Finding differences between discrete-time deep learning survival models

Piotr Magnuszewski
Bialystok University of Technology
Bialystok, Poland *piotr.magnuszewski@sd.pb.edu.pl*

Malgorzata Kretowska
Bialystok University of Technology
Bialystok, Poland *m.kretowska@pb.edu.pl*

Abstract

The use of deep learning methods has gained momentum in the domain of survival analysis. Different models have been proposed to handle time-to-event data. Neural networks are used to find complex relationships between features, improving the predictive capabilities of deep learning models. When conducting experiments, one might want to reduce the number of methods that need to be examined because of the computational resources required for model training. Establishing families of deep learning methods that behave in a similar way might be beneficial for such scenarios. In this paper, we establish a way to measure differences between deep learning discrete-time survival analysis models. The proposed method is based on SHAP values. We conducted experiments for three datasets and five discrete-time survival analysis models. We proposed a special kind of plot that helps visualize the impact of features on the model outputs over time intervals. Based on the obtained results, we performed Friedman and Wilcoxon tests to examine statistically significant differences between the models.

Keywords: survival analysis, discrete survival time, SHAP values, deep learning, time-dependent discrimination index

1. Introduction

Certain problems require us to predict the times it will take for an event to occur. It could be patient survival, machine part failure, or customer churn. Usually, we work with data samples that are described by covariates and time. In practice, for some of the data, we might not have an event observed. Various methods have been proposed for handling samples with incomplete observations (censored samples). The most well-known include the Cox proportional hazards (Cox PH) model [2]. There are various downsides to using strictly statistical methods. Most importantly, basic models might not be able to find complex relationships between features. The limitations stem from the model's construction, e.g. the assumption of linear effects of features. To overcome such issues, new approaches to solve the problem are proposed.

Recently, a growing number of deep learning methods have emerged among new models [18]. These methods are used to analyze standard censored survival data, where time is treated as a continuous variable and a single type of event is considered (e.g., predicting the survival of brain cancer patients based on MRI images [20] or forecasting postoperative survival for patients with gastric cancer [19]). Additionally, models have been proposed for predicting the time to more than one type of event (known as competing risks data) [13], as well as discrete-time models that divide survival time into distinct intervals, treating time as a discrete variable and focusing on the occurrence of the event of interest within these intervals [16]. Notable models in this area include DeepHit [13], DeepSurv [9], PMF [11], Nnet-survival [11], the neural multi-task logistic regression model [4], or BCESurv model [12].

The downside of deep learning is that models are usually difficult to explain. This means that we cannot easily find the reason why a given method produced specific predictions. To solve this problem, several tools have been proposed, including ICE [5], LIME [15] and SHAP [14]. The SHAP values represent the impact each feature had on output values relative to the average output values. The advantage of SHAP is that it can be applied to various models as it does not impose restrictive limitations.

In this work, we present preliminary results of comparisons between deep learning discrete-time survival analysis models using values derived from SHAP. It could be beneficial to construct families of models for which features have similar effects on the model outputs. The comparisons made in this work are based on the percentage contributions of the mean absolute SHAP values of the features. For better visualization, we prepared special plots that capture the contribution percentages with Kendall's Tau correlation (named "arrow plots"). We performed the Friedman test and the Wilcoxon signed rank test to examine statistically significant differences between the models. The analysis was performed on three real datasets.

The rest of the work is divided into 6 sections. In Section 2 we briefly explain what survival analysis is. Then in Section 3 we describe deep learning methods used in the domain of discrete-time survival analysis. In Section 4 we introduce the key concepts of model explainability and in Section 5 model performance evaluation. In Section 6, experiments are described and results presented. We finish with Section 7, where we share the conclusions and further steps.

2. Survival analysis

Survival analysis, or time-to-event analysis [13], refers to a set of statistical methods used to examine the time until a specific event occurs. Each data sample includes a defined start time (the beginning of observation) and, ideally, an end time marking the event. The primary focus is the duration between these two points. However, in real-world scenarios, either the event time or even the start time may be unobserved, which is known as censoring.

Censoring is an important aspect of survival analysis, as it reflects incomplete observation of the event of interest. Two primary types are typically distinguished: left-censoring, which occurs when the event takes place before the onset of observation, and right-censoring, when the event has not occurred by the end of the observation period. The latter is common in practice and may result from the termination of the study before the event is observed or the loss of follow-up. Importantly, censored observations should not be excluded from analysis, as they provide valuable partial information on survival time and contribute meaningfully to model robustness.

In survival analysis, each data sample is typically characterized by three components: covariates x, time t, and a censoring indicator δ. Given a dataset with N observations, let i denote the index of a sample. Then, each sample can be represented as (x_i, t_i, δ_i), where x_i denotes the feature vector, t_i represents the observed time defined as the duration from the start of observation to either the occurrence of the event or censoring and $\delta_i \in \{0, 1\}$ is the event indicator, with $\delta_i = 0$ signifying that the observation was censored, and $\delta_i = 1$ otherwise.

Two fundamental functions are commonly used in survival analysis [2, 18]: the *survival function* and the *hazard function*. The survival function describes the probability of surviving beyond time t. It is formally defined as $S(t) = \Pr(T > t)$, where T is a non-negative random variable representing the time until the event. Intuitively, $S(t)$ can be interpreted as the proportion of individuals in the original population who are expected to survive beyond time t.

The *hazard function* $\lambda(t)$ (also denoted as $h(t)$) characterizes the probability of the event occurring at time t, conditional on the event not occurring up to that time. It is formally defined as

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{\Pr(t \leq T < t + \Delta t \mid T \geq t)}{\Delta t}. \tag{1}$$

As Δt approaches zero, the hazard function yields the instantaneous failure rate at time t.

3. Deep learning discrete-time survival analysis methods

In addition to purely statistical approaches, researchers have tried to use deep learning methods in the domain of survival analysis. The key idea behind deep learning is the use of neural networks that are capable of finding complex relationships between the data. Machine learning methods have been argued to often outperform statistical models when it comes to predictive power [18].

Oftentimes, we consider the events on a discretized time scale [16]. There are several reasons for dividing time into intervals. Sometimes studies are conducted in such a way that samples are gathered at specific times [17]. Apart from that, time discretization can simplify the survival prediction problem. Existing deep learning architectures can be adapted to predict probability mass functions or discretized hazard functions (as in [11]). In this work we will denote the number of time intervals by J.

The PMF model [11] uses a neural network to parameterize a probability mass function f for discrete-time data. The network has J outputs, which correspond to the number of time intervals. The network outputs are described by $\phi(x) = \{\phi_1(x), \cdots, \phi_J(x)\}$, where each output represents the value of f for a given time interval. The value of the probability mass function f represents the chance that an event will occur at time t_j. It has been noted that the probability mass function of the PMF model is equivalent to a softmax function with ϕ_{J+1} set to 0 [11]. This would mean that the neural network has an additional output that represents a time interval past any observed event or censoring. If we set ϕ_{J+1} to 0 then we effectively disallow the network from predicting events past a certain time.

The LogisticHazard model [11] (also known as Nnet-survival) applies a neural network to parametrize a hazard function λ. The network has J outputs that are transformed by a sigmoid function. The application of the sigmoid function ensures that the output values of the network will be in the range of (0,1). The transformed network outputs correspond to the values of hazard function for each time interval. The resulting hazard function is given by [11]:

$$\lambda(t_j|x) = \frac{1}{1 + \exp[-\phi_j(x)]}, \tag{2}$$

where t_j represents the j-th time interval, x is a feature vector, and ϕ_j is the j-th output of the neural network.

DeepHit [13] is a discrete-time survival analysis model. It uses a neural network to parameterize a joint probability distribution function. What distinguishes DeepHit from other models is its ability to handle data with more than one event of interest ($K \geq 1$). Note that DeepHit treats censoring as a separate type of event. The network's architecture consists of K risk-specific subnetworks. The subnets are connected using an additional shared network. It allows the model to learn the relationships that are common among risks. DeepHit uses a single output softmax layer to ensure that the model learns the joint distribution of events y [13], given by

$$y = [y_{1,1}, \cdots, y_{1,J}, \cdots, y_{K,1}, \cdots, y_{K,J}]. \tag{3}$$

The N-MTLR (Neural Multi-Task Logistic Regression) was introduced in [4]. It is based on the MTLR model [21], which uses logistic regression to approximate the survival function. To better understand N-MTLR, one might first want to learn about MTLR. The main premise of the MTLR method is to consider a simple classification task for each of the J time intervals. A logistic regression model can be fitted to determine the probability of survival beyond time t_j, given the feature vector x. The MTLR model contains J logistic regression modules, one for each time interval. It has been noted that the outputs of the modules are not independent. The authors of MTLR have encoded the survival times of samples as a binary survival sequence y consisting of J elements from $\{0, 1\}$ [21]. Starting from the time interval t_j, the sequence is filled with ones. The time intervals before t_j are filled with zeros. MTLR learns the joint predictions of survival status by considering the probability of observing specific survival sequences. This is done by solving an optimization problem that includes both the log-likelihood and regularization terms. The N-MTLR model introduces greater flexibility in terms of the effects of features by using a neural network instead of the linear combination of features from MTLR. As noted by the author of N-MTLR, the main advantage of this method is the introduction of greater flexibility compared to MTLR.

The BCESurv model [12] stems from the concept of binary classifiers. When considering survival analysis, one might try to fit a binary classifier for the time interval t_j to determine the samples that experience the event or not. Expanding on the idea, we might construct an ensemble of classifiers, one for each time interval. When considering an interval t_j, BCESurv discards samples that were censored before t_j. Instead of a classifier ensemble, BCESurv uses a neural network. The network has J outputs, one for each time interval. The method estimates the probabilities for each time interval simultaneously. The authors of the method proposed a special loss function, based on which the model is fitted to the data.

4. Model explainability

Deep learning methods can often be perceived as a black box, meaning that we cannot easily tell why a model has produced certain outputs for given data samples. Efforts have been made to construct tools that might help us to understand the behavior of the models. It should be noted that explainability is not the same as interpretability. The former tries to justify why a model has produced certain output values without looking inside the model, while the latter concerns the understanding of inner workings of a model and the ability to follow the process that resulted in given output values. Several methods for model explainability have been proposed, including the Individual Contribution Expectation (ICE) method [5], Local Interpretable Model-agnostic Explanations (LIME) [15] and Shapley Additive Explanations SHAP values [14]. In our work, we will focus on the last one.

The SHAP values proposed in [14] are used to explain the behavior of various machine learning models. The method is based on Shapely values which are a game-theoretic approach for explaining contributions among "players" who cooperate towards a certain goal. SHAP examines the individual impact of each feature on the model's output values. The strength of this method lies in its ability to explain various types of deep learning models.

SHAP values are measured relative to baseline (or average) output values. This means that we have to select a set of background samples based on which the average values will be calculated and a set of samples for which to get the SHAP values. As a result, we get a set of explanations. For each of the input samples, SHAP produces an impact value the feature's value had on the model's output. It is possible to use SHAP on models that have multiple outputs.

An implementation of SHAP for the Python programming language is available in the `shap` package. It provides all the necessary classes used for calculation and visualization purposes. The `shap` package provides an implementation of `Explainer` class that expects a model and background samples as input. After being initialized, a set of samples must be provided for which to produce SHAP values. After having prepared the explanation values, one might want to visualize them. The `shap` package provides several tools for plotting the data. These include the "beeswarm" plot and the "waterfall" plot.

5. Model performance evaluation

In various situations, we might want to compare different survival analysis models with each other. This can be done using various metrics that are specifically designed for the types of predictions that are considered in the domain of survival analysis. The most common ones include the Harrel's C-index [7], the integrated Brier score (IBS) [6], and the time-dependent concordance index C^{td} [1]. In the context of this work, C^{td} will be considered.

C^{td} was first introduced in [1] as a tool to examine concordance. The main premise behind the method is to measure model's discrimination ability in contexts where certain assumptions are not held. The authors note that C^{td} is an extension of the C-index. C^{td} takes into account the predicted survival functions of data samples. The values get aggregated over time.

The authors of C^{td} define it as a weighed average of AUC values. For each time interval, the probability of correctly predicting survival function values for sample pairs is checked. Only pairs in which one sample experiences the event and the other does not are considered. The AUC of C^{td} is defined by [1]:

$$AUC(t_{(k)}) = Pr\{S(t_{(k)}|X_i(t)) < S(t_{(k)}|X_j(t))|D_i(t_{(k)}) = 1\&D_j(t_{(k)}) = 0\}, \tag{4}$$

where $D_a(t_{(k)})$ is an indicator representing whether a sample experienced the event at time $t_{(k)}$, $\&$ represents an intersection operator, and $S(t_{(k)}|X_a(t))$ is a survival function value at time $t_{(k)}$ given time dependent covariates $X_a(t_{(k)})$ of a sample. Based on the AUC values, we can calculate the value of C^{td} for the whole time range using formula given in [1]:

$$C^{td} = \frac{\Sigma_{k=1}^{K} AUC(t_{(k)}) \cdot w(t_{(k)})}{\Sigma_{k=0}^{K} w(t_{(k)})}. \tag{5}$$

The weight associated with the given time interval is equal to the probability of selecting a pair of samples where one experiences the event at time $t_{(k)}$ and the other does not (and did not experience it prior to $t_{(k)}$).

6. Experiments

Our experiments were performed for 3 datasets: METABRIC, SUPPORT, and FLCHAIN. Data were transformed so that categorical features were encoded and numerical ones standardized. We selected five models for testing. These were PMF, LogisticHazard, DeepHit, N-MTLR and BCESurv. Each of the models was trained 200 times. During the process, we collected SHAP values and Kendall's Tau correlation coefficients. Based on the results, we performed statistical tests to determine the degree of differences between the models.

6.1. Datasets

All datasets have been taken from the `pycox` Python package (version 0.3.0). We selected the METABRIC, the SUPPORT and the FLCHAIN datasets. They often appear in literature (e.g. [9, 11, 13]), and can be regarded as benchmarks in the domain of survival analysis. We encoded categorical features using the one-hot encoding method. An example of such a feature is "race" from the SUPPORT dataset. Binary features were kept unchanged. Numerical features were standardized.

The METABRIC (The Molecular Taxonomy of Breast Cancer International Consortium) dataset contains information about breast cancer patients. More than half of the cases have an observed death event. Each of the patients is described by a gene and protein expression profile. The dataset contains 1904 samples, each having 9 features. These include values like age and hormone treatment indicators. No feature values are missing. Around 42% of samples are censored.

The SUPPORT (Study to Understand Prognoses Preferences Outcomes and Risks of Treatment) dataset contains information about hospitalized patients. In total, there are 8873 samples, each having 14 features. No values are missing because the dataset has been preprocessed as described in [9]. Patients are characterized by features such as age, heart rate, temperature, and white blood cell count. In the dataset, approximately 32% of samples do not have an event observed (are censored).

The FLCHAIN (Free Light Chain) dataset is based on patient data from Olmsted County, Minnesota. There are 6524 samples, each having 8 features. For the purpose of our work, we have removed the `sample.yr` feature, which represents the year in which the blood sample was collected from a patient. FLCHAIN has the highest percentage of censored samples ($\sim$70%) among the datasets used in our work.

6.2. Structure of the experiments

The experiments were constructed using Python (version 3.12.3). We have selected 5 models from the `pycox` package (version 0.3.0). These were DeepHit, PMF, N-MTLR, BCESurv, and LogisticHazard. We also used the `shap` package (version 0.47.0) for SHAP values.

Each of the models was parameterized with a neural network with two hidden layers, each having 32 neurons. Batch normalization was set to true and dropout rate was set to 0.1. For each model, the learning rate was selected based on learning rate finders provided by the `pycox` package. The model training process included a callback for early stopping. The number of epochs was limited to 256. Each model was trained for 200 rounds. In each round, the datasets were randomly split into training-validation-test subsets (in the proportion of 8:1:1). All models were trained on the same data split for a given round. Both training and validation sets were used in the model fitting process. All of the selected models required discrete-time samples. For that reason, the duration values from the datasets had to be divided into time intervals. In our experiments, the number of time spans has been set to 10. These were named starting from `t0` to `t9`. We used the discrete time label transformer from the `pycox` package with the discretization scheme set to "equidistant".

In each training round, the SHAP values and the C^{td} values were collected. The latter ones were calculated based on predictions made by the models for test sets. Each model produced a survival function for data samples. Greater function values represent higher chance of survival.

`Explainer` objects from the `shap` package were used to determine the impact of each feature on outputs of the models. For this purpose a set of background samples had to be selected. We sourced 200 samples from the training set. Furthermore, samples for SHAP value calculation had to be provided to the `Explainer` object. These were also selected from the training set, but avoiding an overlap with the background samples. The use of training sets as the basis for the calculation of SHAP values ensures that the characteristics of the data on which the models were trained are reflected in the results.

Based on the SHAP values, we calculated the mean absolute values for each feature. They can be perceived as the magnitude of impact the feature has on the model's outputs. The mean absolute value can be calculated using the formula:

$$\mu(S) = \frac{\Sigma_{i=1}^{n} |s_i|}{n}, \tag{6}$$

where S represents the SHAP values, s_i is the i-th SHAP value from S, and n is the number of values in S.

Based on the mean absolute SHAP values, we calculated the contribution percentages each feature had on the models' outputs. The percentages can be calculated using the formula:

$$c_m = \frac{\mu(S_m)}{\Sigma_{k=1}^{M} \mu(S_k)}, \tag{7}$$

where c_m is the m-th feature's contribution percentage, S_m are the SHAP values obtained for the m-th feature, and M is the number of features. We can observe that contribution percentages of the features add up to 1.

Apart from the percentages, we also gathered Kendall's Tau correlation [10] values for each of the features. Kendall's Tau can be viewed as a measure of concordance or agreement between pairs of numbers in two rankings [8]. It takes into account the relative ordering of value pairs. Kendall's Tau is based on the number of concordant and discordant pairs. The correlation values range from -1 to 1, where -1 represents complete disagreement and 1 is perfect agreement. The value of 0 signifies that there is no correlation between the features' values. An advantage of Kendall's Tau is that it can be easily interpreted and might be more preferable to Spearman's rank [3]. The correlation values in our experiments were calculated between the SHAP and the feature values. The results were averaged over the training rounds to reduce variance. An overview of the value collection process can be seen in Figure 1.

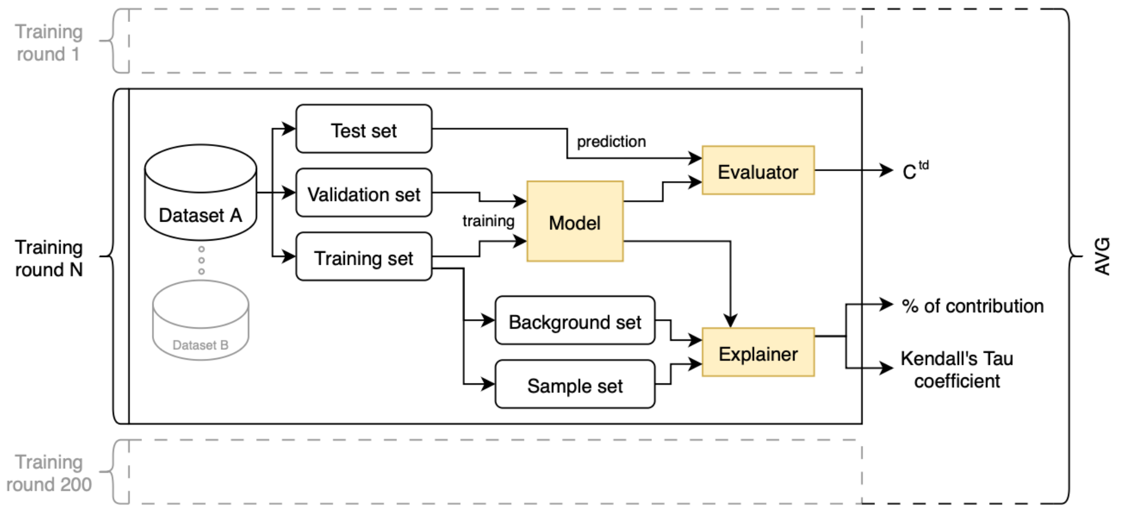

Fig. 1. The outline of value collection part of the experiments

We applied the Friedman test to the obtained feature contribution percentages of the models. For multiple comparisons, the Wilcoxon signed-rank test with the Bonferroni correction was used. A significance level of 0.05 was adopted for all of the tests. We calculated the percentage of statistically significant differences between the models for all pairwise comparisons.

6.3. Results

Each of the selected models was evaluated on a test set. We gathered C^{td} values that were averaged over training rounds. Standard deviation values were also collected. The results we obtained for each dataset are presented in Table 1.

Table 1. The C^{td} results obtained for the three datasets

	METABRIC		SUPPORT		FLCHAIN	
Model	mean C^{td}	std C^{td}	mean C^{td}	std C^{td}	mean C^{td}	std C^{td}
LogisticHazard	<u>0.5353</u>	0.0636	**0.5010**	0.0248	<u>0.7519</u>	0.0210
N-MTLR	0.5655	0.0345	0.4536	0.0167	**0.7857**	0.0175
DeepHit	**0.6129**	0.0540	0.4828	0.0396	0.7775	0.0271
PMF	0.5670	0.0338	<u>0.4516</u>	0.0202	0.7831	0.0203
BCESurv	0.5833	0.0360	0.4576	0.0229	0.7679	0.0271

In the table, we have marked the highest mean C^{td} scores for each of the datasets with bold font and the lowest with an underline. We can see that for the METABRIC dataset, DeepHit performed the best and LogisticHazard the worst. When considering the standard deviation of C^{td}, LogisticHazard varied the most between the training rounds, with DeepHit taking second place. When considering the SUPPORT dataset, we see that the LogisticHazard model performed the best, based on the C^{td} value. DeepHit was the second best and PMF had the worst score amongst the models. For the FLCHAIN dataset, the LogisticHazard model scored the lowest C^{td} value among the models and N-MTLR the highest. The results suggest that, depending on the selected dataset, different models might perform the best and that the LogisticHazard model tends to have the lowest score values. One might want to avoid models that tend to give poor results.

As a result of the experiments, we prepared visualizations (named "arrow plots") that captured both the contribution percentages c_m of the features (points) and the correlation values between the features and the SHAP values (arrows). A separate plot was prepared for each feature. The reason for creating the visualizations was to make it easier to observe the differences between the models. Two arrow plots for the METABRIC dataset can be seen in Figures 2 and 3. The x-axis represents the time intervals. On the y-axis, feature contribution percentages are presented. The arrows on the plots correspond the the Kendall's Tau correlation values.

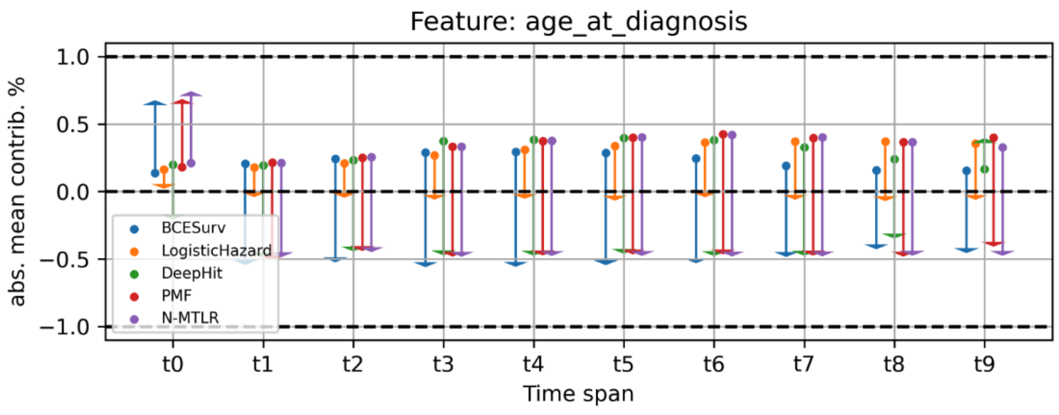

Fig. 2. The arrow plot obtained for the **age_at_diagnosis** feature of the METABRIC dataset

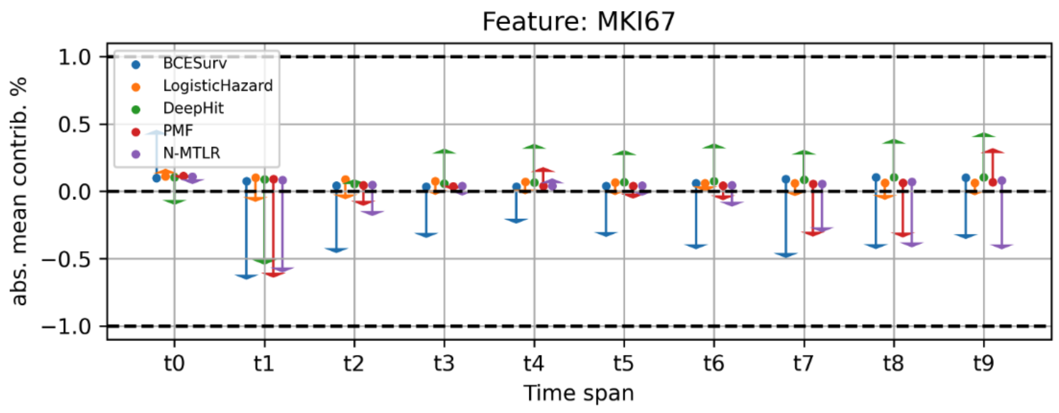

Fig. 3. The arrow plot obtained for the **MKI67** feature of the METABRIC dataset

In Figure 2, we can observe that the age at diagnosis had high contribution percentages for most of the time spans and models. Apart from time interval $t0$, all models had strong negative correlation for the age feature and model output values. In time intervals $t7$ to $t9$ we can see that there were larger differences between the models than for earlier time spans. Based on the direction of arrows we can infer that with an increase in age of a patient, the survival curve estimate values diminished. That might seem intuitive, as with greater age the human immune system might become weaker and the body might not be able to fight with an illness. Furthermore, we can see that in the first time interval there is a disagreement between the models. BCESurv, PMF, and N-MTLR have positive correlation values.

In contrast to Figure 2, we can see that the contribution percentages presented in Figure 3 have smaller values (below 25%). Furthermore, there seems to be a disagreement between the models when considering the correlation coefficient values. The arrow plot might suggest that the relationship between MKI67 gene expression indicator and the outputs of the models is non-obvious. In addition, we can observe that for most of the time intervals, the pair BCESurv-DeepHit had the largest differences in the correlation values between each other. For most of the intervals, DeepHit had on opposite correlation value sign than other models.

In addition to arrow plots for individual features of the METABRIC dataset, we prepared arrow plots for time intervals. A plot for the $t8$ time span can be seen in Figure 4. We might

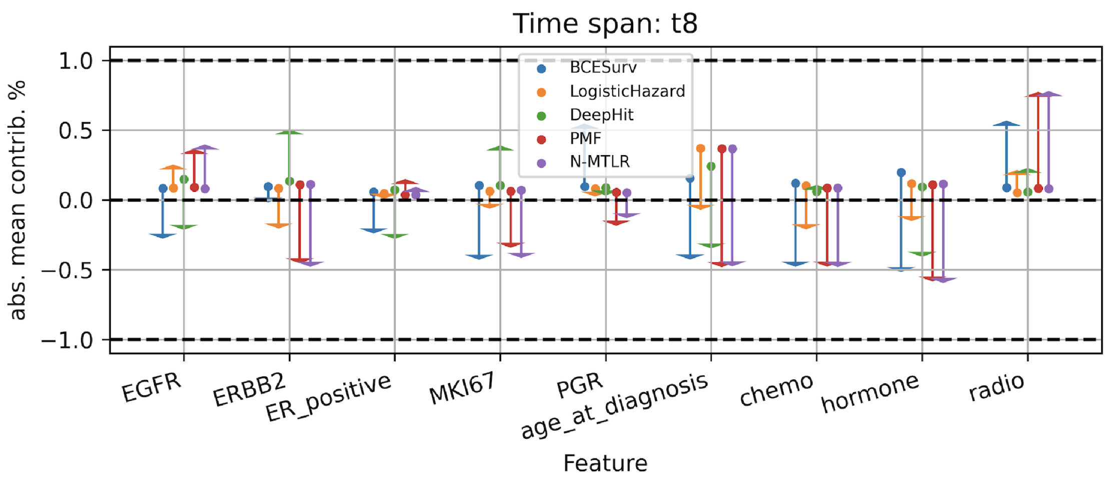

Fig. 4. The arrow plot for the $t8$ time interval for the METABRIC dataset

observe that for some of the features there is agreement between the models as to the impact of the feature on the survival probability. Age at diagnosis, hormone therapy indicator, and radiotherapy indicator have high correlation values. LogisticHazard and DeepHit had lower values than other models for these features. Furthermore, there is a disagreement between the models for the remaining features.

During our experiments, we also prepared arrow plots for the SUPPORT dataset. The plots can be seen in Figures 5 and 6. Based on them, we can observe that majority of the models seem to agree as to the influence of the age feature on model outputs. The arrows point downwards for most of the time intervals, meaning that the correlation coefficients were negative. It seems probable that with an increase in age, the probability of survival decreases. We can see that most of the disagreement between the models is present in the $t0$, $t8$ and $t9$ time spans. Taking into account Figures 2 and 5, we might observe that in both cases the most divergent results are present in the first time interval $t0$.

Figure 6 presents the contribution of respiration rate to the model outputs. We can see that the percentage values are close to zero. This means that the respiration rate feature has little effect on the model outputs. Despite the correlation values being low, there seems to be a positive trend, due to arrows pointing upwards, most often for BCESurv and DeepHit. It might be possible that those patients who have a higher respiration rate tend to have slightly better survival outcomes.

Apart from arrow plots, we performed Friedman tests and Wilcoxon pairwise comparison tests. The former gave statistically significant results for all features apart from radiotherapy

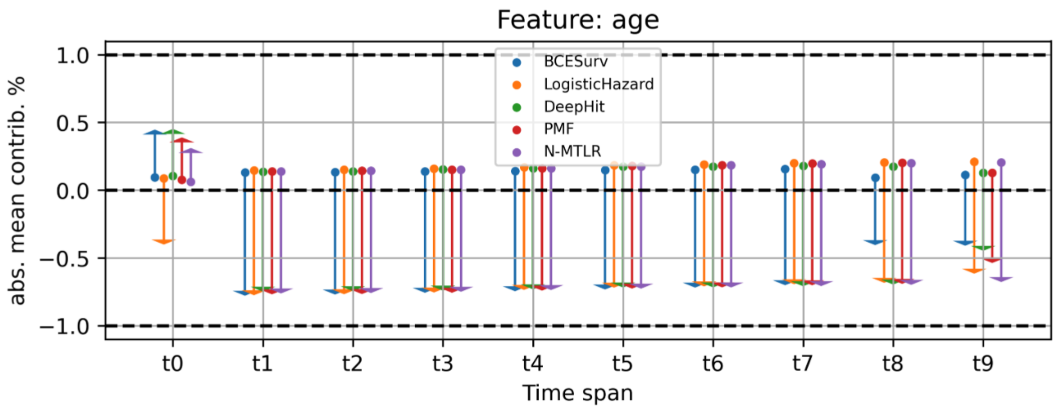

Fig. 5. The arrow plot obtained for the **age** feature of the SUPPORT dataset

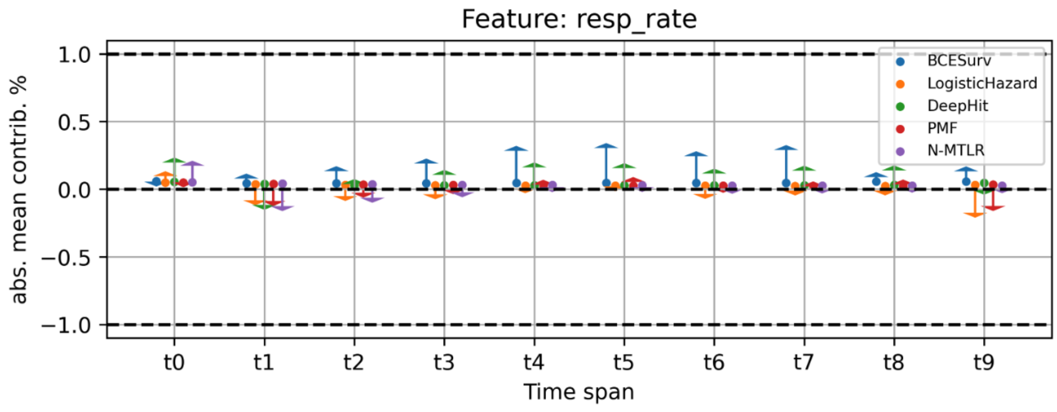

Fig. 6. The arrow plot obtained for the **resp_rate** (respiration rate) feature of the SUPPORT dataset

indicator from the METABRIC dataset. The Wilcoxon tests were based on the feature contribution percentages. The obtained results are presented in Figure 7 in the form of three heatmaps. Each cell contains a percentage value, calculated as the fraction of all of the Wilcoxon tests

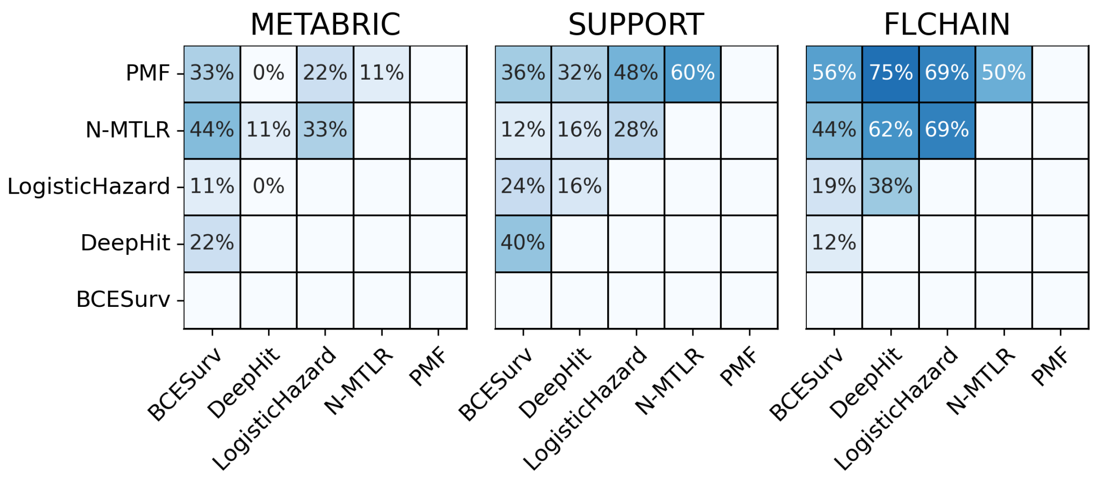

Fig. 7. The percentages of statistically significant differences between the models

performed between a pair of models that were statistically significant. We can observe that there are contrasting values for model pairs between datasets, especially for PMF-DeepHit, NMTLR-BCESurv, and PMF-NMTLR. It is interesting to see that the pair LogisticHazard-BCESurv scored one of the lowest percentage values for each dataset. This might suggest that there are similarities in how both of those models generate predictions based on the data. What is more, we can observe that for the FLCHAIN dataset, many model pairs scored high difference percentages. The FLCHAIN dataset had the highest percentage of censored data (around 70%) among the examined datasets. This might suggest that the fraction of censored data in a dataset significantly impacts the model training process.

7. Conclusions

The goal of our work was to examine differences between discrete-time deep learning survival models. We prepared a special kind of visualization, named arrow plots, which were used to help with the observation of feature importance values and correlation values. Additionally, we performed Friedman and Wilcoxon tests to find whether there were any significant differences between the models.

Based on the results presented in out work, we might observe that there are differences between the models. Analyzing the outcomes of statistical comparisons, one can see that some model pairs, namely PMF-DeepHit, NMTLR-BCESurv, and PMF-NMTLR, had one of the highest percentage value changes between tested datasets. We could observe that the percentages of differences between the models tend to become higher for the FLCHAIN dataset, which had the highest percentage of censored samples. This could suggest that the degree of censoring of a dataset has significant impact on the model training process.

The arrow plots presented in our work have shown that for some time intervals DeepHit had an opposite correlation value sign than other models. Despite this, DeepHit was able to score one of the best scores based on C^{td} for all three datasets. It could suggest that, when compared to other models, DeepHit is able to find different relations that might positively impact the predictive capabilities of the model. Another observation based on the arrow plots is that the contribution percentage values might have a tendency to diverge towards the first and the last time intervals. In addition, the correlation values in these intervals might have a different sign compared to other time spans. This could be due to the equidistant division of time into intervals. It might also suggest that it is particularly difficult to correctly model the most extreme time intervals.

Our method can be applied to various survival analysis models. The flexibility of use can be regarded a strength. However, the method requires the time to be divided into discrete intervals. Moreover, our method does not calculate the precise degree of similarity between the models but only the percentage of statistically significant differences. It could be beneficial to further develop the similarity measure. The results of our experiments provide early insight into the comparison of different survival models which could pave the way for more extensive research.

In continuation of this work, we might test additional model explainability methods and employ other approaches to measure the degree of differences between models. Apart from that, it could be beneficial to collect measurements for additional datasets and allow the selection of the most optimal model hyperparameters (e.g. via grid-search). More work is needed to establish whether we can use model explainability methods to judge the differences between discrete-time survival analysis models.

Acknowledgments

This work was supported by Bialystok University of Technology under the grant WZ/WI-IIT/4/2023 founded by Ministry of Science and Higher Education. GitHub Copilot was used during the code writing process.

References

1. Antolini, L., Boracchi, P., Biganzoli, E.: A time-dependent discrimination index for survival data. Stat Med. 24 (24), 39273944 (2005)
2. Cox, D.R.: Regression models and life-tables. Journal of the Royal Statistical Society Series B: Statistical Methodology. 34 (2), 187202 (1972)
3. Croux, C., Dehon, C.: Influence functions of the Spearman and Kendall correlation measures. Stat Methods Appt. 19 (4), 497515 (2010)
4. Fotso, S.: Deep neural networks for survival analysis based on a multi-task framework. (2018)
5. Goldstein, A., Kapelner, A., Bleich, J., Pitkin, E.: Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. J. Comput. Graph. Statist. 24 (1), 4465 (2015)
6. Graf, E., Schmoor, C., Sauerbrei, W., Schumacher, M.: Assessment and comparison of prognostic classification schemes for survival data. Stat Med. 18 (1718), 25292545 (1999)
7. Harrell, F.E.: Evaluating the yield of medical tests. JAMA: The Journal of the American Medical Association. 247 (18), 2543 (1982)
8. Hauke, J., Kossowski, T.: Comparison of values of pearsons and spearmans correlation coefficients on the same sets of data. QUAGEO. 30 (2), 8793 (2011)
9. 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 Med Res Methodol. 18 (1), 24 (2018)
10. Kendall, M.G.: A new measure of rank correlation. Biometrika. 30 (12), 8193 (1938)
11. Kvamme, H., Borgan, Ø.: Continuous and discrete-time survival prediction with neural networks. Lifetime Data Anal. 27 (4), 710736 (2021)
12. Kvamme, H., Borgan, Ø.: The brier score under administrative censoring: Problems and solutions. (arXiv:1912.08581), (2019)
13. Lee, C., Zame, W., Yoon, J., Van Der Schaar, M.: DeepHit: A deep learning approach to survival analysis with competing risks. Proceedings of the AAAI Conference on Artificial Intelligence. 32 (1), (2018)
14. Lundberg, S., Lee, S.-I.: A unified approach to interpreting model predictions. (2017)
15. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you?: explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 11351144. ACM, San Francisco California USA (2016)
16. Tutz, G., Schmid, M.: Modeling discrete time-to-event data. Springer International Publishing, Cham (2016)
17. Wen, C., Chen, Y.: Discrete-time survival data with longitudinal covariates. Stat Med. 39 (29), 43724385 (2020)
18. Wiegrebe, S., Kopper, P., Sonabend, R., Bischl, B., Bender, A.: Deep learning for survival analysis: a review. Artif Intell Rev. 57 (3), 65 (2024)
19. Wu, M., Yang, X., Liu, Y., Han, F., Li, X., Wang, J., Guo, D., Tang, X., Lin, L., Liu, C.: Development and validation of a deep learning model for predicting postoperative survival of patients with gastric cancer. BMC Public Health. 24 (1), 723 (2024)
20. Xu, X., Prasanna, P.: Brain Cancer Survival Prediction on Treatment-Naïve MRI using Deep Anchor Attention Learning with Vision Transformer. In: 2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI). pp. 15. IEEE, Kolkata, India (2022)
21. Yu, C.-N., Greiner, R., Lin, H.-C., Baracos, V.: Learning patient-specific cancer survival distributions as a sequence of dependent regressors. In: Proceedings of the 25th International Conference on Neural Information Processing Systems. pp. 18451853. Curran Associates Inc., Granada, Spain (2011)