Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

Case-Base Neural Networks: survival analysis with time-varying, higher-order interactions

Jesse Islam[1] | Maxime Turgeon[2] | Robert Sladek[3] | Sahir Bhatnagar[4]

[1]Department of Quantitative life sciences, McGill University
[2]Department of Statistics, University of Manitoba
[3]Department of Human Genetics, McGill University
[4]Department of Biostatistics, McGill University

Correspondence
Email: jesse.islam@mail.mcgill.ca

In survival analysis, covariate interactions are difficult to model without prior knowledge using regression-based methods. Neural network-based survival models have been developed for data-driven interaction modeling, however, these approaches lack implementations for time-varying interactions or a flexible baseline hazard. To address this, we extend the case-base sampling framework and propose Case-Base Neural Networks (CBNN), which introduce time directly into the model, using commonly available neural network components for a simple implementation. First, we case-base sample the survival data, converting the survival time of each participant into discrete moments in time. This allows us to estimate the probability an event occurs at a given moment using a neural network to get the hazard. We compare CBNN to survival methods based on regression and neural networks in two simulations and two real data applications with a fixed neural network architecture. We report metrics for each model over follow-up time using both a proper (Index of Prediction Accuracy) and non-proper (inverse probability censoring weights-adjusted concordance index) scoring rule. A simple simulation shows CBNN outperforming the other neural network methods. The complex simulation highlights the ability of CBNN to model both a complex baseline hazard and time-varying interactions, outperforming all competitors. The first real data application shows CBNN outperforming all neural network competitors, while a second real data application shows competitive performance. CBNN has shown that it is more consistent in prediction performance across time in the simulations and real data applications compared to the competing neural network approaches.

KEYWORDS:
survival analysis, machine learning, case-base, neural network

1 | INTRODUCTION

The widespread use of the Cox proportional hazards model has led to a focus on survival models based on hazard ratios and relative risks rather than on survival curves and absolute risks[1]. In clinical settings, Cox models are frequently prioritized over smooth-in-time accelerated failure time (AFT) models[1], which can estimate absolute risks by modeling the hazard directly through user specification of a baseline hazard distribution. The user may forgo this choice by implementing a flexible set of parameters using splines, as proposed by Royston and Palmar[2]. These regression models also require user-specified interactions, which may limit risk prediction accuracy.

Over longer follow-up, it is challenging to maintain the proportional hazards assumption as the risk associated with each covariate may not be constant with time. Previous studies related to breast cancer have discovered time-varying interactions with covariates of interest, like tumor size[3 4]. Another limitation of regression-based models is the requirement of prior knowledge; the analyst must know potential time-varying interactions and explicitly evaluate them.

Data-driven neural network approaches bypass the need for prior knowledge of interaction terms. DeepSurv is a semi-parametric proportional hazards neural network model that implements the Cox partial log-likelihood as a custom loss function[5]. This results in a stepwise absolute risk curve that cannot accommodate time-varying interactions. Compared to Cox regression, DeepSurv has shown a small improvement in performance using a single layer neural network on the Study to Understand Prognoses Preferences and Risk Treatments (SUPPORT) dataset[6]. Farraggi et al. suggested a modification to the loss function required for a Cox neural network that can model non-proportional hazards[7]. Wang et al. took the concept of extreme learning machines and applied them in a survival context, removing the need for tuning the number of layers and nodes[8]. A Cox-based neural network combined with Local Interpretable Model-Agnostic Explanations (LIME)[9] interprets the association of each Single Nucleotide Polymorphism (SNP)[10] to a phenotype.

DeepHit requires each survival time of interest to be specified in the model and directly estimates survival curves, rather than deriving a hazard function[11]. It assumes an inverse Gaussian distribution as the baseline hazard and its concordance index (c-index) score outperforms DeepSurv[5] on the Molecular Taxonomy of Breast Cancer International Consortium (METABRIC) dataset[12].

Ripley et al. presented seven different distributions and modeling paradigms based on neural networks[13]. They provided the methodology behind parametric log-logistic and log-normal neural network models. Like DeepHit, these models are restricted to one of the seven baseline hazard distributions, reducing the flexibility of the average survival prediction.

Deep Survival Machines (DSM) provides a flexible, parametric survival model using neural networks with a mixture of distributions as the baseline hazard[14]. DSM allows the user to specify a mixture of distributions for a flexible baseline hazard. On both the SUPPORT and METABRIC datasets, DSM outperforms DeepSurv and DeepHit[14]. However, like DeepHit and DeepSurv, DSM cannot model time-varying interactions.

These methods need custom loss and activation functions that are difficult to implement in practice. We note the complexity of these alternative neural network approaches; DeepSurv, DeepHit, and DSM all require custom loss functions[5 11 14], while DSM requires user implementations for distributions beyond Log-Normal or Weibull and a two-phase learning process[14]. Regression-based approaches require prior specification of all interaction terms, which makes it challenging to model covariate effects that change over time. In this article, we propose Case-Base Neural Networks (CBNN) as a method that models time-varying interactions and a flexible baseline hazard using commonly available neural network components. Our approach to modeling the full hazard uses case-base sampling[1], which allows probabilistic models to predict survival outcomes. Following the case-base framework, we can use transformations of time as a feature (covariate) to specify different baseline hazards. For example, by including splines of time as covariates, we can approximate the Royston-Parmar flexible baseline hazard model[2]. An alternative approach to Royston-Parmar is case-base with logistic regression (CBLR), which provides some flexibility in modeling, but still requires user specification of time-varying interactions and baseline hazard, while CBNN can model both without user specification; only a sufficiently deep network is required.

In Section 2, we describe how case-base sampling and neural networks are combined both conceptually and algebraically, along with our hyperparameter choice and software implementation. Then, in Section 3, we describe our metrics and compare the performance of CBNN, DeepSurv, DeepHit, DSM, Cox regression, and case-base using logistic regression on simulated data. Next, Section 4 describes the real-data analysis, while in Section 5, we explore the implications of our results and contextualize them within neural network survival analysis in a single event setting.

2 | CASE-BASE NEURAL NETWORK METHODOLOGY, METRICS, AND SOFTWARE

In this section, we define case-base sampling, which converts the total survival time into discrete person-specific moments (person-moments). Then, we detail how neural networks can be used within this framework, explicitly incorporating time as a feature while adjusting for the sampling bias. Finally, we report on the software versions used. A package is available for use at [GITHUBLINK]. The entire code base to generate the analyses in this paper is available at [GITHUB LINK].

2.1 | Case-base sampling

Case-base sampling is an alternative framework for survival analysis[1]. In case-base sampling, we sample from the continuous survival time of each person in our dataset to create a *base series* of *person-moments*. This *base series* complements the *case series*, which contains all person-moments at which the event of interest occurs.

For each person-moment sampled, let X_i be the corresponding covariate profile $\left(x_{i1}, x_{i2}, ..., x_{ip}\right)$, T_i be the time of the person-moment, and Y_i be the indicator variable for whether the event of interest occurred at time T_i. We estimate the hazard function $h(t \mid X_i)$ using the sampled person-moments. Recall that $h(t \mid X_i)$ is the instantaneous potential of experiencing the event at time t for a given set of covariates X_i, assuming $T_i \geq t$.

Now, let b be the (user-defined) size of the *base series*, and let B be the sum of all follow-up times for the individuals in the study. If we sample the *base series* uniformly, then the hazard function of the sampling process is equal to b/B. Therefore, we have the following equality [a]:

$$\frac{P\left(Y_i = 1 \mid X_i, T_i\right)}{P\left(Y_i = 0 \mid X_i, T_i\right)} = \frac{h\left(T_i \mid X_i\right)}{b/B}. \tag{1}$$

The odds of a person-moment being a part of the *case series* is the ratio of the hazard $h(T_i \mid X_i)$ and the uniform rate b/B. Using (1), we can see how the log-hazard function can be estimated from the log-odds arising from case-base sampling:

$$\log\left(h\left(t \mid X_i\right)\right) = \log\left(\frac{P\left(Y_i = 1 \mid X_i, t\right)}{P\left(Y_i = 0 \mid X_i, t\right)}\right) + \log\left(\frac{b}{B}\right). \tag{2}$$

We may estimate the hazard function if we adjust for the bias introduced when sampling a fraction of the study base B by adding the offset term $\log\left(\frac{b}{B}\right)$ ((2)). Next, we propose using neural networks to model the odds.

2.2 | Neural networks to model the hazard function

After case-base sampling, we pass all features, including time, into any user-defined feed-forward component, to which an offset term (to adjust for bias from case-base sampling) is added, then passed through a sigmoid activation function (Figure 1). We use the sigmoid function as its inverse is the odds, which we can use to calculate the hazard. The general form for the neural network using CBNN is:

$$P\left(Y = 1|X, T\right) = \text{sigmoid}\left(f_\theta(X, T) + \log\left(\frac{B}{b}\right)\right), \tag{3}$$

where T is a random variable representing the event time, X is the random variable for a covariate profile, $f_\theta(X, T)$ represents any feed-forward neural network architecture, $\log\left(\frac{B}{b}\right)$ is bias term set by case-base sampling, θ is the set of parameters learned by the neural network and $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$. By allowing a neural network to approximate a higher-order polynomial of time, the baseline hazard specification is now data-driven, where user-defined hyperparameters such as regularization, number of nodes, and layers control the flexibility of the model. We provide a detailed description of the choices we made in the next sub-section.

[a] We are abusing notation here, conflating hazards with probabilities. For a rigorous treatment, see Saarela & Hanley (2015) section 3[15].

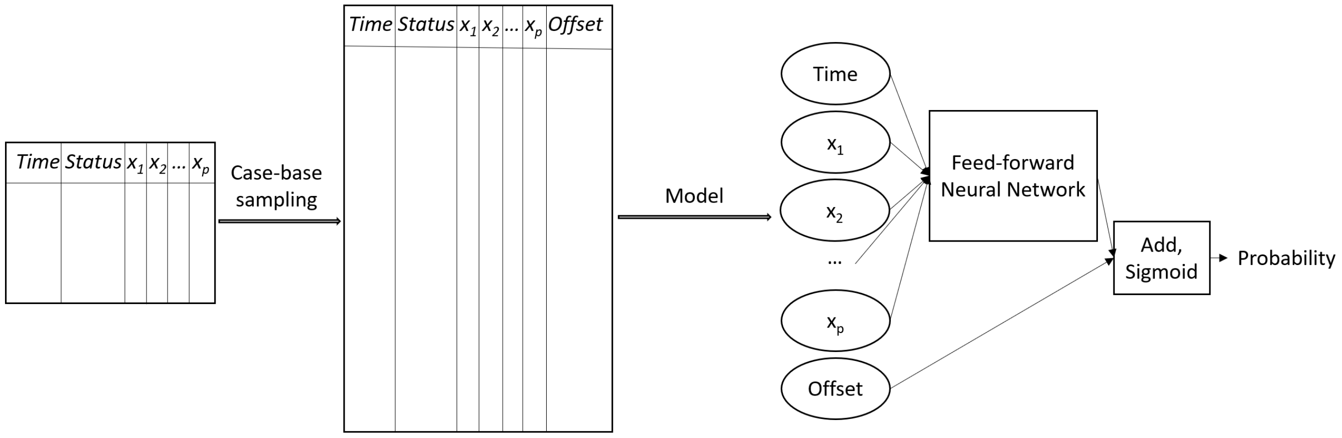

FIGURE 1 Steps involved in CBNN from case-base sampling to the model framework we use for training. The first step is case-base sampling, completed before training begins. Next, we pass this sampled data through a feed-forward neural network. We add the offset and pass that through a sigmoid activation function, whose output is a probability. Once the neural network model completes its training, we can convert the probability output to a hazard, using it for our survival outcomes of interest.

The following derivation shows how our probability estimate is converted to odds:

$$\begin{aligned}
\log\left(h(t \mid X_i)\right) &= \log\left(\frac{\operatorname{sigmoid}\left(f_\theta(X,T) + \log\left(\frac{B}{b}\right)\right)}{1 - \operatorname{sigmoid}\left(f_\theta(X,T) + \log\left(\frac{B}{b}\right)\right)}\right) + \log\left(\frac{b}{B}\right) \\
&= \log\left(\frac{\frac{\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)}{\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)+1}}{1 - \frac{\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)}{\exp\left(f_\theta(X,T)+\log\left(\frac{B}{b}\right)\right)+1}}\right) + \log\left(\frac{b}{B}\right) \\
&= \log\left(\exp\left(f_\theta(X,T) + \log\left(\frac{B}{b}\right)\right)\right) + \log\left(\frac{b}{B}\right) \\
&= f_\theta(X,T) + \log\left(\frac{B}{b}\right) + \log\left(\frac{b}{B}\right) \\
&= f_\theta(X,T).
\end{aligned}$$

We use binary cross-entropy as our loss function[16]:

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(\hat{f}_\theta(x_i,t_i)) + (1-y_i)\cdot\log(1-\hat{f}_\theta(x_i,t_i)),$$

where $\hat{f}_\theta(x_i,t_i)$ is our estimate for a given covariate profile and time, y_i is our target value specifying whether an event occurred, and N represents the number of individuals in our training set.

Backpropagation is used to optimize the parameters in the model with an appropriate minimization algorithm (e.g. Adam, RMSPropagation, stochastic gradient descent)[16]. For our analysis, we use Adam. Note that the size of the *case series* is fixed as the number of events, but we can make the *base series* as large as we want. A ratio of 100:1 *base series* to *case series* is sufficient[1]. We pass our feed-forward neural network through a sigmoid activation function (Figure 1). Finally, we can convert this model output to a hazard. When using our model for predictions, we manually set the offset term to 0 in the new data, as we account for the bias during the fitting process.

Since we are directly modeling the hazard, we can readily estimate the risk function (F) at time t for a covariate profile X, viz.

$$F(t|X) = 1 - \exp\left(- \int_0^t h(u|X)\,\mathrm{d}u \right). \tag{4}$$

We use Riemann's Sum[17] to approximate the integral in (4).

2.3 | Hyperparameter selection

Neural networks provide flexibility when defining the architecture and optimization parameters of our models. These hyperparameter choices can affect the estimated parameters and were chosen during a set of initial simulations to determine if CBNN can learn complex interactions in practice. We set a max number of epochs to be 2000, batch size as 512, learning rate as 10^{-3}, decay as 10^{-7}, patience as 10 epochs, $\{50, 50, 25, 25\}$ nodes in each hidden layer with 50% dropout at each layer, a validation split of 20%, a testing split of 10, a minimum delta loss on the validation set of 10^{-7} over 10 epochs, and Adam[16] as our optimizer. These choices may permit the model to approximate higher-order interactions while preventing over-fitting[18]. We fix a train-validation-test split that allows us to update the weights with a subset of the data (training), assess performance at each epoch (validation) and gauge performance of the final model (test). We select the best weights after training based on validation loss[16].

2.4 | Software implementation

R[19] and python[20] are used to evaluate methods from both languages. We fit the Cox model using the **survival** package[21], the CBLR model using the **casebase** package[22], the DeepSurv model using the **survivalmodels** package[23], the DSM model using **DeepSurvivalMachines**[14] and the DeepHit model using **pycox**[11]. We made the components of CBNN using the **keras**[24] package and the **casebase** package for the sampling step. The **simsurv** package[25] is used to simulate our simulation studies, while **flexsurv**[26] is used to fit a flexible baseline hazard using splines for our complex simulation. We use the implementation of c-ipcw from the python package **sksurv**[27]. The **riskRegression** package[28] is used to get the Brier score and IPA metrics. We changed the **riskRegression** package to be used with any user supplied risk function F. To ensure that both R and Python-based models are running in unison on the same data through our simulations and bootstrap, we use the **reticulate** package[29].

3 | SIMULATION STUDIES

In this section, we use simulated data to evaluate the performance of CBNN and compare our approach with existing methods based on regression (Cox, CBLR) and neural network (DeepHit, DeepSurv, DSM) methods. We specify a linear combination of each covariate as the linear predictor in regression-based approaches (Cox, CBLR), which contrasts with neural network approaches that allow for non-linear interactions. We simulate data under a simple exponential model, and a complex baseline hazard with time-varying interactions, each with 10% random censoring. For both settings, we simulate three covariates:

$$z_1 \sim \text{Bernoulli}(0.5) \qquad z_2 \sim \begin{cases} N(0, 0.5) & \text{if } z_1 = 0 \\ N(10, 0.5) & \text{if } z_1 = 1 \end{cases} \qquad z_3 \sim \begin{cases} N(8, 0.5) & \text{if } z_1 = 0 \\ N(-3, 0.5) & \text{if } z_1 = 1. \end{cases}$$

The DeepHit-specific hyperparameter alpha is set to 0.5 (equal weight between its negative log-likelihood and ranking losses[11]). We change the **DeepSurvivalMachines**[14] package to include dropout and a minimum delta loss during the fitting process. For DSM, we define a mixture model of six Weibull distributions for the baseline hazard. All other hyperparameters are held constant across all neural network methods in both the simulation studies and real data applications.

Besides the methods mentioned above, we include the Optimal model in our comparisons using CBLR. That is, we include the exact functional form of the covariates in a CBLR model (referred to as Optimal for simplicity). We calculate t-based 95% confidence intervals using 100 replications of the simulated data For all analyses, we use 80% for training and 20% for the test set. 20% of the training set is kept for validation at each epoch. We predict risk functions F using (4) for individuals in the test set, which are used to calculate our c-ipcw and IPA scores.

3.1 | Performance metrics

We use two metrics to assess the performance of the different methods of interest: 1) index of prediction accuracy (IPA)[30] and 2) inverse probability censoring weights-adjusted concordance index (c-ipcw)[31], which we define below. Both these time-dependent metrics provide transparency as to when in follow-up time each model may perform better than the others.

3.1.1 | Index of prediction accuracy (IPA)

The IPA is a function of the Brier score $(BS(t))$[32], which is defined as

$$BS(t) = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\left(\widehat{F}(X_i,t)-1\right)^2 \cdot I_{T_i\le t,\delta_i=1}}{\widehat{G}(T_i)} + \frac{\left(\widehat{F}(X_i,t)\right)^2 \cdot I_{T_i>t}}{\widehat{G}(t)}\right), \tag{5}$$

where $\delta_i = 1$ shows individuals who have experienced the event, N represents the number of samples in our dataset over which we calculate $BS(t)$, $\widehat{G}(t) = P[C > t]$ is a non-parametric estimate of the censoring distribution, C is censoring time and T_i is an individual's survival or censoring time. The Brier score provides a score that accounts for the information loss because of censoring. There are three categories of individuals that may appear within the dataset once we fix our t of interest. Individuals who experienced the event before t are present in the first term of the equation. The second term of the equation includes individuals who experience the event or are censored after t. Those censored before t are the third category of people. The inverse probability censoring weights (IPCW) adjustment $(G(\cdot))$ is to account for these category three individuals whose information is missing. The IPA score as a function of time is given by

$$\text{IPA}(t) = 1 - \frac{BS_{model}(t)}{BS_{null}(t)},$$

where $BS_{model}(t)$ represents the Brier score over time t for the model of interest, and $BS_{null}(t)$ represents the Brier score if we use an unadjusted Kaplan-Meier curve as the prediction for all observations[30]. Note that IPA has an upper bound of one, where positive values show an increase in performance over the null model and negative values show that the null model performs better. These scores demonstrate how performance changes over follow-up time.

A potential artifact of IPA is that the score is unstable at earlier and later survival times. This is because of a near equivalent Brier score between each model and the null model. At small values, a difference of 0.1 creates a more significant fold change than at larger values. As the Brier score is potentially small at the start and at the end of their respective curves, we may see instability of the IPA score at those ends.

3.1.2 | Inverse probability censoring weights-adjusted concordance index

The c-ipcw is a non-proper, rank-based metric that does not depend on the censoring times in the test data[31]. The c-ipcw is given by

$$c - ipcw(t) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\delta_i\left\{\widehat{G}(T_i)\right\}^{-2} I(T_i < T_j, T_i < t) I\left(\widehat{F}(t|X_i) > \widehat{F}(t|X_j)\right)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\delta_i\left\{\widehat{G}(T_i)\right\}^{-2} I(T_i < T_j, T_i < t)}, \tag{6}$$

where the follow-up period of interest is (0,t), $I(\cdot)$ is an indicator function, $\widehat{F}(X_i,t)$ is the risk function estimated for everyone in the study at time t and c-ipcw can compare the performance of different models, where a higher score is better. Note that the c-ipcw may produce misleading performance, as it ranks based on survival times, not event status[33]. This metric is considered an unbiased population concordance measure because of the IPCW adjustment[31].

in figure 2 i changed C_{IPCW} to c-ipcw, as capital C is used for censoring. in the plots, would it make sense to leave it as a small letter?

3.2 | Simple simulation: constant baseline hazard

We simulate data from a simple model that primarily depends on a constant baseline hazard:

$$h(t \mid X_i) = \lambda \cdot e^{\beta_1 z_1 + \beta_2 z_2 + \beta_3 z_3},$$

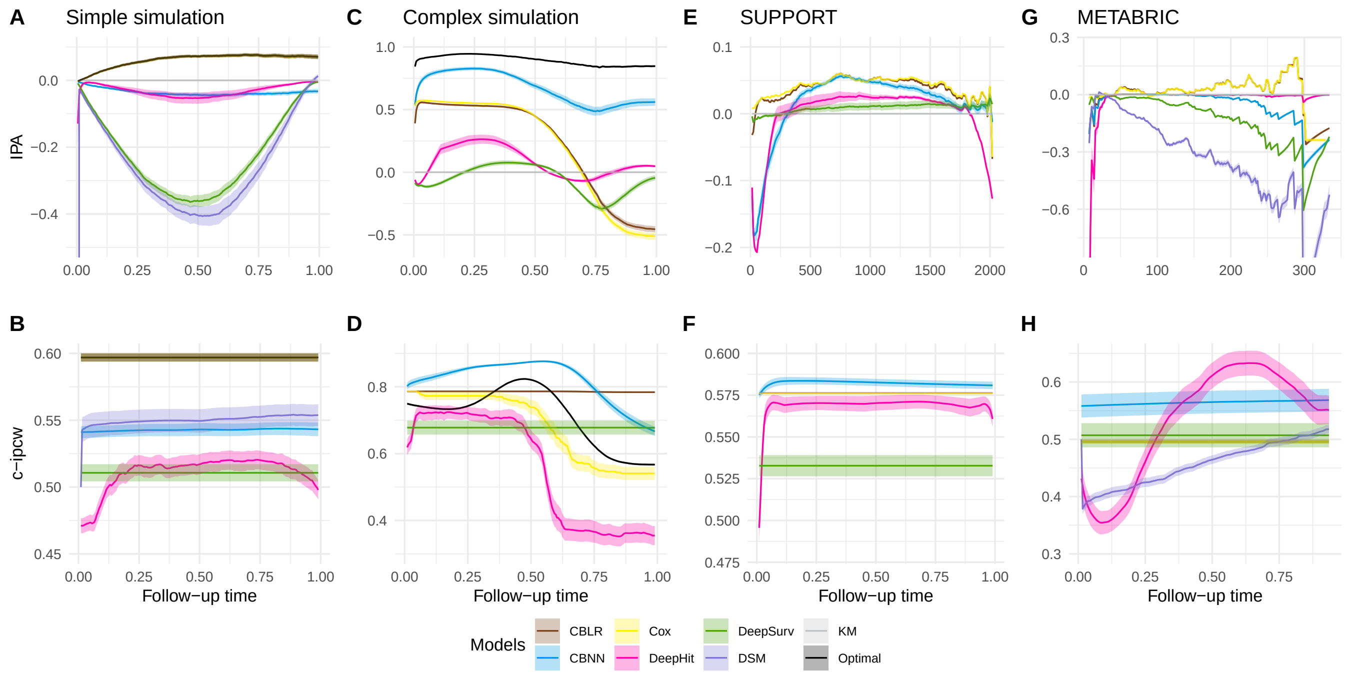

FIGURE 2 Summarizes the simple simulation (A, B), complex simulation (C, D), SUPPORT case study (E, F) and METABRIC case study (G, H) results. The first row shows the IPA score for each model in each study over follow-up time. Negative values mean our model performs worse than the null model and positive values mean the model performs better. The second row shows the c-ipcw for each model in each study over follow-up time. A score of 1 is the maximum performance for either metric. Each model-specific metric in each study shows a 95-percent confidence interval over 100 iterations. The models of interest are case-base with logistic regression (CBLR), Case-Base Neural Networks (CBNN), Cox, DeepHit, DeepSurv, Deep Survival Machines (DSM), Optimal (a CBLR model with the exact interaction terms and baseline hazard specified) and Kaplan-Meier (to serve as a baseline, predicting the average for all individuals).

with minimal covariate effects given by $\beta_1 = 0.1, \beta_2 = 0.1, \beta_3 = 0.1$ and $\lambda = 1.0$. Once we simulate survival times, we introduce 10% random censoring.

3.2.1 | Performance comparison in simple simulation

Figure 2 A, B and Table 1 A show the results for the simple simulation. The regression-based methods (CBLR, Cox, Optimal) outperform the neural network ones in the simple simulation setting. Among the neural network approaches, CBNN outperforms all other methods in terms of both IPA and c-ipcw (Figure 2 A, B). Specifically, we see CBNN is consistent across time with smaller confidence intervals compared to DeepHit, DeepSurv and DSM.

In this simple setting, the regression models are much closer to the Optimal model, while the neural network models perform worse than the null model. The wide confidence bands in c-ipcw suggest the neural network models may be over-parameterized.

3.3 | Complex simulation: flexible baseline hazard, time-varying interactions

This simulation demonstrates performance with the presence of a complex baseline hazard and a time-varying interaction. Originally used to demonstrate the spline-based hazard model proposed by Royston and Parmar[2], the breast cancer dataset provides a complex hazard from which we simulate, available in the **flexsurv** R package[26]. To increase the complexity of our data-generating mechanism for this simulation, we design the model as follows:

$$h(t \mid X_i) = \sum_{i=1}^{5}(\gamma_i \cdot \psi_i) + \beta_1(z_1) + \beta_2(z_2) + \beta_3(z_3) + \tau_1(z_1 \cdot z_2 \cdot t) + \tau_2(z_1 \cdot z_3) + \tau_3(z_2 \cdot z_3),$$

TABLE 1 Four tables representing performance at certain follow-up times for the simple simulation, complex simulation, SUPPORT and METABRIC. Each table shows performance for each method in each study at 25%, 50%. 75% and 100% of follow-up time. The bold elements show the best model for each study, at each follow-up time of interest. These tables are included to provide exact measures at certain intervals. The models of interest are: Cox, case-base with logistic regression (CBLR), DeepSurv, DeepHit, Case-Base Neural Networks (CBNN), Optimal (a CBLR model with the exact interaction terms and baseline hazard specified) and Deep Survival Machines (DSM).

A: Simple	IPA				c-ipcw			
Method	25	50	75	100	25	50	75	100
Cox	**0.05 (0.05,0.06)**	**0.07 (0.07,0.08)**	**0.08 (0.07,0.08)**	**0.07 (0.06,0.08)**	**0.6 (0.59,0.6)**	**0.6 (0.59,0.6)**	**0.6 (0.59,0.6)**	**0.6 (0.59,0.6)**
CBLR	**0.05 (0.05,0.06)**	**0.07 (0.07,0.08)**	**0.08 (0.07,0.08)**	**0.07 (0.06,0.08)**	**0.6 (0.59,0.6)**	**0.6 (0.59,0.6)**	**0.6 (0.59,0.6)**	**0.6 (0.59,0.6)**
DeepSurv	-0.26 (-0.27,-0.25)	-0.36 (-0.38,-0.34)	-0.22 (-0.24,-0.21)	0 (-0.01,0)	0.51 (0.5,0.52)	0.51 (0.5,0.52)	0.51 (0.5,0.52)	0.51 (0.5,0.52)
DeepHit	-0.03 (-0.04,-0.02)	-0.05 (-0.07,-0.04)	-0.03 (-0.04,-0.02)	-0.01 (-0.01,0)	0.52 (0.51,0.52)	0.52 (0.51,0.52)	0.52 (0.51,0.53)	0.5 (0.49,0.5)
CBNN	-0.03 (-0.04,-0.03)	-0.04 (-0.05,-0.04)	-0.04 (-0.05,-0.03)	-0.03 (-0.04,-0.03)	0.54 (0.54,0.55)	0.54 (0.54,0.55)	0.54 (0.54,0.55)	0.54 (0.54,0.55)
Optimal	**0.05 (0.05,0.06)**	**0.07 (0.07,0.08)**	**0.08 (0.07,0.08)**	**0.07 (0.06,0.08)**	**0.6 (0.59,0.6)**	**0.6 (0.59,0.6)**	**0.6 (0.59,0.6)**	**0.6 (0.59,0.6)**
DSM	-0.28 (-0.3,-0.27)	-0.4 (-0.43,-0.37)	-0.28 (-0.3,-0.25)	0.01 (0.01,0.02)	0.55 (0.54,0.56)	0.55 (0.54,0.56)	0.55 (0.54,0.56)	0.55 (0.55,0.56)

B: Complex	IPA				c-ipcw			
Method	25	50	75	100	25	50	75	100
Cox	0.55 (0.54,0.56)	0.45 (0.44,0.46)	-0.19 (-0.21,-0.17)	-0.51 (-0.54,-0.48)	0.77 (0.76,0.79)	0.74 (0.72,0.76)	0.56 (0.54,0.58)	0.54 (0.52,0.56)
CBLR	0.53 (0.52,0.54)	0.45 (0.44,0.46)	-0.15 (-0.16,-0.13)	-0.46 (-0.48,-0.43)	0.79 (0.78,0.79)	0.79 (0.78,0.79)	0.79 (0.78,0.79)	**0.78 (0.78,0.79)**
DeepSurv	0.03 (0.01,0.05)	0.06 (0.05,0.07)	-0.27 (-0.29,-0.25)	-0.04 (-0.07,-0.02)	0.68 (0.66,0.7)	0.68 (0.66,0.7)	0.68 (0.66,0.7)	0.68 (0.66,0.7)
DeepHit	0.26 (0.22,0.3)	0.07 (0.06,0.08)	-0.06 (-0.08,-0.04)	0.05 (0.04,0.06)	0.72 (0.7,0.74)	0.65 (0.62,0.67)	0.37 (0.33,0.4)	0.35 (0.33,0.38)
CBNN	0.83 (0.81,0.84)	0.69 (0.67,0.71)	0.49 (0.45,0.52)	0.56 (0.53,0.59)	**0.86 (0.85,0.86)**	**0.87 (0.87,0.88)**	**0.8 (0.79,0.81)**	0.67 (0.65,0.68)
Optimal	**0.94 (0.94,0.95)**	**0.9 (0.9,0.91)**	**0.84 (0.84,0.85)**	**0.85 (0.84,0.85)**	0.74 (0.74,0.75)	0.82 (0.82,0.82)	0.63 (0.63,0.63)	0.57 (0.56,0.57)

C: SUPPORT	IPA				c-ipcw			
Method	25	50	75	100	25	50	75	100
Cox	**0.04 (0.04,0.04)**	**0.05 (0.05,0.05)**	**0.04 (0.04,0.04)**	-0.06 (-0.07,-0.06)	**0.58 (0.58,0.58)**	**0.58 (0.58,0.58)**	**0.58 (0.58,0.58)**	**0.58 (0.58,0.58)**
CBLR	**0.04 (0.04,0.04)**	**0.05 (0.05,0.05)**	**0.04 (0.04,0.04)**	-0.07 (-0.07,-0.06)	**0.58 (0.58,0.58)**	**0.58 (0.58,0.58)**	**0.58 (0.58,0.58)**	**0.58 (0.58,0.58)**
DeepSurv	0.01 (0,0.01)	0.01 (0.01,0.02)	0.01 (0.01,0.02)	**0.01 (0.01,0.02)**	0.53 (0.53,0.54)	0.53 (0.53,0.54)	0.53 (0.53,0.54)	0.53 (0.53,0.54)
DeepHit	0.02 (0.01,0.03)	0.02 (0.02,0.03)	0.02 (0.02,0.02)	-0.13 (-0.13,-0.12)	0.57 (0.57,0.57)	0.57 (0.57,0.57)	0.57 (0.57,0.57)	0.56 (0.56,0.57)
CBNN	0.03 (0.03,0.04)	**0.05 (0.04,0.05)**	0.03 (0.02,0.03)	-0.01 (-0.02,0)	**0.58 (0.58,0.59)**	**0.58 (0.58,0.58)**	**0.58 (0.58,0.58)**	**0.58 (0.58,0.58)**

D: METABRIC	IPA				c-ipcw			
Method	25	50	75	100	25	50	75	100
Cox	0.00 (-0.01,0.00)	**0.03 (0.03,0.04)**	**0.06 (0.05,0.06)**	-0.23 (-0.25,-0.23)	0.5 (0.49,0.5)	0.5 (0.49,0.5)	0.5 (0.49,0.5)	0.5 (0.49,0.5)
CBLR	**0.00 (0.00,0.00)**	**0.03 (0.03,0.03)**	**0.06 (0.05,0.06)**	-0.18 (-0.18,-0.17)	0.5 (0.49,0.5)	0.5 (0.49,0.5)	0.5 (0.49,0.5)	0.5 (0.49,0.5)
DeepSurv	-0.02 (-0.02,-0.01)	-0.08 (-0.08,-0.08)	-0.26 (-0.27,-0.25)	-0.22 (-0.23,-0.21)	0.51 (0.49,0.53)	0.51 (0.49,0.53)	0.51 (0.49,0.53)	0.51 (0.49,0.53)
DeepHit	**0.00 (0.00,0.00)**	0.00 (0.00,0.00)	-0.01 (-0.01,0.00)	**0.00 (0.00,0.00)**	0.45 (0.43,0.47)	**0.6 (0.57,0.62)**	**0.63 (0.6,0.65)**	0.55 (0.53,0.58)
CBNN	**0.00 (0.00,0.00)**	-0.01 (-0.01,0.00)	-0.09 (-0.1,-0.09)	-0.24 (-0.25,-0.23)	**0.56 (0.54,0.58)**	0.56 (0.54,0.58)	0.57 (0.55,0.59)	**0.57 (0.55,0.59)**
DSM	-0.14 (-0.14,-0.13)	-0.31 (-0.32,-0.29)	-0.52 (-0.55,-0.49)	-0.52 (-0.57,-0.48)	0.42 (0.42,0.43)	0.46 (0.45,0.47)	0.49 (0.48,0.5)	0.52 (0.51,0.53)

where $\gamma_1 = 3.9, \gamma_2 = 3, \gamma_3 = -0.43, \gamma_4 = 1.33, \gamma_5 = -0.86, \beta_1 = 1, \beta_2 = 1, \beta_3 = 1, \tau_1 = 10, \tau_2 = 2, \tau_3 = 2$ and ψ are basis splines. To generate the *gamma* coefficients, let $g\colon \mathbb{R} \to \mathbb{R}$ be a smoothing method for variable t by a projection on to a set of basis functions on splines: i used gt as the function for splines. Specifically, natural cubic spline function of log time https://www.rdocumentation.org/packages/flexsurv/versions/2.1/topics/flexsurvspline . therefore, should i use g(log(t)) and and replace "smoothing method" with natural cubic spline? SAHIR: will fix any issues in this section in the final version.

$$g(t) = \sum_{l=1}^{m} \psi_l(t)\gamma_l. \tag{7}$$

An initial model with three knots ($m = 5$) is fit using the *flexsurvspline* function with an intercept-only model on the data, which returns the coefficients (γ) and basis of time (ψ) we use in our model. Note that we fix these values for the analysis. The β coefficients represent direct effects, τ_2 and τ_3 represent interactions and τ_1 is a time-varying interaction.

3.3.1 | Performance comparison in complex simulation

Figure 2 C, D and Table 1 B show the performance over time on a test set in the complex simulation. Apart from the Optimal regression model, CBNN outperforms the competing models when examining IPA, and up to the 75-percentile of follow-up time for c-ipcw. We expected the Optimal model to perform best in both metrics. However, this was not the case for c-ipcw and may be due to an artifact of concordance-based metrics, where a misspecified model may perform better than a correctly specified one[33]. We attribute this difference in performance to the flexibility of CBNN in terms of time-varying interactions and baseline hazard, flexibility the other neural network models do not have, explaining their drop in performance.

4 | APPLICATION TO SUPPORT AND METABRIC DATA

Our complex simulation demonstrates the superior performance of CBNN in ideal conditions with clean data. To obtain a more realistic performance assessment, we compared models using two real datasets with a time-to-event outcome. The first case study examines the SUPPORT dataset[6]. The second case study examines the METABRIC dataset[12]. We use the same hyperparameters as in the simulation studies. As we do not know the true model for the real data, we exclude the Optimal model. We split the datasets keeping 20% of the observations as a test set. 20% of the training set is kept aside for validation at each epoch. We predict risk functions for everyone in the test set, which are used to calculate our metrics. We conduct 100 bootstrap re-samples for the real data applications to obtain confidence intervals.

4.1 | Performance evaluation using the SUPPORT dataset

The SUPPORT dataset tracks the time until death for seriously ill patients at five American hospitals[6]. We use a pre-processed version of the dataset made available in the DeepSurv package[5]. This dataset contains 9104 samples and 14 covariates (age, sex, race, number of comorbidities, presence of diabetes, presence of dementia, presence of cancer, mean arterial blood pressure, heart rate, respiration rate, temperature, white blood cell count, serum's sodium, and serum's creatinine)[5]. Patients with missing features were excluded and 68.10% of the patients died during the 5.56-year study period[5].

Figure 2 E, F and Table 1 C demonstrates the performance over time on a test set. The regression models (CBLR, Cox) perform best considering IPA, followed by CBNN from the 25th to 100th percentile of follow-up time. We note a drop in performance for CBNN before the 25th percentile of follow-up. For c-ipcw, CBNN outperforms the competing models consistently over follow-up time. Note that performance is similar for all models, aside from DeepSurv whose c-ipcw is lower than the rest (Figure 2 E, F and Table 1 C).

4.2 | Performance evaluation using the METABRIC dataset

METABRIC is a 30-yearlong study aiming to discover the molecular drivers of breast tumors, following 2000 individuals with breast cancer until death[12]. They described these growths as primary invasive breast carcinomas, with the goal of discovering both genetic and clinical risk factors for breast cancer survival[12]. We used the processed dataset made available through DeepSurv[5], which includes 1980 patients of which 57.72% die due to breast cancer within a median 10 years of follow-up[5]. There are 11 covariates in total: 4 RNA-Seq gene expressions (MKI67, EGFR, PGR, and ERBB2) and 5 clinical features (hormone treatment indicator, radiotherapy indicator, chemotherapy indicator, ER-positive indicator, and age at diagnosis)[5].

Figure 2 G, H and Table 1 D shows the performance on a test set over time. The IPA scores suggest that regression models outperform competing models on this dataset, as all the neural network models are equal to or perform worse than KM over follow-up time. Our CBNN model is comparable to KM until around the 50-percentile of follow-up time, after which CBNN, DeepSurv and DSM drop in performance. The c-ipcw produces a different ranking. Our CBNN model outperforms the other models up to the 25^{th} percentile of follow-up time, whereas DeepHit performs best from the 50^{th} to 75^{th} percentile of follow-up. With this metric, CBNN and DeepHit outperform the regression models. This disagreement between IPA and c-ipcw between may be due to the misspecification issue of concordance-based metrics[33]. The neural network models may be over-parameterized, as shown by the wide confidence bands in c-ipcw.

5 | DISCUSSION

Our method, CBNN, models survival outcomes by using neural networks on case-base sampled data. While estimating the hazard function, we incorporate follow-up-time as a feature, providing a data-driven estimation of a flexible baseline hazard and time-varying interactions. Based on our complex simulation results (Figure 2 C, D and Table 1 B), CBNN outperforms the competitors when time-varying interactions and a complex baseline hazard are present.

Each competing neural network model we evaluated has limitations or model-specific parameters. DeepSurv is a proportional hazards model and does not estimate the baseline hazard[5]. DeepHit requires an alpha hyperparameter, is restricted to a single distribution for the baseline hazard, and models the survival function directly[11]. DSM requires the user to specify the component distributions in the mixture for a flexible baseline hazard to be fit[14]. These alternative neural network approaches match on time, while CBNN models time directly.

In the simple simulation, all neural network-based approaches resulted in a loss of performance based on IPA, while the regression-based approaches did not. We attribute this to potential over-parameterization in the neural network models. The wide confidence intervals suggest over-fitting may be the reason for the loss in performance, even with dropout. Both DeepSurv and DSM are affected, while DeepHit and CBNN are less so.

In the complex simulation, CBNN has a distinct advantage over the other methods while examining the IPA score. The regression models do show improvement over the null model, while the competing neural network approaches do not. The time-varying interactions and complex baseline hazard may have caused a drop in performance for the competing neural network methods, compared to the regression models without interactions.

In the SUPPORT application, the overall performance considering IPA is near identical. The regression models outperform CBNN, which performs better than the competing neural network models. As the confidence intervals are tight, it is unlikely that the loss in performance is because of over-fitting. Flexibility in both interaction modeling and baseline hazard is not particularly helpful with the SUPPORT dataset.

In the METABRIC application IPA results, DSM and DeepSurv had a steep drop in performance. As DeepSurv does not depend on the baseline hazard during the fitting process and DSM is the most flexible competing model in our comparison, the baseline hazard is unlikely to cause this drop. With tight confidence intervals, over-fitting is unlikely as well. Further research would be required to determine the cause of this drop in performance. We do not see the same drop in CBNN demonstrating that, in the METABRIC application, CBNN outperforms DeepSurv and DSM. Considering IPA score, regression models consistently outperform the neural network models. From a biological perspective, the 30-year follow-up time in the METABRIC study may contain competing causes of death. The signal from the start of the study may not match the signal towards the end, potentially explaining the drop in performance as we reach later survival times with fewer individuals.

Concordance-based measures are commonly used in survival analysis to compare models and we opt to keep them in our analyses. However, c-ipcw is a non-proper metric and may cause misspecified models to appear better than they should[33]. The model rankings between c-ipcw and IPA differed for both the complex simulation and METABRIC application. In the complex simulation, the c-ipcw of a correctly specified model (Optimal) underperforms CBNN and CBLR; models without the exact interaction terms specified. We base our METABRIC assessment of this disagreement between metrics on the complex simulation. The IPA score ranks regression models first, followed by DeepHit and CBNN, but the c-ipcw score ranks regression models after DeepHit and CBNN. Wide confidence intervals for neural network models in c-ipcw show potential misspecification, suggesting that regression models perform best in the METABRIC application.

During our testing for the complex simulation and SUPPORT application, DSM did not converge with our choice of hyperparameters and model design. As the goal is to fix the design and compare performance, we did not change any shared hyperparameters. For DSM, a different number and combination of component distributions did not help the model converge, though we did not conduct an exhaustive test. This may be a limitation to their method or early-stage implementation issues. We include this method where it did function as it shares a similar goal of a flexible baseline hazard.

If the user suspects that there may be unknown time-varying interactions and a complex baseline hazard, CBNN should strongly be considered. Once we perform case-base sampling and adjust for the sampling bias, we can use a sigmoid activation function to predict our hazard function. The user can approach their problem as they would for probabilistic modeling while interpreting their prediction in a survival context. In terms of performance, we saw that CBNN consistently performs better than the other neural network-based approaches over follow-up time. This may be because of the hyperparameters chosen, as they differ from each of the respective methods papers. CBNN only requires the use of standard components in machine learning libraries (add layer and sigmoid activation function). Due to the simplicity in its implementation and by extension user experience, we expect CBNN to be both a user-friendly approach to data-driven survival analysis and easily extendable to any feed-forward neural network frameworks.

Though CBNN can model time-varying interactions and a flexible baseline hazard, all models require substantial amounts of data to learn real, complex relationships effectively. Larger datasets will appear with time, but the cost of generating longitudinal datasets remains. A viable alternative strategy is the use of transfer learning, where independent and identically distributed data can function as a preliminary model to learn related features in a parallel dataset, initializing weights for our survival task[34].

This paper describes CBNN in the single event setting. A next step would be to extend this approach to competing risks. A promising use case would be high-dimensional data with complex correlation structures, such as imaging data.

References

1. Hanley JA, Miettinen OS. Fitting smooth-in-time prognostic risk functions via logistic regression. *The International Journal of Biostatistics* 2009; 5(1).

2. Royston P, Parmar MK. Flexible parametric proportional-hazards and proportional-odds models for censored survival data, with application to prognostic modelling and estimation of treatment effects. *Statistics in medicine* 2002; 21(15): 2175–2197.

3. Coradini D, Daidone MG, Boracchi P, et al. Time-dependent relevance of steroid receptors in breast cancer. *Journal of clinical oncology* 2000; 18(14): 2702–2709.

4. Hilsenbeck SG, Ravdin PM, Moor CAd, Chamness GC, Osborne CK, Clark GM. Time-dependence of hazard ratios for prognostic factors in primary breast cancer. *Breast cancer research and treatment* 1998; 52(1): 227–237.

5. Katzman JL, Shaham U, Cloninger A, Bates J, Jiang T, Kluger Y. DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network. *BMC Medical Research Methodology* 2018; 18(1): 24.

6. Knaus WA, Harrell FE, Lynn J, et al. The SUPPORT prognostic model: Objective estimates of survival for seriously ill hospitalized adults. *Annals of Internal Medicine* 1995; 122(3): 191–203.

7. Faraggi D, Simon R. A neural network model for survival data. *Statistics in medicine* 1995; 14(1): 73–82.

8. Wang H, Li G. Extreme learning machine Cox model for high-dimensional survival analysis. *Statistics in medicine* 2019; 38(12): 2139–2156.

9. Ribeiro MT, Singh S, Guestrin C. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386* 2016.

10. Sun T, Wei Y, Chen W, Ding Y. Genome-wide association study-based deep learning for survival prediction. *Statistics in medicine* 2020; 39(30): 4605–4620.

11. Lee C, Zame WR, Yoon J, Schaar Mvd. DeepHit: A Deep Learning Approach to Survival Analysis With Competing Risks.. *AAAI Conference on Artificial Intelligence* 2018: 2314–2321. Software from pycox version 0.2.2.

12. Curtis C, Shah SP, Chin SF, et al. The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature* 2012; 486(7403): 346–352.

13. Ripley RM, Harris AL, Tarassenko L. Non-linear survival analysis using neural networks. *Statistics in medicine* 2004; 23(5): 825–842.

14. Nagpal C, Li XR, Dubrawski A. Deep survival machines: Fully parametric survival regression and representation learning for censored data with competing risks. *IEEE Journal of Biomedical and Health Informatics* 2021. Software downloaded March 10, 2021.

15. Saarela O, Hanley JA. Case-base methods for studying vaccination safety. *Biometrics* 2015; 71(1): 42–52.

16. Gulli A, Pal S. *Deep learning with Keras*. Packt Publishing Ltd . 2017.

17. Hughes-Hallett D, Gleason AM, McCallum WG. *Calculus: Single and multivariable*. John Wiley & Sons . 2020.

18. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 2014; 15(1): 1929–1958.

19. R Core Team . R: A Language and Environment for Statistical Computing. WebPage; 2020. version 4.0.3.

20. Van Rossum G, Drake FL. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace . 2009. Version 3.6.13.

21. Terry M. Therneau , Patricia M. Grambsch . *Modeling Survival Data: Extending the Cox Model*. New York: Springer . 2000. R package version 3.2-11.

22. Bhatnagar S, Turgeon M, Islam J, Saarela O, Hanley J. casebase: Fitting Flexible Smooth-in-Time Hazards and Risk Functions via Logistic and Multinomial Regression. WebPage; 2020. R package version 0.9.0.

23. Sonabend R. survivalmodels: Models for Survival Analysis. WebPage; 2021. R package version 0.1.8.9000.

24. Allaire J, Chollet F. keras: R Interface to ’Keras’. WebPage; 2021. R package version 2.7.0.

25. Brilleman SL, Wolfe R, Moreno-Betancur M, Crowther MJ. Simulating Survival Data Using the simsurv R Package. *Journal of Statistical Software* 2020; 97(3): 1–27. R package version 1.0.0doi: 10.18637/jss.v097.i03

26. Jackson C. flexsurv: A Platform for Parametric Survival Modeling in R. *Journal of Statistical Software* 2016; 70(8): 1–33. R package version 2.0doi: 10.18637/jss.v070.i08

27. Pölsterl S. scikit-survival: A Library for Time-to-Event Analysis Built on Top of scikit-learn. *Journal of Machine Learning Research* 2020; 21(212): 1-6. Version 0.14.0.

28. Gerds TA, Kattan MW. *Medical Risk Prediction Models: With Ties to Machine Learning (1st ed.)*. Chapman and Hall/CRC . 2021. R package version 2021.10.10.

29. Ushey K, Allaire J, Tang Y. reticulate: Interface to ’Python’. WebPage; 2020. R package version 1.22.

30. Kattan MW, Gerds TA. The index of prediction accuracy: an intuitive measure useful for evaluating risk prediction models. *Diagnostic and prognostic research* 2018; 2(1): 1–7.

31. Uno H, Cai T, Pencina MJ, D’Agostino RB, Wei LJ. On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. *Statistics in medicine* 2011; 30(10): 1105–1117.

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

33. Blanche P, Kattan MW, Gerds TA. The c-index is not proper for the evaluation of-year predicted risks. *Biostatistics* 2019; 20(2): 347–357.

34. Bozinovski S. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica* 2020; 44(3).

□