 doi:10.20944/preprints202104.0529.v1

https://doi.org/

Deep Learning-based Survival Analysis for Ultra High-dimensional Survival Data

Hao, L.; Kim, J.; Kwon, S.; Ha, I.D.

Article

Deep Learning-based Survival Analysis for High-dimensional Survival Data

Lin Hao [1], Juncheol Kim[1], Sookhee Kwon [1] and Il Do Ha [1,2,*]

[1] Department of Statistics, Pukyong National University, Busan, 48513, Korea
[2] Department of Artificial Intelligence Convergence, Pukyong National University, Busan, 48513, Korea
* Correspondence: idha1204@gmail.com; Tel.: +82-51-629-5536

Abstract: As the development of high-throughput technologies, more and more high-dimensional or ultra high-dimensional genomic data are generated. Therefore, how to make effective analysis of such data becomes a challenge. Machine learning (ML) algorithms have been widely applied for modelling nonlinear and complicated interactions in a variety of practical fields such as high-dimensional survival data. Recently, the multilayer deep neural network (DNN) models have made remarkable achievements. Thus, a Cox-based DNN prediction survival model (DNNSurv model) [1], which was built with Keras and Tensorflow, was developed. However, its results were only evaluated to the survival datasets with high-dimensional or large sample sizes. In this paper, we evaluate the prediction performance of the DNNSurv model using ultra high-dimensional and high-dimensional survival datasets, and compare it with three popular ML survival prediction models (i.e., random survival forest and Cox-based LASSO and Ridge models). For this purpose we also present the optimal setting of several hyper-parameters including selection of tuning parameter. The proposed method demonstrates via data analysis that the DNNSurv model performs overall well as compared with the ML models, in terms of three main evaluation measures (i.e., concordance index, time-dependent Brier score and time-dependent AUC) for survival prediction performance.

Keywords: censored data; machine learning; deep learning; DNNSurv; survival analysis

1. Introduction

Survival time (i.e., time-to-event) is defined as the time to an event of interest, such as time to death, time to recurrence, and time to employment. One important characteristic of the survival data is that it is often censored, which leads to incomplete outcomes. Due to the presence of censoring, statistical analysis for survival data is usually much more complicated than regular statistical analysis [2]. Many statistical methods have been developed for survival analysis by using typically non-parametric or semi-parametric statistical methods. Particularly, as the development of high throughput technologies, more and more high-dimensional (HD) or ultra high-dimensional (ultra HD) genomic data are generated [3]. Unlike regular cases, the HD case is often observed in biomedical data such as genomic data, i.e., the number of covariates (p) (e.g., gene features) is usually much larger than the sample size (n) (i.e. $p \gg n$), leading to a challenging problem [3]. In this paper, we consider HD case as well as ultra HD case where p is extremely large (e.g., $p > 10^5$).

Recently, machine learning (ML) algorithms have been widely applied for modelling nonlinear and complicated interactions, and improving predictability, in a variety of practical fields, and it can well handle incomplete data in survival analysis for HD survival data [4]. Particularly, neural network algorithm has been applied to survival analysis for a long time. The multilayer deep neural network (DNN) model has recently made remarkable achievements for complex and HD cases with complete data [5–7]. Nevertheless, the application of deep learning to survival analysis for censored data is still limited because the existing DNN survival modelling approaches use a single hidden layer only [1]. Faraggi and Simon [8] proposed a single hidden layer feed-forward neural network which is usually regarded as a nonlinear extension of the Cox proportional hazards (PH) model. However,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

it did not outperform the classical Cox model in a research with prostate cancer survival data [9,10]. Katzman et al. [11] restudied such single layer model in a deep learning framework (named DeepSurv) and it showed that, in terms of the the Harrell's [12] concordance index (C-index), the novel network performed better than the regular Cox model and random survival forest [13] model in a research of breast cancer. Ching et al. [14] proposed a single hidden layer deep learning package (Cox-nnet) to predict patient prognosis from high-throughput omics survival data which is also an extension of neural network for the Cox model. It was demonstrated that the neural network survival model performed similar or better than the regular Cox model, the penalized Cox model and the random survival forest model using TCGA (the cancer genomic atlas) cancer data with high throughout gene expressions. To overcome such restriction, very recently, Sun et al. [1] developed a multi-hidden-layer Cox-based DNN survival model (DNNSurv model), to predict the progression of an age-related macular degeneration (AMD) disease, and compared it with other survival models based on machine learning. It showed that in a research of AMD progression, the DNNSurv model not only outperformed several other survival models (e.g., penalized Cox model and random survival forest model) in terms of the evaluation metrics (e.g., C-index), but also successfully obtained the patient-specific predictor importance measures using the local interpretable model-agnostic explanation (LIME) method [15]. However, it was only concerned to the survival datasets with both HD and large sample size.

In this paper, we evaluate the prediction performance of the DNNSurv model using several HD and ultra HD datasets for survival analysis, and compare it with three popular ML survival prediction models (i.e., random survival forest model, Cox-based LASSO and Ridge models). Here, we also present the optimal setting of several hyper-parameters including selection of tuning parameter. The DNNSurv model is built with Keras [16] and Tensorflow [17] to make sure that the computation is stable and efficient. Keras is an open-source software library and is often used to define and train deep learning models. Several backends are supported by Keras, and Tensorflow is used as the backend engine of Keras. Keras contains numerous commonly used building blocks for neural-network, such as layers, activation functions, and optimizers, which makes the work much easier for writing deep neural network code. The DNNSurv model [1] is compatible with both GPUs and CPUs, via Keras and Tensorflow.

The paper is organized as follows. In Section 2, we review the machine and deep learning survival methods, including prediction evaluation measures for survival analysis. In Section 3, we present the setting of hyper-parameters, together with a cross validation procedure how to find the optimal tuning parameter, and we then assess the performance of four survival prediction models (DNNSurv, Random survival forest, Cox-based LASSO and Cox-based Ridge) using several real HD and ultra HD survival datasets. Discussion is given in Section 4.

2. Machine and deep learning methods for survival analysis

Let T be a non-negative continuous random variable which represents the time-to-event. The survival function and hazard function are denoted by $S(t)$ and $\lambda(t)$, respectively. For each individual i $(i = 1, \ldots, n)$, let T_i be the survival time and let C_i be the corresponding censoring time. Then observable random variables are given by

$$Y_i = \min(T_i, C_i) \text{ and } \delta_i = I(T_i \leq C_i), \tag{1}$$

where δ_i is censoring indicator.

Let $x = (x_1, \ldots, x_p)^T$ be a vector of covariates for an individual and let $\lambda(t; x)$ be the hazard function at time t for an individual with covariates x. Under the Cox PH model, the hazard function for an individual takes the form

$$\lambda(t; x) = \lambda_0(t) \exp(x^T \beta), \tag{2}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

where $\lambda_0(t)$ is the unspecified baseline hazard function at time t under $x = 0$, and $\beta = (\beta_1, \ldots, \beta_p)^T$ is a vector of regression parameters corresponding to covariates x. The term $x^T\beta$ does not include the intercept term and it is called the linear predictor or prognostic index. In this paper, the models applied for the analysis of HD or ultra HD survival datasets are based on the Cox PH model except for the random survival forest (RSF) model.

2.1. Methods

For the HD data (e.g., high throughput genomic data) with $p \gg n$, standard statistical methods can not be applied directly. The same problems arise in the case of survival data [33]. To overcome such problems, various improved methods (e.g., penalized-based methods, random forests and deep learning) have been developed. Below we review the machine and deep learning methods for survival analysis of the HD time-to-event data.

2.1.1. Penalized Cox models

Penalized Cox models are often used for processing the HD survival data. The commonly used penalized Cox models include the Cox-based LASSO (Cox-LASSO) and Cox-based ridge (Cox-Ridge) models [18,19], which are used for minimizing the negative partial log likelihood of the Cox model with different penalty functions. The partial log-likelihood of the Cox model is given by:

$$\ell(\beta) = \sum_{r \in D} \left\{ x_r^T \beta - \log \left(\sum_{i \in R_r} \exp(x_i^T \beta) \right) \right\}, \tag{3}$$

where D is the set of all events, y_r is the rth $(r = 1, \cdots, E)$ smallest distinct event time among the Y_i's, E is the number of distinct events, x_r is the corresponding covariate vector and $R_r = \{i : Y_i \geq y_r\}$ is the set of individuals who are at risk at time y_r. Then we can obtain the penalized maximum likelihood estimates of the regression parameters β corresponding to the two methods:

$$\begin{aligned} \widehat{\beta}_{\text{LASSO}} &= \underset{\beta}{\operatorname{argmin}} \left\{ -\frac{\ell(\beta)}{n} + \gamma \|\beta\|_1 \right\}, \\ \widehat{\beta}_{\text{Ridge}} &= \underset{\beta}{\operatorname{argmin}} \left\{ -\frac{\ell(\beta)}{n} + \gamma \|\beta\|_2^2 \right\}, \end{aligned} \tag{4}$$

where $\|\beta\|_1 = \sum_{k=1}^{p} |\beta_k|$ is L_1-norm penalty, $\|\beta\|_2 = \sum_{k=1}^{p} (\beta_k^2)^{1/2}$ is L_2-norm penalty, and γ is the turning parameter, which is used for the adjustment of regularization. Particularly, there is no regularization when $\gamma = 0$, and it tends to be more regularized when $\gamma \to \infty$. Note here that LASSO penalty performs well for selecting significant variables among a variety of genes, but the limit is that it can only select at most n variables for $p \gg n$ cases because of the convex optimization problem. On the other hand, for ridge penalty, it is more suitable for solving multicollinearity problems between covariates, but is not proper for the variable selection problem [4].

2.1.2. Random survival forest

Breiman [20] proposed the random forest algorithm, and showed that randomizing the base learning process can improve the performance of ensemble learning. Figure 1 shows a simple representation of a random forest. The random survival forest (RSF) algorithm [20] is an extension of the random forest to survival analysis with censored data. Because of the fact that some parametric methods used for survival analysis are based on restrictive assumptions, it makes the survival analysis much more difficult. However, the RSF method can handle these problems automatically. It is based on random bootstrap samples using the training dataset. For each bootstrap sample, it randomly selects candidate variables at each node of the tree when growing trees. Moreover, the candidate variables

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

are used to split the node that maximize the survival difference between child nodes [13]. Note here that different from the random forest algorithm, the splitting rule in the RSF used for growing a tree should consider both survival time and censoring indicator due to censoring. For survival data, the log-rank splitting rule is often used to split nodes by maximizing the log-rank test statistic [21].

The main ideas of the RSF algorithm are growing a survival tree and building the ensemble cumulative hazard function which is the average of a cumulative hazard functions (usually we use the Nelson-Aalen cumulative hazard functions) [22]. The RSF is available using "randomForestSRC" R package.

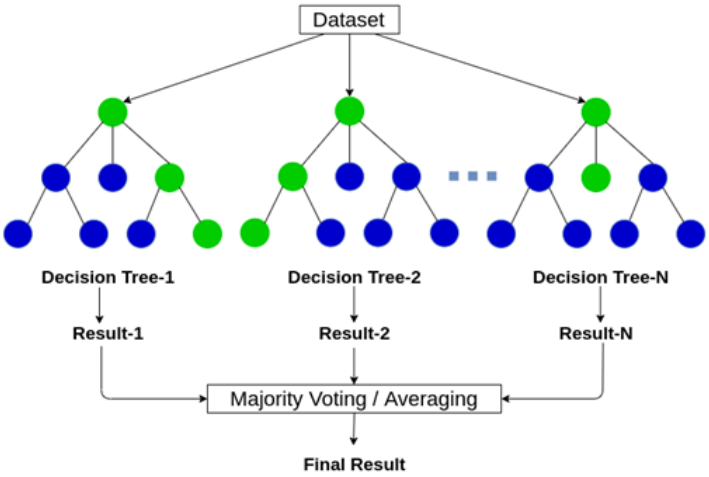

Figure 1. Diagrammatic representation of a random forest[1]

2.1.3. DNNSurv model

The DNN model is well known for its capacity in learning complex covariate structures such as non-linearity or interactions [23]. By the universal approximation theorem [24,25], the neural network algorithm can be extremely effective even though it consists of very simple architecture such as just one single hidden layer. The DNNSurv model [1] was built by the combination of the DNN survival model and the regular Cox PH model, and it can be applied to the HD or ultra HD survival datasets. The DNNSurv model is constructed as follows. The corresponding hazard function is of the form

$$\lambda(t;x) = \lambda_0(t)e^{g(x;\beta)}, \tag{5}$$

where $g(x;\beta)$ is an unknown function with a vector of parameters β, indicating the prognostic index which can be nonlinear. In other words, it is the extension of the linear predictor in the regular Cox PH model, and it becomes the Cox model when $g(x;\beta) = x^T\beta$. As a result, the DNNSurv model can be used for various nonlinear covariate structures [1]. Furthermore, because of the presence of tied events, which means that more than one events occur from different individuals at the same time, the DNNSurv model applies the Efron's approach [26] to approximate the partial log-likelihood $\ell(\beta;x)$. It is defined by

$$\ell(\beta;x) = \frac{1}{N_D}\sum_{r\in D}\left\{\sum_{i\in K_r} g(x_i;\beta) - \sum_{s=0}^{k_r-1}\log\left(\sum_{i\in R_r} e^{g(x_i;\beta)} - \frac{s}{k_r}\sum_{i\in K_r} e^{g(x_i;\beta)}\right)\right\}, \tag{6}$$

[1] For further references see https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

where D is the set of all events with size N_D, y_r is the rth $(r = 1, \cdots, E)$ smallest distinct event time, K_r is the set of individuals who fail at time y_r, k_r is the size of K_r, and R_r is the risk set at time y_r. On the other hand, the loss function of DNNSurv model with L_1 penalty, which is used for handling HD covariates, is defined as

$$\text{Loss} = -\ell(\beta; x) + \gamma \|\beta\|_1, \tag{7}$$

where γ is the tuning parameter.

A simple structure of the DNNSurv model includes one input layer, two hidden layers and one output layer, as shown in Figure 2. For each individual, the vector of covariates x is input into the input layer and a scalar prognostic index $g(x; \beta)$ is output from the output layer with weights. For the hidden layer, the model of the lth layer can be written as $a^{(l)} = f^{(l)}(w_0^{(l)} + w^{(l)} a^{(l-1)})$, which is constructed by weight w and the activation function f. The activation function of the DNNSurv model is the scaled exponential linear units (SeLU) [27], which is defined by

$$f(x) = a \cdot \text{ReLU}(x) + \lambda I(x < 0) b(e^x - 1), \tag{8}$$

where ReLU [28] is the rectified linear unit with the form of $f(x) = \max(0, x)$, and a and b are constants.

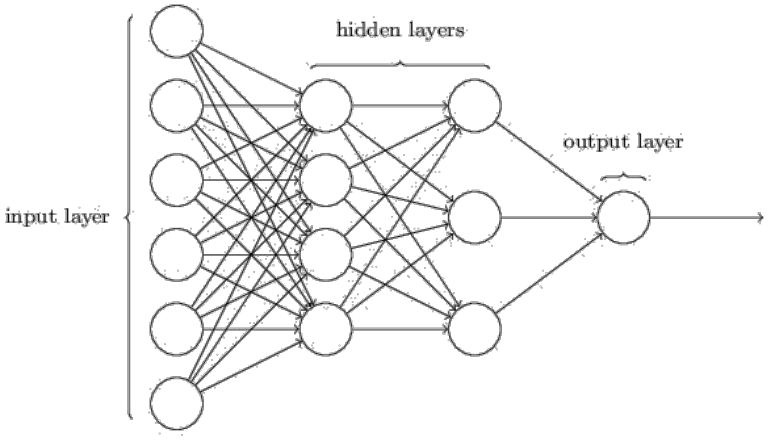

Figure 2. A schematic diagram of the DNNSurv structure [1]

The mini-batch stochastic gradient descent algorithm [29] is applied to obtain $\hat{\beta}$ to minimize the loss function (7), which is much faster than the standard stochastic gradient descent in terms of minimizing of the loss function. From (6) and (7) the loss function of the lth $(l = 1, \cdots, L)$ batch is given by

$$-\ell^l(\beta; x) + \gamma \|\beta\|_1 . \tag{9}$$

Here,

$$\ell^l(\beta; x) = \frac{1}{N_D^l} \sum_{r \in D^l} \left\{ \sum_{i \in K_r^l} g(x_i; \beta) - \sum_{s=0}^{k_r^l - 1} \log \left(\sum_{i \in R_r^l} e^{g(x_i;\beta)} - \frac{s}{k_r^l} \sum_{i \in K_r^l} e^{g(x_i;\beta)} \right) \right\}, \tag{10}$$

where N_D^l, D^l, K_r^l, k_r^l and R_r^l are the corresponding terms for the lth batch similar to those defined in equation (6). Then β can be updated through the following gradient decent formula contributed by the lth batch:

[1] For further references see https://ieeexplore.ieee.org/document/8737773

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

$$\begin{aligned} \beta_{l+1} &\leftarrow \beta_l - \eta\Delta_l \\ \text{with } \Delta_l &= -\nabla_\beta \ell^l(\beta; x) + \gamma\nabla_\beta \|\beta\|_1 , \end{aligned} \tag{11}$$

where η is the learning rate. The process will be repeated for n_e (epoch size) times until convergence. The selection of DNN hyper-parameters, which mainly include the number of hidden layers, the number of nodes in each hidden layer, activation function, turning parameter, batch size, the number of epochs and the learning rate, will be presented in detail in Section 3.

2.2. Evaluation measures of survival prediction

Three popular survival accuracy metrics, i.e. C-index and time-dependent Brier score and AUC (area under the curve), are used to evaluate the performance of the survival prediction models. Below we describe the three measures.

2.2.1. C-index

Let T_1 and T_2 be survival times of different two subjects. The C-index [12] is definede by

$$C = P\left(\widehat{T}_1 > \widehat{T}_2 | T_1 > T_2\right), \tag{12}$$

where $\widehat{T}_1$ and $\widehat{T}_2$ are estimated survival times of T_1 and T_2, respectively, which can often be obtained by the estimation of the risk or prognostic scores $g(x;\beta)$. It is used to measure the proportion of the pairs where the predicted outcomes are concordant with the observed outcomes. The C-index can be estimated by [3,12]

$$\widehat{C} = \frac{\sum_i \sum_j \left\{\delta_i I(Y_i < Y_j) I\left(\widehat{g}(x_i;\widehat{\beta}) > \widehat{g}(x_j;\widehat{\beta})\right) + 0.5 \cdot I\left(\widehat{g}(x_i;\widehat{\beta}) = \widehat{g}(x_j;\widehat{\beta})\right)\right\}}{\sum_i \sum_j \left\{\delta_i I(Y_i < Y_j) + I\left(\widehat{g}(x_i;\widehat{\beta}) = \widehat{g}(x_j;\widehat{\beta})\right)\right\}}. \tag{13}$$

The range of the value for C-index is from 0 to 1, and a larger value indicates a better performance.

2.2.2. Time-dependent Brier score

The definition of the time-dependent Brier score [30,31] is given by

$$BS(t) = E\left\{I(t) - S(t;x)\right\}^2, \tag{14}$$

where $I(t) = I(T > t)$ indicates the event status at time point t. The Brier score $BS(t)$ indicates the mean squared error of the difference between the survival function $S(t;x_i)$ and the event status $U(t)$. Thus, the Brier score is estimated based on the mean squared error between the predicted survival function $\widehat{S}(t;x_i)$ and the observed event status $Y_i(t) = I(Y_i > t)$ at a specific time point t, where $Y_i = \min(T_i, C_i)$. The estimated Brier score [30,31] is given by

$$\widehat{BS}(t) = \frac{1}{n}\sum_{i=1}^{n} \widehat{w}_i(t)\left\{Y_i(t) - \widehat{S}(t;x_i)\right\}^2, \tag{15}$$

where $\widehat{w}_i(t)$ is the inverse probability of censoring weights (IPCW) [31], which is given by

$$\widehat{w}_i(t) = \frac{(1 - Y_i(t))\delta_i}{\widehat{G}(Y_i-)} + \frac{Y_i(t)}{\widehat{G}(t)}, \text{ with } \widehat{G}(t) = \widehat{P}(C > t), \tag{16}$$

where $\widehat{G}(t) = \widehat{P}(C > t)$ with censoring time C, and $\widehat{G}(Y_i-)$ indicates the estimated survival function just prior to Y_i for the censoring time C. Note that the lower Brier score indicates the better performance.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

2.2.3. Time-dependent AUC

The receiver operating characteristic (ROC) curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. Here, TPR and FPR are equal to sensitivity and 1-specificity, respectively. In survival analysis, the ROC depends on time t [32], with the following two definitions

$$\begin{aligned} Se &= \text{Sensitivity}(c,t) = P\{M > c | T \leq t\}, \\ Sp &= \text{Specificity}(c,t) = P\{M \leq c | T > t\}, \end{aligned} \tag{17}$$

where c is an arbitrary threshold or cut-off value, M is a diagnostic test or marker (here, $M = g(x;\beta)$). The corresponding estimates of time-dependent sensitivity and specificity are given by

$$\begin{aligned} \widehat{Se} &= P\left\{\hat{g}(x;\widehat{\beta}) > c | T \leq t\right\}, \\ \widehat{Sp} &= P\left\{\hat{g}(x;\widehat{\beta}) \leq c | T > t\right\}. \end{aligned} \tag{18}$$

Therefore, we can determine the ROC curve, and furthermore we can obtain the corresponding AUC at each time point t. The value of AUC is between 0 and 1, and the discriminant ability is much more stronger with a higher AUC.

3. Analysis of high- and ultra high-dimensional survival data

We use both high-dimensional (HD) and ultra HD datasets to evaluate the performance of the DNNSurv model compared with other three survival prediction models (i.e., RSF, Cox-LASSO and Cox-Ridge).

3.1. Real survival datasets

We consider three datasets which are presented by Wang and Li [3] for the evaluation of the four survival models (i.e., DNNSurv, RSF, Cox-LASSO and Cox-Ridge). The datasets are summarized in Table 3.1. They consist of ultra HD (EMTAB386 and GSE49997 datasets) and HD (TCGAmirna dataset), which are available from the "curatedOvarianData" R package.

The short introductions of the three datasets are as follows.

- The EMTAB386 dataset contains angiogenic mRNA and microRNA gene expression signature on 129 advanced stage, high grade serous ovarian cancers, which consists of 129 samples and 10357 gene features (G).
- The GSE49997 dataset contains the expression values of 204 epithelial ovarian cancer patients, which consists of 194 samples and 16048 gene features (G).
- The TCGAmirna dataset contains 554 patients with high-grade serous ovarian cancer, which consists of 554 samples and 799 gene features (G).

In addition, as shown in Table 3.1, we also use the corresponding covariates (G+C) with both gene features (G) and clinical variables (C) in each dataset. Note that the time to event is overall survival time for each dataset. Before starting the survival analysis, we preprocess all the datasets, by including the elimination of the missing values and the columns with the same values, and normalization for continuous covariates.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

[H]

Dataset	Type of covariates	Sample size (n)	No. covariates (p)	Censoring rate
EMTAB386	G	129	10357	43.4%
GSE49997	G	194	16048	70.6%
TCGAmirna	G	554	799	47.4%
EMTAB386	G + C	107	10362	44.9%
GSE49997	G + C	193	16055	71.0%
TCGAmirna	G + C	187	814	32.1%

G: gene features only; G + C: gene features and clinical variables; No. covariates: the number of covariates.

3.2. Performance evaluation

For all datasets, we evaluate the performance of the DNNSurv model using the three evaluation measures mentioned in Section 2.2 (i.e., C-index, time-dependent Brier score and time-dependent AUC), and compare it with the performance of the three other models (i.e., RSF, Cox-LASSO and Cox-Ridge). For the performance evaluation of all four models, the 10-fold cross validation (CV) is performed for each real dataset in Table 3.1. The final results of each evaluation measures are based on the average of the results of 10 test datasets. Below we present the 10-fold CV procedure for the four models.

- For the DNNSurv model, i) the first step is to perform a 10-fold CV grid search method to select an optimal tuning parameter γ^* which maximizes the C-index. Here, the 10-fold CV procedure is as follows. Denote the full dataset by f, and denote CV training and test datasets by $f_{-k}(= f - f_k)$ and f_k, respectively, for $k = 1, \ldots, 10$. For each γ and k, we find the estimator $\widehat{\beta}_{f_{-k}}(\gamma)$ using the training dataset f_{-k}. Then, for each γ we compute the CV estimates, i.e. $\widehat{C}(\gamma)$, based on the C-index in (2.11):

$$\widehat{C}(\gamma) = \frac{1}{10}\sum_{k=1}^{10}\frac{1}{M_k}\left\{\frac{\sum_{i\in f_k}\sum_{j\in f_k}\left\{\delta_i I(Y_i < Y_j) I\left(\widehat{g}_k(x_i;\widehat{\beta}_{f_{-k}}) > \widehat{g}_k(x_j;\widehat{\beta}_{f_{-k}})\right) + 0.5\cdot I\left(\widehat{g}_k(x_i;\widehat{\beta}_{f_{-k}}) = \widehat{g}_k(x_j;\widehat{\beta}_{f_{-k}})\right)\right\}}{\sum_{i\in f_k}\sum_{j\in f_k}\left\{\delta_i I(Y_i < Y_j) + I\left(\widehat{g}_k(x_i;\widehat{\beta}_{f_{-k}}) = \widehat{g}_k(x_j;\widehat{\beta}_{f_{-k}})\right)\right\}}\right\}. \tag{19}$$

  where M_k is the sample size of the kth subset and $\widehat{\beta}_{f_{-k}} = \widehat{\beta}_{f_{-k}}(\gamma)$. Thus, we find γ^* (i.e., optimal tuning parameter) that maximizes $\widehat{C}(\gamma)$.

  ii) In the second step, given γ^*, we perform a 10-fold CV for each dataset, i.e., we train the model on the training dataset, then obtain the values of three measures (i.e., C-index, time-dependent AUC and Brier score (BS)) by the test dataset.

  For further understanding of our CV procedures with i) and ii), a flowchart is presented in Figure 3.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

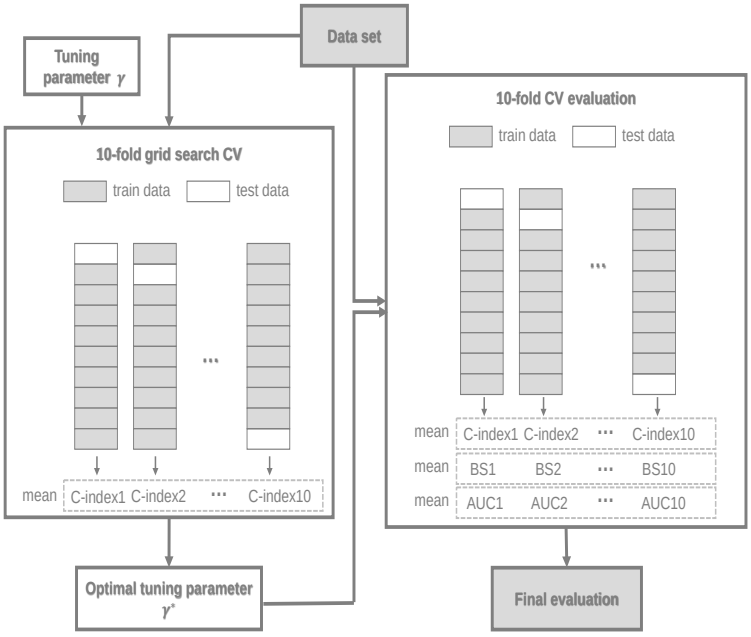

Figure 3. A flowchart of 10-fold CV procedure of the DNNSurv model

- For the RSF model, we train the model by using the log-rank splitting rule for survival analysis. The corresponding parameters we select are as follows: the number of trees are 500, the number of variables randomly selected as candidates for splitting a node is $\sqrt{p}$, and the size of terminal node is 3.
- For Cox-LASSO and Cox-Ridge models, the "glmnet" R package is applied. For L_1 penalty (in Cox-LASSO model) and L_2 penalty (in Cox-Ridge model), a 10-fold CV (by cv.glmnet() R function) is first used, respectively, in the training dataset to select the optimal tuning parameter γ^{**} (denoted as lambda.min in the R package) that gives the minimum mean cross-validated error (cvm). After the γ^{**} is determined, we train each model (Cox-LASSO or Cox-Ridge) in the training dataset, and then validate each model in the test dataset.

Furthermore, we have controlled the hyper-parameters in order to boost the performance of the DNNSurv model. The settings of proper hyper-parameters for each type of covariates are summarized in Table 3.2.

[H]

Type of covariates	Hyper-parameters	EMTAB386	GSE49997	TCGAmirna
G	No. hidden	2	2	2
	No. nodes	30	32	64
	L_1	0.05	0.08	0.02
	AF	SeLU	SeLU	SeLU
	LR	0.0001	0.0001	0.0001
	epoch size	60	60	60
	batch size	4	8	4
G + C	No. hidden	2	2	2
	No. nodes	50	32	64
	L_1	0.02	0.1	0.1
	AF	SeLU	SeLU	SeLU
	LR	0.0001	0.0001	0.0001
	epoch size	60	60	60
	batch size	8	8	8

G: gene features only; G + C: gene features and clinical variables; No. hidden: the number of hidden layers; No. nodes: the number of nodes per hidden layer; L_1: tuning parameter for L_1 penalty; AF: activation function; LR: learning rate.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

3.3. Results

Figures 4 and 5 show the prediction performance of the four models (i.e., DNNSurv, RSF, Cox-LASSO and Cox-Ridge) in terms of C-index based on 10 test datasets using the 10-fold CV. Here, the covariates of each dataset in Figures 4 and 5, respectively, indicate the gene features (G) and both gene features and clinical variables (G+C). As shown in Figures 4 and 5, we can see that the DNNSurv model gives the best performance among the three other models (i.e., RSF, Cox-LASSO and Cox-Ridge) in terms of C-index. In particular, we can also obtain a conclusion that for each dataset, the performance of the DNNSurv model in Figure 5 is better in terms of C-index than that in Figure 4; this means that the performance of the dataset with G+C covariates is better than that with G only.

Figures 6 and 7 report the performance evaluation of the four models in terms of the time-dependent Brier score on the 10 test datasets using the 10-fold CV. The difference of the two figures is that despite gene features, there are additional clinical variables included in the datasets of Figure 7. As shown in Figures 6 and 7, at each time point t, the value of Brier score is the average of the results of Brier score generated by 10-fold CV under each model. According to Figures 6 and 7, the Brier score of the DNNSurv model at each time point t is consistently lower than the three other models on the GSE49997 dataset. It means that the DNNSurv model is superior as compared to the three other models for this dataset. For the TCGAmirna dataset, at each time point t, the value of Brier score for the DNNSurv model is a little smaller than the value under other models, which means that the performance of the DNNSurv model is slightly better than other three models in terms of time-dependent Brier score. However, the performance of the DNNSurv model does not outperform enough for the EMTAB386 dataset because it seems that the Cox-Ridge model performs better than the DNNSurv model among this dataset. Furthermore, the overall trends of the DNNSurv model of the last two datasets from Figures 6 and 7 are very similar.

Figures 8 and 9 also present the time-dependent AUC for four models on the 10 test dataset. The structures of the datasets in Figures 8 and 9 are the same as the Figures 6 and 7, respectively. For each model, the value of AUC at each time point t is the average of the AUC at each time point t generated by 10-fold CV. From Figures 8 and 9, we can see that at almost all time points among all the datasets, the AUC values of the DNNSurv model is consistently higher than those of the three other models. The results demonstrate that the DNNSurv model performs the best as compared to the three other models (i.e., RSF, Cox-LASSO and Cox-Ridge) in terms of the time-dependent AUC. Moreover, we find that for each dataset, the performance of the DNNSurv model according to the time-dependent AUC in Figure 9 is overall better than that in Figure 8.

In addition, Table 3.3 summarizes the 10-fold CV C-index, and the Brier score and AUC at the specified time point in all four survival models under two covariate cases (G and G+C) for each dataset. Here, the results are mean and standard deviation (SD) of the C-index and the Brier score and AUC values at 5 year, based on the 10 test datasets. From Table 3.3, we find that in the three datasets, the performance of the DNNSurv model is overall the best among the four models in terms of the three evaluation measures. In particular, in terms of C-index, the DNNSurv model performs better in G+C case which contains additional clinical variables than in G case. These facts again confirm the results from Figures 4 and 5.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

In summary, we can see that the DNNSurv model performs overall the best as compared to the three ML models (i.e., RSF, Cox-LASSO and Cox-Ridge) in terms of the three evaluation measures under all the datasets we used here.

Dataset	Type	Measure	DNNSurv	RSF	Cox-LASSO	Cox-Rid
EMTAB386	G	C-index (SD)	0.603 (0.100)	0.509 (0.102)	0.499 (0.117)	0.490 (0.0
		5Y-BS (SD)	0.259 (0.085)	0.375 (0.171)	0.234 (0.060)	0.227 (0.0
		5Y-AUC (SD)	0.552 (0.123)	0.394 (0.187)	0.491 (0.162)	0.445 (0.1
	G + C	C-index (SD)	0.687 (0.110)	0.569 (0.102)	0.488 (0.156)	0.606 (0.1
		5Y-BS (SD)	0.334 (0.230)	0.503 (0.178)	0.434 (0.228)	0.266 (0.1
		5Y-AUC (SD)	0.639 (0.130)	0.454 (0.139)	0.469 (0.200)	0.524 (0.1
GSE49997	G	C-index (SD)	0.608 (0.143)	0.562 (0.149)	0.448 (0.170)	0.490 (0.1
		3.5Y-BS (SD)	0.231 (0.078)	0.507 (0.137)	0.679 (0.151)	0.364 (0.0
		3.5Y-AUC (SD)	0.598 (0.161)	0.539 (0.137)	0.562 (0.144)	0.562 (0.1
	G + C	C-index (SD)	0.676 (0.132)	0.500 (0.190)	0.567 (0.133)	0.455 (0.1
		3.5Y-BS (SD)	0.239 (0.039)	0.458 (0.253)	0.320 (0.054)	0.607 (0.1
		3.5Y-AUC (SD)	0.588 (0.140)	0.515 (0.154)	0.522 (0.133)	0.521 (0.1
TCGAmirna	G	C-index (SD)	0.570 (0.092)	0.552 (0.047)	0.516 (0.037)	0.555 (0.0
		8.5Y-BS (SD)	0.100 (0.029)	0.108 (0.041)	0.218 (0.290)	0.098 (0.0
		8.5Y-AUC (SD)	0.601 (0.124)	0.479 (0.144)	0.519 (0.041)	0.520 (0.1
	G + C	C-index (SD)	0.683 (0.079)	0.566 (0.063)	0.513 (0.095)	0.588 (0.0
		8.5Y-BS (SD)	0.141 (0.060)	0.156 (0.167)	0.176 (0.053)	0.141 (0.0
		8.5Y-AUC (SD)	0.588 (0.146)	0.475 (0.211)	0.542 (0.148)	0.483 (0.1

Type: type of covariates; G: gene features only; G + C: gene features and clinical variables; iY-BS: i-year Brier score ($i = 3.5, 5, 8.5$); iY-AUC: i-ye ($i = 3.5, 5, 8.5$); SD: standard deviation; DNN: deep neural network; DNNSurv: Cox-based DNN survival model; RSF: random survival forest; Cox- Cox-based LASSO; Cox-Ridge: Cox-based Ridge.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

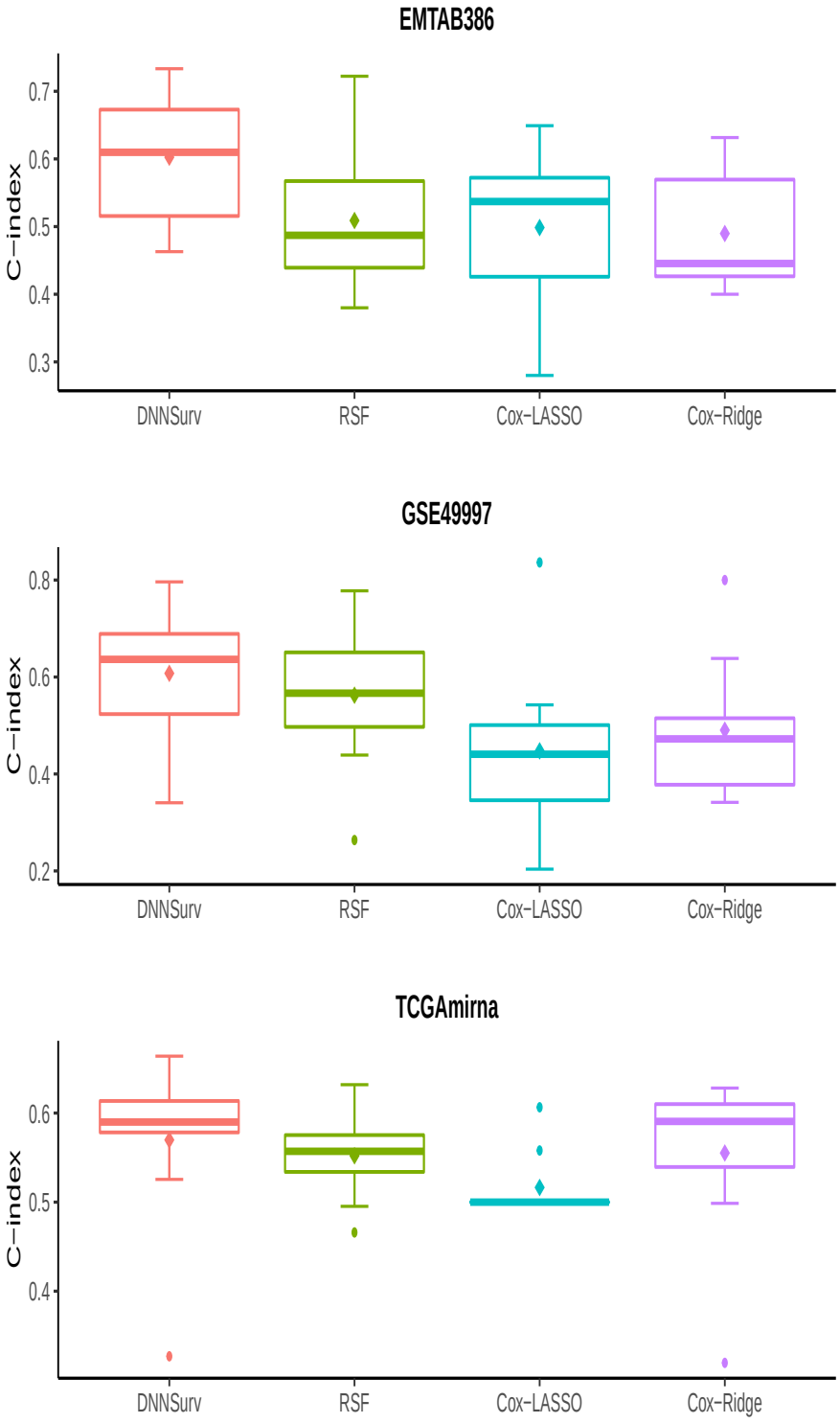

Figure 4. Boxplots of the C-index for four models on three test datasets which contain gene features only (G)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

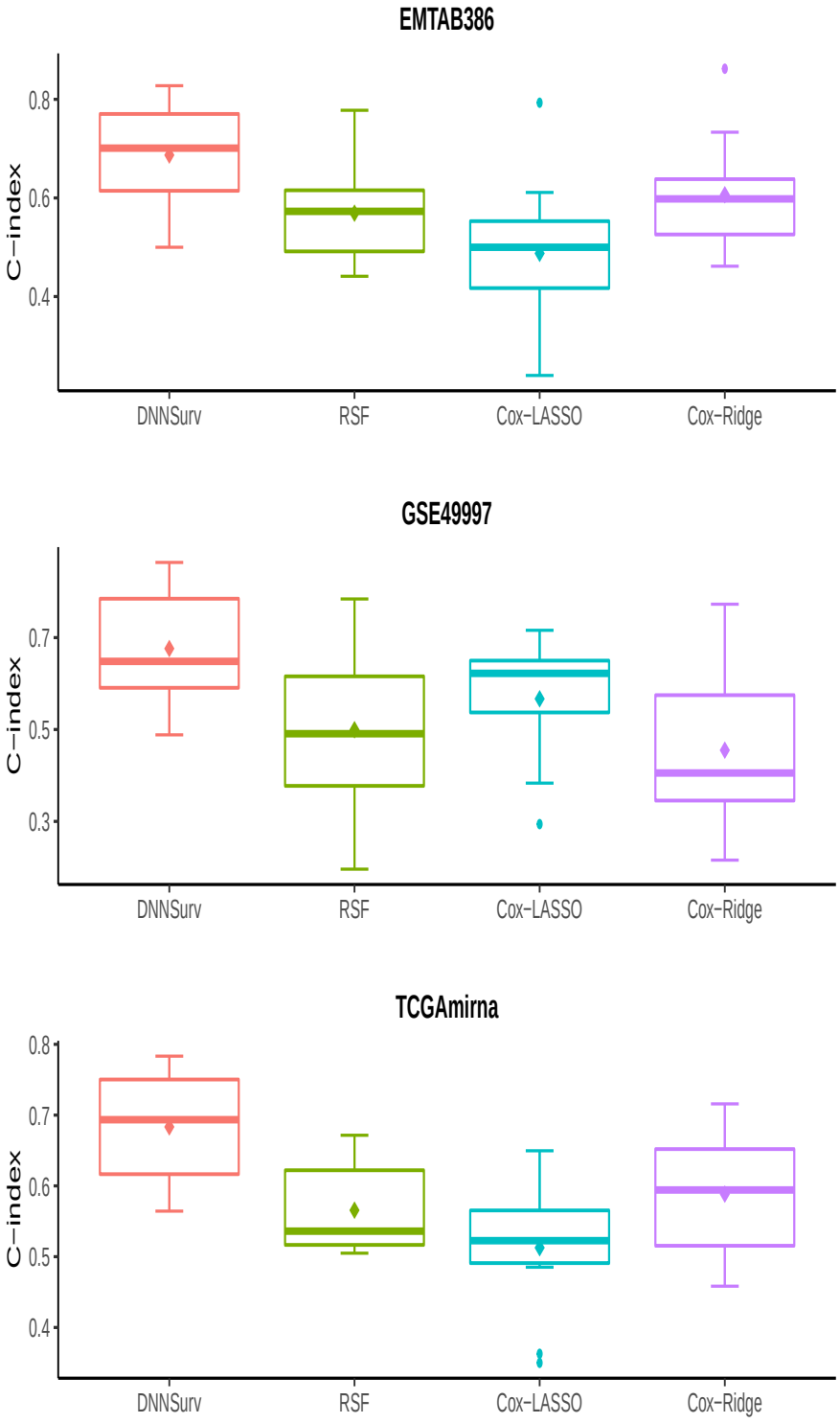

Figure 5. Boxplots of the C-index for four models on three test datasets which contain both gene and clinical variables (G+C)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

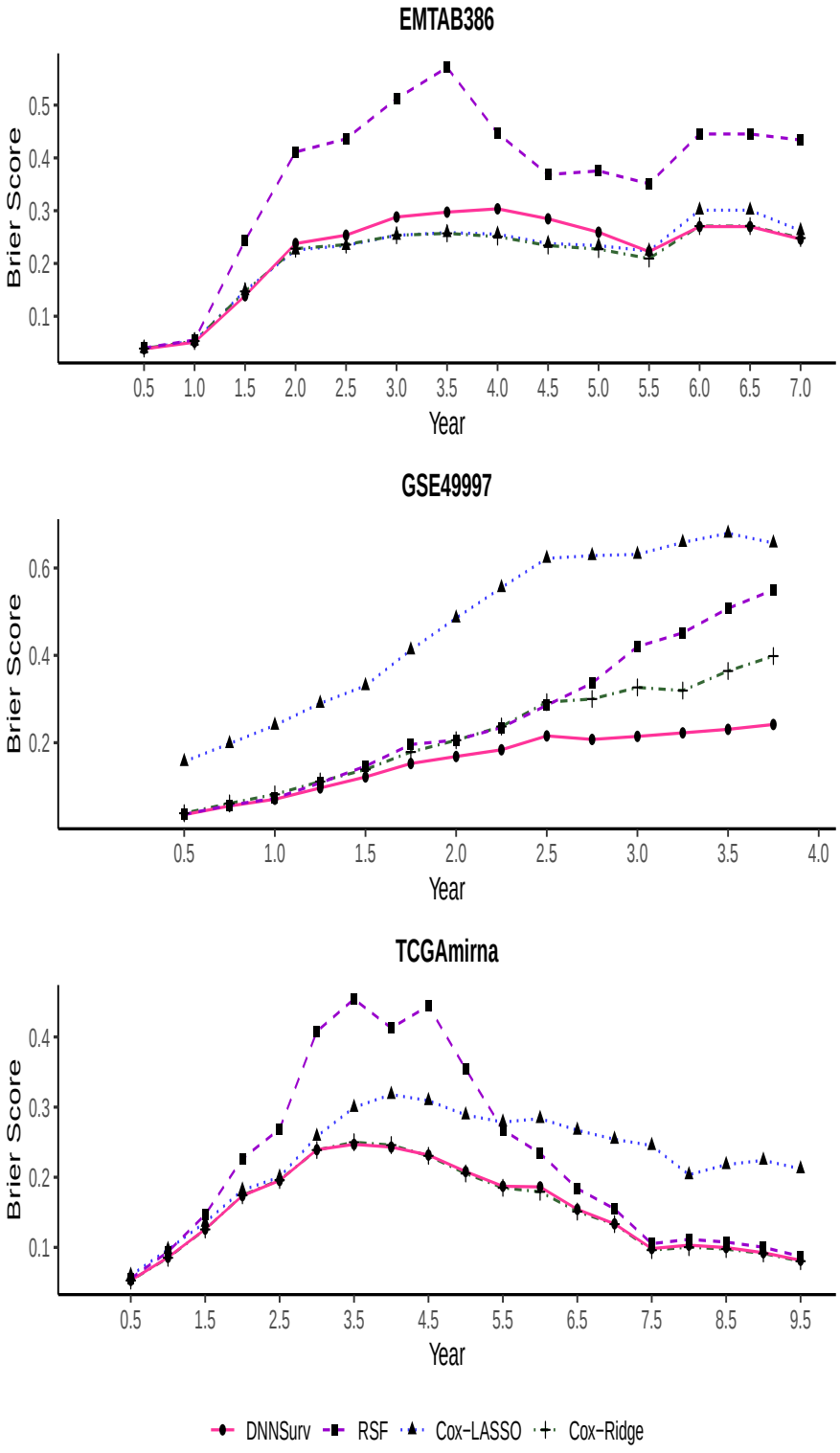

Figure 6. The time-dependent Brier score for four models on three test datasets which contain gene features only (G)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

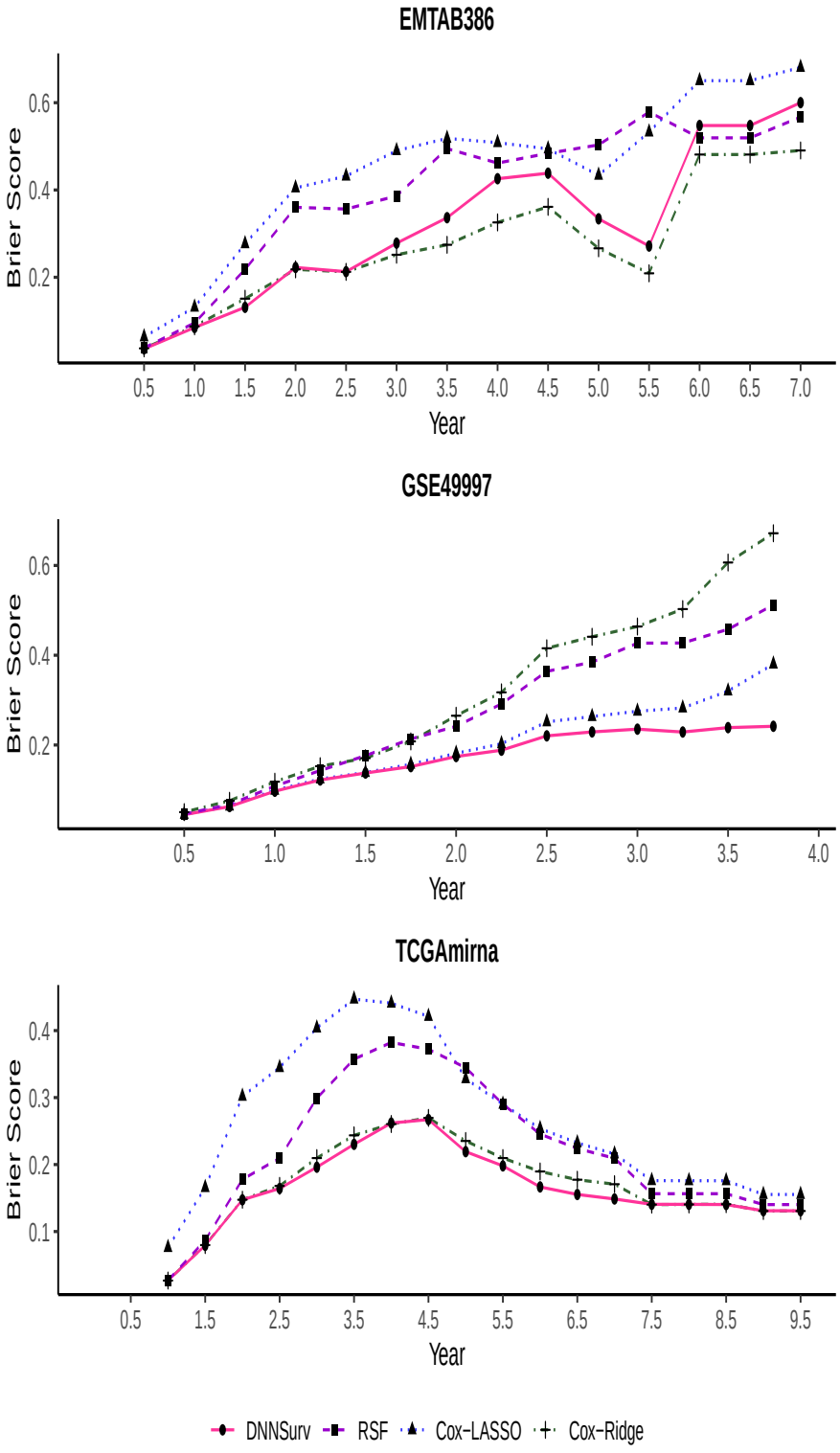

Figure 7. The time-dependent Brier score for four models on three test datasets which contain both gene and clinical variables (G+C)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

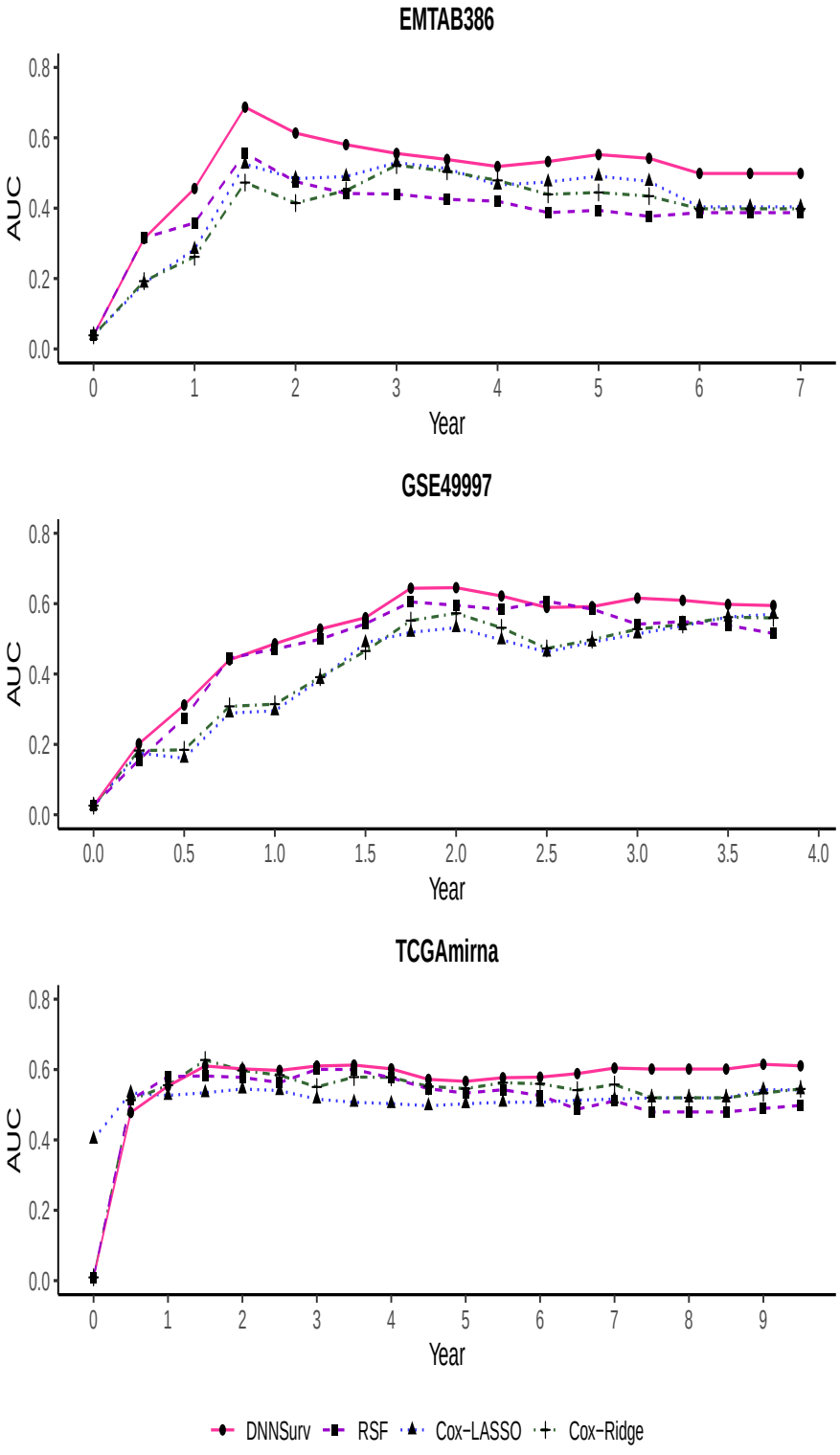

Figure 8. The time-dependent AUC for four models on three test datasets which contain gene features only (G)

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 April 2021 doi:10.20944/preprints202104.0529.v1

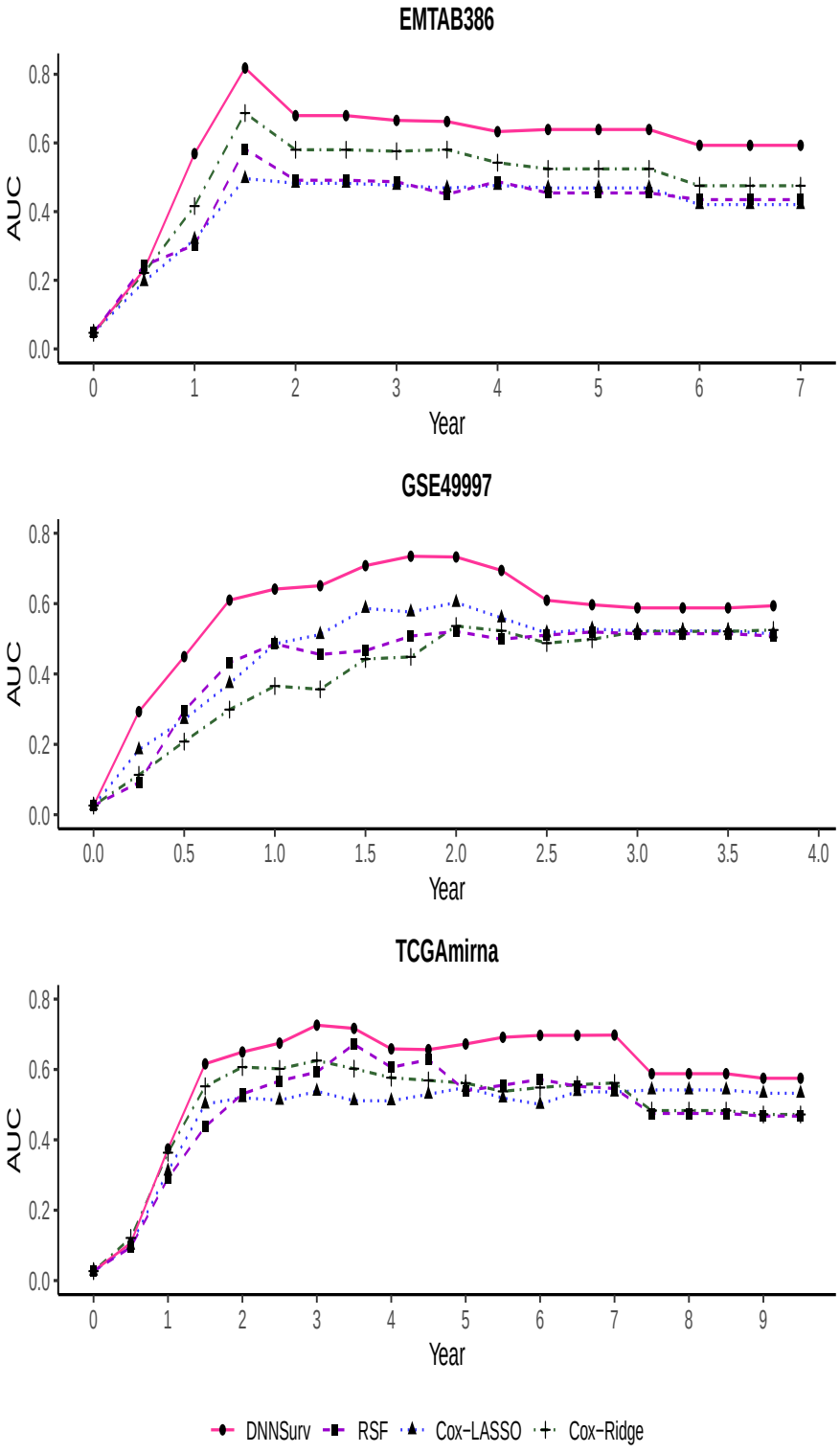

Figure 9. The time-dependent AUC for four models on three test datasets which contain both gene and clinical variables (G+C)

4. Discussion

In this paper, we have successfully applied the DNNSurv model to the real HD or ultra HD survival datasets, and have effectively evaluated its ability to make accurate dynamic survival prediction. The results of the data analysis demonstrate that the DNNSurv model outperforms the three ML survival models (i.e., RSF, Cox-LASSO and Cox-Ridge) in terms of the three evaluation measures (i.e., C-index, and time-dependent Brier score and AUC).

However, there are still some limitations in the DNNSurv model. For example, it takes much times to run the DNNSurv model using Keras and Tensorflow, which is often computationally expensive. The setting of hyper-parameters can be sensitive to the prediction performances. Developing an unified procedure for finding optimal hyper-parameters including tuning parameter would be an interesting future research. Furthermore, extension of the DNNsurv model to advanced survival models (e.g., frailty model [2]) with clustered time-to-event data would be also an interesting further work.

Author Contributions: Hao, L. and Kim, J. implemented the algorithms, and Hao, L. and Kwon, S. conducted data analysis. Hao, L. and Ha, I.D. wrote an initial draft, and Ha, I.D. verified the results and supervised this research. All authors revised the paper.

Funding: This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2020R1F1A1A01056987).

The datasets mentioned in Section 3 are public available through their R package on Bioconductor (https://www.bioconductor.org/).

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

References

1. 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.
2. Ha I.D.; Jeong J.H.; Lee Y. *Statistical modelling of survival data with random effects: h-likelihood approach*; Springer, Singapore, 2017; pp. 7–67.
3. Wang H.; Li G. Extreme learning machine Cox model for high-dimensional survival analysis. *Statistics in Medicine* **2019**, *38(12)*, 2139–2156.
4. Lee S.; Lim H. Review of statistical methods for survival analysis using genomic data. *Genomics & Informatics* **2019**, *17(4)*, e41.
5. Min S.; Lee B.; Yoon S. Deep learning in bioinformatics. *Briefings in Bioinformatics* **2017**, *18(5)*, 851–869.
6. Miotto R.; Wang F.; Wang S.; et al. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics* **2018**, *19(6)*, 1236–1246.
7. Poplin R.; Varadarajan A.V.; Blumer K.; et al. Prediction of cardiovascular risk factors from retinal fundus photographs via deep learning. *Nature Biomedical Engineering* **2018**, *2(3)*, 158–164.
8. Faraggi D.; Simon R. A neural network model for survival data. *Statistics in Medicine* **1995**, *14(1)*, 73–82.
9. Xiang A.; Lapuerta P.; Ryutov A.; et al. Comparison of the performance of neural network methods and Cox regression for censored survival data. *Computational Statistics & Data Analysis* **2000**, *34(2)*, 243–257.
10. Sargent D.J. Comparison of artificial neural networks with other statistical approaches. *Cancer* **2001**, *91(S8)*, 1636–1642.
11. Katzman J.L.; 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)*, 1–12.
12. Harrell F.E.; Lee K.L.; Mark D.B. Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine* **1996**, *15(4)*, 361–387.
13. Ishwaran H.; Kogalur U.B.; Blackstone E.H.; Lauer M.S. Random survival forests. *Annals of Applied Statistics* **2008**, *2(3)*, 841–860.
14. Ching T.; Zhu X.; Garmire L.X. Cox-nnet: an artificial neural network method for prognosis prediction of high-throughput omics data. *PLoS Computational Biology* **2018**, *14(4)*, e1006076.

15. Ribeiro M.T.; Singh S.; Guestrin C. Why should i trust you? Explaining the predictions of any classifier. Paper presented at: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, 2016; 1135–1144.
16. Chollet F. 2015; Keras, GitHub. https://github.com/fchollet/keras.
17. Abadi M.; Barham P.; Chen J.; et al. Tensorflow: a system for large-scale machine learning. Paper presented at: Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, Savannah, 2016; 265–283.
18. Tibshirani R. The lasso method for variable selection in the Cox model. *Statistics in Medicine* **1997**, *16(4)*, 385–395.
19. Hoerl A.; Kennard R. Ridge regression. *Encyclopedia of Statistical Sciences* **2006**, *8*, 129–136.
20. Breiman L. Random forests. *Machine Learning* **2001**, *45(1)*, 5–32.
21. Segal M.R. Regression trees for censored data. *Biometrics* **1988**, 35–47.
22. Ishwaran H.; Kogalur U.B.; Chen X.; Minn A.J. Random survival forests for high-dimensional data. *Statistical Analysis and Data Mining* **2011**, *4*, 115–132.
23. LeCun Y.; Bengio Y.; Hinton G. Deep learning. *Nature* **2015**, *521(7553)*, 436.
24. Cybenko G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* **1989**, *2(4)*, 303–314.
25. Hornik K.; Stinchcombe M.; White H. Multilayer feedforward networks are universal approximators. *Neural Networks* **1989**, *2(5)*, 359–366.
26. Efron B. The efficiency of Cox's likelihood function for censored data. *Journal of the American Statistical Association* **1977**, *72(359)*, 557–565.
27. Klambauer G.; Unterthiner T.; Mayr A.; Hochreiter S. Self-normalizing neural networks. Advances in Neural Information Processing Systems, 10010 N Torrey Pines Rd, La Jolla, California, USA, 2017; 971–980.
28. Sutskever I.; Martens J.; Dahl G.; Hinton G. On the importance of initialization and momentum in deep learning. Paper presented at: Proceedings of the International Conference on Machine Learning, Atlanta, USA, 2013; 1139–1147.
29. Hinton G.; Srivastava N.; Swersky K. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent; CSE 250C Machine Learning Theory Lecture: University of California, San Diego, 2012; 14.
30. 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.
31. Gerds T.A.; Schumacher M. Consistent estimation of the expected Brier score in general survival models with right-censored event times. *Biometrical Journal* **2006**, *48(6)*, 1029–1040.
32. Heagerty P.J.; Lumley T.; Pepe M.S. Time-dependent ROC curves for censored survival data and a diagnostic marker. *Biometrics* **2000**, *56(2)*, 337–344.
33. Witten D.W.; Tibshirani R. Survival analysis with high-dimensional covariates. *Statistical Methods in Medical Research* **2010**, *19(1)*, 29–51.