerical variables.

A percentage $m\%$ of missing data are introduced in a dataset of $|E|$ policyholders. We randomly select $m\% \cdot |E|$ policyholders of the dataset that will be uniformly distributed; at each $X_i$ selected, a missing value is introduced.

This is a simulation study where different percentages of missing values are inserted in each dataset as shown in Table 6. The percentages of missing values introduced are 10%, 15%, and 20%.

**Table 6.** FRSF Performance for Different Percentages of Missing Values.

| **Datasets** | **Introduction of Missing Values** | | | |
|---|---|---|---|---|
| | **Without** | **10%** | **15%** | **20%** |
| | | **% Decrease in C-Index** | | |
| n_100 | 0.5583 | 0.614 | 0.573 | 0.945 |
| n_50 | 0.5479 | −0.162 | −0.072 | −0.038 |
| n_30 | 0.5438 | −0.054 | −0.128 | −0.254 |
| n_10 | 0.5392 | −0.058 | 0.726 | 0.698 |

In this experiment, for each dataset, a fivefold cross-validation is performed five times. The Friedman aligned test accepts the null hypothesis of equality of the FRSF results before and after introducing missing values with a confidence level of 99%. This confirms the robustness of the algorithm in datasets with missing data.

#### 3.4.2. Results of the FRSF Processing Data Noise

This experiment studies how noise can modify the FRSF performance. The experiment is divided into two tests. In the first, noise is inserted in the response variable, and in the second test, outlier values are inserted into the explanatory variables.

Introducing Noisy Data into the FRSF

Noise is introduced into the response variable, $T_i$. Using a uniform distribution, 10% of the observations are altered with a value that is chosen randomly among all the possible time values. Noise data are introduced only in the training dataset.

Figure 4 shows the c-index differences of the FRSF with original data and the FRSF after introducing noise in the time variable. In the displayed results, the worst scenario is a 3.7% decrease in the c-index for the n_30 size dataset. On the other hand, the differences in the median are close to 0% across all datasets.

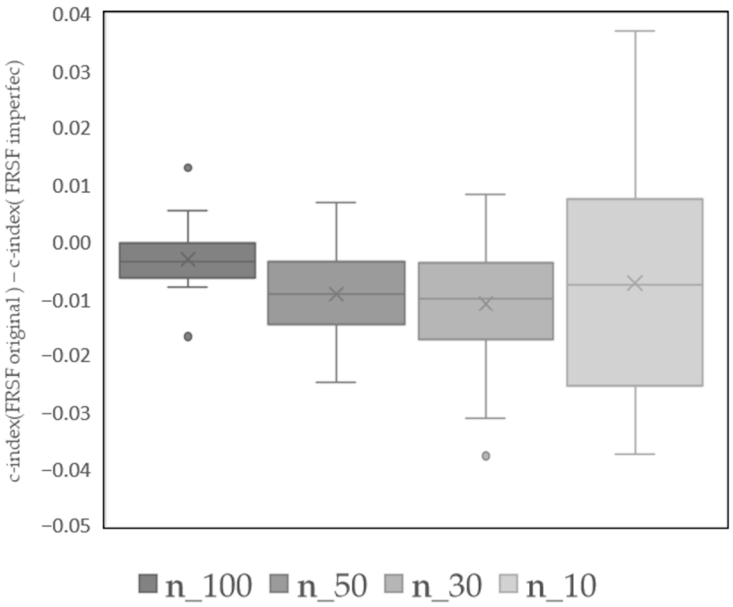



**Figure 4.** FRSF performance with noisy data in different datasets.

For each dataset, a fivefold cross-validation is performed five times. The Wilcoxon ranked test is applied to the FRFS results; it does not show significant differences with a confidence level of 90%. These results validate that the FRSF has good performance in insurance datasets with noisy data.

Effect of Outliers

In this experiment, outlier values are inserted in the explanatory variables of the training dataset. The interquartile method for calculating outliers is used. The lower $q_1$, median $q_2$, and upper $q_3$ quartiles of each numerical attribute are used; additionally, min is the lowest value and max is the highest one.

1. Select a numerical explanatory variable $X_{num\_i}$ at each dataset.
2. For the selected explanatory variable, calculate $v^* = \min\{v/q_3 + (v - 0.5)\ iq \leq \max\ (X_{num_i}) \leq q_3 + v.iq\}$, where $iq = q_3 - q_1$ (inter-quartile range for $X_{num\_i}$).
3. Select 1% of $X_{num\_i}$ values.
4. Define $v_1 = v^* + 0.5$ , $v_2 = v^* + 1$, and $v_3 = v^* + 1.5$.
5. For each selected value, $X_{num\_i}$ is altered by a randomly selected value in the interval $[q_3 + v_i.iq; q_3 + (v_i + 0.5).iq]$ for $i = 1, 2, 3$. At each experiment, a different value is obtained depending on $v_1$, $v_2$ , and $v_3$.

Then, three new datasets with outliers are obtained for each original dataset according to $v_1$, $v_2$, and $v_3$. The results are shown in Table 7.

In this experiment, for each dataset, a fivefold cross-validation is performed five times. The Friedman aligned test accepts the null hypothesis of equality of the results of the FRSF results before and after introducing outliers with a 95% confidence level. The results confirm the good performance of the algorithm in the presence of outliers.

**Table 7.** FRSF Performance with Outlier Examples.

| Datasets | Outliers | | | |
|---|---|---|---|---|
| | **Without** | **v1** | **v2** | **v3** |
| | | **% Decrease in C-Index** | | |
| n_100 | 0.5583 | −0.058 | 0.007 | −0.071 |
| n_50 | 0.5479 | −0.888 | −0.922 | −0.855 |
| n_30 | 0.5438 | −0.498 | −0.558 | −0.603 |
| n_10 | 0.5392 | −0.323 | −0.382 | −0.572 |

### *3.5. Application of the FST to an Insurance Dataset and Comparison of the FST with Other Survival Models*

In this experiment, the FST is applied to a life insurance portfolio for lapses prediction.

In Figure 5, the results validated the findings of [16], where FST performance is better than those of RST_logrank and RST_cind (RST means an RSF algorithm in which the number of trees is 1).

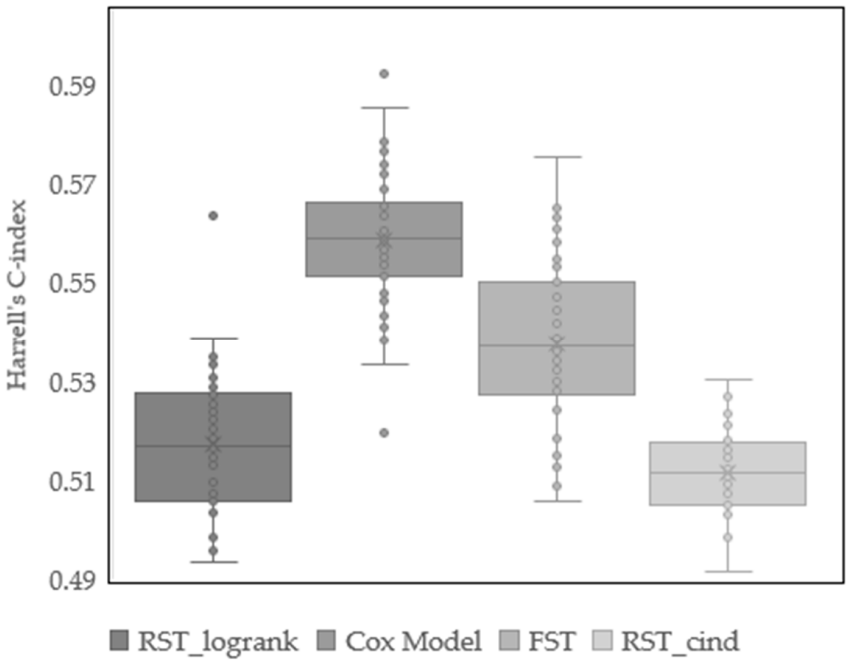



**Figure 5.** FST performance compared with other survival models using the original dataset (n_100).

However, it was also observed that a single fuzzy survival learner is not enough to overcome the performance of the Cox proportional model. For this reason, the following section shows the behaviour of FRSF, which is a multi-tree ensemble that outperforms the traditional Cox model.

### *3.6. Application of the FRSF in Insurance and Comparing the FRSF with Other Survival Models*

In this experiment, the application of the FRSF to an insurance portfolio to predict lapses is presented. Additionally, we compare FRSF's performance with other state-of-the-art methods constructed using survival trees to measure FRSF's effectiveness. We also compare it against Cox's traditional model.

Figure 6 shows the better results of FRSF among the state-of-the-art methods. The median of all differences is greater than zero, which means that FRSF's c-index is higher than that of each other technique compared.

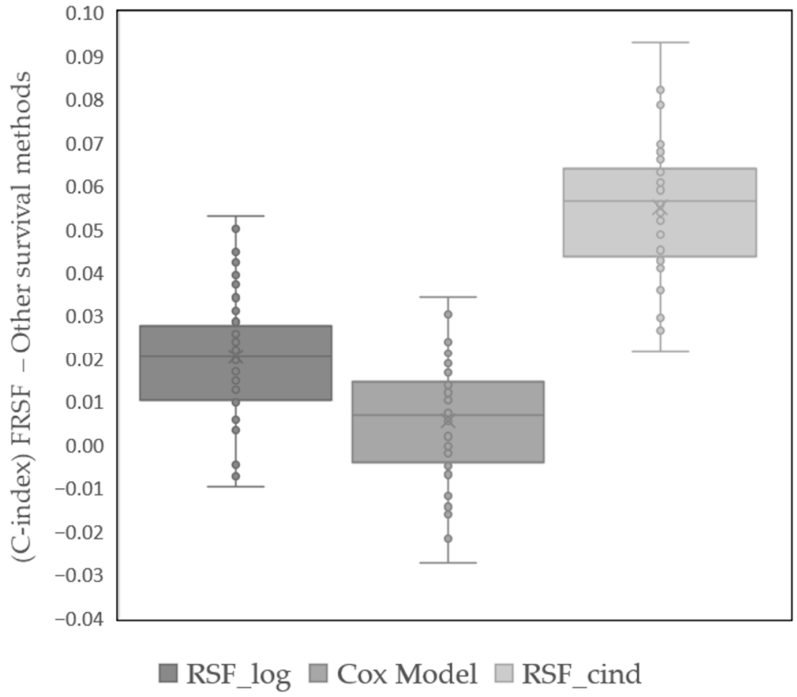



**Figure 6.** C-index difference between FRSF and each survival model using the original dataset (n_100).

Compared to the results of the previous experiment, it is clearly demonstrated how the multi-FST ensemble in the FRSF provides better performance than a single base learner.

In this experiment, the original dataset (n_100) is used. To validate the results in this experiment, a fivefold cross-validation is performed 10 times. The Friedman aligned test rejects the null hypothesis of equality of the algorithms' results with a confidence level of 99.9%, placing the FRSF in the top of the ranking. Following a post hoc Holm test for each pair of comparisons between the control algorithm FRSF and the other survival techniques, the null hypothesis is rejected with a 99%, 99%, and 93% confidence level for the RSF_log, Cox, and RSF_cind models, respectively.

### *3.7. Computational Time*

This experiment is a comparison of the computational time among the tree-based survival algorithms used in this study.

Table 8 shows the CPU times in minutes of RSF_log, RSF_cind, and FRSF. They come from an average of 50 runs of each algorithm; we use a computer with Intel(R) Core(TM) i7-1065G7 1.50 GHz, 16 GB RAM, Windows 10.

**Table 8.** Computational Time for Different Datasets and Number of Trees.

| Datasets | nº_Trees | FRSF | RSF_log | RSF_cind |
|---|---|---|---|---|
| n_100 | 10 | 186.2 | 2.2 | 2.1 |
| n_50 | 10 | 47.3 | 0.9 | 0.8 |
| n_30 | 10 | 15.9 | 0.4 | 0.3 |
| n_10 | 10 | 12.8 | 0.3 | 0.2 |
| n_100 | 1 | 4.5 | 0.1 | 0.1 |
| n_50 | 1 | 5.6 | 0.1 | 0.1 |
| n_30 | 1 | 4.4 | 0.1 | 0.1 |
| n_10 | 1 | 3.2 | 0.1 | 0.1 |

## 4. Discussion

The FRSF, thanks to the introduction of the fuzzy sets, makes it possible to process missing data both in the learning and prediction stages.

The FRSF's performance is robust to the presence of imprecise data such as noise in the response variable and missing values or outliers in the explanatory variables. The performed statistical tests validate the generalisation error that is not lower when original and imperfect c-index results are compared.

Furthermore, when we work with the FRSF algorithm, the performance with imperfect data is better than those of the FST, Cox, and RSF algorithms by definition, as FRSF improves these algorithms' performance using original data as it is demonstrated in Section 3.6.

In the results shown in the previous section, it is verified that a single tree, as in the FST algorithm, does not show better performance compared to the traditional Cox model. Thus, it is necessary to propose an algorithm that ensembles multiple trees.

In FRSF, the computational time is much higher than those of the RSF_log or RSF_cind, whereas it is expected between RSF log and RSF_cind that there are no important differences; however, the FRSF performs better than the Cox model with a relatively small number of ensembles, and this behaviour is not observed in the RSF algorithm.

In all experiments, a piecewise linear membership function is used; for future work, other membership functions may be proposed. Additionally, the average operator is used as the aggregation function for the FRSF ensemble; in the future, it may be proposed to move to other functions, such as maximum, minimum, or even other aggregation methods or aggregation strategies in the defuzzification step, as proposed in [25].

## 5. Conclusions

In this article, we propose the new FRSF algorithm as an ensemble of multiple fuzzy survival trees to solve the time-to-event problem.

The introduction of fuzzy logic in the FRSF algorithm makes it possible to work with imprecise data without lower generalisation capacity. Specifically, insurance datasets can present missing, outliers, or noisy data, and the generalisation capacity does not decrease.

One of the main advantages of FRSF is that no a priori assumption of the value imputed to a missing value is necessary. In practical terms, the FRSF can process missing data automatically.

The FRSF algorithm shows an improvement in performance compared to other tree-based algorithms and the Cox proportional model when insurance datasets are used.

The FRSF and FST algorithms are applied to a real portfolio of a life insurance company for predicting lapses rates.

The computational time for FRSF is higher than for other right-censored machine learning algorithms such as RSF. This is the price paid for having better performance and the possibility of processing missing data.

Additionally, all the comparative results of the experiments are validated by statistical methods.

More accurate lapses rates could yield a significant variation in the determination of the best estimate and solvency capital requirement for insurance companies. In commercial terms, new competitive products could be designed, as well as a differentiated treatment for policyholders regarding their lapse probability.

The FRSF is in Python code, which facilitates its manipulation and future extension; the code is available at https://github.com/Jorandra/FRSF_survival_fuzzy_prediction (accessed on 23 November 2022).

Future work on this topic may focus on alternative membership functions for elements at each node or different aggregation functions for the ensemble. In order to overcome the FRSF's limitation in computational time consumption, it could be implemented in other computer languages to increase the processing speed.

**Author Contributions:** Conceptualization, J.L.A. and J.L.V.; data curation, J.L.A.; formal analysis, J.L.V.; investigation, J.L.A.; supervision, J.L.V.; visualization, J.L.A.; writing—original draft, J.L.A.; writing—review and editing, J.L.A. and J.L.V. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** https://github.com/Jorandra/FRSF_survival_fuzzy_prediction (accessed on 23 November 2022).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. International Accounting Standards Board. *IFRS17:Insurance Contracts*. 2017. Available online: https://www.ifrs.org/supporting-implementation/supporting-materials-by-ifrs-standards/ifrs-17/ (accessed on 23 November 2022).
2. EIOPA. *EIOPA Report on the Fifth Quantitative Impact Study (QIS5) for Solvency II*; EIOPA: Frankfurt, Germany, 2011.
3. Cox, D.R. Regression Models and Life-tables. *J. R. Stat. Soc. Ser. B* **1972**, *34*, 187–202. [CrossRef]
4. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
5. Peto, R.; Peto, J. Asymptotically Efficient Rank Invariant Test Procedures. *J. R. Stat. Soc. Ser. A* **1972**, *135*, 185. [CrossRef]
6. Ishwaran, H.; Kogalur, U.B.; Blackstone, E.H.; Lauer, M.S. Random Survival Forests. *Ann. Appl. Stat.* **2008**, *2*, 841–860. [CrossRef]
7. Kurt Omurlu, I.; Ture, M.; Tokatli, F. The Comparisons of Random Survival Forests and Cox Regression Analysis with Simulation and an Application Related to Breast Cancer. *Expert Syst. Appl.* **2009**, *36*, 8582–8588. [CrossRef]
8. Aleandri, M.; Eletti, A. Modelling Dynamic Lapse with Survival Analysis and Machine Learning in CPI. *Decis. Econ. Financ.* **2021**, *44*, 37–56. [CrossRef]
9. Ptak-Chmielewska, A.; Matuszyk, A. Application of the Random Survival Forests Method in the Bankruptcy Prediction for Small and Medium Enterprises. *Argum. Oeconomica* **2020**, *2019*, 127–142. [CrossRef]
10. Schmid, M.; Wright, M.N.; Ziegler, A. On the Use of Harrell's C for Clinical Risk Prediction via Random Survival Forests. *Expert Syst. Appl.* **2016**, *63*, 450–459. [CrossRef]
11. Begenova, S.; Avdeenko, T. The Research of Fuzzy Decision Trees Building Based on Entropy and the Theory of Fuzzy Sets. In Proceedings of the Collection of Selected Papers of the IV International Conference on Information Technology and Nanotechnology, Samara, Russia, 24–27 April 2018; pp. 296–303.
12. Begenova, S.B.; Avdeenko, T.V. Building of Fuzzy Decision Trees Using ID3 Algorithm. *J. Phys. Conf. Ser.* **2018**, *1015*, 022002. [CrossRef]
13. Olaru, C.; Wehenkel, L. A Complete Fuzzy Decision Tree Technique. *Fuzzy Sets Syst.* **2003**, *138*, 221–254. [CrossRef]
14. Cintra, M.; Monard, M.; Camargo, H. A Fuzzy Decision Tree Algorithm Based on C4.5. *Mathw. Soft Comput.* **2013**, *20*, 56–62.
15. Idris, N.F.; Ismail, M.A. Breast Cancer Disease Classification Using Fuzzy-ID3 Algorithm with FUZZYDBD Method: Automatic Fuzzy Database Definition. *PeerJ Comput. Sci.* **2021**, *7*, e427. [CrossRef] [PubMed]
16. Andrade, J.L.; Valencia, J.L. A Fuzzy Survival Tree (FST). In *Building Bridges between Soft and Statistical Methodologies for Data Science. SMPS 2022. Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland, 2023; Volume 1433, pp. 16–23.
17. Julian, S. Random-Survival-Forest 2021. Available online: https://github.com/julianspaeth/random-survival-forest (accessed on 20 November 2022).
18. Pena-Reyes, C.A.; Sipper, M. Fuzzy CoCo: A Cooperative-Coevolutionary Approach to Fuzzy Modeling. *IEEE Trans. Fuzzy Syst.* **2001**, *9*, 727–737. [CrossRef]
19. Harrell, F.E. Evaluating the Yield of Medical Tests. *JAMA J. Am. Med. Assoc.* **1982**, *247*, 2543–2546. [CrossRef]
20. Liao, T.W.; Celmins, A.K.; Hammell, R.J. A Fuzzy C-Means Variant for the Generation of Fuzzy Term Sets. *Fuzzy Sets Syst.* **2003**, *135*, 241–257. [CrossRef]
21. Aliev, R.A.; Pedrycz, W.; Guirimov, B.G.; Aliev, R.R.; Ilhan, U.; Babagil, M.; Mammadli, S. Type-2 Fuzzy Neural Networks with Fuzzy Clustering and Differential Evolution Optimization. *Inf. Sci.* **2011**, *181*, 1591–1608. [CrossRef]
22. Cintra, M.; Monard, M.; Cherman, E.; de Arruda, C.H. On the Estimation of the Number of Fuzzy Sets for Fuzzy Rule-Based Classification Systems. In Proceedings of the 2011 11th International Conference on Hybrid Intelligent Systems, HIS, Melacca, Malaysia, 5–8 December 2011; pp. 211–216.
23. Bozorg-Haddad, O.; Solgi, M.; Loáiciga, H.A. *Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization*; Wiley: Hoboken, NJ, USA, 2017.
24. Cintra, M.; Camargo, H.; Martin, T. Optimising the Fuzzy Granulation of Attribute Domains. In Proceedings of the 2009 International Fuzzy Systems Association World Congress and 2009 European Society for Fuzzy Logic and Technology Conference, IFSA-EUSFLAT 2009—Proceedings, Lisbon, Portugal, 20–20 July 2014; pp. 742–747.

25. Bonissone, P.; Cadenas, J.M.; Carmen Garrido, M.; Andrés Díaz-Valladares, R. A Fuzzy Random Forest. *Int. J. Approx. Reason.* **2010**, *51*, 729–747. [CrossRef]
26. Wright, M.N.; Ziegler, A. Ranger: A Fast Implementation of Random Forests for High Dimensional Data in *C++* and *R*. *J. Stat. Softw.* **2017**, *77*, i01. [CrossRef]
27. Poelsterl, S. Scikit-Survival: A Library for Time-to-Event Analysis Built on Top of Scikit-Learn. *J. Mach. Learn. Res.* **2020**, *21*, 1–6.
28. Holm, S. A Simple Sequentially Rejective Multiple Test Procedure. *Scand. J. Stat.* **1979**, *6*, 65–70.
29. Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biometrics* **1945**, *1*, 80–83. [CrossRef]
30. Andrade, J.L.; Valencia, J.L. Modeling Lapse Rates Using Machine Learning: A Comparison between Survival Forests and Cox Proportional Hazards Techniques. *An. Inst. Actuar. Esp.* **2021**, *27*, 161–183. [CrossRef] [PubMed]