Survival Analysis of Heart Failure Patients

Souradeep Sen
Department of Computer Science,
University of Exeter,
Exeter, UK
souradeepsen2711@gmail.com

August 17, 2023

Abstract

The study aims to compare the performance of Deep Learning (DL) architectures for predicting mortality in heart failure (HF) patients, against traditional survival analysis techniques. The aim is to examine if deep learning can help estimate survival probabilities based on contextual and historical features from clinical data, and how these predictions perform against traditional techniques. By using longitudinal patient data from Electronic Health Records (EHR), an examination of the performance of machine learning risk prediction models is conducted against conventional survival analysis in HF patients. The findings may have important implications for clinical practice, healthcare resource allocation, and future research in risk prediction modeling for HF patients.

1 Introduction

Heart failure is a clinical condition interfering with the heart's ability to pump blood, leading to a reduction in systemic circulation performance. This condition is widespread globally - as of 2017, approximately 64 million people worldwide are estimated to be affected by heart failure [**?**]. Hence, accurately predicting risk is crucial for improving patient outcomes.

In traditional survival analysis, accounting for time-variant patient characteristics may require experimentation and extensive domain knowledge. It can also be limited in its ability to handle complex non-linear dependencies. Deep learning is an exciting tool in this aspect due to its innate ability to handle complex non-linear relationships [**?**]. As deep learning is generally data-hungry, the advent of Electronic Heath Records (EHR) data seems promising to be used in conjunction with this framework. This study attempts to examine if deep learning can compete with traditional survival analysis in terms of prediction accuracy using real-world EHR data.

The paper is organized as follows. Section 2 contains a brief review of related work. Section 3 presents a more rigorous understanding of survival analysis, before building up to how discrete-time survival likelihood may be expressed in terms of hazard rates [**?**] and can be parameterized by a neural network. A suitable loss function is chosen from the available literature and is derived as per [**?**]. Section 4 delves into Uncertainty Quantification, by way of Monte Carlo (MC) dropout [**?**] and looks at explainability of the model(s) built through the use of SHAP values [**?**]. Section 5 discusses the model architecture used for the paper. Section 6 looks at the dataset used in the paper. Section 7 establishes the evaluation criteria. Section 8 focuses on results by means of experiments over the chosen dataset and compares the architecture to existing solutions from deep learning and traditional survival analysis. Section 9 is reserved for proposed extensions to the model and further work that could be done.

2 Related Work

Deep learning methods in conjunction with classical machine learning, have recently started gaining traction in clinical settings - see [**?**], [**?**], [**?**], [**?**] and [**?**]. They are starting to be adopted in the field of survival analysis as well. A deep neural network model with learned medical feature embedding is proposed in [**?**] to address high dimensionality and temporality in electronic health record (EHR) data. Here, a convolutional neural network is used to capture non-linear longitudinal evolution of EHRs and local temporal dependency for risk prediction, and embed medical features to account for high dimensionality. Experiments show promising results in predicting risks for congestive heart failure.

The study by [**?**] focuses on personalized predictive modeling. They aim to create patient-specific models by using similar patient cohorts to capture unique traits. While CNNs show promise in measuring patient similarity, they struggle with incorporating time-dependent and context cues from EHRs. The authors introduce a solution called time-fusion CNN, by generating patient vectors to gauge similarity and subsequently making personalized disease prediction. Authors in [**?**] explore dynamically updating CNN models as new data accumulates for real-time mortality risk prediction. Meanwhile, Caicedo et al. (2019) investigate maintaining interpretability for deep learning models deployed in an ICU setting.

Extensions to the classical Cox proportional hazards model were first proposed in [**?**] and later in [**?**]. More complex architectures like RNNs were applied to ingest time-varying patient data anad generate risk scores in [**?**]. Discrete time survival predictions were addressed in [**?**], [**?**] and [**?**]. This

study aims to consider incidences as time-to-event to enable probabilistic risk prediction for mortality. The use of EHR such as those available in MIMIC-IV, allows access to comprehensive longitudinal data, which captures the entire cycle of a host of medical factors such as patients' diagnoses and treatments.

3 Survival Analysis

3.1 Basics

Survival analysis tries to estimate a survival distribution representing the probability of an event of interest, (in this case, patient death), to occur beyond a certain time in the future [**?**]. One way to specify the survival distribution is through the survival function. The survival function defines the probability of surviving till point t [**?**].

$$S(t) = P(T > t),\ 0 < t < \infty \tag{1}$$

It can be thought of as the complement of the cumulative distribution function $F(t)$.

$$S(t) = P(T > t) = 1 - P(T \leq t) = 1 - F(t),\ 0 < t < \infty \tag{2}$$

Another way to specify the survival distribution is through the hazard function, which denotes the instantaneous rate of failure. For the continuous-time scenario, the hazard function and the survival function are related as follows

$$\begin{aligned} f(t) &= \frac{d}{dt}F(t) = -\frac{d}{dt}S(t) \\ h(t) &= \frac{f(t)}{S(t)} \end{aligned} \tag{3}$$

where $f(t)$ is the probability mass function (PMF). This says that the hazard is the probability of the subject experiencing an event at time t, provided that the subject is alive till time t.

3.2 A Brief Review of Traditional Fitters

In the context of continuous time survival models, the Cox Proportional Hazards model has long been the 'gold-standard' for survival analysis [**?**]. As seen in [**?**], extensions have been made for Cox models to learn non-linear hazards although the proportional hazards assumption remains. The assumption of the Cox proportional hazards model is that the covariates can be linearly combined to form a hazard ratio, which is multiplied with the baseline hazard

$$h(t|x) = h_0(t)e^{w^T x} \tag{4}$$

where h_0 is the baseline hazard. Here x is a vector of subject covariates and w is the corresponding weight vector. The equation showcases a linear model, and $e^{w^T x}$ is the hazard ratio, which is a multiplicative factor on the baseline hazard. Proportional hazards models are learned by maximizing Cox's partial likelihood in classical survival analysis [**?**]. An expanded formulation is provided in Appendix A.3.

Accelerated failure time models (AFT) are a popular parametric alternative to the Cox Proportional Hazards (PH) model [**?**]. They are especially useful in discriminating between groups where the survival times of one can be uniformly shifted back or forth to get the survival times of another. Unlike Cox PH models, where the baseline hazard is unspecified, AFT models directly model the survival times [**?**].

At the cost of higher fitting times, Random Survival Forest (RSF) models deliver high discriminative power with impressive calibration [**?**]. Building on Breiman's approach in [**?**], this method uses the log-rank test between survival cohorts to determine the best splits [1].

3.3 Data Setup

Before moving to the discrete setting, some formal notation for the data is introduced. The data is assumed to be right-censored (starting times for all subjects are known, but ending times are not). Hence, the data, $\mathcal{D}$ can be represented as a set of tuples $\{(x_i, t_i, d_i)\}_{i=1}^{N}$ [**?**]. Here, $x_i \in \mathbb{R}^d$ are covariates for patient i. t_i is the time of an event or censoring such that $t_i = min(T_i, C_i)$, where T_i and C_i respectively denote the times of event and censoring. A subject is assumed to have either experienced an event or have been censored, but not both. d_i is an indicator that signifies whether t_i is event time or censoring time. $d = 1$ for a subject that experiences the event (uncensored) while $d = 0$ for a subject that is censored before experiencing the event. More formally, $d = \mathbb{1}\{T_i \leq C_i\}$. Later in the paper, experiments with time-variant covariates will necessitate the use of a null masking matrix, $\mathcal{M}$ [**?**].

3.4 Discrete-Time Survival Analysis

For hazard and survival calculation in a discrete-time setting, the following formulation from [**?**] is presented. Let $\mathbb{T} = \{\tau_1, \tau_2, \dots\}$ denote the timestamps, i.e. the indices of the discrete times corresponding to different subjects in the data. The event timestamp is $T^* \in \mathbb{T}$. The definitions of PMF and survival function follow as

$$\begin{aligned} f(\tau_j) &= P(T^* = \tau_j), \\ S(\tau_j) &= P(T^* > \tau_j) = \sum_{k>j} f(\tau_k) \end{aligned} \tag{5}$$

It can be seen that the hazard at time τ_j, $h(\tau_j)$, is just the probability of an event happening at time τ_j, given the subject has survived till the previous time step τ_{j-1}.

$$h(\tau_j) = P(T^* = \tau_j | T^* > \tau_{j-1}) = \frac{f(\tau_j)}{S(\tau_{j-1})} \tag{6}$$

$$\begin{aligned} &\implies h(\tau_j) = \frac{S(\tau_{j-1}) - S(\tau_j)}{S(\tau_{j-1})} \\ &\implies S(\tau_j) = (1 - h(\tau_j))S(\tau_{j-1}) \end{aligned}$$

[1] RSF models have a selection bias towards covariates with many possible splits or missing values. Improvements towards mitigating this have been made in the form of Conditional Inference Forests which use more nuanced criteria to split nodes, via unbiased recursive partitioning [**?**].

Recursively, the survival function can be parameterized wholly in terms of the hazard function as

$$S(\tau_j) = \prod_{k=1}^{j} (1 - h(\tau_k)) \tag{7}$$

Equation 7 will be useful in later derivations. Now, coming to likelihoods, if there were no censoring involved, the likelihood of observing n failures (events) is of the form

$$L(t_1, t_2, \ldots, t_n) = f(t_1)f(t_2)\ldots f(t_n) = \prod_{i=1}^{n} f(t_i) \tag{8}$$

As per [**?**], for an observed event, the pdf is retained. But for a right-censored observation, it is replaced by the survival function, as that subject is known only to have survived past that time. The likelihood then becomes

$$L(t_1, t_2, \ldots, t_n) = \prod_{i=1}^{n} f(t_i)^{\delta_i} S(t_i)^{1-\delta_i} = \prod_{i=1}^{n} h(t_i)^{\delta_i} S(t_i) \tag{9}$$

3.5 Loss Function(s)

The architecture incorporates two loss functions - one being the negative log likelihood of the data [**?**], another being a lower bound on the concordance index [**?**]. For the derivations, see Appendix A.4

These two loss functions are linearly combined based on a hyperparameter $\alpha \in [0, 1]$ to make the total loss

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{nll} + (1 - \alpha)\mathcal{L}_{lbo} \tag{10}$$

3.6 Neural Network Parameterization

As hazards are conditional probabilities (see Eq. 6), they must lie within $[0, 1]$. A handy function for this is the sigmoid non-linearity

$$g(x) = \frac{1}{1 + e^{-x}} \tag{11}$$

For a neural network taking x_i (the covariates of patient i) as input and producing m outputs denoting m discrete timesteps in the patient's journey, applying the sigmoid function effectively transforms the outputs into valid hazards.

$$h(\tau_k | x_i) = g(\phi_j(x_i)) = \frac{1}{1 + e^{-\phi_j(x_i)}} \;\forall\; k \in [\tau_{j-1}, \tau_j) \tag{12}$$

where $\phi_j(x_i)$ is the output of the neural network at node j corresponding to the hazard function over the time period $[\tau_{j-1}, \tau_j)$.

Returning to the problem at hand, for such a parameterization, the continuous time scale $\mathbb{T}$ needs to be discretized into m pieces. A simple way to do this is to equally divide up the set $\mathbb{T}$ into m equal parts. The hazards across these m timesteps can be cumulatively multiplied to find the survival function as seen in Eq. 24.

4 Uncertainty Quantification and Explainability

This implementation uses Monte-Carlo Dropout [**?**] as a means of uncertainty quantification and SHAP to explain predictions.

4.1 MC Dropout

Dropout was originally introduced in 2014 [**?**] as a method to regularize neural networks by randomly dropping units along with their connections. This prevents units from co-adapting too much, thereby forcing their neighbours to 'learn' the signal in a more 'robust' fashion. Referring to the original paper, for a standard feed-forward neural network with $\mathbb{L}$ layers, the output of the i^{th} neuron in the l^{th} layer, where $l \in \mathbb{L}$

$$\begin{aligned} z_i^l &= \mathbf{w}_i^T \mathbf{a}^{l-1} + \mathbf{b}_i \\ \mathbf{a}_i^l &= \sigma^l(z_i^l) \end{aligned} \tag{13}$$

where $\mathbf{a}_i^l$ is the activated (having passed through the activation function) output from the i^{th} neuron in the l^{th} layer. With dropout, the feed forward operation is modified slightly.

$$r_j^{l-1} \sim Bernoulli(1 - p)$$

$$\hat{\mathbf{a}}^{l-1} = r^{l-1} \times \mathbf{a}^{l-1} \tag{14}$$

$$\begin{aligned} z_i^l &= \mathbf{w}_i^T \hat{\mathbf{a}}^{l-1} + \mathbf{b}_i \\ \hat{\mathbf{a}}_i^l &= \sigma^l(z_i^l) \end{aligned} \tag{15}$$

It can be seen in 14 that units are dropped randomly according to a Bernoulli distribution with probability p, which is known as the dropout rate and is a tunable hyperparameter. Dropout is generally reserved for training. Libraries like PyTorch allow users to prime networks for training or evaluation.

When applied during the testing or prediction phase as well, a probability distribution of the output is created, allowing for probabilistic inference instead of point inference as present in regular neural networks. Over many iterations, this essentially means that the network used for prediction is slightly different each time, resulting in slightly different outputs - thereby allowing users to generate a probability distribution of outputs.

4.2 SHAP

Being uncertainty-aware ties in with the idea of interpretability of modern deep learning methods, which is often a barrier towards the adoption of such methods into highly regulated industries such as finance or medicine. SHAP (SHapley Additive exPlanations) values [**?**], extended from the field of cooperative game theory provide a unified, model-agnostic method to explain the contributions of covariates to a model's output. The idea is to estimate a model's output with a simple *explanation* model. For simple models like linear regression, the model is it own explanation model. For complex models such as deep learning models, a separate explanation model needs to be made.

Let f, and g be the original and the simplified explanation models respectively. Local methods [**?**] are used to explain a prediction $f(x)$. For an original input x mapped to a *'simplified'* input x' [2] through a mapping function $x = h_x(x')$ such that $g(z') \approx f(h_x(z'))$ whenever $z' \approx x'$. At this point, the definition of an additive feature attribution method must be introduced. The useful property of such models is that they attribute an effect ϕ_i to each feature x_i, such that the addition of all the effects approximates the output $f(x)$. They are linear functions of binary variables.

$$g(z') = \phi_0 + \sum_{i=1}^{n} \phi_i z'_i \tag{16}$$

where $z' \in \{0,1\}^n$, with n being the number of simplified input features. For explaining deep learning models, DeepLIFT was proposed in [**?**], with the idea to explain differences in output from some 'background' (or reference) output in terms of the difference in input from the background input. A value $C_{\Delta x_i \Delta t}$ is assigned to Δx such that $\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t$. This is called the summation-to-delta property. Here, $t = f(x)$ is the model output. $\Delta t = f(x) - f(r)$ refers to the deviation from the reference output and $\Delta x = x - r$ is the deviation from the reference input. Taking $\phi_i = C_{\Delta x_i \Delta t}$ and $\phi_0 = f(r)$, this explanation model satisfies the conditions to be an additive feature attribution method [**?**].

Shapley values assign an importance to each feature representing the effect the inclusion of that feature has on the model. To compute this effect, models are trained with the feature included - $f_{S\cup\{i\}}$ and with the feature withheld - $f_S(x_S)$. Here S refers to all possible subsets without the feature i, i.e. $S \subseteq F\backslash\{i\}$. The Shapley values are the weighted averages of all possible differences.

$$\phi_i = \sum_{S \subseteq F\backslash\{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)] \tag{17}$$

5 Model Architecture

This study attempts to model survival functions from both a time-invariant perspective (using means of covariates) and from a time-variant perspective (using historical data for covariates). The latter takes some more preprocessing and requires a slightly more complex architecture.

5.1 Time-Invariant

The simpler time-invariant version of the architecture is a multi-layer perceptron (MLP), otherwise referred to as a fully-connected feed-forward neural network. Consider a batch of input covariates $x \in \mathbb{R}^{n\times d}$, where n is the batch size and d is the number of input covariates. The basic architecture is as outlined in Table 1.

Layer Type	Shape/Rate
Dense (nn.Linear)	(d, h)
ReLU (nn.ReLU)	-
Batch Normalization (nn.BatchNorm1d)	h
Dropout (nn.Dropout)	0.5
Dense (nn.Linear)	(h, h)
ReLU (nn.ReLU)	-
Batch Normalization (nn.BatchNorm1d)	h
Dropout (nn.Dropout)	0.5
Dense (nn.Linear)	(h, m)

Table 1: General Architecture - Time-Invariant

Here, m is the output size, denoting the number of discretized time indices across time $max(\mathbb{T})$, while h denotes the number of hidden nodes in the linear layers.

5.2 Time-Varying

First, a patient image is built for each subject in the dataset. Consider a batch of input covariates $x \in \mathbb{R}^{n\times t_s\times d}$, where t_s is the number of time-steps along the subjects' journey(s). For this study, the number of time steps considered for all patients is kept constant and this is achieved by sampling the patient's journey over a fixed number of points. n and d retain their meaning from earlier.

A convolutional neural network is used to identify patterns in the patient-image, whose outputs are then fed into a fully-connected network. The architecture is elaborated in Table 2.

Layer Type	Shape/Rate
Convolutional (nn.Conv2d)	$(1, c_1)$
ReLU (nn.ReLU)	-
Convolutional (nn.Conv2d)	(c_1, c_2)
ReLU (nn.ReLU)	-
Flatten (nn.Flatten)	-
Dense (nn.Linear)	(fc, h)
ReLU (nn.ReLU)	-
Batch Normalization (nn.BatchNorm1d)	h
Dropout (nn.Dropout)	0.5
Dense (nn.Linear)	(h, h)
ReLU (nn.ReLU)	-
Batch Normalization (nn.BatchNorm1d)	h
Dropout (nn.Dropout)	0.5
Dense (nn.Linear)	(h, m)

Table 2: General Architecture - Time-Variant

Here, kernel_size = 3, stride = 1 and padding = 1 [3].

6 Dataset

The study uses the large publicly available database MIMIC-IV [**?**], which consists of critical care data from hospital and

[2] Note that inputs are often simplified to their binary representation. An example may be x'=1 for all non-zero values of x and 0 when $x = 0$.
[3] Note that $fc = (c_2 \times input\_data.shape[2] \times input\_data.shape[3])$

ICU admissions for almost 300,000 patients admitted to intensive care units at the Beth Israel Deaconess Medical Center (BIDMC). For a more thorough treatment of the data, see MIMIC-IV website.

6.1 Preprocess

To prepare the data, all admissions associated with a Heart Failure ICD-10 code [?] are selected - this forms the base pool of patients. Admission and discharge times are collected along with static covariates such as patients' gender, age and date of death. Patients' ethnicity is collected and grouped into six broad categories - Native, Asian, Black, Hispanic, White and Other [4].

Time-variant records such as BMI, Weight and Height are collected from Online Medical Records (OMR). Such records have an associated chart-time or chart-date, denoting when they were collected. Lab tests corresponding to cholesterol, sodium intake, lymphocyte levels [?] and hemoglobin levels [?] were considered. Medication administered from the classes of angiotensin-converting enzyme blockers, angiotensin receptor blockers, calcium channel blockers and beta blockers are also taken as features. Vital signs such as temperature, heartrate, respiratory rate, O_2 saturation, systolic and diastolic blood pressure are also taken. Patients who have at least one record for all of the above datasets (OMR, lab test, medication and vital signs) are retained.

For the time-varying version, the time difference between each subsequent observation is noted. Only patients who have at least 10 distinct time steps are retained. This of course, brings down the number of patients slightly as compared to the time-invariant version. For both versions, comorbidities such as diabetes, hypertension and obesity are included as static factors, i.e., the column corresponding to a particular comorbidity is marked as 1 throughout if they have been diagnosed with the comorbidity at least once in their journey.

6.2 Censoring

Patients' earliest admission time is considered as their 'start' date. If their death date is not captured in the data [5], their last known discharge time is taken as their 'end' date - these patients are considered to be censored. Otherwise, their date of death is taken to be the 'end' date - these are the uncensored patients. The distribution of event/censoring times is quite similar across both censored and uncensored cohorts - see Figure 1.

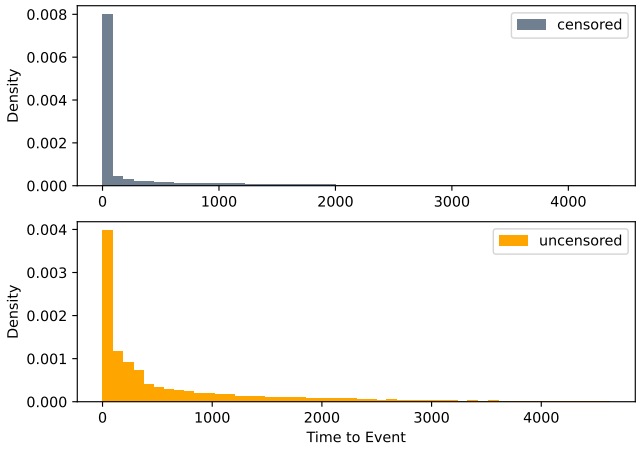

Figure 1: Cohort-wise Histogram showing Censored vs. Uncensored subjects

6.3 Data Standardization

Patients are randomly distributed into train, test and validation pools. Columns with zero variance in the training data are discarded from all three datasets - train, test and validation. Across each pool, their survival time is discretized into $m = 15$ buckets or time intervals. For example, a patient whose survival time belongs to the j^{th} bucket has experienced the event (or been censored) between the time interval $[\tau_j, \tau_{j+1})$ $\forall j \in [1, 2, \cdots, 15]$. For the time-invariant version, simple pipelines for imputation and scaling are carried out.

For the time-varying version, the process is slightly more involved. First, a patient-level cumulative time difference column is introduced, to keep track of when medications or test were administered - for every patient, this column starts with 0. Next, a null mask $\mathcal{M}$ is created, as in [?]. As neural networks cannot natively handle missing data, the matrices need to be imputed (strategies will be discussed later). However, it can be reductive to ignore the fact that some data is missing in the original dataset. Therefore, a compromise is reached by creating a null mask before imputing the missing values. This null mask is then appended to the dataset before being fed to the neural network. The expectation is for the network to consider imputed values as missing when their corresponding mask element $\mathcal{M}_{idj} = 0$, denoting the j^{th} observation of covariate d for patient i is missing.

As convolutional neural networks expect input of the same shape, the patient images need to be quantized to have the same width [6] and the same height. The latter is achieved by sampling the multivariate time series data at set points during the patients history. For this study, a patient's history is divided into 10 equal parts, and all their covariates are interpolated over these divisions. Each column is normalized as

[4] To avoid multi-collinearity issues arising later, the 'OTHER' variable is dropped after one-hot encoding. Similary, 'F' variable corresponding to gender (female) is also dropped after one-hot encoding.

[5] Out-of-hospital mortality is captured from state records. If the individual survived for at least one year after their last hospital discharge, then the death date will have a NULL value.

[6] This is already done as all the patients have the same number of covariates.

per the L2 norm. Finally, they are reshaped to be fed into a PyTorch CNN model.

7 Evaluation Metrics

As survival analysis differs from ordinary linear regression in the aspect that not all the survival times are known (right-censoring), there is merit in reviewing the evaluation metrics for this task. The two metrics used in the experimental setup are the time-dependent concordance index in favour of the regular concordance index and the Brier score along with its aggregated version, the integrated Brier score.

7.1 Time-Dependent Concordance Index (td-concordance index)

The C-index, often referred to as 'Harrell's C-Index' or simply as the C-statistic is a measure of the discriminative capacity of a model. In essence, it is the generalization of the area under the ROC (Receiver Operator Characteristic) curve in a survival analysis setting [**?**] [**?**]. The formula for the c-index is given as

$$c = \frac{1}{|\mathcal{E}|} \sum_{i:C_i=1} \sum_{T_j>T_i} \mathbb{1}_{\eta_j>\eta_i} \tag{18}$$

where η_i denotes the predicted survival time for subject i. As c-index is rank-based index, it can be substituted with the survival probability of subject i as well. $|\mathcal{E}|$ is the number of pairs that *can* be compared.

It can be interpreted as the fraction of all pairs of subjects whose predicted survival times are correctly ordered among all subjects that can actually be ordered [**?**]. A pair (i, j) is considered 'comparable' if the one with the lower observed time is uncensored, that is, when $T_i < T_j$, then $d_i = 1$. A pair is considered 'concordant' when the model predicts a higher risk for the patient with a lower survival time. Thus, it follows that

$$c - index = \begin{cases} 1.0, & \text{perfect concordance,} \\ 0.5, & \text{equivalent to random classification,} \\ 0.0, & \text{perfect anti-concordance.} \end{cases}$$

There are limitations with the traditional c-index as highlighted by [**?**], an important one being the assumption that the risk scores do not change over time. The study in [**?**] proposes a time-dependent concordance index which essentially takes a weighted average of time-specific C-index values across the entire time scale.

7.2 Brier Score

Contrasting the discriminatory power of the C-index, the Brier score provides a measure of how well the model is calibrated. It represents the distance between observed and predicted survival probability - hence 0 is the most desirable value. Given the data, $\mathcal{D}$, a set of tuples $\{(x_i, t_i, d_i)\}_{i=1}^N$ and the predicted survival function $\hat{S}(t, x_i),\ \forall t \in \mathbb{R}^+$, the Brier score (without right-censored observations) assumes a form similar to the mean-squared-error.

$$BS(t) = \frac{1}{N} \sum_{i=1}^{N} (\mathbb{I}_{T_i>t} - \hat{S}(t, x_i))^2 \tag{19}$$

With the occurence of right-censored data, the formula needs to be adjusted by the inverse probability of censoring weights method [**?**]. Let $\hat{G}(t) = P[C > t]$ be the estimator of the conditional survival function of the censoring times calculated using the Kaplan-Meier method, where C is the censoring time.

$$BS(t) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{(0 - \hat{S}(t, x_i))^2 \cdot \mathbb{I}_{T_i \le t, d_i=1}}{\hat{G}(T_i)} + \frac{(1 - \hat{S}(t, x_i))^2 \cdot \mathbb{I}_{T_i>t}}{\hat{G}(t)} \right) \tag{20}$$

An aggregated version of this score provides a simpler summary of the calibration of a model.

$$IBS(t_{max}) = \frac{1}{t_{max}} \int_0^{t_{max}} BS(t)dt \tag{21}$$

8 Experiments and Fits

A series of experiments were conducted on this data from both a time-invariant and a time-varying perspective. The survival distributions were first examined with a non-parametric fitter, the Kaplan-Meier fitter (see: Figure 7). A short write-up on these methods can be found in the Appendix A. All relevant code for this study can be found on github.

Next, a range of parametric, semi-parametric, tree-based and neural network-based methods were applied to the data and their evaluation criteria noted. To establish confidence intervals on said criteria, fits were run multiple times - all time-invariant models apart from Cox Proportional Hazards model and Weibull Accelerated Failure Time model were run 10 times. Thus, distributions of their cindex and integrated Brier scores were generated.

One of the earlier aims of this study was to see if hybrid deep learning, that is the supplementation of deep learning with classical machine learning techniques would yield better performance compared to traditional techniques. In that light, although methods such as clustering and PCA are introduced into the pipeline, the highest discriminatory power (comparable to traditional methods) is generally observed if the data is directly fed into the neural network(s). Due to this shortcoming, the time-variant version has not been equipped with any clustering utility as opposed to its time-invariant counterpart. Some tests were carried out to see the effect of having multiple clusters and preprocessing the data with PCA. Details are in Table 3. The details of experiments comparing the two architectures with other off-the-shelf methods are listed in Table 4.

Extension	Hyperparameters	C-Index	IBS
Clustering	clusters = 1	0.6957	0.3995
Clustering	clusters = 2	0.6857	0.4075
Clustering	clusters = 3	0.6786	0.3915
Clustering	clusters = 4	0.6724	0.4108
Clustering	clusters = 5	0.6703	0.4141
PCA	n_components = 1	0.4891	0.4147
PCA	n_components = 2	0.5071	0.4127
PCA	n_components = 5	0.4927	0.4133
PCA	n_components = 10	0.4826	0.4049
Clustering & PCA	clusters = 2, n_components = 2	0.5103	0.4131

Table 3: Evaluation metrics - Degradation with extensions. Note these metrics were collected on one specific run. Reruns may yield slightly different results

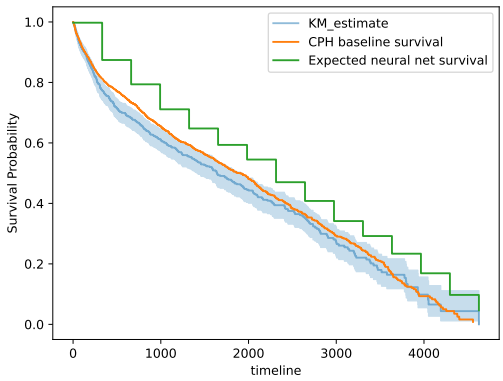

Figure 4: Comparison of expected survival curve against baseline survival and empirical survival

The time-invariant model performs better than the other neural networks, however it cannot reach the discriminative power of traditional or tree-based or traditional fitters. The time-variant counterpart seems to perform on par with Random Survival Forest in terms of c-index (discriminative power), thus outperforming the traditional fitters. Both time-invariant and time-variant methods have quite a large Brier score (indicating poor calibration) in comparison to off-the-shelf methods. The Figures 2 and 3 showcase this situation.

This implementation is equipped with a wrapper for the Python library `shap` which allows explanation of particular observations. SHAP allows the introduction of a notion of an 'expected' [7] survival curve, which can be compared against baseline survival of traditional fitters and the empirical survival distribution from non-parametric fitters such as the Kaplan-Meier fitter. Figure 4 shows this. This leads to the question of statistical similarity of the curves thus generated. A few tests involving the Kolmogorov-Smirnov test are carried out in the Appendix A.5. From here, it may be natural to want to investigate the factors that cause an individual subject's survival to deviate (at any particular timestep) from the 'expected' survival at that time. Figure 5 illustrates how a particular subject's survival curve is shifted from the expected curve. Figure 6 shows the breakdown for one patient at a particular discrete timestep.

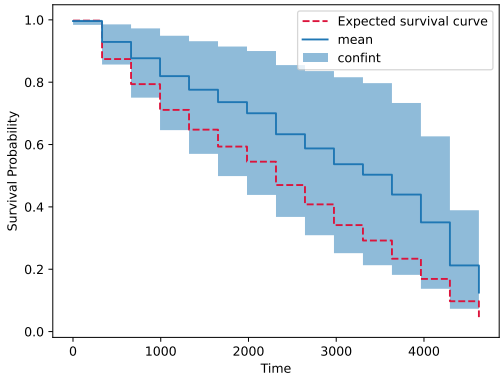

Figure 5: An individual subject's survival curve differs from the expected survival curve

Furthermore, this implementation generates confidence intervals by applying Monte-Carlo dropout. For any particular subject, the returned survival curve natively has confidence bands associated with it. They can be seen again in Figure 5.

9 Further Work

Neural network methods capable of temporal learning (forms of RNN) can be applied to this problem to see if the discriminative power can be increased. This study has not applied several modern neural network optimizations like Nesterov momentum and learning rate scheduling - further work can focus on these aspects.

Deeper architectures may be applied to the problem, however, longer training times may discourage this. This study has used evaluation metrics implemented in [**?**]. Currently the time-dependent concordance index calculation appears to take a good amount of time to generate results. This can be a point for further improvement. From a modularization perspective, classes for Time-Invariant and Time-Variant processes are completely separate at the moment. Perhaps later

[7] In the literature, expected survival generally refers to the mean or median survival time and not a time-varying metric.

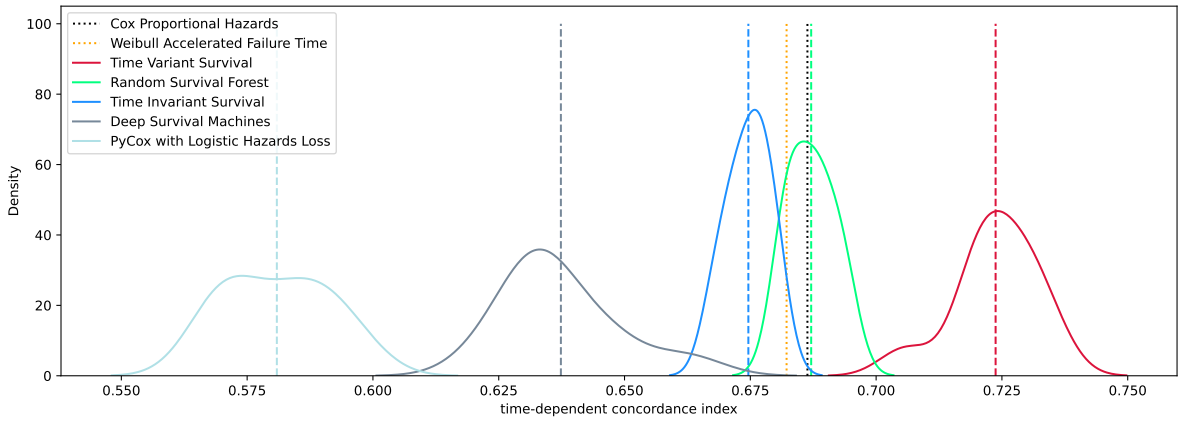

Figure 2: Distribution of C-index - All models run for 10 iterations.

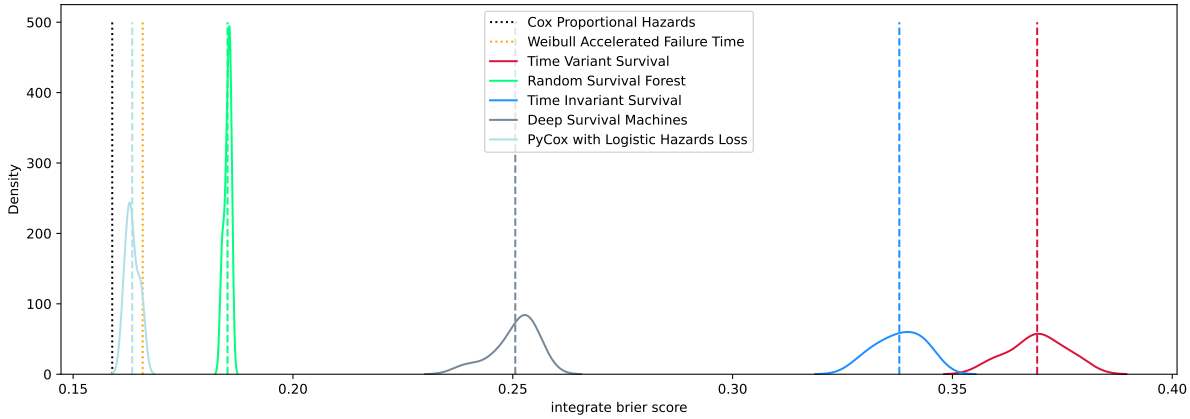

Figure 3: Distribution of IBS - All models run for 10 iterations.

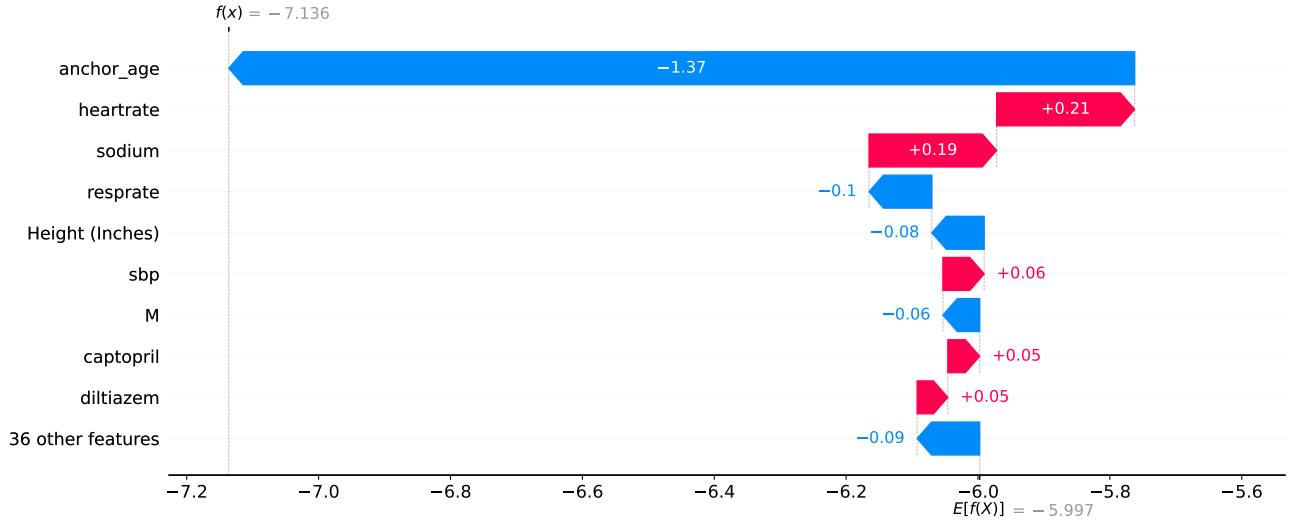

Figure 6: Deviation from expected output for single subject shown on predictor scale. Passing the values through sigmoid function will convert them to response scale, i.e. discrete hazards

Model	Hyperparameters	C-Index	IBS
Cox Proportional Hazards (CPH)	penalizer:0.1, step_size:0.1	0.6863	**0.1589**
Weibull Accelerated Failure Times (AFT)	penalizer:0.01	0.6822	0.1658
Random Survival Forest	n_estimators:100, min_samples_split:10, min_samples_leaf:15, n_jobs:-1, random_state:1234, oob_score:True	0.6874 (0.6814, 0.6935)	0.1853 (0.1838, 0.1858)
PyCox (Logistic Hazard loss function [**?**])	discretization=15, layers=[256,256], batch_norm:True, dropout:0.5, batch_size:256, epochs:500	0.5811 (0.5681, 0.5949)	0.1631 (0.1616, 0.1655)
Deep Survival Machines	k:6, distribution:LogNormal, learning_rate:1e-3, layers:[100, 100]	0.6336 (0.6251, 0.6557)	0.2523 (0.2422, 0.2553)
Time-Invariant Survival	clusters:1, discretizations:15, hidden_size:25, alpha:0.01, batch_size:256, num_epochs:400, learning_rate:0.001, shuffle:true, patience:20, dropout:0.5, mc_iter = 100	0.6751 (0.6683, 0.6801)	0.3380 (0.3303, 0.3445)
Time-Variant Survival	discretizations:15, hidden_size:25 , alpha:0.01 , batch_size:256, num_epochs:100, learning_rate:0.001, patience:20, dropout_rate:0.5, mc_iter:100	**0.7234** (0.7123 , 0.7334)	0.3691 (0.3599, 0.3778)

Table 4: Evaluation metrics - Best metrics are written in **bold**. Both metrics are reported as `Mean (lower 90% confidence interval, upper 90% confidence interval)`.

they can inherit common functionalities from a parent class.

For evaluation criteria, another popular metric in the survival analysis literature is cumulative dynamic AUC [**?**]. Future iterations can focus on this metric along with the two others mentioned in this study to see if it yields a different assessment of the models' suitability. In future, if the clustering utility is investigated in more detail, one might find that some parts of the code break due to the random initialization of clusters sometimes producing cohorts whose duration indices do not match the number of requested discretizations. This is a bug and may warrant investigation.

10 Conclusion

In conclusion, it is seen that the time-invariant version can be benchmarked by semi-parametric and fully-parametric models, resulting in discriminatory power close to these traditional fitters. The time-variant model fares marginally better and can be benchmarked by the Random Survival Forests model. In both cases, comparable performance is achieved with the off-the-shelf models; however consistently[8] outperforming these models may take further hyperparameter tuning and architecture changes.

Both the time-invariant and time-varying architectures are sacrificing calibration for discriminatory power. Again, the Figures 2 and 3 show this. Depending on the situation, the metric to which more importance is given may change. Therefore, careful consideration needs to be taken before attempting to apply the method(s) proposed in this study. Another criterion to consider that often goes hand in hand with neural networks and large scale data is training time. Deep learning methodologies can compete with traditional methods, provided proper hyperparameter tuning and enough training time. That being said, it is often seen that training time (number of epochs) is often not linearly correlated with better performance (in terms of discriminatory power).

11 Declarations

Declaration of Originality: I am aware of and understand the University of Exeter's policy on plagiarism and I certify that this assignment is my own work, except where indicated by referencing, and that I have followed the good academic practices.

Declaration of Ethical Concerns: Although this research utilizes patient data, it is anonymized and thus does not fall under the category of 'human subjects'. With this in mind, ethical approval has still been sought and received under Worktribe ethics application ID 2669275.
[]

A Mathematical Formulations

Some formal mathematical formulations of the methods used in this paper are elaborated in the following sections.

[8]It must be noted that the performance for both RSF and TVS is quite volatile

A.1 The Kaplan Meier fitter

This estimator, first proposed in [?], is the product over the failure times of the conditional probabilities of surviving to the next failure time. It is given as

$$\hat{S}(t) = \prod_{t_i \leq t} (1 - \hat{q}_i) = \prod_{t_i \leq t} (1 - \frac{d_i}{n_i}) \tag{22}$$

, where n_i is the number of subjects at risk at time t_i, and d_i is the number of individuals who fail at that time [?]. When plotted, it produces the standard step curve that comes up in all introductory material to survival analysis.

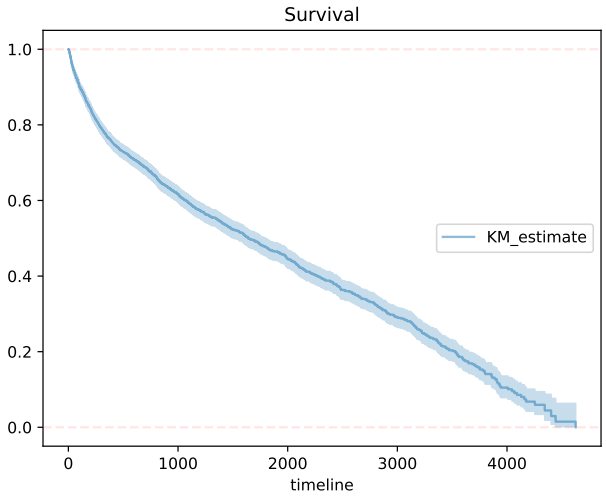

Figure 7: Kaplan-Meier curve of Survival for entire cohort

A.2 More Survival Analysis Basics

Classical expression for hazard is given as

$$h(t) = \lim_{\delta \to 0} \frac{P(t < T < t + \delta | T > t)}{\delta} \tag{23}$$

From equation 3, it can be further simplified as

$$h(t) = -\frac{d}{dt} S(t) \frac{1}{S(t)}$$
$$\implies S(t) = exp\left(-\int_0^t h(u)du\right) \tag{24}$$

This relationship produces the survival function from a corresponding hazard function [?].

$$H(u) = \int_0^t h(u)du$$

is known as the cumulative hazard, which is also extensively studied throughout the literature.

A.3 Cox Partial Likelihood

A general formulation for the Cox partial likelihood is given following [?]. Consider failure time t_i. The set of all subjects in the trial "at risk" for failure at this time is denoted by $j : T_j > T_i$, where T denotes the observed survival time. The probability of patient i failing at this time is

$$L_i = \frac{h(t_i|x_i)}{\sum_{j:T_j > T_i} h(t_i|x_j)} \tag{25}$$

where $h(t|x)$ denotes the hazard for patient with covariates x at time t. The likelihood of the entire cohort is thus,

$$L = \prod_{i:C_i=1} L_i = \prod_{i:C_i=1} \frac{h(t_i|x_i)}{\sum_{j:T_j > T_i} h(t_i|x_j)} \tag{26}$$

where $C_i = 1$ denotes an observed (uncensored) event. The assumption of a proportional hazards model reduces this to

$$L = \prod_{C_i=1} \frac{e^{w^T x_i}}{\sum_{j:T_j > T_i} e^{w^T x_j}} \tag{27}$$

It is noted that the baseline hazard h_0 is canceled from both the numerator and the denominator [?]. The log likelihood therefore becomes

$$\ell = \sum_{i:C_i=1} \left(w^T x_i - log \sum_{j:T_j > T_i} e^{w^T x_j} \right) \tag{28}$$

Interestingly, this negative of this log-likelihood is used as the loss function for DeepSurv [?] and the Faraggi-Simon net [?].

A.4 Loss Function(s)

Negative Log-Likelihood - hazard function
The following derivation is taken from [?]. For notational convenience, let $\kappa(t) \in \{0, \ldots, m\}$ define the index of the discrete time t, meaning $t = \tau_{\kappa(t)}$. Thus, the likelihood contribution for individual i is seen to be

$$\begin{aligned} L_i &= f(t_i)^{\delta_i} S(t_i)^{1-\delta_i} \\ &= [h(t_i) S(\tau_{\kappa(t_i)-1})]^{d_i} [(1 - h(t_i)) S(\tau_{\kappa(t_i)-1})]^{1-d_i} \\ &= h(t_i)^{d_i} [1 - h(t_i)]^{1-d_i} S(\tau_{\kappa(t_i)-1})] \\ &= h(t_i)^{d_i} [1 - h(t_i)]^{1-d_i} \prod_{j=1}^{\kappa(t_i)-1} [1 - h(\tau_j)] \end{aligned} \tag{29}$$

For all the individuals, the combined log-likelihood is

$$L = \prod_{i=1}^{n} L_i = \prod_{i=1}^{n} \left(h(t_i|x_i)^{d_i} [1 - h(t_i|x_i)]^{1-d_i} \prod_{j=1}^{\kappa(t_i)-1} [1 - h(\tau_j|x_i)] \right) \tag{30}$$

From here, the loss function for a batch can be constructed as the negative log likelihood.

$$log(L) = \sum_{i=1}^{n} \left(d_i log[h(t_i|x_i)] + (1 - d_i) log[1 - h(t_i|x_i)] + \sum_{j=1}^{\kappa(t_i)-1} log[1 - h(\tau_j|x_i)] \right) \tag{31}$$

To adjust for varying batch sizes, the mean negative log likelihood is taken as the loss.

$$\mathcal{L}_{nll} = -\frac{1}{n}\sum_{i=1}^{n}\left(d_i log[h(t_i|x_i)] + (1-d_i)log[1-h(t_i|x_i)] + \sum_{j=1}^{\kappa(t_i)-1} log[1-h(\tau_j|x_i)]\right)$$
$$= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{\kappa_{t_i}}(y_{ij}log[h(\tau_j|x_i)] + (1-y_{ij})log[1-h(\tau_j|x_i)]) \tag{32}$$

This can now be minimized by gradient-based methods, thus making it a useful loss function for a neural network to work with. Here, y_{ij} is an indicator variable corresponding to 1 if and only if an event is experienced by the individual i at time t_j. Formally, $y_{ij} = \mathbb{1}(t_i = \tau_j, d_i = 1)$. Hence, y is a sparse matrix consisting of mostly 0 with 1 only present when the time t_j represents an observed event $d_i = 1$ for individual x_i.

The loss can be thought of as the negative log likelihood of Bernoulli data $\theta^p(1-\theta)^{1-p}$, where $\theta = h(\tau_j|x_i)$ and $p = y_{ij}$ [**?**], and can be computed using existing functions in the PyTorch library.

Lower-Bound on C-Index
The following loss function is taken from [**?**]. The sigmoid function defined as $\sigma = (1+e^{-z})^{-1}$ is an approximation to the indicator function $\mathbb{1}_{z>0}$. However, it is not a lower bound. For this, the scaled version of its log is taken.

$$\mathbb{1}_{z>0} \geq log\ [2\sigma(z)]/log\ 2$$
$$\implies \mathbb{1}_{z>0} \geq 1 + (log\ \sigma(z)/log\ 2) \tag{33}$$

For a more rigourous treatment of this inequality, see the original paper [**?**]. It follows that the lower bound on the concordance index then becomes

$$c = \frac{1}{|\mathcal{E}|}\sum_{i:C_i=1}\sum_{T_j>T_i}\mathbb{1}_{\eta_j>\eta_i}$$
$$c \geq \frac{1}{|\mathcal{E}|}\sum_{i:C_i=1}\sum_{T_j>T_i} 1 + (log\ \sigma(\eta_j - \eta_i)/log\ 2) \tag{34}$$

where η_i is the survival probability of the i^{th} subject. It follows from the exposition around Equation 18 that the η_i can also represent survival times as c-index is rank-based. The negative of this lower-bound is chosen to be the loss function, which can therefore be minimized using gradient-based methods (again, with the help of a library like PyTorch).

$$\mathcal{L}_{lbo} = -\frac{1}{|\mathcal{E}|}\sum_{i:C_i=1}\sum_{T_j>T_i} 1 + (log\ \sigma(\eta_j - \eta_i)/log\ 2) \tag{35}$$

A.5 Statistical Test for Measuring Similarity of Curves

The Kolmogorv-Smirnov test [**?**] is used to test for this, with the null hypothesis being that the curves are 'similar', i.e. the generating process for both the curves is the same. As standard, two critical values of 0.01 and 0.05 are chosen. It is seen that the p-value for the test between CPH and the neural network is above 0.05; therefore the null hypothesis cannot be rejected and it follows that the baseline survival from a traditional fitter such as Cox Proportional Hazards is statistically similar to the 'expected' survival generated from the NN[9]. The case for comparison with the empirical distribution is more of an edge case, allowing rejection of the null hypothesis at critical value of 0.05 but not at 0.01.

B Tools Used

B.1 Multi Layer Perceptron

The multi-layer perceptron, arose from the field of psychology [**?**] formed the foundation of neural networks, although later implementations gradually tore away from the biological foundations and now bear only a slight similarity to the functioning of a human brain. For ordinary regression tasks, an input vector $\mathbf{x}$, weight vector $\mathbf{w}$ and bias vector $\mathbf{b}$,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

could be used to estimate a target variable $\mathbf{y}$ as

$$\mathbf{y} = \sigma(\mathbf{w}^T\mathbf{x} + \mathbf{b}) \tag{36}$$

where σ is the sigmoid function for logistic regression and just the identity for linear regression. Now, a perceptron without any hidden layers (as was the original) is effectively similar to ordinary regression. With the addition of hidden layers and non-linear activation functions, multi-layer perceptrons become much more powerful. This network is trained iteratively by backpropagating [**?**] errors through the connections over multiple epochs until some reasonable notion of convergence is achieved.

B.2 Convolutional Neural Networks

CNNs are a type of regularized feed-forward network that learn via filter (kernel) optimization. This type of network has roots in biological vision. The first application of backpropagation to CNNs was in [**?**]. They are useful for preserving spatial arrangement within the data and enforcing local connectivity. CNNs have found success in field of image and audio processing and even time series forecasting. Because of the shared-weight architecture, CNNs require far less connections than an equivalent fully-connected network.

[9] Refers to the expected survival curve generated from the time-invariant neural network in this study

C Diagnoses and Medication

C.1 ICD-10 Codes

Heart Failure

39891, 40200, 40201, 40210, 40211, 40290, 40291, 40400, 40401, 40402, 40403, 40410, 40411, 40412, 40413, 40490, 40491, 40492, 40493, 4280, 4281, 42820, 42821, 42822, 42823, 42830, 42831, 42832, 42833, 42840, 42841, 42842, 42843, 4289, E8726, E8745, I0981, I110, I119, I130, I131, I1310, I1311, I132, I50, I501, I502, I5020, I5021, I5022, I5023, I503, I5030, I5031, I5032, I5033, I504, I5040, I5041, I5042, I5043, I508, I5081, I50810, I50811, I50812, I50813, I50814, I5082, I5083, I5084, I5089, I509, I9713, I97130, I97131, T8622, T8632, Y625

Hypertension

G932, H4005, H40051, H40052, H40053, H40059, I10, I15, I150, I158, I159, I270, I272, I2720, I2721, I2722, I2723, I2724, I2729, I873, I8730, I87301, I87302, I87303, I87309, I8731, I87311, I87312, I87313, I87319, I8732, I87321, I87322, I87323, I87329, I8733, I87331, I87332, I87333, I87339, I8739, I87391, I87392, I87393, I87399, I973, K766, O10, O100, O1001, O10011, O10012, O10013, O10019, O1002, O1003, O104, O1041, O10411, O10412, O10413, O10419, O1042, O1043, O109, O1091, O10911, O10912, O10913, O10919, O1092, O1093, O11, O111, O112, O113, O114, O115, O119, O12, O13, O131, O132, O133, O134, O135, O139, O16, O161, O162, O163, O164, O165, O169, P292, P2930, R030

Obesity

E66, E6601, E6609, E661, E662, E668

Diabetes

E101, E09, E090, E0900, E0901, E091, E0910, E0911, E092, E0921, E0922, E0929, E093, E0931, E09311, E09319, E0932, E09321, E093211, E093212, E093213, E093219, E09329, E093291, E093292, E093293, E093299, E0933, E09331, E093311, E093312, E093313, E093319, E09339, E093391, E093392, E093393, E093399, E0934, E09341, E093411, E093412, E093413, E093419, E09349, E093491, E093492, E093493, E093499, E0935, E09351, E093511, E093512, E093513, E093519, E09352, E093521, E093522, E093523, E093529, E09353, E093531, E093532, E093533, E093539, E09354, E093541, E093542, E093543, E093549, E09355, E093551, E093552, E093553, E093559, E09359, E093591, E093592, E093593, E093599, E0936, E0937, E0937X1, E0937X2, E0937X3, E0937X9, E0939, E094, E0940, E0941, E0942, E0943, E0944, E0949, E095, E0951, E0952, E0959, E096, E0961, E09610, E09618, E0962, E09620, E09621, E09622, E09628, E0963, E09630, E09638, E0964, E09641, E09649, E0965, E0969, E098, E099, E10, E1010, E1011, E102, E1021, E1022, E1029, E103, E1031, E10311, E10319, E1032, E10321, E103211, E103212, E103213, E103219, E10329, E103291, E103292, E103293, E103299, E1033, E10331, E103311, E103312, E103313, E103319, E10339, E103391, E103392, E103393, E103399, E1034, E10341, E103411, E103412, E103413, E103419, E10349, E103491, E103492, E103493, E103499, E1035, E10351, E103511, E103512, E103513, E103519, E10352, E103521, E103522, E103523, E103529, E10353, E103531, E103532, E103533, E103539, E10354, E103541, E103542, E103543, E103549, E10355, E103551, E103552, E103553, E103559, E10359, E103591, E103592, E103593, E103599, E1036, E1037, E1037X1, E1037X2, E1037X3, E1037X9, E1039, E104, E1040, E1041, E1042, E1043, E1044, E1049, E105, E1051, E1052, E1059, E106, E1061, E10610, E10618, E1062, E10620, E10621, E10622, E10628, E1063, E10630, E10638, E1064, E10641, E10649, E1065, E1069, E108, E109, E11, E110, E1100, E1101, E111, E1110, E1111, E112, E1121, E1122, E1129, E113, E1131, E11311, E11319, E1132, E11321, E113211, E113212, E113213, E113219, E11329, E113291, E113292, E113293, E113299, E1133, E11331, E113311, E113312, E113313, E113319, E11339, E113391, E113392, E113393, E113399, E1134, E11341, E113411, E113412, E113413, E113419, E11349, E113491, E113492, E113493, E113499, E1135, E11351, E113511, E113512, E113513, E113519, E11352, E113521, E113522, E113523, E113529, E11353, E113531, E113532, E113533, E113539, E11354, E113541, E113542, E113543, E113549, E11355, E113551, E113552, E113553, E113559, E11359, E113591, E113592, E113593, E113599, E1136, E1137, E1137X1, E1137X2, E1137X3, E1137X9, E1139, E114, E1140, E1141, E1142, E1143, E1144, E1149, E115, E1151, E1152, E1159, E116, E1161, E11610, E11618, E1162, E11620, E11621, E11622, E11628, E1163, E11630, E11638, E1164, E11641, E11649, E1165, E1169, E118, E119, E13, E130, E1300, E1301, E131, E1310, E1311, E132, E1321, E1322, E1329, E133, E1331, E13311, E13319, E1332, E13321, E133211, E133212, E133213, E133219, E13329, E133291, E133292, E133293, E133299, E1333, E13331, E133311, E133312, E133313, E133319, E13339, E133391, E133392, E133393, E133399, E1334, E13341, E133411, E133412, E133413, E133419, E13349, E133491, E133492, E133493, E133499, E1335, E13351, E133511, E133512, E133513, E133519, E13352, E133521, E133522, E133523, E133529, E13353, E133531, E133532, E133533, E133539, E13354, E133541, E133542, E133543, E133549, E13355, E133551, E133552, E133553, E133559, E13359, E133591, E133592, E133593, E133599, E1336, E1337, E1337X1, E1337X2, E1337X3, E1337X9, E1339, E134, E1340, E1341, E1342, E1343, E1344, E1349, E135, E1351, E1352, E1359, E136, E1361, E13610, E13618, E1362, E13620, E13621, E13622, E13628, E1363, E13630, E13638, E1364, E13641, E13649, E1365, E1369, E138, E139, N251, O240, O2401, O24011, O24012, O24013, O24019, O2402, O2403, O241, O2411, O24111, O24112, O24113, O24119, O2412, O2413, O243, O2431, O24311, O24312, O24313, O24319, O2432, O2433, O244, O2441, O24410, O24414, O24415, O24419, O2442, O24420, O24424, O24425, O24429, O2443, O24430, O24434, O24435, O24439, O248, O2481, O24811, O24812, O24813, O24819, O2482, O2483, O249, O2491, O24911, O24912, O24913, O24919, O2492, O2493, P700, P702, R7303, Z131, Z833, Z8632

Medication

Type	Medication Name
Calcium channel blocker	nifedipine, nisoldipine, verapamil, amlodipine, diltiazem, felodipine, isradipine, nicardipine
ARB	losartan, olmesartan, telmisartan, valsartan, azilsartan, candesartan, eprosartan, irbesartan
ACE inhibitor	lisinopril, moexipril, perindopril, quinapril, ramipril, trandolapril, benazepril, captopril, enalapril, fosinopril
beta blocker	bisoprolol, metoprolol, nadolol, nebivolol, propranolol, atenolol, acebutolol

Table 6: Instances of patients being administered these medicines are included in the model(s).

C.2 Miscellaneous

For the experiments involving Time-Invariant or Time-Variant models, here are the configurations:

- Optimizer - Adam
- Early Stopping - with patience 20; Refer Table 4
- Train-Test-Validation Split - 0.6:0.2:0.2