university of groningen

faculty of science and engineering

A Custom Distribution Fitted Random Forest for Accelerated Failure Time Modelling

Muhammad Rizki Duwinanto

university of groningen / faculty of science and engineering

University of Groningen

A Custom Distribution Fitted Random Forest for Accelerated Failure Time Modelling

Master's Thesis

To fulfill the requirements for the degree of
Master of Science in Artificial Intelligence
at the University of Groningen under the supervision of
Dr. Marleen B. Schippers (Artificial Intelligence, University of Groningen),
Dr. Jiapan Guo (Artificial Intelligence, University of Groningen)
Dr. Tamas Szili-Török, M.D (University Medical Center Groningen)
and
Ömer Tarik Özyilmaz (University Medical Center Groningen)

Muhammad Rizki Duwinanto (s5764971)

August 30, 2025

Contents

Acknowledgments

First and foremost, I would like to praise Allah the Almighty, the Most Gracious, and the Most Merciful for His blessing given to me during my study and in completing this thesis.

Next, I would like to express my sincere gratitude to my supervisors, Dr. Marleen Schippers, Dr. Jiapan Guo, Dr. Tamas Szili-Török, M.D, and Ömer Tarık Özyılmaz. I am especially grateful to Omer for being an amazing supervisor who encouraged me when I was close to giving up on the project, and to Tamas for his invaluable feedback and guidance throughout this work. I would also like to thank Marleen and Jiapan for their excellent teaching and continuous support during the thesis.

Of course, I would also like to thank my family, who have supported me throughout this journey, especially my sister, my brother-in-law, and my parents. Without their unwavering love, encouragement, and prayers, I would not have been able to complete this thesis. They gave me the emotional strength and motivation to keep moving forward and helped me pursue this degree with confidence.

I would like to say a special thanks to my friends who directly supported me in completing this thesis: Ivaylo, Mortaza, Tim, Assaf, Shakoush, Elena, Bayu, David, Max Verbeek, Oliver, Yuxiang, Chanyuang, and many others whose names I cannot mention one by one. Your encouragement, insights, and companionship made this challenging journey much more enjoyable and meaningful.

I also want to express my profound gratitude to the Indonesian Endowment Fund for Education (LPDP) for granting me the scholarship that made my studies possible. Without their financial support, pursuing this degree abroad would have been extremely difficult.

I would like to thank everyone in my family and friends in Indonesia, the Netherlands and Groningen for their support and encouragement. Your belief in me fueled the perseverance needed to complete this thesis.

I would like to thank everyone reading this thesis, because this thesis helped to decide my next life step and realise that it gave me a life lesson more than just a grade.

Lastly, I am thankful to everyone who has directly or indirectly supported me throughout this journey. Each encouragement, piece of advice, and act of kindness has contributed to the completion of this thesis.

Abstract

Survival Analysis is crucial in healthcare for predicting the occurrence of a specific event in a population, such as a patient's death or a particular outcome. A major challenge in survival analysis is handling censored data, where the event of interest has not occurred during the observation period. Most survival analysis methods are currently focused on predicting the survival risk of an event within a certain time, using Random Survival Forest and Neural Networks. To predict the occurrence of time-to-event directly, the Accelerated Failure Time model can be used. Existing methods, such as XGBoost AFT, have already been explored but lack a more data-driven approach. In this research, a random forest is implemented using the Accelerated Failure Time (AFT) loss function to explore the custom-distribution fitting. This research aims to improve the accelerated failure loss function using parametric fitting for three distributions in logarithmic time space of Logistic, Extreme and Normal and non-parametric fitting using Gaussian Mixture Models before the training. Finally, the bootstrapping method is used to avoid overfitting during the distribution fitting process. Custom distribution fitting dramatically enhanced the C-index and metrics, but bootstrapping yielded no and worse results. The usage of Gaussian mixture models also creates worse training and testing performances. The datasets used are the Support and NHANES datasets. The best AFT Random Forest models are configured using the extreme function and Logistic distribution hyperparameter with a custom-fitted distribution without bootstrapping. Compared to other best machine learning models, they underperform to XGBoost and Random Survival Forest during training, especially on the Support Dataset. However, the comparison is more comparable on NHANES, with the test C-Index higher on AFT Forest than Random Survival Forest. In conclusion, Accelerated Failure Random Forest represents a promising step toward more effective direct time-to-event prediction models.

1 Introduction

Artificial intelligence is crucial today for enhancing various aspects of life, including business, social, educational, and healthcare. In healthcare, machine learning algorithms have been proven to change the practice of medicine and the delivery of healthcare [1] [2]. Some examples of the machine learning algorithms that are used in healthcare are support vector machines [3], neural networks [4], and decision trees [5]. The benefit of the application of these tools can enable medical professionals to make more accurate diagnoses, identify diseases earlier and therefore use less aggressive or more focused treatments [6]. These algorithms can be used for machine learning tasks, such as classification and survival analysis, in medical fields.

Survival analysis is a study to predict the occurrence of an event and the time until the event [7]. The time until the event is important to know for practitioners to enhance medical care and the prognosis of a disease. There are several reasons to study the time until an event or a failure time using survival analysis, which are to study the distribution of the data, the risks, and to understand the mechanism of failure. The examples of survival analysis are to study the survival times of cancer patients, the time to pregnancy after gynaecological surgery and the interval of occurrence between long-covid symptoms. The benefits of survival analysis in healthcare are to allow researchers in medical healthcare to describe the progression of a disease and the effects of treatment. It is also beneficial for other fields such as economics and banking. The survival analysis usually studies data from a continuous distribution with censored data, data that exceeds the time of the study. This censoring happens mostly because of the limitations of the study time researchers have.

The most common survival analysis graphical methods are Kaplan-Meier [8], which provide a simple explanation of survival analysis by providing a descriptive analysis. Another common method is Cox Regression [9], which is widely used in a scenario that depends on a predictor variable by assuming a proportional hazard for all the groups. The method, however, only produces a survival risk within a certain time rather than a time-to-event directly.

Some methods in survival analysis predict time-to-event directly, for example, Accelerated Failure Time (AFT) [7]. Accelerated Failure Time assumes that predictors act multiplicatively on a failure time. The Accelerated Failure Time is known to utilise a parametric distribution as an assumption that researchers have to learn before using the methods. Nevertheless, Accelerated Failure Time is known to mimic a biological process that no other methods can do [10].

These survival analyses can be extended and explored through many machine learning methods. The aim of this is to study the effects of each feature that a subject of the study or patient has on the survival function [11]. The methods that have been explored are the Neural Network [12], Survival Support Vector Machine [11] and a Survival Tree [13]. The Survival Tree, a method based on regression decision tree methods, is chosen by the researchers due to its flexibility, interpretability to group prognostic factors, and ability to handle censored data [14]. The Survival Tree, nevertheless, is known to have a high bias that can lead to the models not generalising very well and therefore can be optimised by utilising an ensemble technique, a method combining multiple algorithms of decision trees. The examples of these ensemble techniques are bagging [15], which creates many trees as independent learners and aggregates them and boosting [16], which learn sequentially from a tree that is weak learner and applies a correction to the prediction of the weak learner.

There are known methods of survival analysis methods that have utilised ensemble methods. For instance, XGBoost-AFT [17] and Random Survival Forest [18]. These methods have been proven to generate a better prediction than a single survival tree. XGBoost-AFT is known to be the chosen method to generate time-to-event directly. The algorithm is very robust to be used in predicting survival times. However, there are still known limitations of XGBoost-AFT, such as hyperparameter

sensitivity or instability, a lack of non-parametric assumptions as in the original AFT model, which can lead to errors [19].

There is a need for research on how to approach the model using a non-parametric method so that researchers do not have to tune the model and assume the wrong distribution. There is limited research currently to learn Accelerated Failure Time (AFT) parametrically using a Gaussian mixture model (which is parametric because it assumes Gaussianity on the distribution) [19]. There is also still no research currently that combines the non-parametric or semi-parametric methods with the ensemble methods.

In this research, a new method using the Accelerated Failure Time-based survival analysis tree and random forest is proposed. The technique can be applied generally to all censored and uncensored datasets, especially in all clinical data with varied censorship. The proposed Accelerated Failure Time tree and random forest, which are based on XGBoostAFT loss function, also have custom fitting of known distributions and non-parametric fitting using a Gaussian mixture model. We argue that the Accelerated Failure Time-based survival random forest can provide a better performance on clinical data, especially using custom parametric fitting, non-parametric fitting using Gaussian mixture models and bootstrapping fitting.

1.1 Research Questions

To summarise, the main research question would be: *Can we use Survival Tree analysis to predict time-to-event disease outcomes from the clinical data?* This question consists of the following sub-questions:

Q1. How does the proposed Random Forest AFT compare to other existing methods?

Q2. What is the effect of the proposed distribution fitting algorithm on the proposed AFT Random Forest?

Q3. What is the effect of the proposed bootstrap sampling fitting algorithm on the proposed AFT Random Forest?

1.2 Contribution

- Our main contribution is the random forest algorithm for survival analysis, utilising the Accelerated Failure Time loss function to predict time-to-event directly.

- Custom distribution fitting is utilised to get the best parameter for the covariance of the survival time distribution, which contrasts with the existing XGBoost implementation of using gradient boosting to improve the distribution.

- Bootstrap sampling can also be leveraged to get better custom distribution fitting of sigma, which can improve the performance and combat overfitting.

- The random forest also has a non-parametric fitting using Gaussian mixture models to improve distribution fitting.

1.3 Thesis Outline

The thesis is structured as follows. The second chapter explains the background and theoretical framework for the research. The third chapter discusses the novel methodologies, the metrics and the dataset. The setup for the experiments is explained in Chapter 4. The results and discussion are discussed in Chapters 5 and 6, respectively. Chapter 7 summarises the conclusion, limitations and implications of the research.

2 Background Literature

The background literature needed for the methodologies is explained in this section. The subsection 2.1 describes the existing survival analysis methods. The subsection 2.2 describes the decision tree, ensemble methods from the decision tree and their extension for survival analysis.

2.1 Survival Analysis

Survival Analysis is a study of analysing data in which the time until the event is important [7]. The result of the survival analysis is a response variable called the time until the event, which is often referred to as *failure times*, *survival times*, or *event time*. This event is usually continuous, but survival also allows the response variable to be completely undetermined using censoring. This means that if the data is censored, it is known to exceed the study's time limit. Survival analysis is useful for testing and describing treatment interactions, understanding prognostic factors, modelling the absolute effect of the treatment, understanding the time course of therapy, modifying treatment effects, and adjusting imbalances in treatment allocations.

The types of censoring that could happen in a study are *right-censoring*, *left-censoring*, *interval-censoring* [7] [20]. *Right-censoring* is where the data is censored after a certain time limit of the study, while is where the event is known, the study starts. This is the most common type of censoring because the subject is not known to develop an event due to the limitation of study duration. The most common example of these is the survival analysis of a patient after developing a disease of interest. *Left-censoring* or *delayed entry* is the type of censoring where the event is not known to have happened before the study begins. For example, if a patient is not known to have an underlying condition or a history of drug use before the study begins. This type of censoring is often excluded from the study. Interval censoring is when follow-up happens only for a certain range of responses and is always present. This is common in a study that has a periodic follow-up, such as risks of developing a disease in a certain high-risk group in an area for a certain time. The demonstration of different cases of censoring is shown in Table 1. The illustration of different cases of censoring can be explained in Figure 1.

Type	**Lower Bound Time**	**Upper Bound Time**
Uncensored	T	T
Right-Censored	Finite non-negative	∞
Left-Censored	0	Finite non-negative
Right-Censored	Finite non-negative	Finite non-negative

Table 1: Different Types of Censoring [17][20]

The survival analysis usually measures the survival function $S(t)$ [7], which is given by Equation 1.

$$S(t) = \text{Prob}(T > t) = 1 - F(t) \tag{1}$$

Where $F(t)$ is the cumulative distribution for a time until the event variable of T. The probability of response event T happening after time t is the $S(t)$, which is always one at $t = 0$. The survival function must be non-increasing as t increases, as shown in Figure 2.

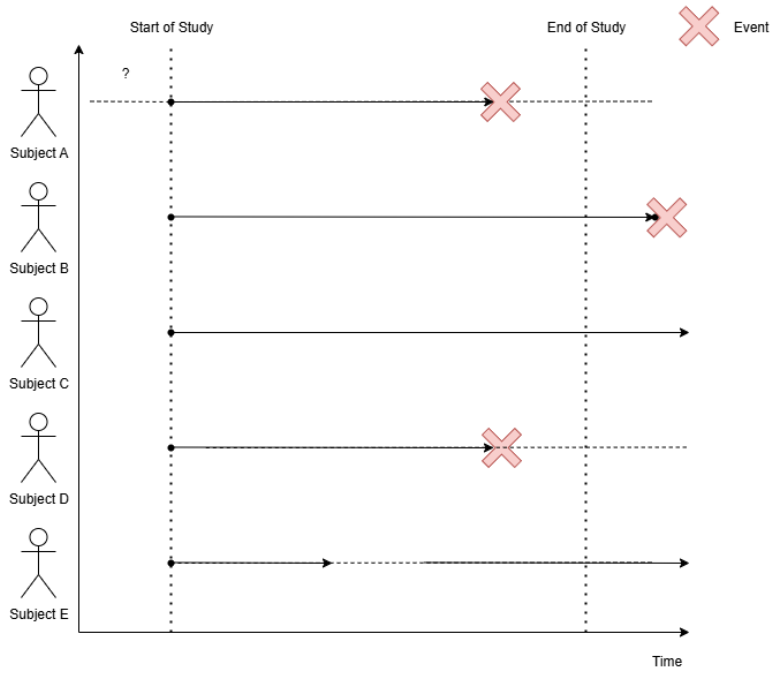

Figure 1: Examples of Censoring. We can see that Subject A is left-censored, while Subject B and C are right-censored. Subject D is the only observed case. Subject E is interval censored.

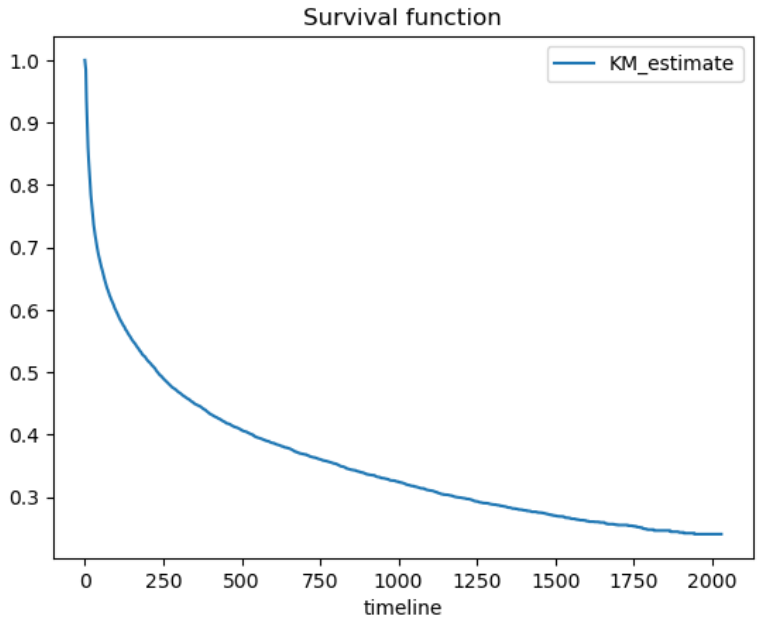

Figure 2: Survival Function $S(t)$ Figures.

Another function that is commonly used in *hazard function* $\lambda(t)$ or *instantaneous event rate*, which is defined by Equation 2.

$$\lambda(t) = \lim_{u \to 0} \frac{\text{Prob}(t < T \leq t + u | T > t)}{u} = \frac{f(t)}{S(t)} \tag{2}$$

Function $f(t)$ is the probability density function (pdf) of the time to event T evaluated at t. The

likelihood that the event will occur within a short window of time t, assuming that it has not yet happened, determines the hazard at time t. The integral under the area of hazard function $\lambda(t)$ is denoted by $\wedge(t)$, which is defined by the negative logarithmic product of survival function $S(t)$. It can be shown in Equation 3. The hazard function graph is illustrated in Figure 3.

$$\wedge(t) = -\log(S(t)) \tag{3}$$

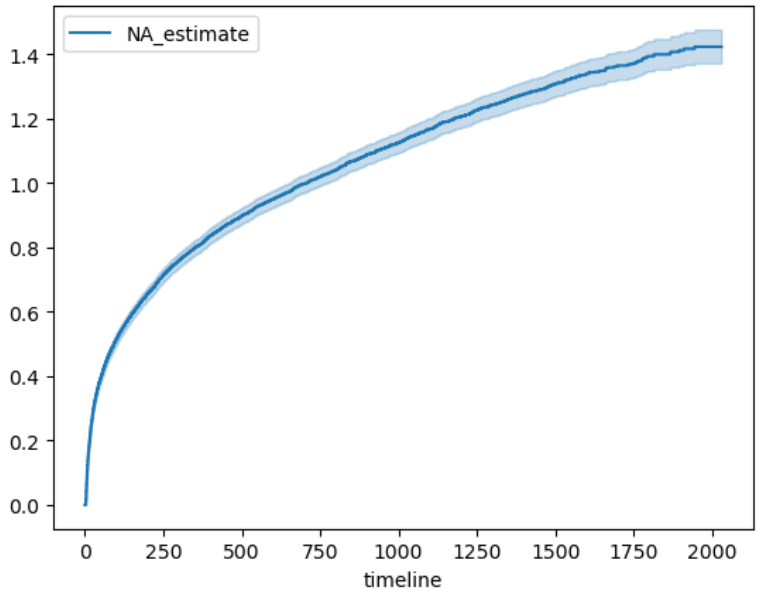

Figure 3: Hazard Function $\lambda(t)$ Figures.

2.1.1 Traditional Methods

Traditional methods such as Kaplan-Meier [8] and Cox-proportional hazard [9] are commonly used in Survival Analysis. Kaplan-Meier estimator [8] is the most common estimator to estimate the survival function $S_{KM}(t)$ [7]. It is non-parametric and utilises a product-limit estimator and is based on conditional probabilities. Kaplan-Meier is described in Equation 4 with d_i being the number of failures at time t_i and n_i being the number of subjects at risk at time t_i.

$$S_{KM}(t) = \prod_{i;t_i \leq t} (1 - \frac{d_i}{n_i}) \tag{4}$$

The Kaplan-Meier Estimator is useful to estimate survival time distribution without making any assumptions; therefore, it is perfect to describe simple survival. Provide graphical illustrations and compare study populations. However, Kaplan-Meier is only descriptive, does not control covariates and does not handle any time-dependent variables [21] [22].

To handle predictor variables of X_i, we can use a parametric survival model. The most commonly used is Cox Regression [9], which is another method to estimate semiparametric survival time. Cox-Regression estimates a hazard function $\lambda(t|x)$ at a given time t for a given feature vector $\mathbf{x}$ as shown in Equation 5.

$$\lambda(t|x) = \lambda(t)\exp(X\beta) \tag{5}$$

β in Equation 5 is the parameter of interest, which is approximated by a partial likelihood. Cox Regression outputs the hazard rate instead of the actual time-to-event. Cox PH also assumes proportional hazards, that the hazard ratio of two groups is always constant, as also shown in Equation 5. Cox Proportional Hazard is known for its flexibility, simple interpretation and its ability to analyse multiple risk factors [23]. However, there are multiple disadvantages of Cox PH, such as its assumption of proportional hazards, which can lead to biases and confusion as it assumes the same hazard ratio for a high-risk group and a low-risk group. It also has limited explanations compared to other methods [24].

2.1.2 Accelerated Failure Time

Accelerated failure time is a method to predict time-to-event directly [7]. It is a semiparametric model that, rather than assuming proportionality of hazards, assumes that the predictors act multiplicatively on a failure time or additively on the log failure time. The survival function $S(t|X)$ at a given time t for a given feature vector $\mathbf{x}$ can be written in Equation 6.

$$S(t|X) = \psi((log(t) - X\beta)/\sigma) \tag{6}$$

In Equation 6, ψ is any standardised survival distribution function and σ is the scale parameter. The model can also be stated to output logarithmic time-to-event as in Equation 7, with ε being the error value that comes from the distribution function.

$$\log(T) = X\beta + \sigma\varepsilon \tag{7}$$

The distribution function here, which outputs a logarithmic form of time $log(T)$, can be any log-distribution. The median survival time can be shown in Equation 8.

$$T_{0.5}|X = \exp(X\beta + \sigma\psi^{-1}(0.5)) \tag{8}$$

Common choices of distribution that is used in AFT are usually extreme, which assumes $\psi(u) = \exp(-\exp(u))$, logistic, which assumes $\psi(u) = [1+\exp(u)]^{-1}$ and normal distribution which assumes $\psi(u) = 1 - \Phi(u)$. The most common AFT model used is the Weibull model, which is often another form of extreme model by negating β and transforming γ (Euler–Mascheroni constant, $\gamma \approx 0.57721...$) into a scale parameter of σ. The survival function model that models Weibull can be shown in Equation 9.

$$S(t|X) = \exp[-\exp((log(t) - X\beta)/\sigma)] \tag{9}$$

The survival functions of log-normal and log-logistic survival times are shown in Equation 10 and Equation 11, respectively.

$$S(t|X) = 1 - \Phi(log(t) - X\beta)/\sigma) \tag{10}$$

$$S(t|X) = [1 + \exp((log(t) - X\beta)/\sigma)]^{-1} \tag{11}$$

Accelerated Failure Time here assumes a certain distribution, which can be less practical for the researcher to describe first the survival and therefore less flexible than Cox Proportional Hazard. The model assumes a certain distribution, which can make it prone to overfitting [25] if the model is not regularised properly. The model has advantages over a Cox Proportional Hazard model, which are more interpretable, can be used for non-proportional hazards data and most importantly, can output time-to-event value directly rather than converting it first from the hazard function.

2.2 Decision Tree and Ensemble Methods

Decision trees are one of the most used machine learning techniques for classification and regression. The most popular algorithm is CART (Classification and Regression Tree), introduced by Breiman [26]. The basic idea is to partition the tree to form groups based on similarities according to the outcome of interest. This is usually done by minimising a measure of node impurity. The CART Algorithm can be used for a regression problem using the Sum of Squared Errors and the Mean Squared Error for the loss function. The function of the Sum of Squared Error and the Mean Squared Error can be expressed in Equation 12 and Equation 13, respectively.

$$\mathrm{SSE}(y,\hat{y}) = \sum_{u=0}^{N} (y_u(n) - \hat{y}_u(n))^2 \tag{12}$$

$$\mathrm{MSE}(y,\hat{y}) = \frac{1}{N}\sum_{u=0}^{N} (y_u(n) - \hat{y}_u(n))^2 \tag{13}$$

The CART Algorithm for Regression can be expressed in Algorithm 1. The algorithm here utilises the exact-greedy based splitting to search for the split that minimises the loss function. There are some advantages of using a CART Algorithm, that it is interpretable and explainable, requires little data preprocessing and ability to handle numerical and categorical data [27]. The simple understanding of a decision tree also mimics how human makes decisions, and therefore it is a choice for an explainable model in Machine Learning. However, a decision tree is prone to overfitting as it yields high bias and low variance output [28]. Thus, it can produce a result that does not generalise well with the dataset.

2.2.1 Survival Tree

Decision trees could be used in a survival analysis ([14]). It is used to extend the decision tree so that it can be used for a case of survival analysis, which is a regression problem with the presence of censoring [13]. There are multiple directions of research on survival trees. Gordon and Olshen proposed a splitting rule based on Kaplan-Meier Estimates ([29]). The tree proposed used a Wasserstein metric between the survival function obtained from the Kaplan–Meier estimator. However, the more popular algorithm is developed from the prior algorithm that uses the log-rank statistic to compare the two groups formed by the children nodes by Ciampi et al. [30]. The log-rank test is the most commonly as splitting criterion and can be illustrated in Equation 14 with $O_{i,j}$ as the observed number of events in the groups at time j, $E_{i,j}$ as the expected value, and $V_{i,j}$ as the variance. The log-rank statistic employs a null hypothesis that two groups have identical hazard functions.

$$Z_i = \frac{\sum_{j=1}^{J}(O_{i,j} - E_{i,j})}{\sqrt{\sum_{j=1}^{J} V_{i,j}}} \tag{14}$$

The log-rank statistic is preferred as a splitting criterion because of its simplicity in splitting data with censoring into two nodes based on the Kaplan-Meier survival function. Log-rank statistic has drawbacks such as instability towards some variables, and tends to have false nodes due to bias of hazard function similar to the drawbacks of Cox Proportional Hazard [31].

There are also a few splitting criteria, such as a negative log-likelihood score ([32]), which uses a splitting criterion based on an exponential model log-likelihood and a Martingale residuals ([33])

Algorithm 1 The CART Algorithm for Regression

Require: Training data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, max depth $d_{\max}$, min samples split $n_{\min}$
Ensure: Regression tree T

function BUILDTREE(D, depth)
  if depth $\geq d_{\max}$ **or** $|D| < n_{\min}$ **then**
    $v \leftarrow \text{MEAN}(\{y_i \mid (\mathbf{x}_i, y_i) \in D\})$
    return Leaf node with value v
  end if
  $(f^*, s^*, L, R) \leftarrow$ FINDBESTSPLIT(D)
  if f^* is **None then**
    $v \leftarrow \text{MEAN}(\{y_i \mid (\mathbf{x}_i, y_i) \in D\})$
    return Leaf node with value v
  end if
  $T_{\text{left}} \leftarrow$ BUILDTREE(L, depth + 1)
  $T_{\text{right}} \leftarrow$ BUILDTREE(R, depth + 1)
  return Decision node with feature f^*, split s^*, left child T_{left}, right child T_{right}
end function
function FINDBESTSPLIT(D)
  best_loss $\leftarrow \infty$
  best_split $\leftarrow$ **None**
  for all features f in D **do**
    for all split values s of f **do**
      Partition D into L and R using f and s
      if L or R is empty **then**
        continue
      end if
      loss $\leftarrow$ MSE/SSE(L, R)
      if *loss* $<$ *best_loss* **then**
        best_loss $\leftarrow$ *loss*
        best_split $\leftarrow (f, s, L, R)$
      end if
    end for
  end for
  return *best_split*
end function

which utilises a splitting criterion from a null Cox model as the outcome for a regression tree algorithm. Nonetheless, there are still some drawbacks to using these algorithms rather than the log-rank statistic, such as cost complexity and interpretability.

Survival Tree splitting must be stopped to avoid overfitting. There are multiple algorithms to stop that, such as RECPAM (Recursive Partition and Amalgamation) from [30] that introduces an amalgamation step to prune back the trees, and Hierarchical group profiles from clustering from [34]. The former merges terminal nodes with similar risk profiles, and the latter uses hierarchical clustering to merge similar survival functions. However, pre-pruning is usually preferred for complexity such as setting maximum tree depth limit, minimum node size or post-pruning, such as other statistical significance tests like log-rank.

There is some research on modelling time-to-event directly using survival trees. For example, [35] proposes classification tree analysis (CTA) survival models using log-rank statistics to predict time-to-event value. However, the result is that classification trees only output binary results rather than time-to-event, and the time-to-event is generated using the survival function of the risk provided. The output of the result is also worse than the Cox regression and naive model. The paper only compared one factor, such as age and systolic blood pressure, to predict cardiovascular disease.

2.2.2 Bagging and Random Forest

To overcome the bias of a single decision tree, usually a technique called bootstrap aggregating or bagging is employed [15]. Bagging is a popular ensemble approach for training individual learners on randomly selected portions of the training dataset. Bagging can be illustrated in Figure 4. Aggregating many learners reduces model variation while maintaining bias [36], given the bias-variance decomposition of error in machine learning models. Therefore, bagging can be a robust way to aggregate many biased decision trees. A random forest algorithm is a training algorithm that employs a bootstrap aggregating method [15]. Random forest can be described in Algorithm 2, which selects a random sample from the data and trains each weak learner or individual learner that is a decision tree. Random forest can also be used by feature bagging or selecting a subset of random features from a group of features. The majority voting for the random forest algorithm for classification uses a majority voting, whereas for regression, average or median is used to determine the final output.

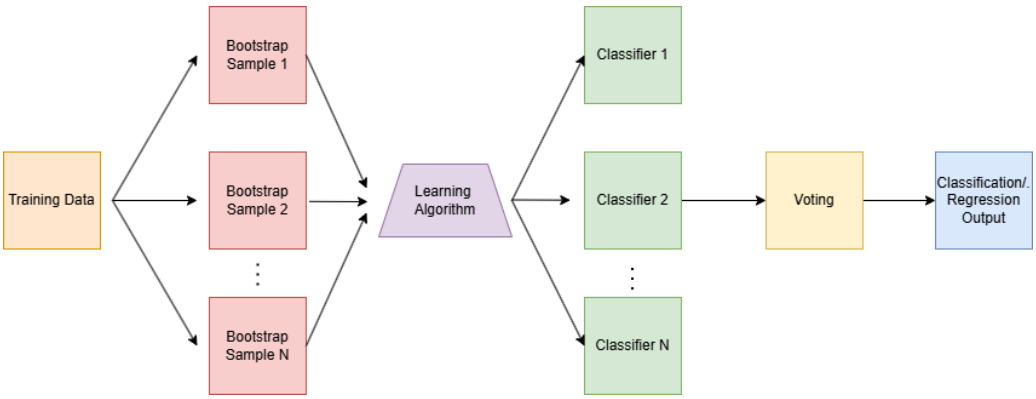

Figure 4: Bagging Illustration

The random forest algorithm is very versatile and to be used for any kind of classification and data, and has a generally high performance as a large number of trees used usually converges and increases performance [15]. It is also scalable as weak learners can be fitted in parallel, therefore reducing the time for the fitting process. The drawbacks of the random forest algorithm are the

Algorithm 2 The Random Forest Algorithm for Regression

Require: Training data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, Number of Trees n_{trees}, Number of features to sample m_{try}
Ensure: A fitted Random Forest model

function RANDOMFORESTFIT(D, n_{trees}, m_{try})
    Initialize an empty list of trees $\mathcal{T}$
    for $t = 1$ to n_{trees} **do**
        Draw a bootstrap sample $D^{(t)}$ of size N from D
        T_t = BUILDTREE($D^{(t)}$)
        Add T_t to $\mathcal{T}$
    end for
    return $\mathcal{T}$
end function
function RANDOMFORESTPREDICT($\mathcal{T}$, $\mathbf{x}$)
    Initialize predictions list $P \leftarrow []$
    for all T_t in $\mathcal{T}$ **do**
        Predict $\hat{y}_t = T_t(\mathbf{x})$
        Add $\hat{y}_t$ to P
    end for
    y' = MEAN(P)
    return y'
end function

complexity of computation and the reduced interpretability of the prediction because of the need for aggregation, as the number of trees is created from a sample of the data.

2.2.3 Boosting and XGBoost

Another form of ensemble learning that can be used in boosting. In contrast to bootstrap aggregating, which grows weak learners from a subset of data from sampling, boosting works by sequentially improving the performance of weak learners based on their performance by increasing the weight of the samples that are incorrectly predicted. The most popular algorithm is the AdaBoost [16] and Gradient Boosting [36][37]. The Adaboost works by reweighting of an incorrect prediction, whereas Gradient Boosting improves it by utilising a minimising loss function [37]. XGBoost (Extreme Gradient Boosting) is the most widely used algorithm that improves a simple gradient boosting [38] by utilising cache access patterns, data compression and sharding, hence improving its scalability. The boosting has advantages that it has a high performance and adaptability to different loss functions. However, it can also be longer to train than bagging for sequential learners. Boosting also lacks the interpretability that a decision tree can provide because the use of loss functions makes a weak learner uninterpretable.

2.2.4 Ensemble Survival Model

Nevertheless, survival trees still present drawbacks such as overfitting and instability from analysing censored data. Ensemble methods of the survival tree that predict survival time have also been explored using XGBoost [17] and Random Survival Forest [18].

XGBoostAFT [17] utilises a loss function for learning accelerated failure time (AFT) models in XGBoost. It can handle not only right-censored data but also left-censored and interval-censored

data. The loss function is derived from an AFT survival function in Equation 7. The loss function handles censoring by utilising the probability distribution function (PDF) and the cumulative distribution function (CDF) that is derived from the choice of distribution assumption, which is log-normal, log-extreme and log-logistic as seen in Equation 10, Equation 9, Equation 11, respectively. XGBoostAFT also predict time-to-event directly, which makes it a versatile tool for a researcher to use. However, there are some drawbacks to using XGBoostAFT. The first one is that XGBoostAFT currently only supports parametric distribution and also does not learn from time distribution directly, which requires fine-tuning the scale parameter σ and the distribution choice. One false assumption can make the prediction wrong. The second one is that it has the drawbacks of XGBoost, that it is not interpretable and explainable directly like a survival tree, as the log-rank can provide.

Random Survival Forest is another random survival forest algorithm that utilises weak learners of decision trees that use a log-rank statistic for splitting [18]. The random survival forest algorithm works the same as the ordinary random forest algorithm described in Algorithm 2 but uses an ensemble cumulative hazard function (CHF) for aggregation. The cumulative hazard function $\hat{H}_h(t)$ is computed from the Nelson–Aalen estimator described in Equation 15 with $d_{i,h}$ and $Y_{i,h}$ as the number of events and the number of individuals happening at time $t_{i,h}$, respectively.

$$\hat{H}_h(t) = \sum_{t_{i,h} \leq t} \frac{d_{i,h}}{Y_{i,h}} \tag{15}$$

Random Survival Forest is proven to be robust to outliers and well-calibrated, as proven by Lu and Ilgin Guler [39]. The algorithm also does not assume any hazard ratio like proportional hazard does, which makes it more flexible. There are some drawbacks to using random survival forest, such as it is computationally expensive to train, less interpretable than a single tree and lacks time-to-event output, which makes it dependent on the Cumulative Hazard Function to format into risks. Moreover, random survival forest has difficulty processing time-varying risk factors, for example, the heart failure exacerbations, which depend on time factors [40].

3 Materials and Methods

The methodology required for experiments is outlined in the following sections. Subsection 3.1 explains the dataset and pre-processing used for the experiment. Subsection 3.2 describes the metric used to measure the performance of the survival model. Subsection 3.3 explains the main model that is introduced in this research, which is the AFT Survival Tree and the Random Forest algorithm.

3.1 Dataset

3.1.1 Support

SUPPORT (Study to Understand Prognoses and Preferences for Outcomes and Risks of Treatments) Dataset [41] is a dataset that consists of 9105 critically ill patients across 5 United States medical centres from 1989-1991 and 1992-1994. In this dataset, 31.89% of the subjects are censored, and the mean survival time for uncensored patients is 206 days. There are 42 features in this dataset. Multiple pre-processings have been done, starting with one-hot encoding of the categorical response. And then uniform imputation is applied for the empty data, which is based on the recommendation of researchers as stated by UC Irvine [42]. The list of imputation values can be seen in Table 32. Finally, there are 14 unimportant features which are dropped. These 14 features are shown in Table 33. The label consists of survival time on a day scale, which is not scaled.

3.1.2 NHANES

National Health and Nutrition Examination Survey (NHANES) is a dataset that consists of single disease databases encompassing 41,474 individuals and 1,191 variables [43]. Used in this study is the dataset of the first National Health and Nutrition Examination Survey (NHANES I) was conducted by the National Centre for Health Statistics between 1971 and 1974. There are 79 features in this dataset, and 14264 patients, 9553 of whom are censored dataset. One-hot encoding is applied to a categorical dataset. No features are dropped from this dataset. The imputation here uses a median imputation from each of the empty data points in a column. The label consists of survival time on a year scale, which is not scaled.

3.2 Metric

3.2.1 C-Index

Time-dependent C-Index (Concordance Index) [44] is a metric that measures the goodness of fit of a survival analysis model by measuring the probability that predicted times $(\hat{t_i}, \hat{t_j})$ of two randomly selected subjects maintain the same order as their true event time (t_i, t_j) [45]. The probability can be defined by $P(\hat{t_i} > \hat{t_j}) | t_i > t_j)$. The C-Index only accounts for pairs (t_i, t_j) that are both uncensored and pairs where one is uncensored and the other is censored at the latter times. The other pairs are considered invalid and are not taken into account. The pairs are called concordant if the order is correct, such that $t_i < t_j$ and $\hat{t_i} < \hat{t_j}$ and discordant if the order is incorrect, such that $t_i < t_j$ and $\hat{t_i} > \hat{t_j}$. If the order of the C-Index formula can be stated in Equation 16.

$$\text{C-Index} = \frac{\text{Number of Concordant Pairs}}{\text{Number of Concordant Pairs} + \text{Number of Discordant Pairs}} \tag{16}$$

With predicted times $(\hat{t_i}, \hat{t_j})$ and true times taken into account (t_i, t_j) and δ_j is the variable that determines if variable j is censored or not, the formula can be rewritten in Equation 17.

$$C = \frac{\sum_{i \neq j} 1_{\hat{t_i} > \hat{t_j}} 1_{t_i > t_j} \delta_j}{\sum_{i \neq j} 1_{t_i > t_j} \delta_j} \tag{17}$$

The C-Index score ranges from 0 to 1, where 1 indicates a perfect order of predicted times compared to the true times and 0 indicates a complete mismatch of predicted times compared to the true times. 0.5 indicates a random order of prediction compared to the true times.

3.2.2 Brier Score and Integrated Brier Score

A Brier score (BS) is a metric that measures the accuracy of a set of probabilistic predictions [46]. A Brier score without right censoring, given a dataset of N samples with data defined as $\vec{x_i}$, censoring defined as δ_i, time-to-event defined as T_i and predicted survival function defined as $\hat{S}(t, \vec{x_i})$ can be defined in Equation 18.

$$BS(t) = \frac{1}{N} \sum_{i=1}^{N} (1_{T_i > t} - \hat{S}(t, \vec{x_i}))^2 \tag{18}$$

With right censoring, however, the equation is adjusted using weighted squared distance by the inverse probability of censoring weights method. Let $\hat{G}(t) = P[C > t]$ be the estimator of the conditional survival function of the censoring times using Kaplan-Meier with C as censoring times, the Brier score becomes Equation 19.

$$BS(t) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{(0 - \hat{S}(t, \vec{x_i}))^2 \cdot 1_{T_i < t, \delta_i = 1}}{\hat{G}(T_i^-)} + \frac{(1 - \hat{S}(t, \vec{x_i}))^2 \cdot 1_{T_i > t}}{\hat{G}(t)} \right) \tag{19}$$

The integrated Brier score is a score that provides the overall calculation of the model's performance at all available times [47]. It is defined as in Equation 20.

$$IBS(t_{max}) = \frac{\int_o^{t_{max}} BS(t) dt}{t_{max}} \tag{20}$$

The integrated Brier score (IBS) is important to measure the calibration of the prediction given survival trees. The integrated Brier score can be seen as an extension of Mean Squared Error for survival analysis.

3.2.3 Mean Absolute Error

The mean absolute error (MAE) is a metric that measures the difference between the predicted value and the true time-to-event value using the Manhattan distance. In other words, MAE can be calculated using the sum of absolute errors. Given a predicted time-to-event $\hat{t_i}$ and true time-to-event t_i from n samples, the MAE can be defined as in Equation 21.

$$MAE = \frac{\sum_{i=1}^{n} |\hat{t_i} - t_i|}{n} \tag{21}$$

The MAE score only calculates the uncensored events. The main benefits of using the MAE scores are to identify discrepancies between the predicted survival times of the model across predictions, regardless of censorship.

3.3 Models

3.3.1 AFT Survival Tree and AFT Random Forest

AFT-Random Forest is an algorithm to predict absolute time-to-event directly using Random Forest with Accelerated Failure Time covariates. The model is derived from XGBoostAFT [17] and the Random Forest Regression Tree based on CART (Classification and Regression Tree) [26]. The AFT-Random Forest uses regression derived from the covariates of the Accelerated Failure Time - Random Forest. The AFT can be derived from Equation 7 and be rewritten the same as XGBoostAFT [17] with Y as the survival time, w as the coefficients, x as the inputs and Z and σ as the random variable of the unknown probability distribution in Equation 22.

$$\ln Y = \langle \mathbf{w}, \mathbf{x} \rangle + \sigma Z \tag{22}$$

It is then used in the same manner as XGBoost-AFT [17] by transforming $\langle \mathbf{w}, \mathbf{x} \rangle$ into an output prediction from the decision trees, which in this case are from a tree of random forest $\mathcal{T}(\mathbf{x})$. The equation becomes Equation 23.

$$\ln Y = \mathcal{T}(\mathbf{x}) + \sigma Z \tag{23}$$

To use it with a random forest, a loss function ℓ_{AFT} using Equation 23 needs to be set. As XGBoost-AFT [17] derives a loss function. The loss function is derived from the likelihood of the training data $\mathcal{D} = \{(\mathbf{x}, y_i)\}_{i=1}^{n}$ and random variables of the distribution Y. It is denoted that f_y is the probability distribution function (PDF) of Y and F_y is the cumulative distribution function (CDF) of Y. The likelihood of $\mathcal{D}$ is the product of the PDF f_y for individual data points, as shown in Equation 24.

$$\ell(\mathcal{D}) = \prod_{i=1}^{n} f_y(y_i) \tag{24}$$

The log likelihood can then be maximised as in Equation 25.

$$\ln \ell(\mathcal{D}) = \sum_{i=1}^{n} \ln f_y(y_i) \tag{25}$$

The loss function is then revised since the censored event is also taken into account.

$$\ln \ell(\mathcal{D}) = \sum_{i=1}^{n} \ln f_y(y_i) + \sum_{i=1}^{n} \ln \left(F_y(\bar{y}) - F_y(\underline{y})\right) \tag{26}$$

Where in the case of right censoring, which this research is focused on, $\bar{y}$ is the last defined time-to-event known and $\underline{y}$ is set to ∞. In other cases, left censoring means that $\bar{y}$ is set to 0 and $\underline{y}$ is set to the last defined time-to-event known and interval censoring sets the $\bar{y}$ and $\underline{y}$ to a time interval.

The AFT survival Regression loss function can be defined as in Equation 27.

$$\ell_{AFT}(y, \mathcal{T}(\mathbf{x})) = \begin{cases} -\ln f_y(Y) & \text{if y is uncensored} \\ -\ln \left(F_y(\bar{y}) - F_y(\underline{y})\right) & \text{if y is censored} \end{cases} \tag{27}$$

The sum of AFT loss as the negative loss function over the training data can be expressed as in Equation 28.

$$\sum_{i=1}^{n} \ell_{AFT}(y, \mathcal{T}(\mathbf{x})) = -\ln \ell(\mathcal{D}) \tag{28}$$

It can be defined that the loss function in terms of the known PDF and CDF of f_z and F_z, respectively, by assuming the distribution of Z and the relation between Y and Z as a monotone increasing function as in Equation 29 with a link function of $s(y.\mathcal{T}(\mathbf{x}))$ as $\frac{(\ln y - \mathcal{T}(\mathbf{x}))}{\sigma}$ to manage the ensemble trees output of log survival time $\ln y$

$$\ell_{AFT}(y,\mathcal{T}(\mathbf{x})) = \begin{cases} -\ln\left(f_z(s(Y))\cdot\frac{1}{\sigma y}\right) & \text{if y is uncensored} \\ -\ln\left(F_z(s(\bar{y})) - F_z(s(\underline{y}))\right) & \text{if y is censored} \end{cases} \tag{29}$$

From the loss function in Equation 29, a simple AFT-Tree can be created that minimises the loss function to split the nodes. The loss function splits the time in log-time space. Hence, $\mathcal{T}(\mathbf{x})$ is calculated as $\mathcal{T}(\mathbf{x}) = \text{mean}(\log y)$. The CART Algorithm shown earlier in Algorithm 1 can be expanded to handle censoring in Algorithm 4.

It is observed in Algorithm 4 that the basic AFT-Tree aggregates the final value of a leaf using a geometric mean of only uncensored time-to-event, which are defined in Equation 30. The design choice of using a geometric mean comes from the output inspired by XGBoost AFT [17] preserves the distribution of the time in the data and also ensures more variance than the mean for a single tree. The tree can also calculate the value of a leaf using the arithmetic average, or create a median survival time from the node using a heterogeneous AFT distribution from that node.

$$\text{Geometric Mean} = \begin{cases} \exp\left(\frac{1}{|E|}\sum_{t\in E}\ln(t_{uncensored})\right), & \text{if not extreme} \\ \exp\left(\frac{1}{|E|}\sum_{t\in E}\ln(t_{uncensored})\right) + \sigma\ln(\ln 2), & \text{if extreme} \end{cases} \tag{30}$$

For the basic AFT-Tree, it is assumed that the probability distribution of z is as in Table 3, similar to XGBoost-AFT [17] without doing the gradient boosting. The σ in this ensemble method is set as a hyperparameter as shown in Algorithm 4.

Table 2: Probability distributions of z[1]

Distribution	**PDF**	**CDF**
Normal	$\frac{\exp\left(-\frac{z^2}{2}\right)}{\sqrt{2\pi}}$	$\frac{1}{2}\left(1+\text{erf}\left(\frac{z}{\sqrt{2}}\right)\right)$
Logistic	$\frac{e^z}{(1+e^z)^2}$	$\frac{e^z}{1+e^z}$
Extreme (Gumbel)	$e^z e^{-\exp(z)}$	$1-e^{-\exp(z)}$

From these AFT trees, an AFT Random Forest can be built. The random forest algorithm is derived from Classification and Regression Tree [26] and follows Algorithm 5. The ensemble methods work by bagging the results of the time-to-event output by using the mean of the outputs of n trees. This is chosen because it preserves the distribution of prediction of the model, so that it does not output a value that overestimates the time of study. The random forest works by random sampling the training data into small subsets to be predicted by each tree using the loss function ℓ_{AFT}, and the data for each tree can be used to approximate the chosen PDF and CDF using distribution fitting and bootstrapping. Feature subsampling is also implemented to take a small subset of features from the training data rather than all of the data.

[1]These correspond to the log-Normal, log-Logistic, and log-Extreme Value (Gumbel) distributions in the AFT model, since z is defined in log-time.

3.3.2 Custom Distribution Fitting

To improve the AFT survival tree results using the PDF and CDF from Table 3, the AFT survival tree can utilise a distribution fitting process where a small percentage of the training data is split to generate a hyperparameter for the distribution. This means σ is not needed as a hyperparameter anymore, and σ and Z are tied to the generated probability distribution function f_z and cumulative distribution function F_z from the distribution fitting process. The new distributions, aside from Table 3, added are Gaussian mixture models. The splitting of the data is done by stratifying censored and uncensored cases and binning across quartiles to ensure similar distribution and number of censored and uncensored cases.

The custom fitting process for the logarithmic distributions of normal, logistic, and extreme uses the Newton-Raphson method [48]. The Newton-Raphson method employs iteration to find the maximum likelihood estimates (MLE) of the parameters of the distribution. The Newton-Raphson method can be defined in Equation 31, with the example to approximate x_{n+1} using the function f and the derivative of the function f, which is f'. The iteration is usually stopped until convergence.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{31}$$

3.3.3 Gaussian Mixture Model Fitting

To improve the fitting of an AFT-Tree and AFT-Random Forest using non-parametric estimation, a Gaussian Mixture Model (GMM) can be used as a custom distribution. The aim is to check whether the Gaussian Mixture Model (GMM) can improve performance compared to other distributions. The Gaussian Mixture Model (GMM) can be expressed as in Table 3.

Table 3: Table of additional probability distribution of z using distribution fitting

Distribution	**PDF**	**CDF**
GMM	$p(x) = \sum_{k=1}^{K} \pi_k \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x-\mu_k)^2}{2\sigma_k^2}\right)$	$P(x) = \sum_{k=1}^{K} \pi_k \left(\frac{1}{2}\left[1 + \text{erf}\left(\frac{x-\mu_k}{\sigma_k\sqrt{2}}\right)\right]\right)$

A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities or normal distributions [49]. It is commonly used as a model of the probability distribution of continuous measurements. Nonetheless, this distribution can mimic crisply the shape of the distribution and therefore can be seen as non-parametric even though it is parametric by assuming the distribution is a collection of Gaussians. The illustrations of Gaussian Mixture Models can be seen in Figure 5. A Gaussian mixture model can be defined as in Equation 32

$$p(x) = \sum_{i=1}^{M} w_i g(x|\mu_i, \Sigma_i) \tag{32}$$

In Equation 32, x is the D-dimensional continous-valued data vector, $w_i, i = 1,..,M$ are the mixture weights and $g(x|\mu_i, \Sigma_i)$ are the component densities. In this research, the Gaussian mixture models have D equal to 1 to model the survival time distribution. With mean vector u_k and covariance σ_k, the component densities can be written in Equation 33.

$$g(x|\mu_k, \Sigma_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(x-\mu_k)^2}{2\sigma_k^2}\right) \tag{33}$$

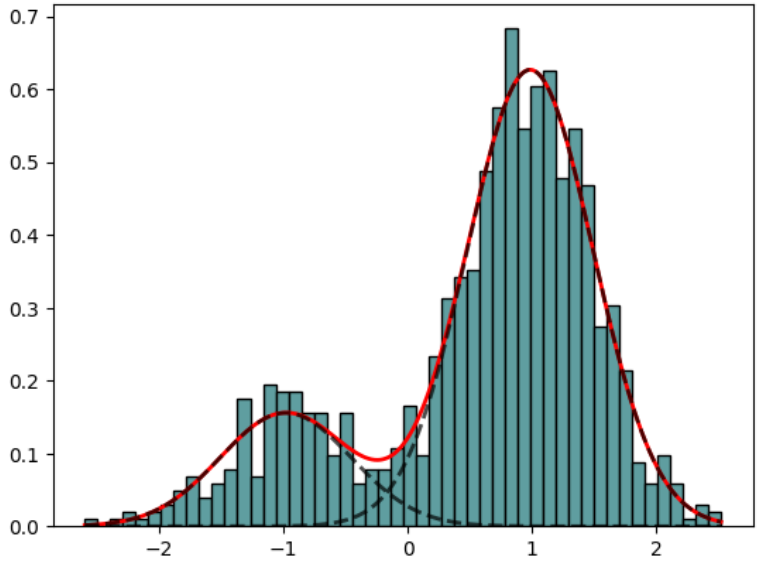

Figure 5: Gaussian Mixture Models on Univariate Distribution Illustration.

The Gaussian mixture models are usually parameterised by the mean vectors u_k, covariance matrices σ_k and mixture weights w_k from all component densities. To know these parameters from the training data, estimation has to be done using several techniques [50]. The most commonly used is the maximum-likelihood (ML) estimation with the Expectation-maximisation (EM) algorithm to get the highest maximum likelihood of Gaussian mixture models parameters of mean vectors u_k, covariance matrices σ_k and mixture weights w_k from each iteration until convergence. The EM algorithm consists of two parts, which are expectation and maximisation. The expectation part consists of computing the posterior probability of each datapoint x as shown in Equation 34.

$$\Pr_{ik} = \frac{w_k\, g(\mathbf{x}_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} w_j\, g(\mathbf{x}_i|\mu_j, \Sigma_j)} \tag{34}$$

The maximisation part consists of updating each mean vector u_k, covariance matrices σ_k and mixture weights w_k from the posterior $\Pr_{ik}$ as shown in Equation 35, Equation 36, Equation 37, respectively.

$$w_k^{\text{new}} = \frac{N_k}{N}, \quad \text{where} \quad N_k = \sum_{i=1}^{N} \Pr_{ik} \tag{35}$$

$$\mu_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^{N} \Pr_{ik} \mathbf{x}_i \tag{36}$$

$$\Sigma_k^{\text{new}} = \frac{1}{N_k} \sum_{i=1}^{N} \Pr_{ik} (\mathbf{x}_i - \mu_k^{\text{new}})(\mathbf{x}_i - \mu_k^{\text{new}})^T \tag{37}$$

The Expectation and Maximisation algorithm will be repeated until convergence is reached or log-likelihood changes by less than a threshold.

The Gaussian Mixture Models are not commonly explored in survival analysis. Accelerated failure time modelling via nonparametric mixtures has been explored by Seo and Kang [19], which explores non-parametric estimation of accelerated failure time using a Gaussian Mixture Model that

accounts for censoring. The result is a model that is more robust to misspecification problems, unlike other parametric AFT models, and does not require a separate procedure, unlike existing non-likelihood-based semi-parametric AFT models. The study opens up a field where assuming survival time distribution for models can be easier for researchers using AFT.

3.3.4 Bootstrap Sampling Fitting

To combat overfitting, rather than a standard train-test split for distribution fitting, bootstrap sampling can also be preferred to predict the parameters of the probability distribution function f_z and the cumulative distribution function F_z. The standard mean bootstrap sampling can be used to approximate the hyperparameters of the chosen probability and the cumulative distribution.

Bootstrap sampling or bootstrapping is a method for estimating the distribution of an estimator or test statistic by resampling one's data or a model estimated from the data [51]. Bootstrapping uses accuracy to estimate. The bootstrapping algorithm can be illustrated in Figure 6.

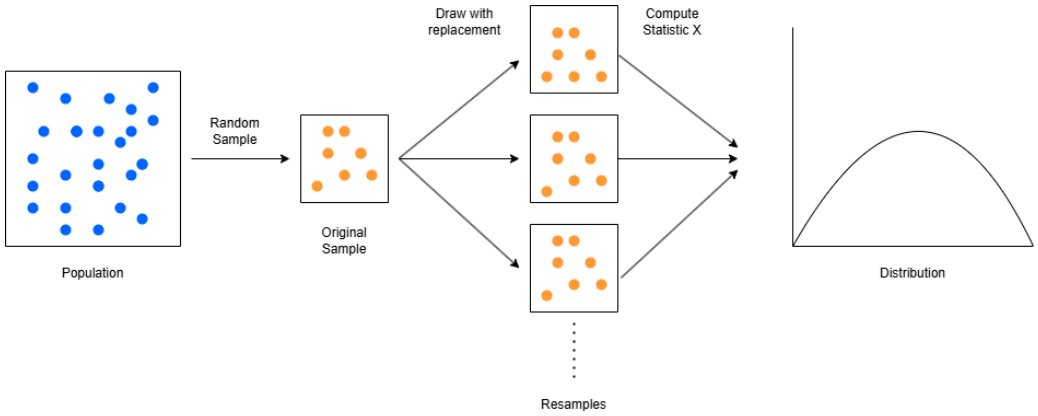

Figure 6: Bootstrapping of a distribution illustration.

Bootstrapping has many advantages, such as simplicity in modelling complex distributions, control and checking the stability of the results and also avoiding overfitting in a distribution [52]. The disadvantage, however that naive use of bootstrapping can lead to inconsistency [53] and also high time-complexity to resample the data and calculate the desired metric.

Algorithm 3 AFT-Tree

Require:
Training data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
max depth $d_{\max}$
min samples split $n_{\min}$
Scale Parameter σ
Ensure: Survival Tree T
function BUILDAFTTREE(D, depth, σ)
  if depth $\geq d_{\max}$ **or** $|D| < n_{\min}$ **then**
    $v \leftarrow$ SETVALUE($\{y_i \mid (\mathbf{x}_i, y_i) \in D\}$)
    return Leaf node with value v
  end if
  $(f^*, s^*, L, R) \leftarrow$ FINDBESTSPLIT(D)
  if f^* is **None then**
    $v \leftarrow$ SETVALUE($\{y_i \mid (\mathbf{x}_i, y_i) \in D\}$)
    return Leaf node with value v
  end if
  if all y_L is **censored or all** y_R is **censored then**
    $v \leftarrow$ SETVALUE($\{y_i \mid (\mathbf{x}_i, y_i) \in D\}$)
    return Leaf node with value v
  end if
  $T_{\text{left}} \leftarrow$ BUILDTREE(L, depth + 1)
  $T_{\text{right}} \leftarrow$ BUILDTREE(R, depth + 1)
  return Decision node with feature f^*, split s^*, left child T_{left}, right child T_{right}
end function
function FINDBESTSPLIT(D, σ)
  $best\_loss \leftarrow \infty$
  $best\_split \leftarrow$ **None**
  for all features f in D **do**
    for all split values s of f **do**
      Partition D into L and R using f and s
      if L or R is empty **then**
        continue
      end if
      $\mathcal{T}(\mathbf{x}) \leftarrow \text{mean}(\log y_D)$
      $loss \leftarrow \ell_{AFT}(Y_L, Y_R, \mathcal{T}(\mathbf{x}), \sigma)$
      if $loss < best\_loss$ **then**
        $best\_loss \leftarrow loss$
        $best\_split \leftarrow (f, s, L, R)$
      end if
    end for
  end for
  return $best\_split$
end function
function SETVALUE($\{y_i \mid (\mathbf{x}_i, y_i) \in D\}$)
  $y_{\text{uncensored}} \leftarrow \{y_i \mid y_i \text{ is uncensored}\}$
  $value \leftarrow \text{geometric-mean}(y_{\text{uncensored}})$
  return $value$
end function

Algorithm 4 AFT-Tree Initialisation Using Custom Distribution and Bootstrapping

Require:
Training data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
Bootstrapped Flag **is_bootstrapped**
Size of Distribution Data L
Ensure: Survival Tree T
function INITAFTTREE(D, depth, **is_bootstrapped**)
    if is_bootstrapped then
        $\sigma \leftarrow$ BOOTSTRAPFITTING(D)
        Tree $\leftarrow$ BUILDAFTTREE(D, depth, σ)
    else
        Partition D into D_{train} and D_{test} of size L from D
        $\sigma \leftarrow$ Parameter from distribution fitting from $y_{D_{test}}$
        Tree $\leftarrow$ BUILDAFTTREE(D_{test}, depth, σ)
    end if
    return Tree
end function
function BOOTSTRAPFITTING(D)
    Initialize σ_t list $P \leftarrow []$
    for $t = 1$ to n_{trees} **do**
        Draw a bootstrap sample $D^{(t)}$ of size N from D
        $\sigma_t \leftarrow$ Parameter from distribution fitting from $y_{D^{(t)}}$
        Add σ_t to P
    end for
    $\sigma \leftarrow$ MEAN(P)
    return σ
end function

Algorithm 5 The AFT Random Forest Algorithm

Require: Training data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$
Number of Trees n_{trees}
Number of features to sample m_{try}
Bootstrapped Flag **is_bootstrapped**
Custom Fitted Flag **is_custom**
Scale Parameter σ
Ensure: A fitted Random Forest model
function RANDOMFORESTAFTFIT(D, n_{trees}, m_{try})
    Initialize an empty list of trees $\mathcal{T}$
    for $t = 1$ to n_{trees} **do**
        Draw a bootstrap sample $D^{(t)}$ of size N from D
        if is_bootstrapped or is_custom then
            T_t = INITAFTTREE($D^{(t)}$, **is_bootstrapped**)
        else
            T_t = BUILDAFTTREE($D^{(t)}$)
        end if
        Add T_t to $\mathcal{T}$
    end for
    return $\mathcal{T}$
end function
function RANDOMFORESTAFTPREDICT($\mathcal{T}$, $\mathbf{x}$)
    Initialize predictions list $P \leftarrow []$
    for all T_t in $\mathcal{T}$ **do**
        Predict $\hat{y}_t = T_t(\mathbf{x})$
        Add $\hat{y}_t$ to P
    end for
    y' = MEDIANSURVIVALTIME(P)
    return y'
end function

4 Experimental Setup

The experiments that we perform to test our hypothesis are described in subsections 4.1, 4.2 and 4.3. The technologies and tools that are used are described in subsection 4.4.

4.1 Experiments 1: Custom Distribution Fitting AFT-Survival Forest

This experiment aims to investigate the effects of custom distribution fitting on parametric settings. This experiment answers the following research question:

Q2. What is the effect of the proposed distribution fitting algorithm on the proposed AFT Random Forest?

To achieve this, we compare models between non-fitted against fitted parametric distributions to see if the added distribution fitting features can improve the fitting process without searching for the sigma. In this experiment, we also hope to compare the effects on all public datasets of Veteran, NHANES and SUPPORT2, which have differing sizes.

4.1.1 Configuration and Hyperparameter Optimisation

The experiment is run with five cross validations for each log distribution, which are normal, logistic and extreme. Another experiment is run with five cross validations for distribution with custom fitting. The evaluation is going to be by averaging the C-Index, Integrated Brier Score, and Mean Absolute Error over 10 runs. The hyperparameter tuning used is a random search of 10 combinations of hyperparameters from Table 4 for each distribution function. To ensure that the comparisons are fair, we set the random seed for hyperparameter tuning and cross-validation splitting to ensure reproducibility. The cross-validation is also stratified and binned to ensure a similar survival time and censoring distribution. The best model for each function would then be compared with the custom-fitted distribution with additional hyperparameters from Table 5 to see if there are improvements in the metric.

Hyperparameters	**Value**
Number of trees (`n_trees`)	{50, 100, 150}
Maximum Tree Depth (`max_depth`)	{3, 6, 10}
Minimum Samples to Split Tree (`min_samples_split`)	{2, 10, 50, 100, 200}
Minimum Samples for Leaf (`min_samples_leaf`)	{1, 2, 5, 10, 50, 100, 200}
Sigma (σ)	{0.1, 0.5, 1, 5, 10, 50}
Random Forest sampling split (`percent_len_sample_forest`)	{0.25, 0.37, 0.5, 0.75}
Distribution test split (`test_size`)	{0.1, 0.2, 0.3}

Table 4: Hyper-parameter random search space for the experiment.

4.2 Experiments 2: Boostrapping Distribution AFT-Survival Forest

This experiment aims to investigate the effects of bootstrap sampling distribution fitting on non-parametric and parametric settings, which answers the following research question:

Hyperparameters	Value
Distribution test split (`test_size`)	{0.1, 0.2, 0.3}

Table 5: Hyper-parameter random search space only for custom-fitted distribution.

Q3. What is the effect of the proposed bootstrap sampling fitting algorithm on the proposed AFT Random Forest?

We compare models between bootstrapped against non-bootstrapped distributions to see if the added bootstrap sampling fitting features can improve the fitting process than the ordinary train-test split.

4.2.1 Configuration and Hyperparameter Optimisation

The experiment is run with five cross validations for each log time distribution that supports custom fitting, which are Normal, Logistic and Extreme, and Gaussian Mixture models. Another run also five cross validations with the additional bootstrap process rather than fitting from the traditional split training data. The evaluation is going to be by averaging the C-Index, Integrated Brier Score, and Mean Absolute Error over 10 runs of the best hyperparameter found. The basic hyperparameters are derived from the previous experiments for a custom-fitted distribution model and a basic model. For the bootstrapped version, the best hyperparameters for each function hyperparameter are then configured with the addition of a hyperparameter from Table 19. For the Gaussian mixture models, since there are no basic versions, it is decided to use the best hyperparameter from the best function in experiment one, with the addition of hyperparameters from Table 7. From these experiments, it can be seen whether there are any improvements in using the bootstrapped version or the Gaussian mixture models.

Hyperparameters	Value
Bootstrapping sampling count (`n_samples`)	{100, 200}
Bootstrapping sampling percentage (`test_size`)	{0.25, 0.5, 0.75}

Table 6: Hyper-parameter random search space for the second experiment.

Hyperparameters	Value
Components of Mixture Models (`n_components`)	{2, 5, 10}

Table 7: Hyper-parameter random search space for the second experiment.

4.3 Experiments 3: Survival Analysis Performance with AFT-Survival Forest

This experiment aims to compare the best model from the previous experiments against other existing methods, such as Random Survival Forest [18] and XGBoostAFT [17], which answers the following research question:

Q1. How does the proposed Random Forest AFT compare to other existing methods?

In these experiments, we also compare the performance using metric evaluation, C-Index, Calibration and Mean Absolute Error are averaged across the models of Random Survival Forest [18] and XGBoostAFT [17].

4.3.1 Configuration and Hyperparameter Optimisation

The best AFT Forest models for both dataset selected from the previous hyperparameter tuning from the two previous experiments. For other models such as Random Survival Forest [18] and XGBoost-AFT [17], hyperparameter tuning is done for both models with the five-fold cross validation and random search with 10 tries. The hyperparameter search space for XGBoostAFT and Random Survival Mdoels are outlined in Table 26 and Table 27, respectively. The evaluation is going to be by averaging the C-Index, Integrated Brier Score, and Mean Absolute Error over 10 runs of the best hyperparameter found. In addition to previous metrics, calibration is compared in this experiment.

Hyperparameters	**Value**
Max Depth (`max_depth`)	{3, 6, 10}
Sigma (σ)	{0.1, 0.5, 1, 5, 10, 50}
Learning Rate (`learning_rate`)	{0.01, 0.05, 0.1}
L2 Regularization (λ)	{0.01, 0.1, 1]}
L1 Regularization (α)	{0.01, 0.1, 1]}
Boosting Rounds (`num_boost_round`)	{100, 200, 500}
Early Stopping Rounds (`early_stopping_rounds`)	{10, 20}
Function	{Extreme, Normal, Logistic}

Table 8: Hyper-parameter random search space for XGBoost AFT.

Hyperparameters	**Value**
Max Depth (`max_depth`)	{3, 6, 10}
Min Samples Split (`min_samples_split`)	{2, 10, 50, 100, 200}
Min Samples Leaf (`min_samples_leaf`)	{1, 2, 5, 10, 50, 100, 200}
Number of Trees (`n_trees`)	{50, 100, 150}

Table 9: Hyper-parameter random search space for random survival forest.

4.4 Tools and Technologies

The code for the experiments was written using Python 3.11.5. The code is available at `https://github.com/rizkiduwinanto/Survival-Tree-Analysis`. The libraries used are `numpy` for the tree algorithm and `joblib` for the random forest algorithm. Concordance Index is implemented using `lifelines`. `scikit-survival` is the preferred library for survival analysis metrics such as Integrated Brier Score, whereas `scikit-learn` is the preferred library for MAE (Mean Absolute Error). The distribution of `lifelines` and `scipy` is used for distribution fitting and bootstrap sampling. Due to the sluggishness of the initial tree fitting using the CPU, `CuPy` is used to make the tree run on the

`CUDA` GPU. The forest also utilises multiprocessing using `JobLib`. `seaborn` and `matplotlib` are used for visualisation of the graphs, calibration and event timeline. `wandb` is used for monitoring all the hyperparameter tuning and visualisation.

All experiments for the XGBoost hyperparameter tuning were performed on a machine equipped with an Intel Core i7 CPU, featuring 16GB of memory, a 6-core CPU, and a 6GB NVIDIA GTX 1660 Ti GPU. The storage is 1TB of internal SSD. All experiments for the AFT Forest and Random Survival Forest Hyperparameter tuning using the dataset of NHANES and Support were performed using the RUG Habrok Computation GPU Cluster, which features 36 cores of 2.7 GHz Intel Xeon Gold and 768 GB of memory, as well as 2 Nvidia V100 GPU Accelerators with 32 GB RAM. Additionally, the cluster has 621 GB of RAM disk space.

5 Results

The results of the experiments described in the previous chapter will be presented in the following sections.

5.1 Survival Analysis Performance with AFT-Survival Forest

The best hyperparameters found after random search of 10 hyperparameters of each function are shown in Table 10 for experiments using the SUPPORT dataset and Table 11 for experiments using the NHANES dataset.

Hyperparameters	**Function**		
	Extreme	**Normal**	**Logistic**
Max Depth (`max_depth`)	3	3	6
Min Samples Split (`min_samples_split`)	200	200	200
Min Samples Leaf (`min_samples_leaf`)	100	100	2
Sigma (σ)	0.5	0.5	0.1
Percent Len Sample Forest (`percent_len_sample_forest`)	0.25	0.25	0.25
Number of Trees (`n_trees`)	150	150	100

Table 10: Basic Hyperparameters for the experiment using Support Dataset, grouped by the function type used (Extreme, Normal, Logistic).

Hyperparameters	**Function**		
	Extreme	**Normal**	**Logistic**
Max Depth (`max_depth`)	10	3	10
Min Samples Split (`min_samples_split`)	200	200	50
Min Samples Leaf (`min_samples_leaf`)	50	100	100
Sigma (σ)	5	0.5	5
Percent Len Sample Forest (`percent_len_sample_forest`)	0.75	0.5	1
Number of Trees (`n_trees`)	150	150	150

Table 11: Basic Hyperparameters for the experiment using NHANES Dataset, grouped by the function type used (Extreme, Normal, Logistic).

The best validation metrics from the hyperparameter tuning are presented in Table 12 and Table 13 for the two datasets, respectively. In the support dataset, the model with the extreme function yields the highest average of the concordance index and the lowest mean absolute error. The Brier score is the same in all the functions. In the NHANES dataset, the model with the extreme function has the highest average of the concordance index, but the model with the logistic function has the lowest MAE and Brier score.

Function	C-Index (↑)	Brier Score (↓)	MAE (↓)
Extreme	**0.534 ± 0.017**	**0.547 ± 0.009**	**186.234 ± 3.894**
Logistic	0.518 ± 0.026	0.547 ± 0.009	186.249 ± 3.908
Normal	0.533 ± 0.012	0.547 ± 0.009	186.245 ± 3.913

Table 12: Performance of Models using Support Dataset from 5-fold Cross Validation.

Function	C-Index (↑)	Brier Score (↓)	MAE (↓)
Extreme	**0.555 ± 0.169**	0.484 ± 0.006	5.026 ± 0.03
Logistic	0.464 ± 0.155	**0.483 ± 0.006**	**5.014 ± 0.024**
Normal	0.468 ± 0.146	0.484 ± 0.006	5.021 ± 0.03

Table 13: Performance of Models using NHANES Dataset from 5-fold Cross Validation.

5.2 Custom Distribution Fitting AFT-Survival Forest

The best hyperparameters in Table 10 and Table 11 are also used for the custom-fitted distribution, with the addition of hyperparameters in Table 19.

Hyperparameters	Value
Distribution test split (`test_size`)	0.1

Table 14: Chosen Hyper-parameter for the first experiment.

The results of the experiments comparing the custom-fitted distribution for the Support and NHANES datasets are shown in Table 15 and Table 16, respectively. The custom-fitted distribution model using the extreme function yields the best validation score for a model that predicts the support dataset for C-Index of 0.574, Brier Score of 0.547 and MAE of 186.031. While for training NHANES models, the custom-distributed models with the logistic function have the best C-Index with 0.699 and the best Brier Score with 0.482. The models with the logistic function and without the custom distribution produce the best MAE with 5.014.

The Figure 15 shows the C-Index performance with and without custom distributions on the Support Dataset. It can be observed that a custom-fitted distribution algorithm improves the metric only for the extreme model, with nearly the same hyperparameters, while for the other two distributions (normal and logistic), the metric becomes worse. The Brier scores as observed in Figure 16 show that the Brier score remains stable for all the models. Only the extreme custom fitted distribution models show a lower Brier score, even though it is quite small. The Mean Absolute Error observed in Figure 17 also shows small differences with a slight improvement in extreme and logistic function models.

On the NHANES dataset, the validation c-index yields a higher average on the normal and logistic datasets and a lower average on the extreme dataset, as shown in Figure 18. The average Brier score decreased in extreme and logistic models but not in normal models, as shown in Figure 19. In Figure 20, the mean absolute error improves all the hyperparameters by using the custom distribution fitting.

The performance on the hyperparameters using the separated test dataset can be seen in Table 17 using the Support test dataset and Table 18 using the NHANES test dataset. The highest C-Index

scores are 0.580 and 0.771 for the test Support Dataset and the test NHANES dataset, respectively. The prior uses the custom-distribution algorithm using the extreme function hyperparameter, and the latter uses the logistic custom-fitted distribution. The lowest MAE is also achieved by the same models for both datasets.

Function	**Custom Distribution**	**C-Index** (↑)	**Brier Score** (↓)	**MAE** (↓)
Extreme	No	0.534 ± 0.017	0.547 ± 0.009	186.234 ± 3.894
Extreme	Yes	**0.574 ± 0.008**	**0.547 ± 0.008**	**186.031 ± 3.9**
Logistic	No	0.518 ± 0.026	0.547 ± 0.009	186.249 ± 3.908
Logistic	Yes	0.507 ± 0.007	0.547 ± 0.008	186.167 ± 3.881
Normal	No	0.533 ± 0.012	0.547 ± 0.009	186.245 ± 3.913
Normal	Yes	0.451 ± 0.015	0.547 ± 0.009	186.339 ± 3.89

Table 15: Performance of Models using Support Dataset from 5-fold Cross Validation with and without custom distribution.

Function	**Custom Distribution**	**C-Index** (↑)	**Brier Score** (↓)	**MAE** (↓)
Extreme	No	0.555 ± 0.169	0.484 ± 0.006	5.026 ± 0.03
Extreme	Yes	0.488 ± 0.165	0.482 ± 0.006	5.018 ± 0.029
Logistic	No	0.464 ± 0.155	0.483 ± 0.006	**5.014 ± 0.024**
Logistic	Yes	**0.699 ± 0.068**	**0.482 ± 0.005**	5.019 ± 0.025
Normal	No	0.468 ± 0.146	0.484 ± 0.006	5.021 ± 0.03
Normal	Yes	0.505 ± 0.177	0.484 ± 0.005	5.021 ± 0.029

Table 16: Performance of Models using NHANES Dataset from 5-fold Cross Validation with and without custom distribution.

Function	**Custom Distribution**	**C-Index Test** (↑)	**Brier Score Test** (↓)	**MAE Test** (↓)
Extreme	No	0.557	**0.550**	192.264
Extreme	Yes	**0.580**	0.550	**191.945**
Logistic	No	0.553	0.550	192.284
Logistic	Yes	0.525	0.550	192.310
Normal	No	0.541	0.550	192.311
Normal	Yes	0.475	0.550	192.368

Table 17: Test Performance of Models using Support Dataset with and without custom distribution.

5.3 Boostrapping Distribution AFT-Survival Forest

The best hyperparameters reported in Table 10 and Table 11 are used in both comparisons. For the custom-fitted distribution, the additional hyperparameter is taken from Table 14, while the bootstrapped distribution includes the extra hyperparameter shown in Table 19. For the GMM, the best

Function	Custom Distribution	C-Index Test (↑)	Brier Score Test (↓)	MAE Test (↓)
Extreme	No	0.733	0.485	5.085
Extreme	Yes	0.706	0.482	5.042
Logistic	No	0.701	0.482	5.057
Logistic	Yes	**0.771**	**0.478**	**5.034**
Normal	No	0.638	0.482	5.040
Normal	Yes	0.655	0.482	5.061

Table 18: Test Performance of Models using NHANES Dataset with and without custom distribution.

hyperparameters from both datasets from Table 10 and Table 11 are applied regardless of the function, with the number of mixture components (`n_components`) specified in Table 20.

Hyperparameters	Value
Bootstrapping sampling count (`n_samples`)	100
Bootstrapping sampling percentage (`test_size`)	0.5

Table 19: Chosen Hyper-parameter for the second experiment, specific for bootstrapping.

Hyperparameters	Value
Components of Mixture Models (`n_components`)	5

Table 20: Chosen Hyper-parameter for the second experiment, specific for Gaussian Mixture Models.

Function	Custom Distribution	Bootstrap	C-Index Test (↑)	Brier Score Test (↓)	MAE Test (↓)
Extreme	No	No	0.534 ± 0.017	0.547 ± 0.009	186.234 ± 3.894
Extreme	Yes	No	**0.574 ± 0.008**	**0.547 ± 0.008**	**186.031 ± 3.9**
Extreme	Yes	Yes	0.485 ± 0.017	0.547 ± 0.009	186.313 ± 3.924
Logistic	No	No	0.518 ± 0.026	0.547 ± 0.009	186.249 ± 3.908
Logistic	Yes	No	0.507 ± 0.007	0.547 ± 0.008	186.167 ± 3.881
Logistic	Yes	Yes	0.512 ± 0.034	0.547 ± 0.008	186.29 ± 3.889
Normal	No	No	0.533 ± 0.012	0.547 ± 0.009	186.245 ± 3.913
Normal	Yes	No	0.451 ± 0.015	0.547 ± 0.009	186.339 ± 3.89
Normal	Yes	Yes	0.467 ± 0.022	0.547 ± 0.009	186.299 ± 3.892
GMM	Yes	No	0.437 ± 0.013	0.547 ± 0.009	186.403 ± 3.878
GMM	Yes	Yes	0.489 ± 0.009	0.547 ± 0.009	186.281 ± 3.902

Table 21: Performance of Models using Support Dataset from 5-fold Cross Validation with and without bootstrapping.

As shown in Table 21 and Table 22, which extend Table 15 and Table 16, respectively, the models that yield the best C-index are the same as previous experiments with the extreme custom-fitted distribution without bootstrap for support and the logistic custom-fitted distribution without bootstrap

Function	Custom Distribution	Bootstrap	C-Index Test (↑)	Brier Score Test (↓)	MAE Test (↓)
Extreme	No	No	0.555 ± 0.169	0.484 ± 0.006	5.026 ± 0.03
Extreme	Yes	No	0.488 ± 0.165	0.482 ± 0.006	5.018 ± 0.029
Extreme	Yes	Yes	0.483 ± 0.15	0.483 ± 0.006	5.027 ± 0.024
Logistic	No	No	0.464 ± 0.155	0.483 ± 0.006	**5.014 ± 0.024**
Logistic	Yes	No	**0.699 ± 0.068**	**0.482 ± 0.005**	5.019 ± 0.025
Logistic	Yes	Yes	0.548 ± 0.131	0.483 ± 0.006	5.023 ± 0.024
Normal	No	No	0.468 ± 0.146	0.484 ± 0.006	5.021 ± 0.03
Normal	Yes	No	0.505 ± 0.177	0.484 ± 0.005	5.021 ± 0.029
Normal	Yes	Yes	0.409 ± 0.092	0.484 ± 0.006	5.02 ± 0.025
GMM	Yes	No	0.439 ± 0.12	0.483 ± 0.005	5.024 ± 0.03
GMM	Yes	Yes	0.397 ± 0.122	0.485 ± 0.005	5.022 ± 0.025

Table 22: Performance of Models using NHANES Dataset from 5-fold Cross Validation with and without bootstrapping.

for NHANES. Also similar are the lowest Brier score and MAE, which are similar to the Table 15 and Table 16.

For the Support dataset, it can be seen in Figure 21 that bootstrapping yields a substantially worse C-Index for the model with the extreme distribution. A slight decrease in performance is also observed for the normal distribution when bootstrapping is applied. The logistic distribution remains relatively stable across configurations, albeit worse if the logistic distribution is applied, while the GMM consistently achieves the lowest C-Index regardless of the setting. The best performance is obtained by the extreme distribution with a custom distribution but without bootstrapping. Figure 22 shows that Brier Scores remain nearly identical across all functions and configurations for the support dataset. Figure 23 illustrates that the mean absolute error is also similar across all settings with the extreme distribution, with a custom distribution, but without bootstrapping, obtaining the lowest MAE score.

For the NHANES dataset, it is illustrated in Figure 24 that a lower C-Index is generally obtained using bootstrapping compared to without bootstrapping. Also, a lower C-index is obtained compared to the basic version (without custom distribution), if using bootstrapping, except for the model using the logistic function. The GMM function results are also abysmal compared to other functions as a hyperparameter, with the lowest average C-index yielded by GMM with bootstrapping. The highest C-index is obtained by the custom-fitted logistic distribution without bootstrapping applied. Figure 25 shows that the bootstrap process on the AFT-Survival Tree does not offer a significant advantage compared to the model without bootstrapping. In the Figure 26, the MAE hovers around 5, with the lowest still the logistic without custom-fitted distribution and bootstrapping. Generally, bootstrapping does not improve MAE in a model with any function.

Table 23 presents the test performance of AFT Forest models on the SUPPORT dataset with and without custom distributions. The Extreme distribution with a custom distribution and without bootstrapping achieves the best overall results, with the highest C-Index of 0.580 and the lowest MAE of 191.945. While custom distributions can yield improvements—particularly for the Extreme distribution—bootstrapping, as shown in the comparisons, does not improve when custom distributions are not applied. The lowest validation is also achieved by the Gaussian mixture model function without bootstrapping, with a C-Index of 0.437.

Table 24 presents the test performance of AFT Forest models on the NHANES dataset with

and without custom distributions. The logistic distribution with a custom distribution and without bootstrapping achieves the best overall results, with the highest C-Index of 0.699 and the lowest Brier of 0.482. The lowest MAE is achieved by the logistic without any enhancements. The lowest C-Index is reached by the Gaussian mixture model with bootstrapping, with a C-index of 0.397. Overall, boostrapping does not improve any metrics in NHANES.

Function	Custom Distribution	Bootstrap	C-Index Test (↑)	Brier Score Test (↓)	MAE Test (↓)
Extreme	No	No	0.557	**0.550**	192.264
Extreme	Yes	No	**0.580**	0.550	**191.945**
Extreme	Yes	Yes	0.468	0.550	192.428
GMM	Yes	No	0.442	0.550	192.574
GMM	Yes	Yes	0.492	0.550	192.321
Logistic	No	No	0.553	0.550	192.284
Logistic	Yes	No	0.525	0.550	192.310
Logistic	Yes	Yes	0.554	0.550	192.246
Normal	No	No	0.541	0.550	192.311
Normal	Yes	No	0.475	0.550	192.368
Normal	Yes	Yes	0.499	0.550	192.354

Table 23: Test Performance of Models using Support Dataset with and without bootstrapping.

Function	Custom Distribution	Bootstrap	C-Index Test (↑)	Brier Score Test (↓)	MAE Test (↓)
Extreme	No	No	0.733	0.485	5.085
Extreme	Yes	No	0.706	0.482	5.042
Extreme	Yes	Yes	0.717	0.483	5.069
GMM	Yes	No	0.669	0.482	5.056
GMM	Yes	Yes	0.599	0.483	5.067
Logistic	No	No	0.701	0.482	5.057
Logistic	Yes	No	**0.771**	**0.478**	**5.034**
Logistic	Yes	Yes	0.715	0.484	5.073
Normal	No	No	0.638	0.482	5.040
Normal	Yes	No	0.655	0.482	5.061
Normal	Yes	Yes	0.575	0.482	5.058

Table 24: Test Performance of Models using NHANES Dataset with and without bootstrapping.

5.4 Comparison against other Machine Learning Models

The best AFT-Survival Forest from the two previous experiments is then compared to other machine learning models. The best hyperparameters for the Support and NHANES dataset are summarised in Table 25. Hyperparameter tuning is also done with XGBoost and Random Survival Forest with a similar 5-fold cross-validation with 10 random trials. The chosen hyperparameters are shown in Table 26 and Table 27 for the XGBoost and Random Survival Forest, respectively.

Hyperparameters	**Support**	**NHANES**
Max Depth (max_depth)	3	10
Min Samples Split (min_samples_split)	200	50
Min Samples Leaf (min_samples_leaf)	100	100
Percent Len Sample Forest (percent_len_sample_forest)	0.25	0.37
Number of Trees (n_trees)	150	150
Function	Extreme	Logistic
Test Size (test_size)	0.1	0.1
Custom Distribution (is_custom_dist)	Yes	Yes
Bootstrap (is_bootstrap)	No	No

Table 25: Hyperparameters for the best AFTForest models on Support and NHANES datasets based on the previous experiments and cross-validation.

Hyperparameter	**Support**	**NHANES**
Max Depth (max_depth)	6	6
Sigma (σ)	0.5	1
Learning Rate (learning_rate)	0.1	0.05
L2 Regularization (λ)	0.01	0.01
L1 Regularization (α)	0.1	0.01
Boosting Rounds (num_boost_round)	200	200
Early Stopping Rounds (early_stopping_rounds)	10	10
Function	Extreme	Normal

Table 26: Hyperparameters for XGBoost survival analysis on Support and NHANES datasets. Parameters were optimised through cross-validation.

Hyperparameters	**Support**	**NHANES**
Max Depth (max_depth)	10	10
Min Samples Split (min_samples_split)	50	50
Min Samples Leaf (min_samples_leaf)	1	1
Number of Trees (n_trees)	100	100

Table 27: Hyperparameters for Random Survival Forest experiments on Support and NHANES datasets. Parameters were optimised through cross-validation.

Table 28 and Table 11 show the cross-validation performance using the Support and NHANES datasets, respectively. The XGBoost model achieves the best C-Index for both datasets, with an average of 0.764 for XGBoost and 0.811 for NHANES. The lowest is achieved by the C-Index of the proposed AFT Survival Tree methodology, with an average of 0.574 for XGBoost and 0.699 for NHANES. The overall lowest Brier score is obtained by the Random Survival Forest for both datasets. Also, for both datasets, the overall lowest MAE score is obtained by the AFT Forest, with just over 186 for Support and just over 5 for NHANES.

Model	**C-Index** (↑)	**Brier Score** (↓)	**MAE** (↓)
XGBoost	**0.764 ± 0.004**	0.437 ± 0.004	270.095 ± 18.444
RSF	0.675 ± 0.01	**0.376 ± 0.005**	308.655 ± 19.796
AFT Forest	0.574 ± 0.008	0.547 ± 0.008	**186.031 ± 3.9**

Table 28: Performance of Models using Support Dataset from 5 Cross Validation

Model	**C-Index** (↑)	**Brier Score** (↓)	**MAE** (↓)
XGBoost	**0.811 ± 0.008**	0.162 ± 0.005	14.765 ± 1.136
RSF	0.761 ± 0.008	**0.139 ± 0.005**	7.301 ± 0.163
AFT Forest	0.699 ± 0.068	0.482 ± 0.005	**5.019 ± 0.025**

Table 29: Performance of Models using NHANES Dataset from 5 Cross Validation

Figure 27, Figure 28, and Figure 29 respectively show the cross-validation performance of C-Index, Brier and MAE of models using the Support Dataset. It can be observed that the C-Index of using XGBoost is higher than Random Survival Forest and AFT Forest, which is the worst by far on this dataset. The AFT-Forest yields also worse Brier score compared to XGBoost and Random Survival Forest. However, the mean absolute error of AFT Forest is better than XGBoost and Random Survival Forest, which have a big variance in MAE.

The cross-validation performance of C-Index, Brier and MAE of models using the NHANES dataset is shown Figure 30, Figure 31 and Figure 32, respectively. The best C-Index is achieved by the XGBoost, the Random Survival Forest and the AFT Forest. A large variance is observed in the C-Indexes of the AFT Forest in the figures. The Brier Score is nearly similar to that in Support. The AFT Forest yields the lowest mean absolute error and then followed by Random Survival Forest and XGBoost.

Table 30 and Table 31 respectively show the test dataset performance for the Support and NHANES datasets. The highest C-Index test is achieved by the XGBoost model with a C-index of 0.783 using the Support test dataset and 0.814 using the NHANES test dataset. The lowest C-index for support is achieved by AFT-Forest with a C-index of 0.580. However, for NHANES, the lowest C-index is obtained by Random Survival Forest with a C-index of 0.765. The lowest Brier score is achieved by the random survival forest for both datasets, with 0.382 on support and 0.141 on NHANES. The lowest mean absolute error is achieved by AFT Forest with values of 191.945 on Support and 5.034 on NHANES,

The tree calibration using the Support test dataset is shown in Figure 7 for XGBoost, Figure 8 for Random Survival Forest, Figure 9 for AFT Forest. The tree calibration using the NHANES test dataset is shown in Figure 10, Figure 11, Figure 12 for XGBoost, Random Survival Forest and AFT

Model	C-Index Test (↑)	Brier Score Test (↓)	MAE Test (↓)
XGBoost	**0.783**	0.422	243.179
RSF	0.682	**0.382**	306.219
AFT Forest	0.580	0.550	**191.945**

Table 30: Testing Performance of Models using Support Dataset from 5 Cross Validation

Model	C-Index Test (↑)	Brier Score Test (↓)	MAE Test (↓)
XGBoost	**0.814**	0.164	13.440
RSF	0.765	**0.141**	7.303
AFT Forest	0.771	0.478	**5.034**

Table 31: Testing Performance of Models using NHANES Dataset from 5 Cross Validation

Forest, respectively. The tree calibration is useful to illustrate how well the test predictions performed compared to the true survival time. To emphasise the model usage by practitioners for a certain number of patients, a patient event timeline can help illustrate the prediction from three random uncensored cases, also referred to as actual events, and two random censored cases. The patient event timeline for all models predicted with the dataset Support can be found in Figure 13. For the models trained using the NHANES dataset, Figure 14 illustrates the patient event timelines created by models of XGBoost, random survival forest and AFT Forest.

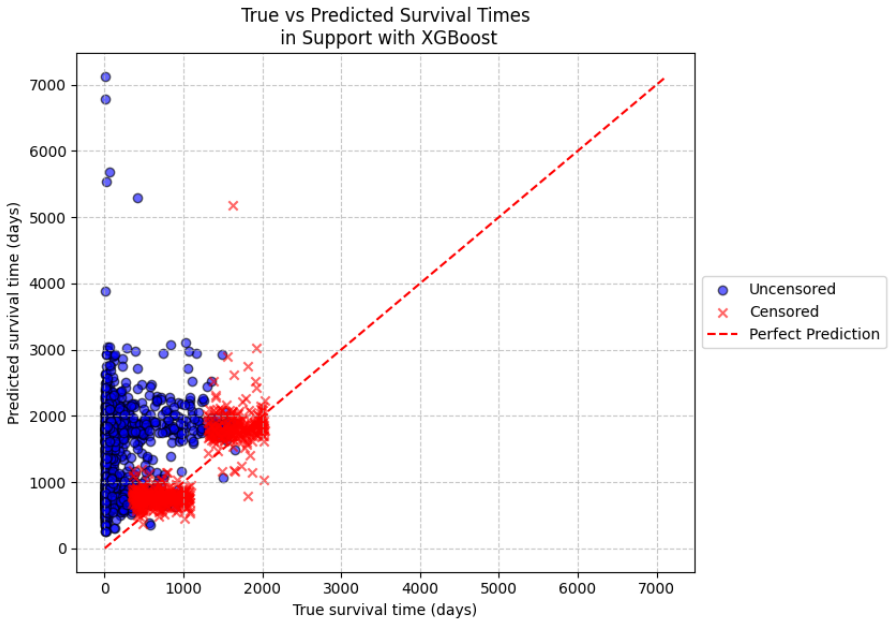

Figure 7: XGBoost model calibration on Support.

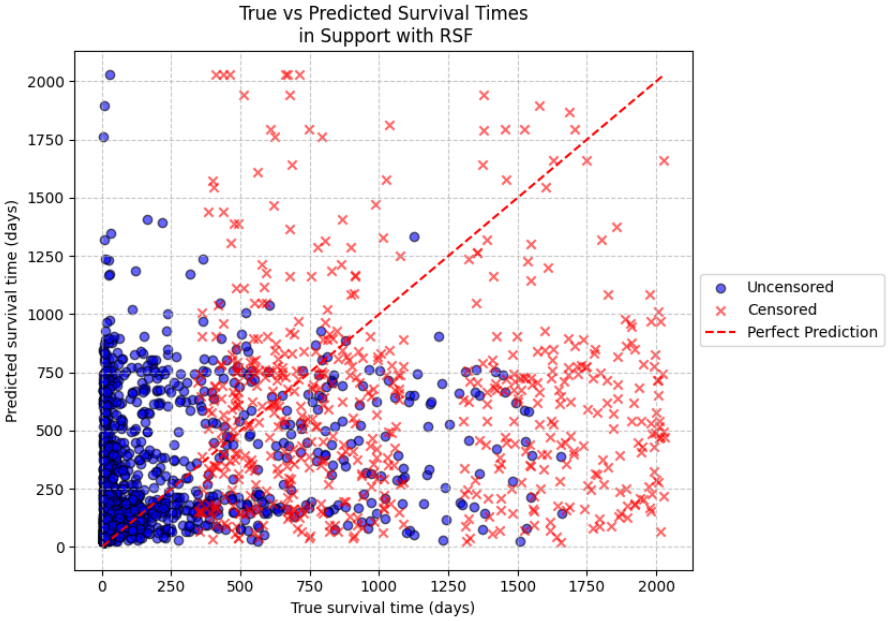

Figure 8: Random Survival Forest model calibration on Support.

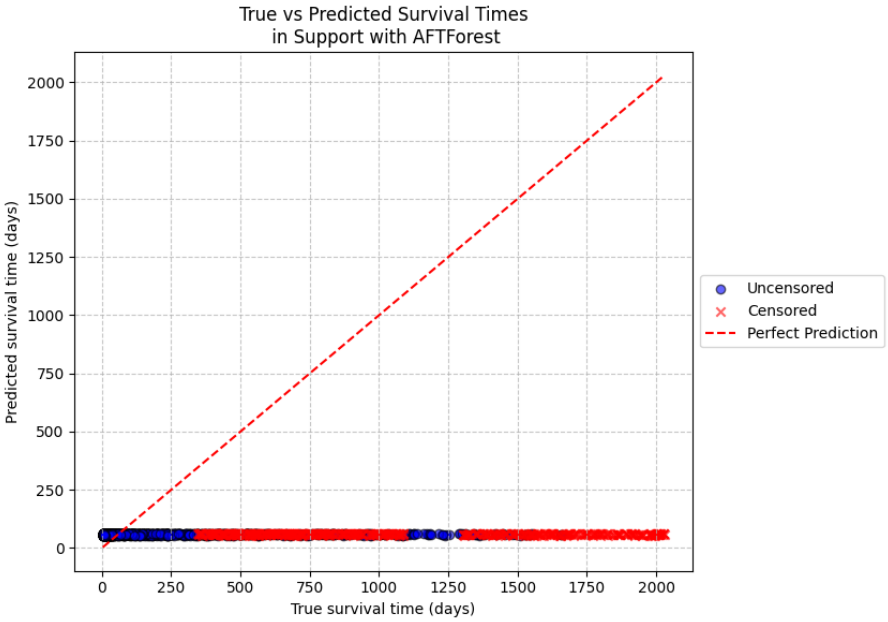

Figure 9: AFT Forest model calibration on Support.

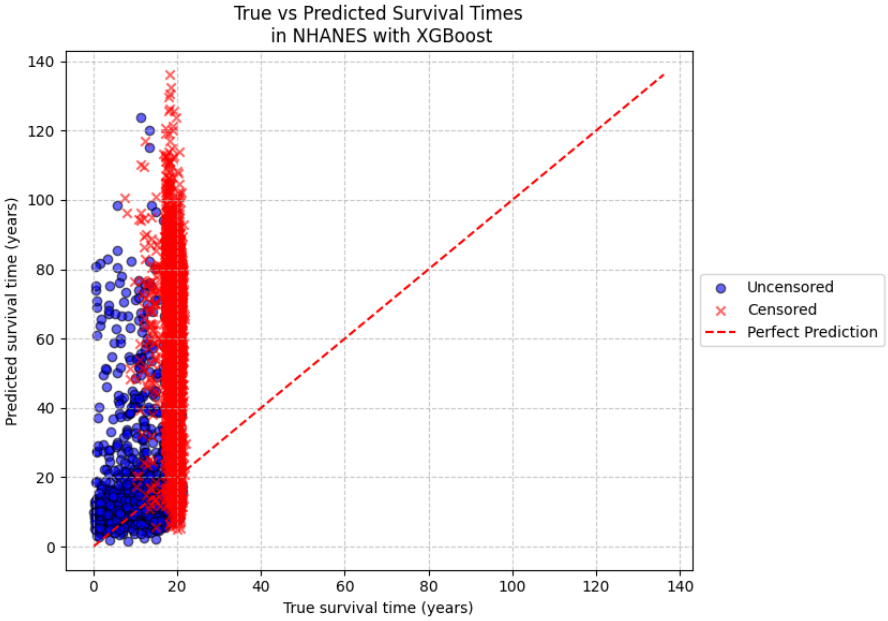

Figure 10: XGBoost model calibration on NHANES.

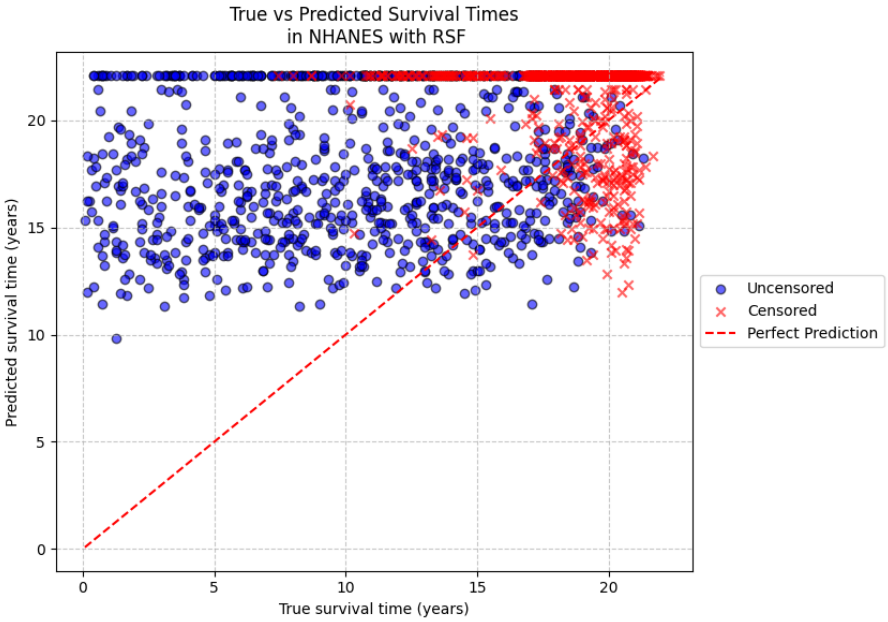

Figure 11: Random Survival Forest model calibration on NHANES.

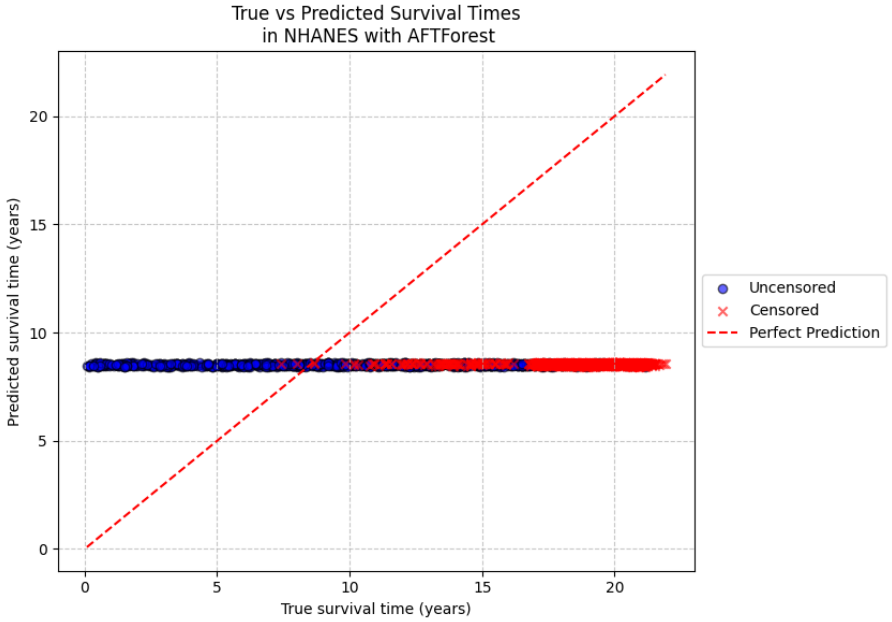

Figure 12: AFT Forest model calibration on NHANES.

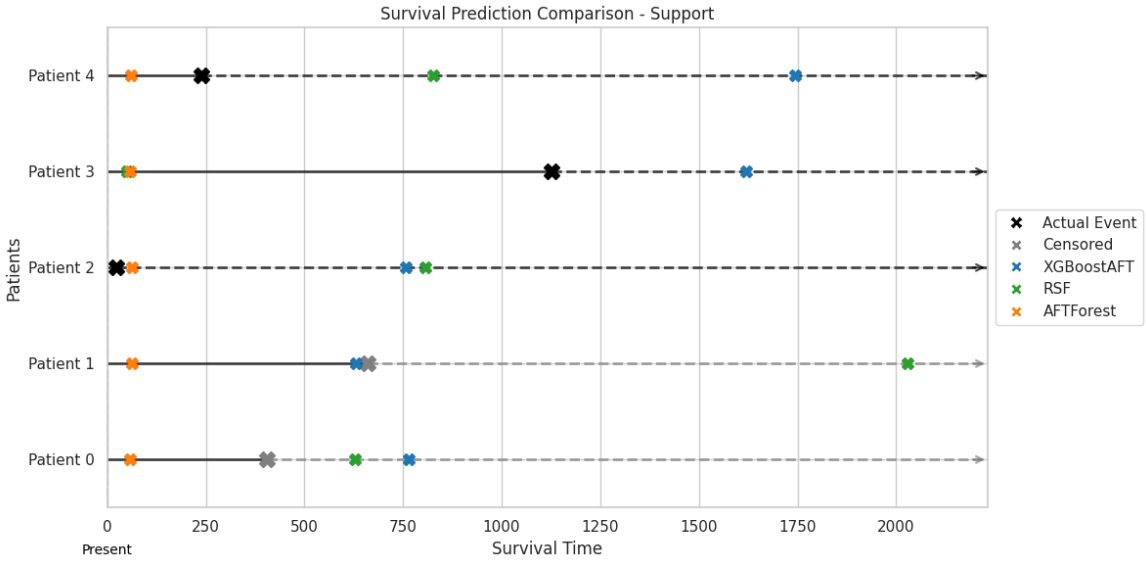

Figure 13: Predicted and ground truths for three random uncensored events and two random censored cases with the models trained on Support.

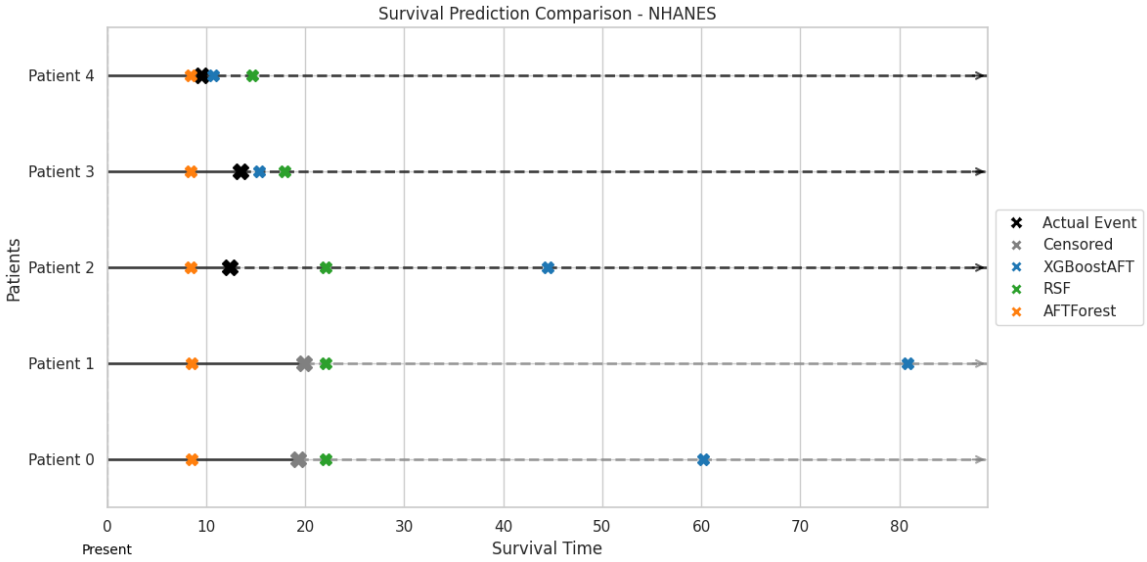

Figure 14: Predicted and ground truths for three random uncensored events and two random censored cases with the models trained on NHANES.

6 Discussion

Several experiments were performed to answer the following research questions:

Q1. How does the proposed Random Forest AFT compare to other existing methods?

Q2. What is the effect of the proposed distribution fitting algorithm on the proposed AFT Random Forest?

Q3. What is the effect of the proposed bootstrap sampling fitting algorithm on the proposed AFT Random Forest?

6.1 Predictive Power of AFT-Random Forest

The result of the model with the dataset of Support and NHANES without a custom distribution, as observed in Table 12 and Table 13, outlines that the overall validation training performance of the AFT Forest model is very low with a C-Index around 0.5, which indicates a random prediction across the validation. In particular, for models with the Support Dataset, the results are more uniform with a C-Index of 0.5. In a training for the NHANES dataset, the C-Index for certain hyperparameter functions, such as logistic and normal, reaches below 0.5 with a large standard deviation of 0.15, which indicates an unstable training procedure. Whereas for the training for the Support dataset, it occasionally reaches just below 0.5, especially for the logistic and normal functions. With the results of the testing dataset shown in Table 30, it is clear that the model is still underfitting, as overfitting usually requires a large overconfidence, and that means a large C-Index during cross-validation, as told by Aliferis and Simon [54].

In some cross-validation results, the C-Index below 0.5 is often obtained, suggesting predictions worse than random. There are several reasons why this is happening. The first reason is due to underfitting into multiple trees, which causes the order of the data to the wrong order as trees are biased. The forest uses the average to aggregate multiple forests according to align with Random Forest [18], and are expected to alleviate overfitting. However, the results are becoming more underfitted because of reduced variance across all trees. One of the causes of reduced variance might be the usage of all the features in the subspace across the trees. This is particularly not tested due to the preference for fair comparison with XGBoost and Random Survival Forest (with hyperparameter setting), which use all the features to build all the trees. The second reason is due to the calculation accelerated loss function itself, which is designed based on research from Barnwal et al. [17], which uses negative log-likelihood, which does not capture the pattern of the data, especially on Support data. The third reason is the uncertainty about the data, where the covariates may not align well with the accelerated failure time assumptions, limiting the model fit. Some methods can alleviate this problem by using quantile-varying covariate effects for accelerated failure time assumptions as outlined by Reeder et al. [55], but it is not implemented in the loss function due to the limited scope of the research.

It is observed that in the AFT Forest with Support that the Brier Score is not a very good metric to differentiate hyperparameters because the value is always the same of around 0.547. This could be because the Brier score, defined in Equation 18, is calculated based on median survival models, not direct time-to-event, and each leaf produces a more uniform value for each leaf. For the AFT Forest with NHANES, the Brier score results are less uniform, although they just differ with a standard deviation of 0.006. Mean absolute error also differs slightly for each hyperparameter tuning result for both trainings with NHANES and Support. This could mean that the values of each leaf, for which are the absolute time-to-event for uncensored, can be similar for all hyperparameters. This is already

alleviated with the geometric mean, instead of the arithmetic mean [9], following the output design of XGBoost AFT [17], but the issue persists, because of the lack of updates from boosting of XGBoost AFT [17]. The lack of a determined scaling formula for time-to-event also causes the uniformity of the Mean Absolute Error.

The overall chosen hyperparameters favour a larger number of trees, as explained in Table 10 and Table 11, with mostly 15 trees as hyperparameters, which aligns with research of random forest from Breiman [15]. The feature subspace use is mostly small for Support, with an average of 0.25, while for NHANES favours a large feature subspace between 0.5 and 1. Both datasets overall favour a larger minimum sample split and leaf with a certain anomaly.

6.2 Effects of Custom Distribution

The custom distribution enhances certain hyperparameter functions and yields better results for extreme distributions in Support AFT Forest training, as well as for logistic and normal distributions in NHANES AFT Forest training, as shown in the C-Index column in Table 15 and Table 16. This also aligns with the test results for the test dataset for both datasets as outlined in Table 17 and Table 18. This proves that custom distribution can improve results if the covariates align with the distribution that is fitted. However, if the distribution cannot be fitted or does not converge according to the Newton-Raphson method, the performance would decrease, causing a lower C-index. The parameter chosen from Table 14 preserves the training data so as not to lose too much information, and splitting data also preserves the distribution for censoring and time using binning, which enables the Newton-Raphson algorithm to determine the correct sigma according to covariates. The correct splitting data also avoids overfitting, which is feared could lower the metrics of the C-Index.

6.3 Effects of Bootstrap Sampling on Custom Distribution

Overall, the bootstrap sampling algorithm does not enhance the certain hyperparameter function compared to only a custom-fitted algorithm. It should improve fitting to avoid overfitting as outlined by Freedman [52], but in reality, it creates underfitting as shown in Table 21 and Table 22. There are several reasons for the underfitting using bootstrap sampling. First, the number of samples of 100, and the percentages of each data as samples of 50 per cent, might not be sufficient. There should be a need to add more samples for sampling. Second, each bootstrap sampling causes a loss of sigma, which is not well aggregated with the average, which causes an underfitting of distributions. Finally, the lack of binning of each random sample to preserve the parameters of each accelerated failure time distribution.

Gaussian mixture models present worse results compared to any other functions on both datasets. This can be because the Gaussian mixture models create overfitting and underfitting on the models without bootstrapping due to 10 per cent data splitting. With bootstrapping, the effect of why bootstrapping fails alleviates this, as shown in the performance of the NHANES model with GMM and bootstrapping. The number of components might need to be tuned because 5 Gaussian components in the distribution might not capture or overcapture the distributions of the data.

6.4 Comparison against other Machine Learning Models

If we compared all the models, XGBoost AFT proposed by Barnwal et al. [17] is much better in terms of metric performance and also time performance. The best models of Accelerated Failure Time Forest are shown to still lag behind XGBoost and Random Survival Forest. Especially in Support,

the C-Index discrepancies are large compared to Random Survival Forest, which also has a large discrepancy to XGBoost. Whereas in NHANES, the discrepancy of the C-Index is smaller, but AFT-Forest still lags in validation performance behind Random Survival Forest and AFT Forest. This proves that boosting, such as with XGBoost, is still better than the bagging process due to the frequent updates on its weak learners' output with gradient updates rather than aggregating multiple weak learners. The algorithm also runs faster and more accurately than AFT-Forest and Random Survival Forest.

The lowest Brier score is achieved by Random Survival Forest on both datasets. The Brier score is proven to be better on the random survival model due to its prediction of risks rather than time-to-event. On the other hand, the lowest mean absolute error is achieved by the Accelerated Failure Time, which shows less difference in uncensored time-to-event. Yet, this does not necessarily a good thing as it could mean that the prediction only plateaus to a bias and a calibration check is needed. The XGBoost on NHANES, particularly, has a high Mean Absolute Error of approximately 14.7, compared to 7.3 of RSF and 5 of AFT Forest as outlined in Table 29. This is because the boosting process might overestimate the prediction of time-to-event due to rigorous gradient updates and the effects of the sigma and learning rate hyperparameters.

The performance using the test dataset also outlines that XGBoost produces the best C-Index of 0.783 and 0.814, for Support and NHANES, respectively. On Support testing, it is followed by Random Survival Forest with a C-Index of 0.783 and AFT-Forest of 0.580. However, on the NHANES testing, the AFT-Forest performs better than Random Survival Forest on test data in Table 31 with 0.771 and 0.765 C-Index, respectively. This means that the AFT loss function can produce a better splitting on NHANES compared to the Log-rank criterion on Random Survival Forest. The possible reasons are the Random Survival Forest's log-rank, which only splits based on the largest risk difference of two groups, and the hyperparameter on Random Survival Forest, which plateaus at 100 trees compared to AFT-Forest's 150 trees. Also on the tuning hyperparameter of Random Survival Forest favours a low minimum sample leaf, despite the same seed for both hyperparameters; more exploration and grid search are needed to compare both models.

Generally, the models trained on NHANES perform better than Support in all of the models on the C-Index metric, as shown in Table 29 for NHANES and Table 28 for Support. There are multiple reasons why NHANES models give better results than Support. The first reason is that NHANES has more features and values, which creates a better splitting decision of the loss function of XGBoost AFT, Log-rank, and AFT-Forest Loss function. The second reason is that the covariates are much pronounced in NHANES, with a median preprocessing for empty values, which can affect it. The third reason is the year time scale, which is much better than the day scale of support, which affects the loss function splitting. The final reason is that NHANES has more censoring than support, which affects the C-index calculation, and the cross-validation is stratified to have equal censoring. This means that for time-to-event prediction, a dataset with a year scale and larger censoring is more preferred, despite also that hidden distributions in data are also a major factor. Therefore, generally in this research, NHANES is more suitable to be used for time-to-event modelling than Support. Another research is needed to determine whether hazard-based models would yield different results.

If we observed the calibration for Support and NHANES, it could be seen that no perfect calibration is reached for any of the models. For XGBoost, it always tends to overestimate with many predictions of above 2000 days, as shown in Figure 7 and Figure 10. As discussed previously, the cause of this is due to rigorous gradient updates. In the XGBoost calibration, the censored cases are grouped for both in Support and NHANES. It could be noted that in Support, the censored cases are more grouped towards the perfect prediction. For Random Survival Forest with Support, the calibrations are more dispersed as seen in Figure 8, especially for higher survival times. There is a systematic

underestimation for long survival times and an overestimation for very short survival times. Censored data (red crosses) appear more spread out in the random survival forest. In contrast, Random Survival Forest with NHANES as seen in Figure 11 tends to underestimate to same value of just above 20 years, with some data scattered. There is a bias that overwhelmed most of the prediction, causing the clustering in the calibration. The calibration of the AFT Forest collapse in a narrow prediction of around 50-100 days for Support and 8-9 years for NHANES as seen in Figure 9 and Figure 12.

In real life, using AFT Random Forest proves unreliable. If a practitioner or researchers use the model on a random patient, as in Figure 13 and Figure 14. The AFT Forest will severely underestimate the time-to-event in Support and NHANES models equally. This renders the approach of using AFT Random Forest still ineffective for practitioners in research. However, it can also be observed that XGBoost and Random Survival Forest mostly overestimate in both models and produce less accurate time-to-event predictions also. This is also a disadvantage that can affect the prognosis in both datasets and confuse practitioners in determining the lifetime of a patient who has the symptoms and a previous disease, despite a large C-Index being present. It should be noted that the C-Index is only discriminatory, and that means a large value means that the ordering is nearly correct. That means in practice, to measure the goodness of models, a large C-Index is not a guarantee. The ideal way to use it is also the combination of other metrics, such as Mean Absolute Error. but also needed.

There are a few reasons why the AFT Forest approach does not work, such as the leaf value design choice of using the geometric mean, which is the exponent of the average log time, and aggregation according to Breiman [15]. It is also tried to have assumed that the leaf is independent and has its own AFT distribution. Then, it is fitted with an AFT distribution using the same algorithm, but the calibration remains the same. There should be more exploration on how to produce a leaf value that is adjunct to the absolute true absolute time-to-event. There is also a need to explore the scaling of the poor calibration to produce a better time-to-event value for both censored and uncensored cases. It could also prove that XGBoost AFT rely more on the boosting process to update the absolute time-to-event by gradient updates rather than just aggregating with average multiple absolute time-to-event [38][17].

7 Conclusion

7.1 Summary of Main Contributions

In conclusion, an accelerated failure time random forest that features a custom distribution, bootstrap sampling and non-parametric fitting using a Gaussian mixture model is implemented. The effects of the custom-fitted distribution produce a better result for certain hyperparameter functions overall. The bootstrap sampling, in contrast, does not present an improvement to the custom-fitted distribution and tends to produce a worse performance. Additionally, the non-parametric fitting fails to improve performance for both training and testing, yielding worse results compared to other distribution fitting methods. Generally, the Accelerated Failure Time random forest still yields worse performance in survival analysis time-to-event prediction compared to XGBoost and Random Survival Forest. Moreover, qualitatively, AFT Forest produces the worst calibration compared to other models, even though all the calibrations for NHANES and Support for all models are still imperfect. Therefore, the AFT Forest model still cannot outperform existing methods such as XGBoost and Random Survival Forest, but this research can be a step toward more effective direct time-to-event prediction models.

7.2 Limitations and Future Work

There are a few limitations of this research that would be addressed. First, the adherence to random hyperparameter tuning due to time and resource constraints, and the necessary testing for the AFT Random Forest. The number of random hyperparameters is constrained to only 10 combinations of hyperparameters due to time and resources, since the AFT-Forest need GPU and CPU to run, despite the optimisation done to speed up the training process of the AFT-Forest.

Second, the lack of a fair comparison also exists due to a random search hyperparameter tuning strategy, despite an equal search space for the number of trees, and the existence of a seed. The comparison should be either to use a grid search hyperparameter search, or a more efficient Bayesian tuning [56] for a fairer comparison.

The future work of the AFT Survival Forest implementation can be to update the algorithm to use boosting rather than bagging, since it is proven that random forests cannot outperform existing XGBoost. With the implementation of the AFT Boosting algorithm, more accurate comparisons with XGBoost can be made.

Also, another future work consideration is to implement the custom-fitting distribution, bootstrapping and non-parametric fitting directly into XGBoost AFT.

The exploration of changing the distribution parameter each time of the node splitting can be an improvement for this research, since without and with a custom distribution and bootstrapping, the distribution parameter or sigma does not change. In XGBoost, L1 and L2 regularisation exist and can be explored further on AFT Forest [38].

Lastly, a better algorithm for determining the leaf value for the absolute time-to-event that encompasses the uncensored and censored data can be explored since this research already tried the arithmetic mean, geometric mean and heterogeneous AFT distribution on its leaf. Better aggregation for the forest can be determined for the survival time, since this research relies on the original implementation.

Bibliography

[1] J. Bajwa, U. Munir, A. Nori, and B. Williams, "Artificial intelligence in healthcare: transforming the practice of medicine," *Future healthcare journal*, vol. 8, no. 2, pp. e188–e194, 2021.

[2] M. Dave and N. Patel, "Artificial intelligence in healthcare and education," *British dental journal*, vol. 234, no. 10, pp. 761–764, 2023.

[3] R. Guido, S. Ferrisi, D. Lofaro, and D. Conforti, "An overview on the advancements of support vector machine models in healthcare applications: a review," *Information*, vol. 15, no. 4, p. 235, 2024.

[4] N. Shahid, T. Rappon, and W. Berta, "Applications of artificial neural networks in health care organizational decision-making: A scoping review," *PloS one*, vol. 14, no. 2, p. e0212356, 2019.

[5] J.-M. Bae, "The clinical decision analysis using decision tree," *Epidemiology and health*, vol. 36, p. e2014025, 2014.

[6] D. B. Olawade, A. C. David-Olawade, O. Z. Wada, A. J. Asaolu, T. Adereni, and J. Ling, "Artificial intelligence in healthcare delivery: Prospects and pitfalls," *Journal of Medicine, Surgery, and Public Health*, p. 100108, 2024.

[7] F. E. Harrell *et al.*, *Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis*. Springer, 2001, vol. 608.

[8] 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.

[9] 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.

[10] M. J. Crowther, P. Royston, and M. Clements, "A flexible parametric accelerated failure time model and the extension to time-dependent acceleration factors," *Biostatistics*, vol. 24, no. 3, pp. 811–831, 2023.

[11] V. Van Belle, K. Pelckmans, J. A. Suykens, and S. Van Huffel, "Survival svm: a practical scalable algorithm." in *ESANN*, 2008, pp. 89–94.

[12] G. Kantidakis, A.-D. Hazewinkel, and M. Fiocco, "Neural networks for survival prediction in medicine using prognostic factors: a review and critical appraisal," *Computational and Mathematical Methods in Medicine*, vol. 2022, no. 1, p. 1176060, 2022.

[13] I. Bou-Hamad, D. Larocque, and H. Ben-Ameur, "A review of survival trees," 2011.

[14] D. Bertsimas, J. Dunn, E. Gibson, and A. Orfanoudaki, "Optimal survival trees," *Machine learning*, vol. 111, no. 8, pp. 2951–3023, 2022.

[15] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[16] P. Beja-Battais, "Overview of adaboost : Reconciling its views to better understand its dynamics," 2023. [Online]. Available: https://arxiv.org/abs/2310.18323

[17] A. Barnwal, H. Cho, and T. Hocking, "Survival regression with accelerated failure time model in xgboost," *Journal of Computational and Graphical Statistics*, vol. 31, no. 4, pp. 1292–1302, 2022.

[18] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer, "Random survival forests," 2008.

[19] B. Seo and S. Kang, "Accelerated failure time modeling via nonparametric mixtures," *Biometrics*, vol. 79, no. 1, pp. 165–177, 2023.

[20] S. Wiegrebe, P. Kopper, R. Sonabend, B. Bischl, and A. Bender, "Deep learning for survival analysis: a review," *Artificial Intelligence Review*, vol. 57, no. 3, p. 65, 2024.

[21] D. Verhaest and S. Baert, "The effects of workplace learning in higher education on employment and match quality: is there an early-career trade-off?" *Empirical Economics*, vol. 55, no. 3, pp. 1229–1270, 2018.

[22] M. Nazarzadeh, Z. Bidel, D. Canoy, E. Copland, M. Wamil, J. Majert, K. S. Byrne, J. Sundström, K. Teo, B. R. Davis *et al.*, "Blood pressure lowering and risk of new-onset type 2 diabetes: an individual participant data meta-analysis," *The Lancet*, vol. 398, no. 10313, pp. 1803–1810, 2021.

[23] S. V. Deo, V. Deo, and V. Sundaram, "Survival analysis—part 2: Cox proportional hazards model," *Indian journal of thoracic and cardiovascular surgery*, vol. 37, pp. 229–233, 2021.

[24] N. Jiang, Y. Wu, and C. Li, "Limitations of using cox proportional hazards model in cardiovascular research," *Cardiovascular Diabetology*, vol. 23, no. 1, p. 219, 2024.

[25] S. Bosson-Amedenu, E. Ayitey, F. Ayiah-Mensah, and L. Asare, "Evaluating key predictors of breast cancer through survival: a comparison of aft frailty models with lasso, ridge, and elastic net regularization," *BMC cancer*, vol. 25, no. 1, p. 665, 2025.

[26] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[27] G. James, D. Witten, T. Hastie, R. Tibshirani *et al.*, *An introduction to statistical learning*. Springer, 2013, vol. 112, no. 1.

[28] D. J. Hand, "Principles of data mining," *Drug safety*, vol. 30, pp. 621–622, 2007.

[29] L. Gordon and R. A. Olshen, "Tree-structured survival analysis." *Cancer treatment reports*, vol. 69, no. 10, pp. 1065–1069, 1985.

[30] A. Ciampi, S. A. Hogg, and L. Kates, "Regression analysis of censored survival data with the generalized f family—an alternative to the proportional hazards model," *Statistics in Medicine*, vol. 5, no. 1, pp. 85–96, 1986.

[31] T. Emura, W.-C. Hsu, and W.-C. Chou, "A survival tree based on stabilized score tests for high-dimensional covariates," *Journal of Applied Statistics*, vol. 50, no. 2, pp. 264–290, 2023.

[32] R. B. Davis and J. R. Anderson, "Exponential survival trees," *Statistics in medicine*, vol. 8, no. 8, pp. 947–961, 1989.

[33] T. M. Therneau, P. M. Grambsch, and T. R. Fleming, "Martingale-based residuals for survival models," *Biometrika*, vol. 77, no. 1, pp. 147–160, 1990.

[34] C.-A. Tsai, D.-T. Chen, J. J. Chen, C. M. Balch, J. F. Thompson, and S.-J. Soong, "An integrated tree-based classification approach to prognostic grouping with application to localized melanoma patients," *Journal of biopharmaceutical statistics*, vol. 17, no. 3, pp. 445–460, 2007.

[35] A. Linden and P. R. Yarnold, "Modeling time-to-event (survival) data using classification tree analysis," *Journal of Evaluation in Clinical Practice*, vol. 23, no. 6, pp. 1299–1308, 2017.

[36] A. J. Ferreira and M. A. Figueiredo, "Boosting algorithms: A review of methods, theory, and applications," *Ensemble machine learning: Methods and applications*, pp. 35–85, 2012.

[37] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[38] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[39] M. Lu and S. Ilgin Guler, "Comparison of random survival forest with accelerated failure time-weibull model for bridge deck deterioration," *Transportation Research Record*, vol. 2676, no. 7, pp. 296–311, 2022.

[40] S. Wongvibulsin, K. C. Wu, and S. L. Zeger, "Clinical risk prediction with random forests for survival, longitudinal, and multivariate (rf-slam) data analysis," *BMC medical research methodology*, vol. 20, pp. 1–14, 2020.

[41] P. Murphy, B. Kreling, E. Kathryn, M. Stevens, J. Lynn, and J. Dulac, "Description of the support intervention," *Journal of the American Geriatrics Society*, vol. 48, no. S1, pp. S154–S161, 2000.

[42] UCI. [Online]. Available: https://archive.ics.uci.edu/dataset/880/support2

[43] C. J. Patel, N. Pho, M. McDuffie, J. Easton-Marks, C. Kothari, I. S. Kohane, and P. Avillach, "A database of human exposomes and phenomes from the us national health and nutrition examination survey," *Scientific data*, vol. 3, no. 1, pp. 1–10, 2016.

[44] T. A. Gerds, M. W. Kattan, M. Schumacher, and C. Yu, "Estimating a time-dependent concordance index for survival prediction models with covariate dependent censoring," *Statistics in medicine*, vol. 32, no. 13, pp. 2173–2184, 2013.

[45] A. Alabdallah, M. Ohlsson, S. Pashami, and T. Rögnvaldsson, "The concordance index decomposition: A measure for a deeper understanding of survival prediction models," *Artificial Intelligence in Medicine*, vol. 148, p. 102781, 2024.

[46] 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.

[47] R. Goswami and A. K. Dey, "Integrated brier score based survival cobra–a regression based approach," *arXiv preprint arXiv:2210.12006*, 2022.

[48] S. Akram and Q. U. Ann, "Newton raphson method," *International Journal of Scientific & Engineering Research*, vol. 6, no. 7, pp. 1748–1752, 2015.

[49] D. Reynolds, "Gaussian mixture models," in *Encyclopedia of biometrics*. Springer, 2015, pp. 827–832.

[50] G. McLachlan and K. Basford, "Mixture models: Inference and applications to clustering, marcel dekker," *Inc. New York*, pp. 10–18, 1988.

[51] J. L. Horowitz, "Bootstrap methods in econometrics," *Annual Review of Economics*, vol. 11, no. 1, pp. 193–224, 2019.

[52] D. A. Freedman, "Bootstrapping regression models," *The annals of statistics*, pp. 1218–1228, 1981.

[53] G. A. Young, "Bootstrap: More than a stab in the dark?" *Statistical Science*, pp. 382–395, 1994.

[54] C. Aliferis and G. Simon, "Overfitting, underfitting and general model overconfidence and under-performance pitfalls and best practices in machine learning and ai," *Artificial intelligence and machine learning in health care and medical sciences: Best practices and pitfalls*, pp. 477–524, 2024.

[55] H. T. Reeder, K. H. Lee, and S. Haneuse, "Characterizing quantile-varying covariate effects under the accelerated failure time model," *Biostatistics*, vol. 25, no. 2, pp. 449–467, 2024.

[56] A. H. Victoria and G. Maragatham, "Automatic tuning of hyperparameters using bayesian optimization," *Evolving Systems*, vol. 12, no. 1, pp. 217–223, 2021.

Appendices

A Dataset

A.1 Support Dataset Preprocessing

Baseline Variable	Normal Fill-in Value
Serum albumin (alb)	3.5
PaO_2/FiO_2 ratio (pafi)	333.3
Bilirubin (bili)	1.01
Creatinine (crea)	1.01
Blood urea nitrogen (bun)	6.51
White blood count (wblc)	9 (thousands)
Urine output (urine)	2502

Table 32: Imputation of Support Dataset [42]

	Variable Name
1	aps
2	sps
3	surv2m
4	surv6m
5	prg2m
6	prg6m
7	dnr
8	dnrday
9	sfdm2
10	hospdead
11	slos
12	charges
13	totcst
14	totmcst

Table 33: Variables Dropped from the Support Dataset

A.2 Additional Visualisation for results

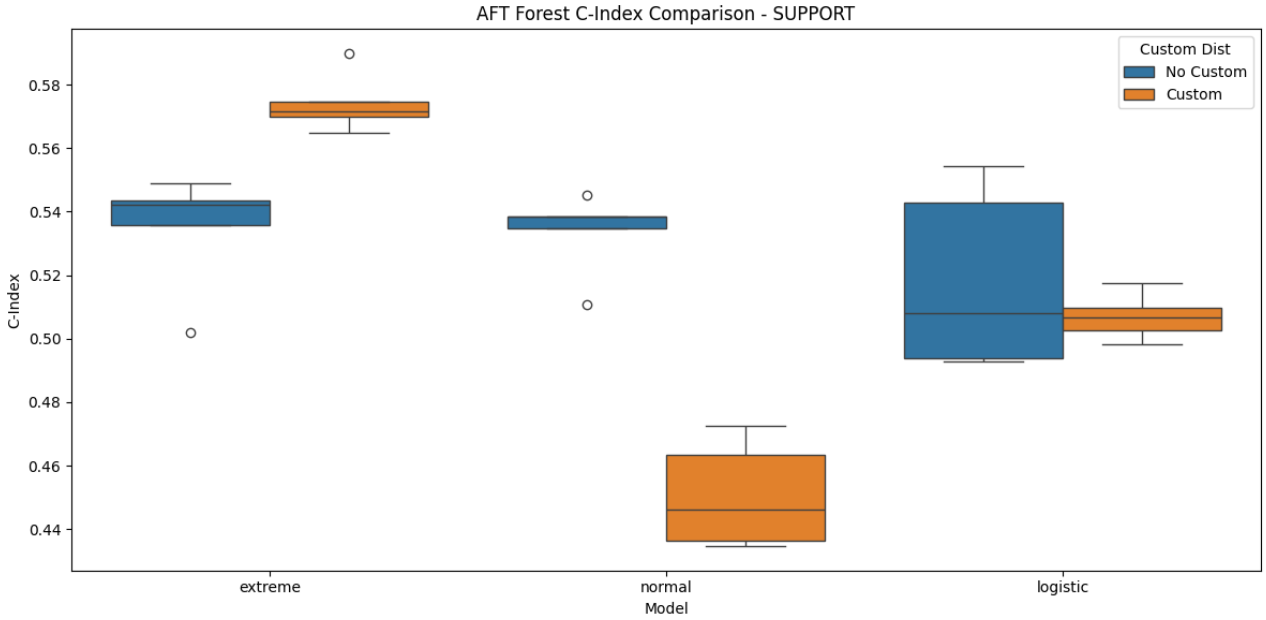

Figure 15: Cross-validation C-Index performance of AFT Forest models on Support with and without custom distributions.

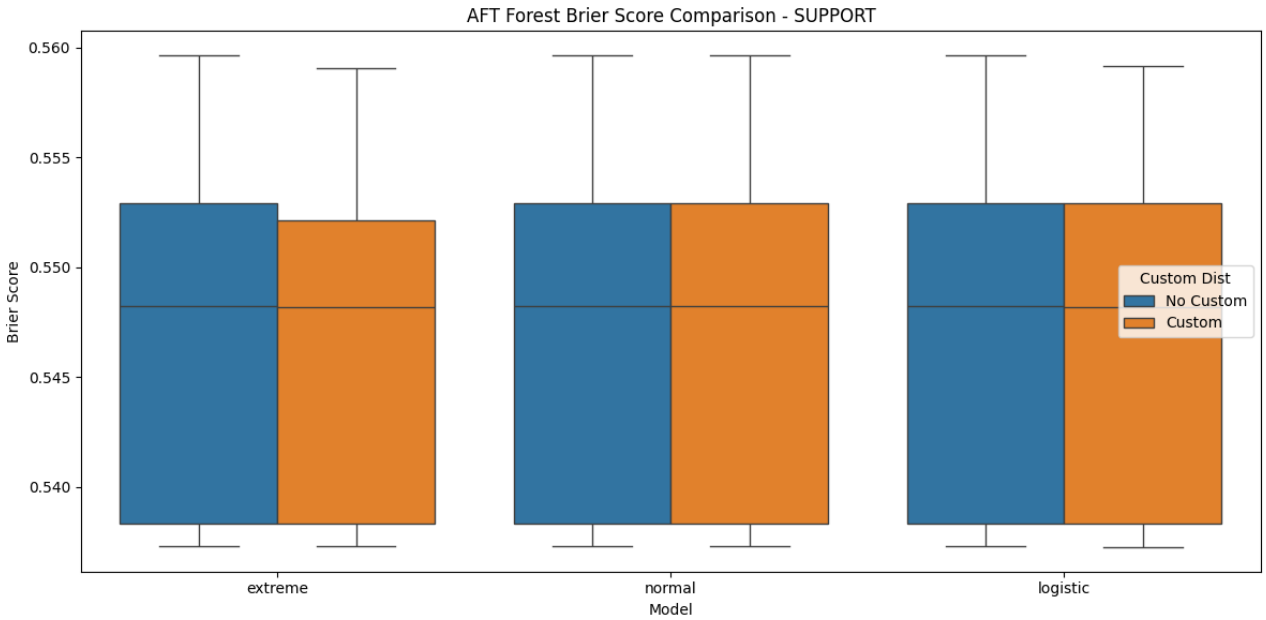

Figure 16: Cross-validation Brier Score performance of AFT Forest models on Support with and without custom distributions.

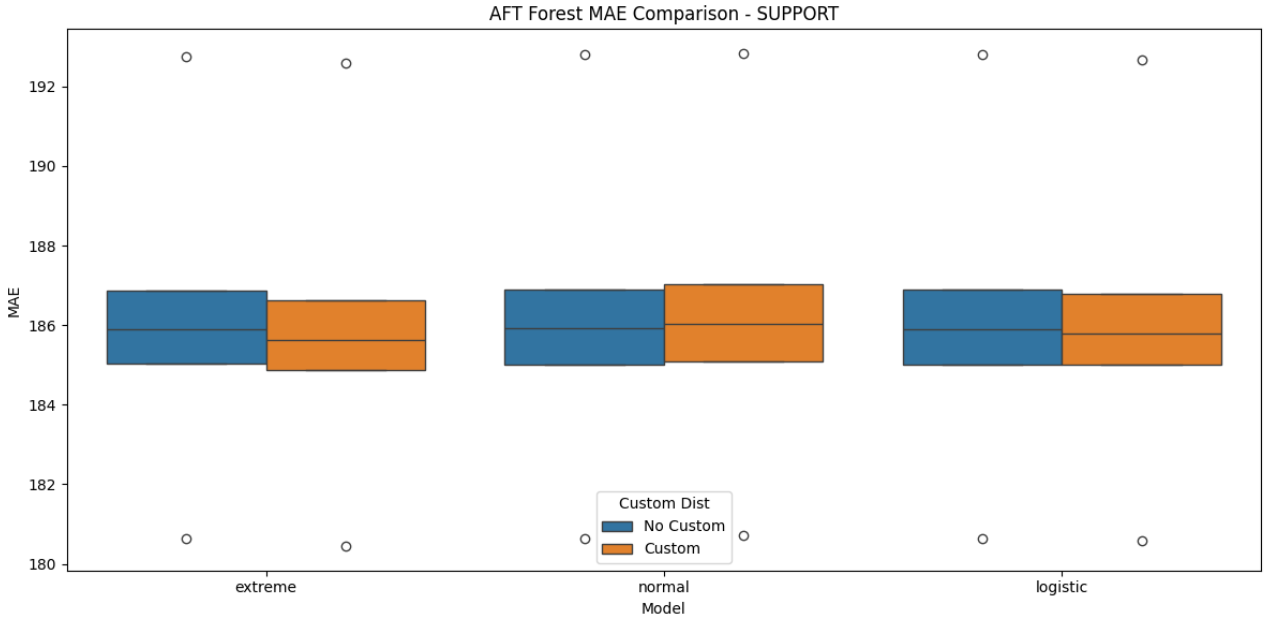

Figure 17: Cross-validation MAE Score performance of AFT Forest models on Support with and without custom distributions.

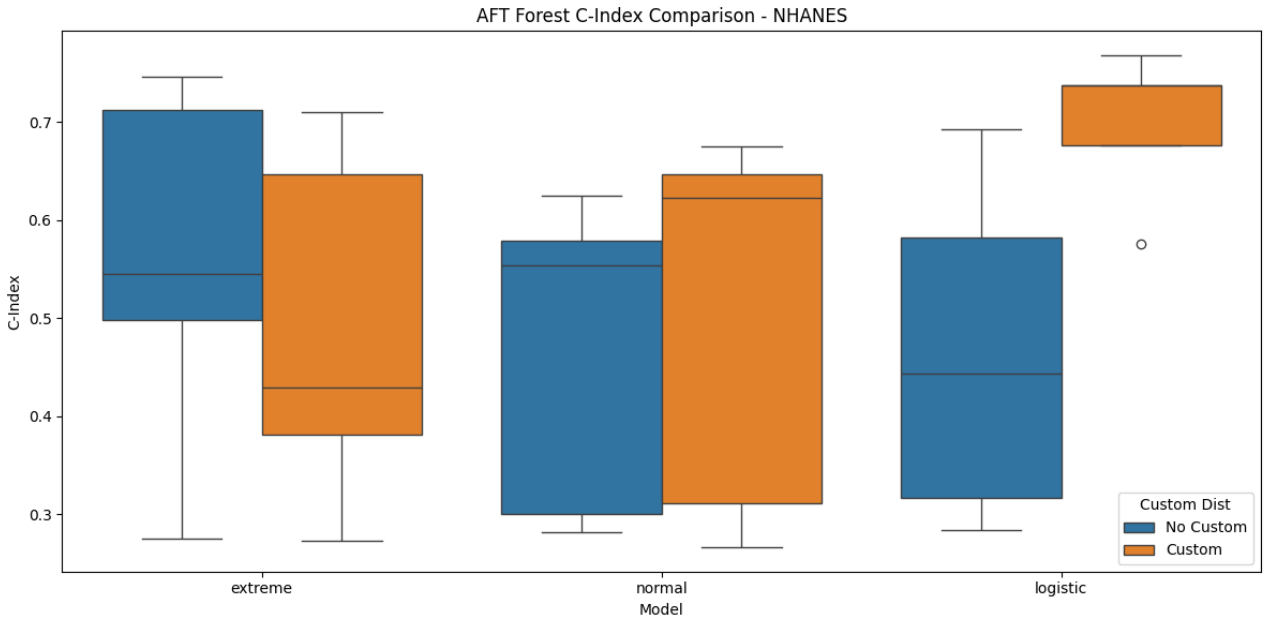

Figure 18: Cross-validation C-Index performance of AFT Forest models on NHANES with and without custom distributions.

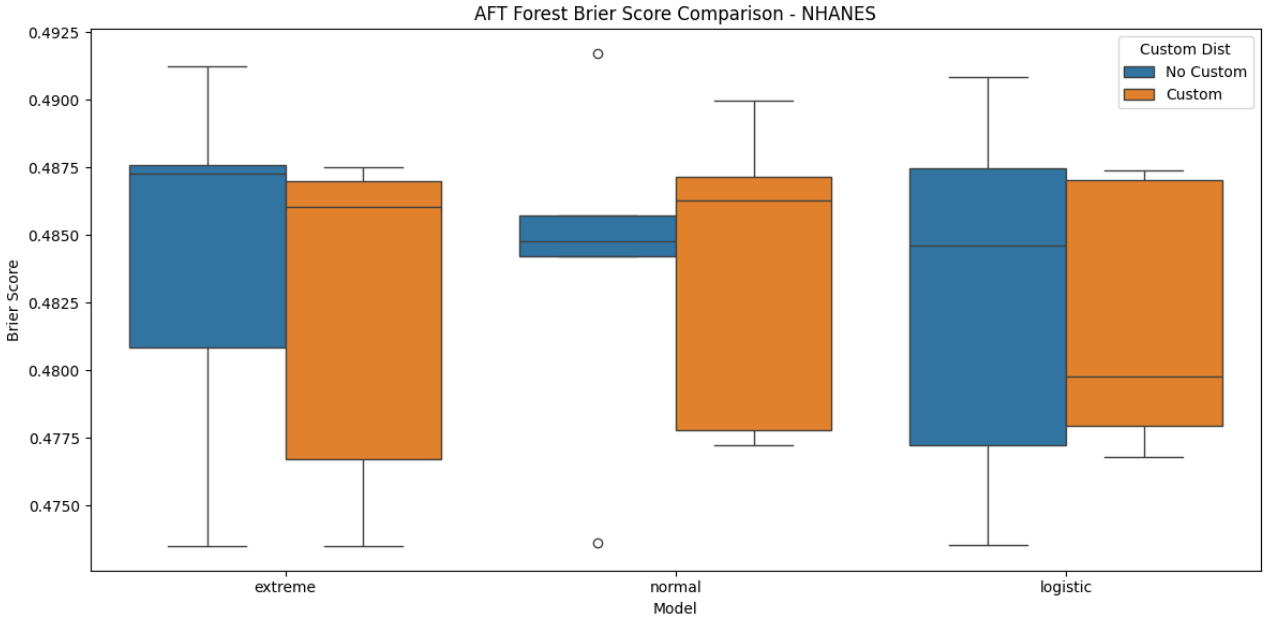

Figure 19: Cross-validation Brier Score performance of AFT Forest models on NHANES with and without custom distributions.

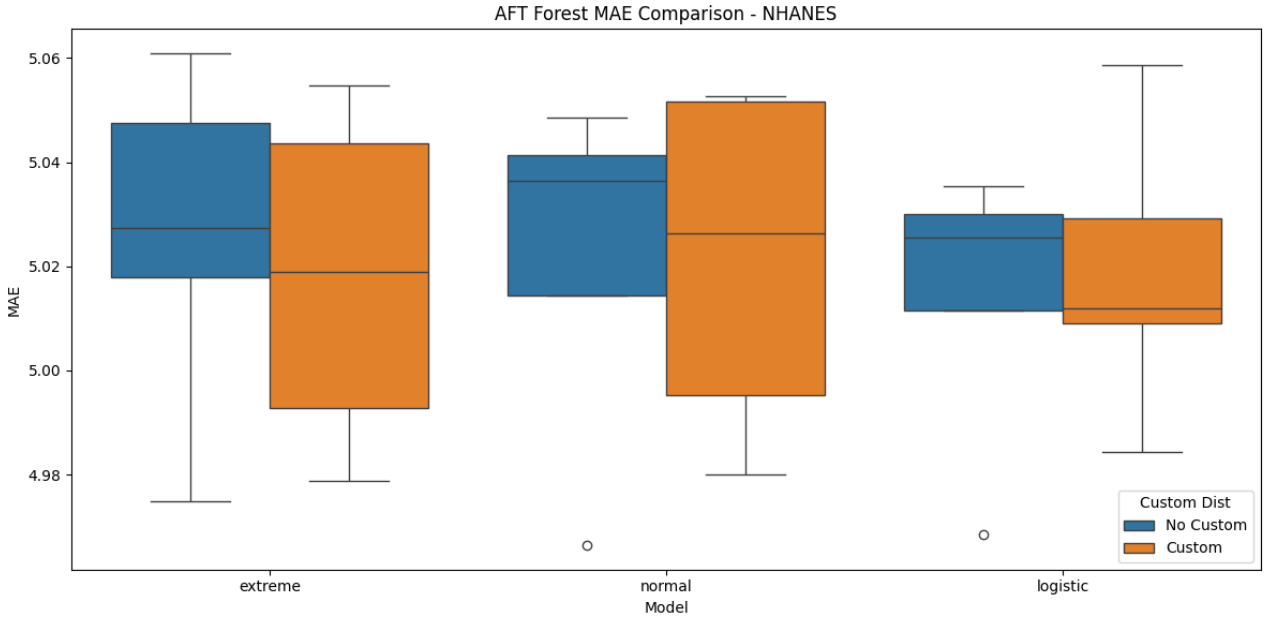

Figure 20: Cross-validation MAE Score performance of AFT Forest models on NHANES with and without custom distributions.

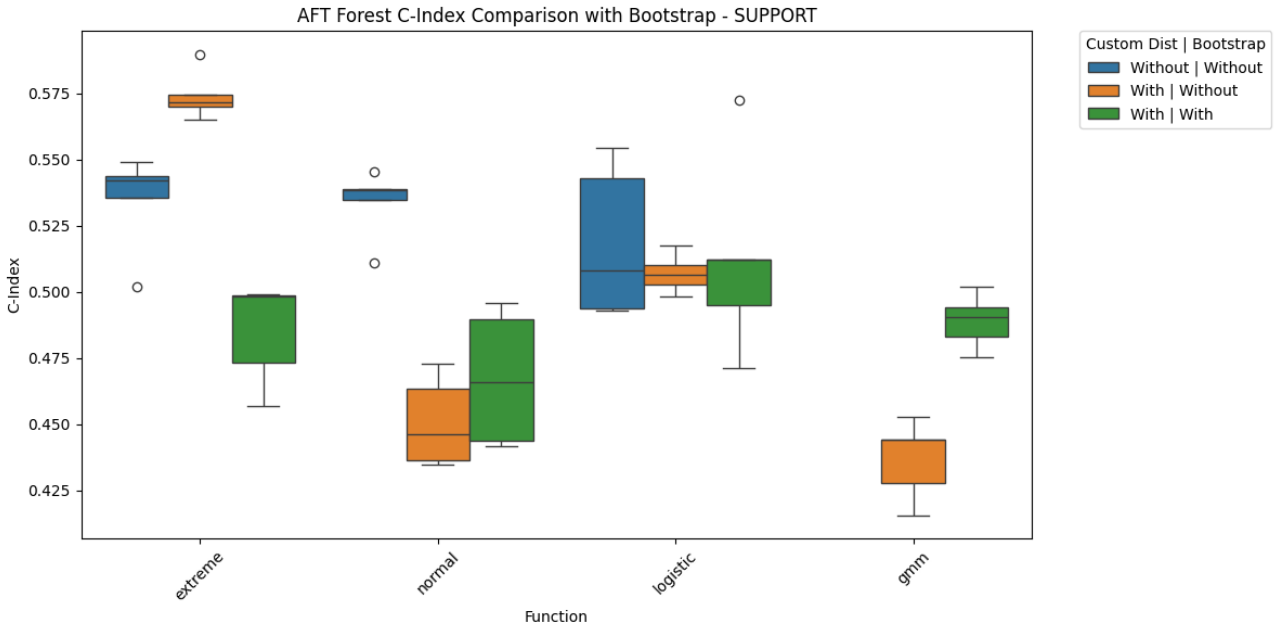

Figure 21: Cross-validation C-Index performance of AFT Forest models on Support with and without custom distributions.

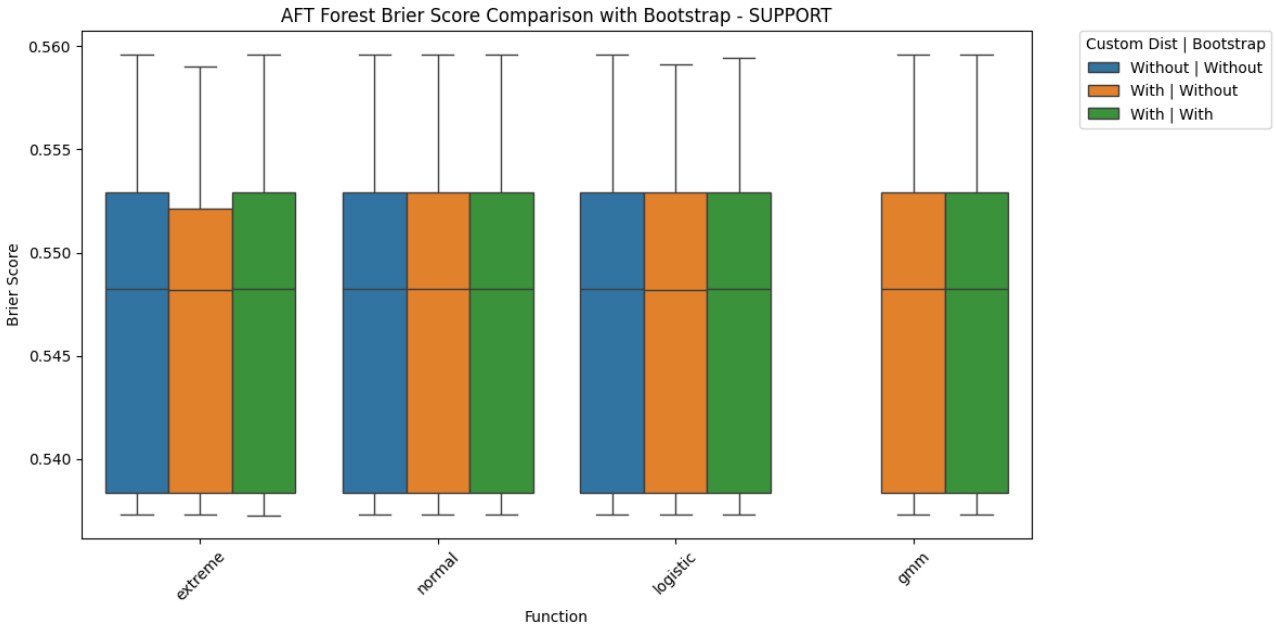

Figure 22: Cross-validation Brier Score performance of AFT Forest models on Support with and without custom distributions.

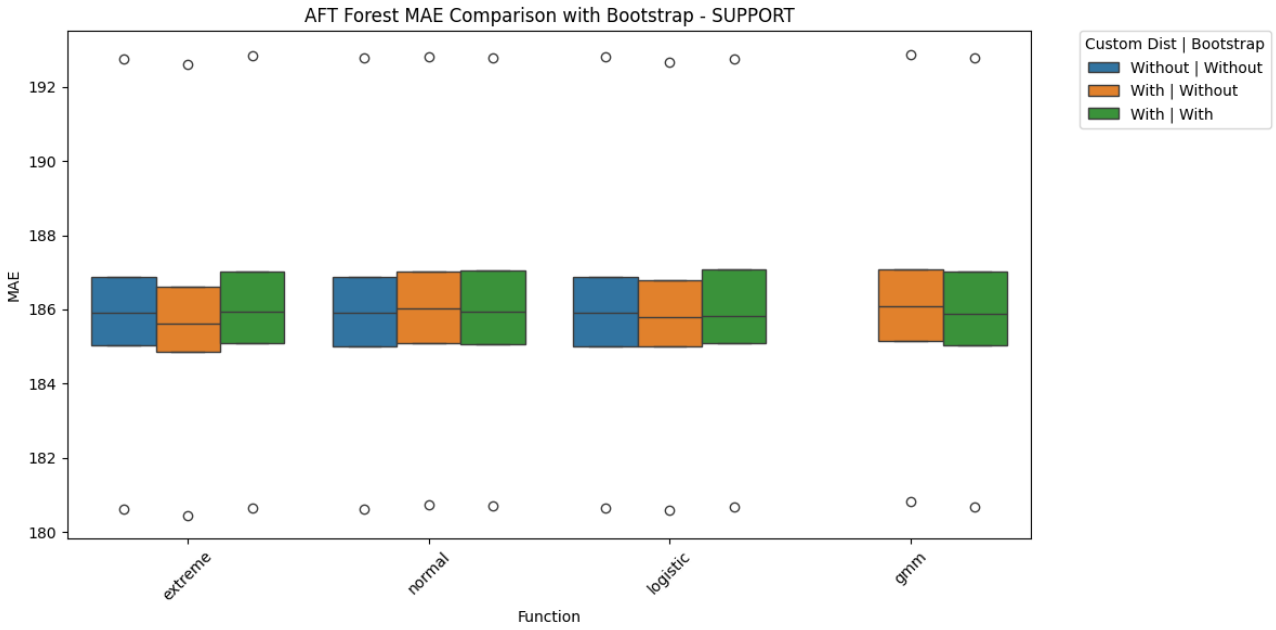

Figure 23: Cross-validation MAE performance of AFT Forest models on Support with and without custom distributions.

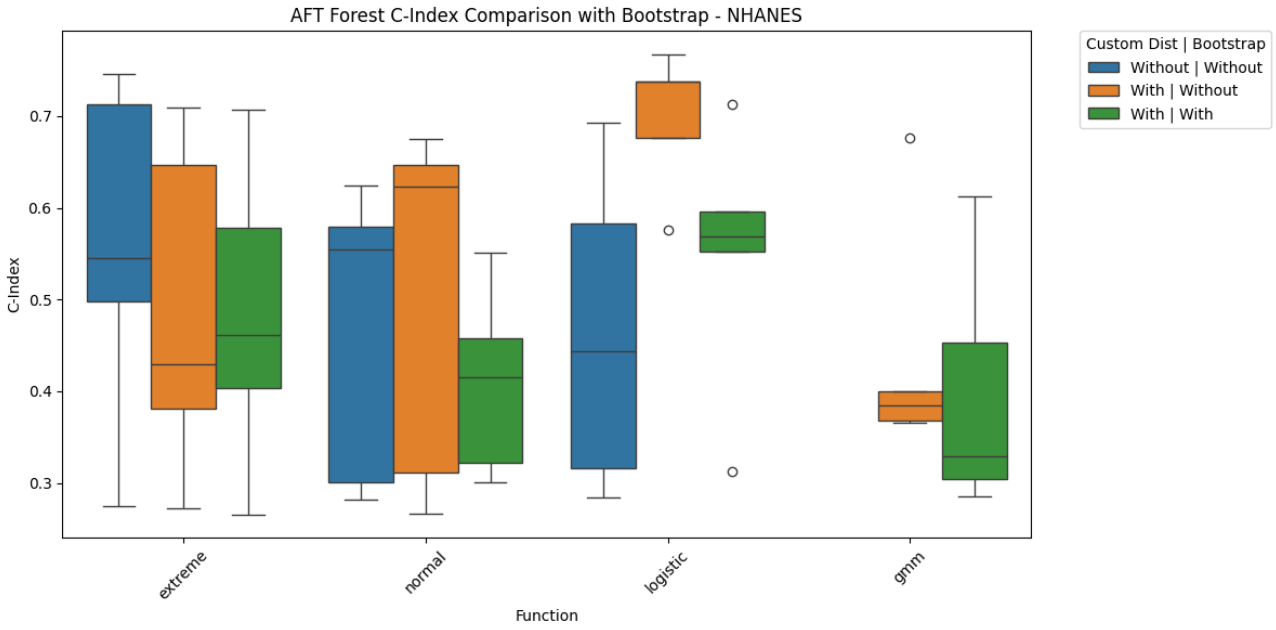

Figure 24: Cross-validation C-Index performance of AFT Forest models on NHANES with and without custom distributions.

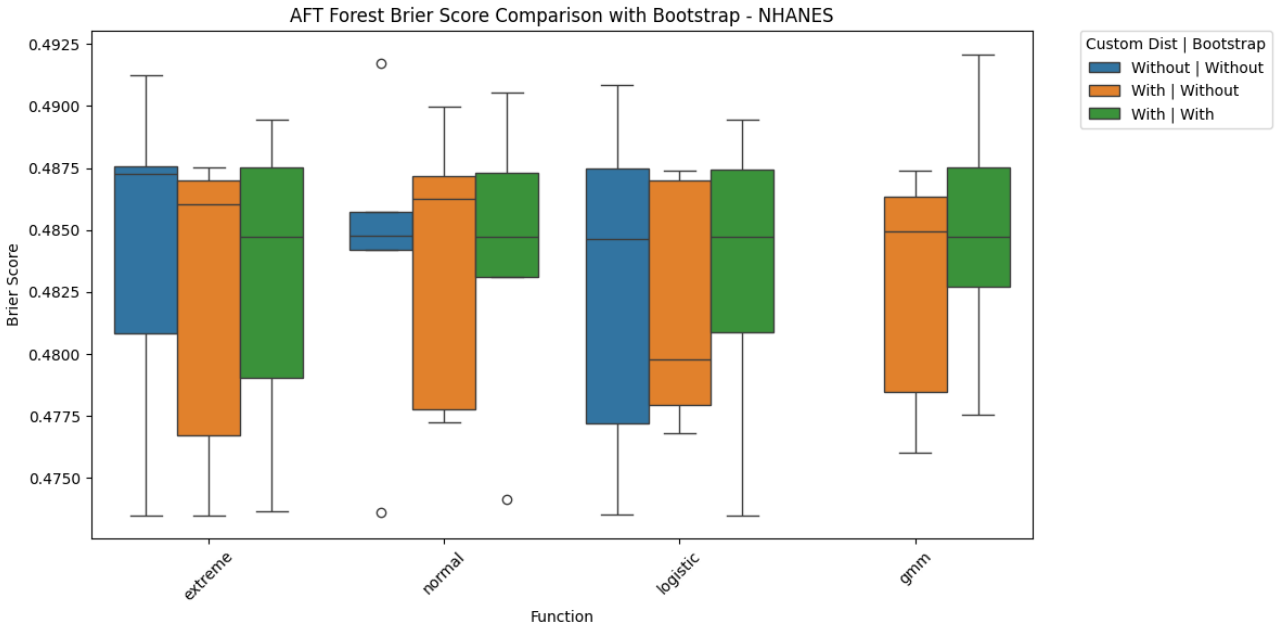

Figure 25: Cross-validation Brier Score performance of AFT Forest models on NHANES with and without custom distributions.

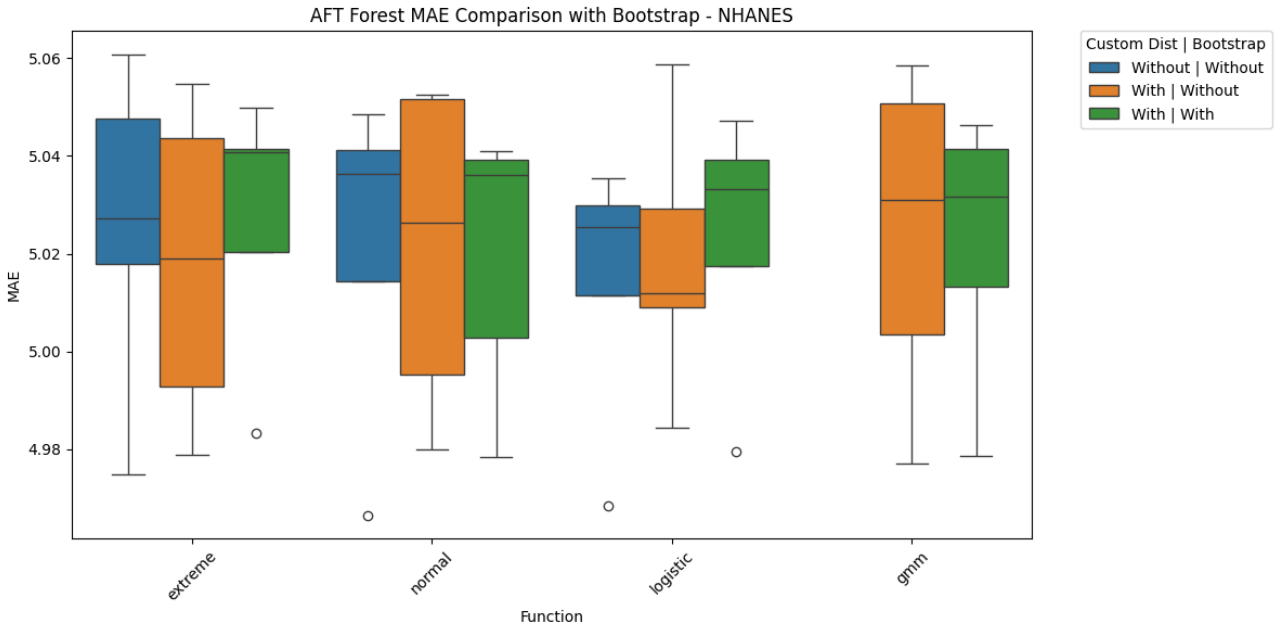

Figure 26: Cross-validation MAE performance of AFT Forest models on NHANES with and without custom distributions.

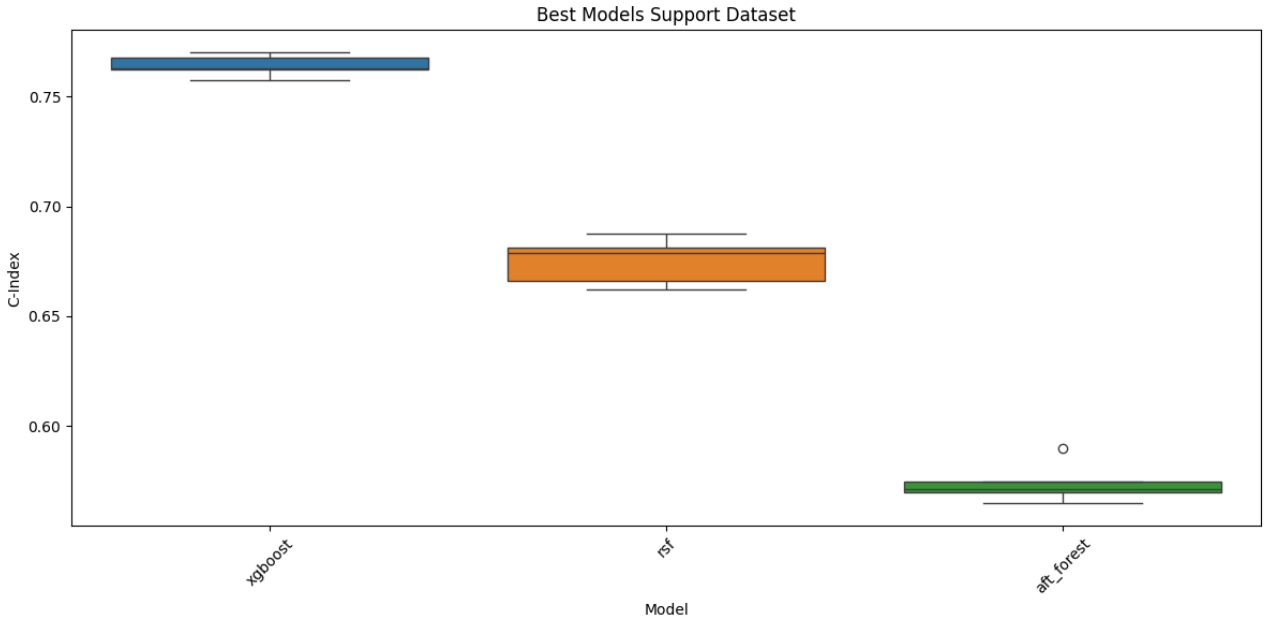

Figure 27: Cross-validation C-index performance of the best models on Support

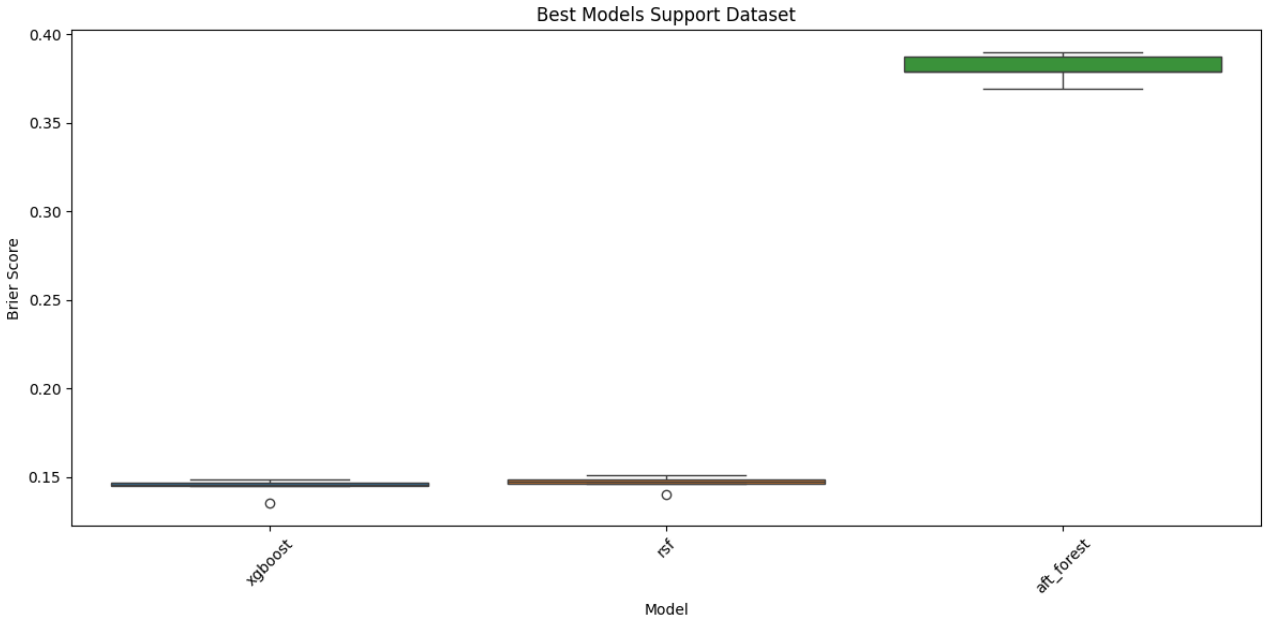

Figure 28: Cross-validation Brier performance of the best models on Support

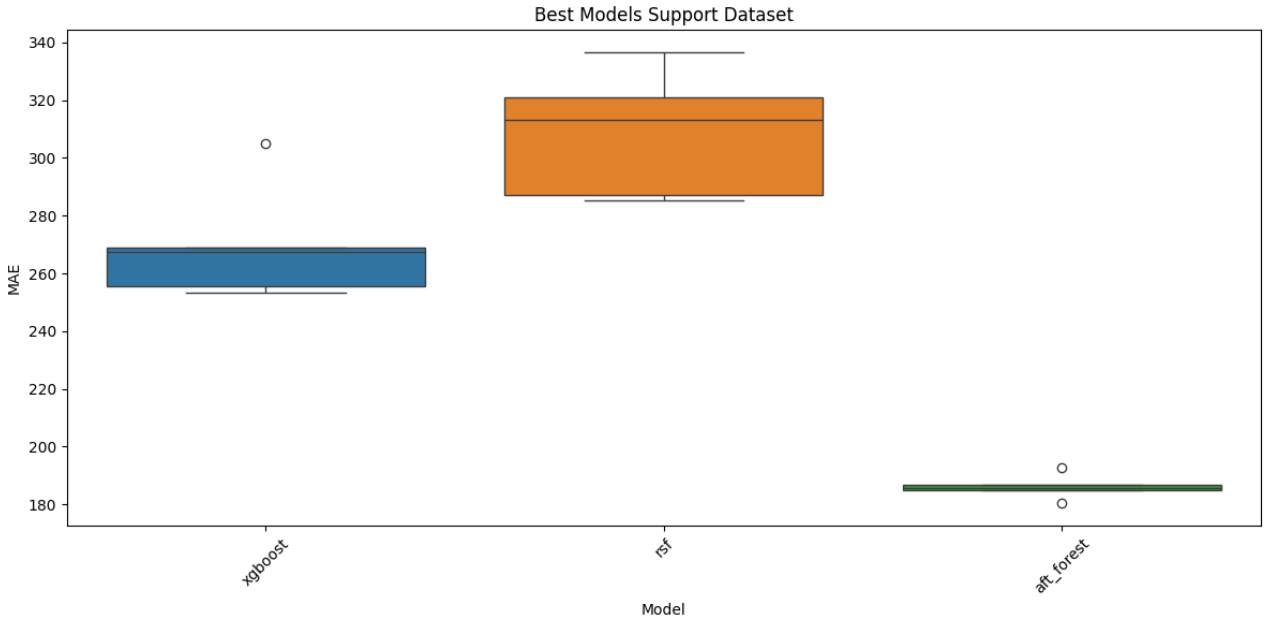

Figure 29: Cross-validation MAE performance of the best models on Support

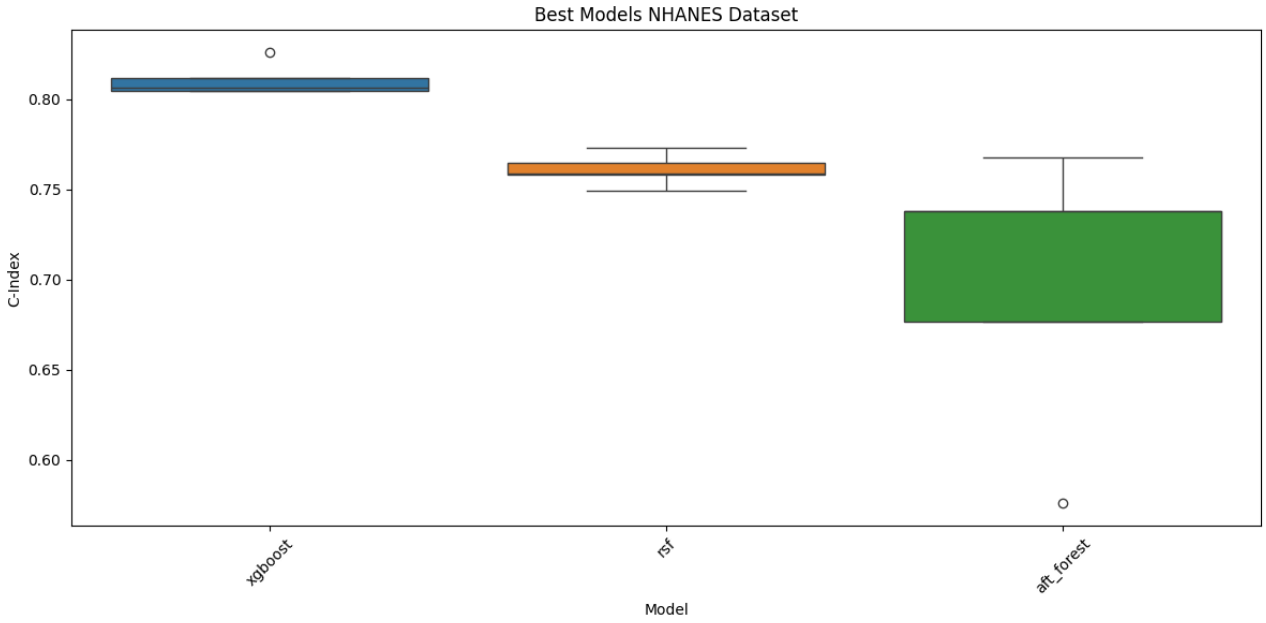

Figure 30: Cross-validation C-Index performance of the best models on NHANES

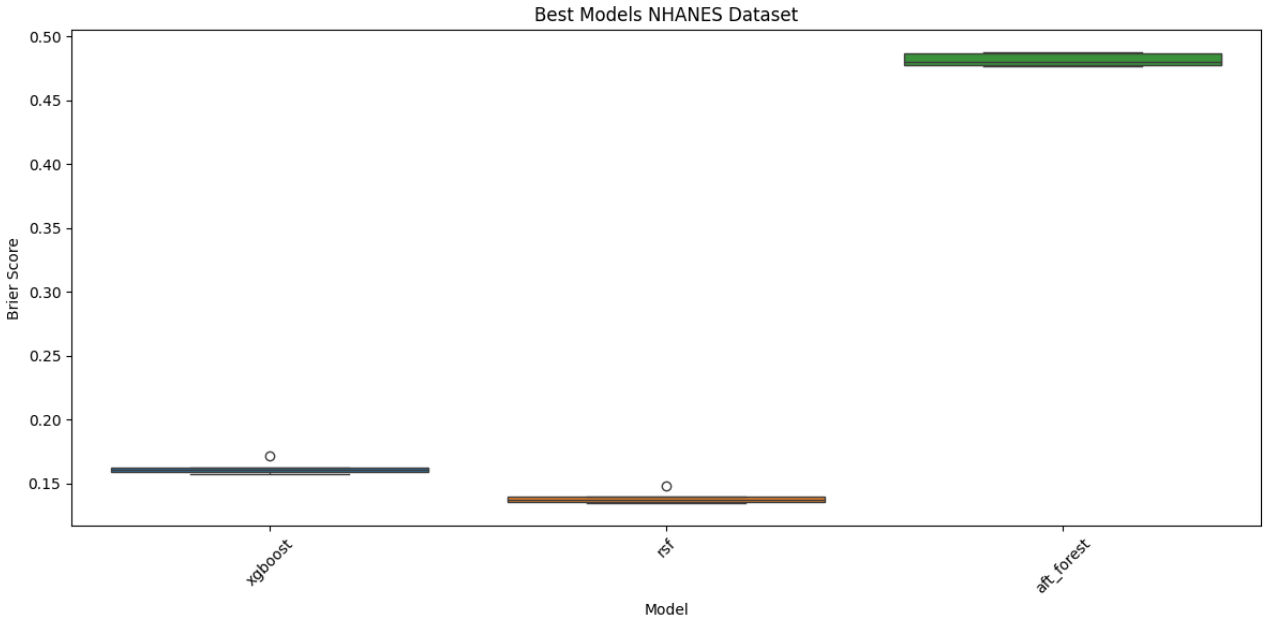

Figure 31: Cross-validation Brier performance of the best models on NHANES

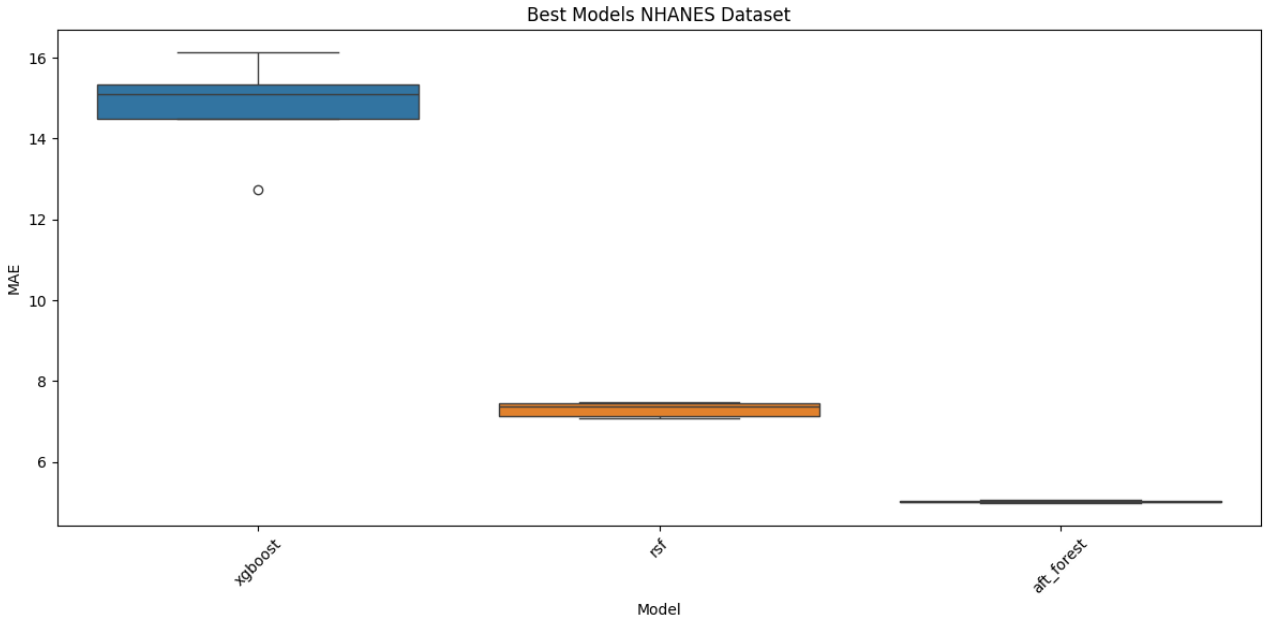

Figure 32: Cross-validation MAE performance of the best models on NHANES