DEEP SURVIVAL & PERSONALIZED TREATMENT RECOMMENDER SYSTEM

Presented by
Animesh Guchhait

This presentation will be based on the following paper

DeepSurv: personalized treatment recommender system using a Cox proportional hazards deep neural network | BMC Medical Research Methodology | Full Text (biomedcentral.com)

RECAP: COX'S PROPORTIONAL HAZARDS MODEL

Hazard Function

$$\lambda(t|x) = \lambda_0(t) \cdot e^{h(x)}.$$

Where risk score, $r(x) = e^{h(x)}$, $\hat{h}_\beta(x) = \beta^T x$ [or equivalently $\hat{r}_\beta(x) = e^{\beta^T x}$].

Partial Likelihood

$$L_c(\beta) = \prod_{i:E_i=1} \frac{\hat{r}_\beta(x_i)}{\sum_{j \in \Re(T_i)} \hat{r}_\beta(x_j)} = \prod_{i:E_i=1} \frac{\exp(\hat{h}_\beta(x_i))}{\sum_{j \in \Re(T_i)} \exp(\hat{h}_\beta(x_j))},$$

The risk set $\Re(t) = \{i : T_i \geq t\}$ is the set of patients still at risk of failure at time t.

NONLINEAR SURVIVAL MODELS

❑ The Faraggi-Simon method:

- ➢ Feed forward neural network that provides the basis for a nonlinear proportional hazards model
- ➢ single hidden layer network with two or three nodes
- ➢ the NN computes nonlinear features from the training data
- ➢ calculates their linear combination to estimate the log-risk function
- ➢ Similar to Cox regression, the network optimizes a modified Cox partial likelihood
- ➢ has not been shown to outperform the linear CPH

❑ Random Survival Forest

- ➢ a tree method that produces an ensemble estimate for the cumulative hazard function

NONLINEAR SURVIVAL MODELS

❑ DeepSurv:

- deep feed-forward neural network which predicts the effects of a patient’s covariates on their hazard rate
- The output of the network $\hat{h}_\theta(x)$ is a single node with a linear activation which estimates the log-risk function in the Cox model
- train the network by setting the objective function to be the average negative log partial likelihood with regularization:

$$l(\theta) := -\frac{1}{N_{E=1}} \sum_{i:E_i=1} \left(\hat{h}_\theta(x_i) - \log \sum_{j \in \Re(T_i)} e^{\hat{h}_\theta(x_j)} \right) + \lambda \cdot ||\theta||_2^2, \tag{4}$$

$N_{E=1}$ is the number of patients with an observable event

The risk set $\Re(t) = \{i : T_i \geq t\}$ is the set of patients still at risk of failure at time t.

NONLINEAR SURVIVAL MODELS

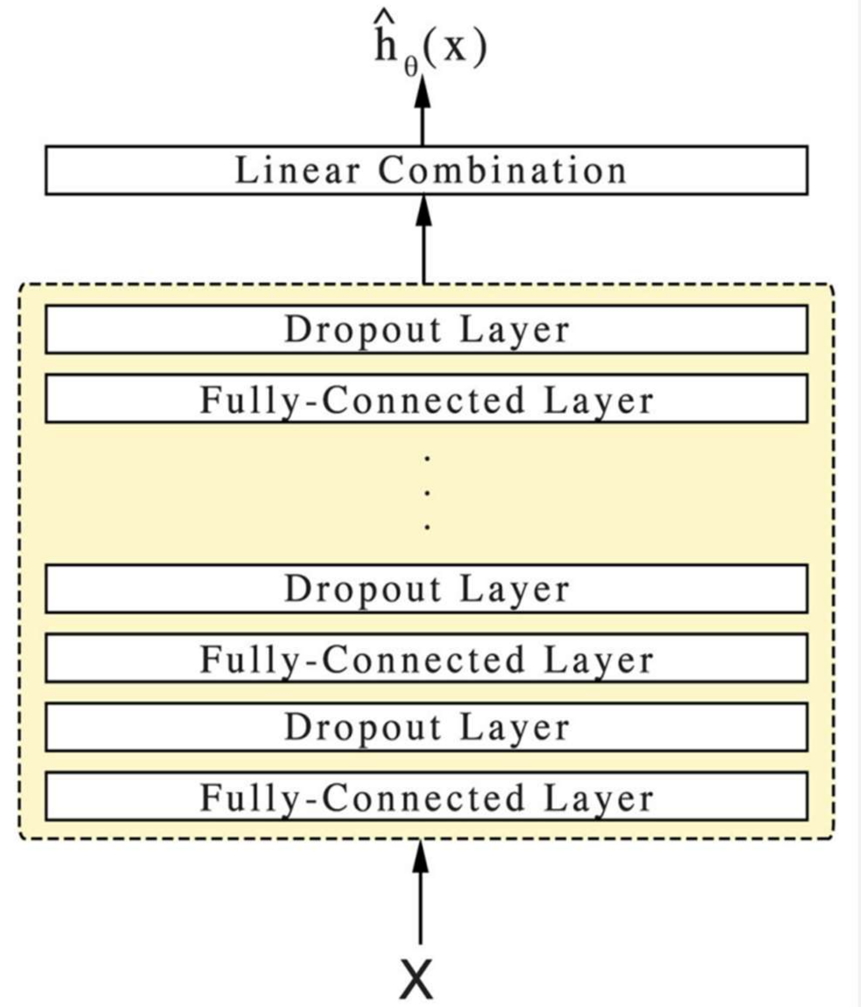
❑ DeepSurv:

- deep feed-forward neural network which predicts the effects of a patient's covariates on their hazard rate
- The output of the network $\hat{h}_\theta(x)$ is a single node with a linear activation which estimates the log-risk function in the Cox model
- train the network by setting the objective function to be the average negative log partial likelihood with regularization:

$$l(\theta) := -\frac{1}{N_{E=1}} \sum_{i:E_i=1} \left(\hat{h}_\theta(x_i) - \log \sum_{j \in \Re(T_i)} e^{\hat{h}_\theta(x_j)} \right) + \lambda \cdot ||\theta||_2^2, \qquad (4)$$

$N_{E=1}$ is the number of patients with an observable event

The risk set $\Re(t) = \{i : T_i \geq t\}$ is the set of patients still at risk of failure at time t.



TREATMENT RECOMMENDER SYSTEM

- patients are subject to different levels of risk based on their relevant prognostic features and which treatment they undergo
- Let all patients in a given study be assigned to one of n treatment groups τ ∈ {0, 1, ... , n − 1}
- We assume each treatment i to have an independent risk function $e^{h_i(x)}$
- So the hazard function becomes

$$\lambda(t; x|\tau = i) = \lambda_0(t) \cdot e^{h_i(x)}.$$

- recommender function

$$\begin{aligned}\mathrm{rec}_{ij}(x) &= \log\left(\frac{\lambda(t; x|\tau = i)}{\lambda(t; x|\tau = j)}\right) = \log\left(\frac{\lambda_0(t) \cdot e^{h_i(x)}}{\lambda_0(t) \cdot e^{h_j(x)}}\right) \\ &= h_i(x) - h_j(x).\end{aligned}$$

if $\mathrm{rec}_{ij}(x)$ gives positive result that means treatment i has higher risk of death than j

TREATMENT RECOMMENDER SYSTEM

- The recommender function can be used to provide personalized treatment recommendations
- DeepSurv's architecture holds an advantage over the CPH because it calculates the recommender function without an a priori specification of treatment interaction terms.
- CPH recommender function
  - Let each patient have a set of n features x_n, in which one feature is a treatment variable $x_0 = \tau$
  - Recommender function is

$$
\begin{aligned}
\mathbf{rec}_{ij}(x) &= \log\left(\frac{\lambda(t;x|\tau = i)}{\lambda(t;x|\tau = j)}\right) \\
&= \log\left(\frac{\lambda_0(t)\cdot e^{\beta_0 i+\beta_1 x_1+\ldots+\beta_n x_n}}{\lambda_0(t)\cdot e^{\beta_0 j+\beta_1 x_1+\ldots+\beta_n x_n}}\right) \\
&= \log\left(e^{\beta_0 i+\beta_1 x_1+\ldots+\beta_n x_n-(\beta_0 j+\beta_1 x_1+\ldots+\beta_n x_n)}\right) \\
&= \beta_0 i - \beta_0 j \\
&= \beta_0(i - j).
\end{aligned}
$$

CPH determines whether the treatment is effective and, if so, then recommending it to all patients

RESULTS ON THE DATASETS

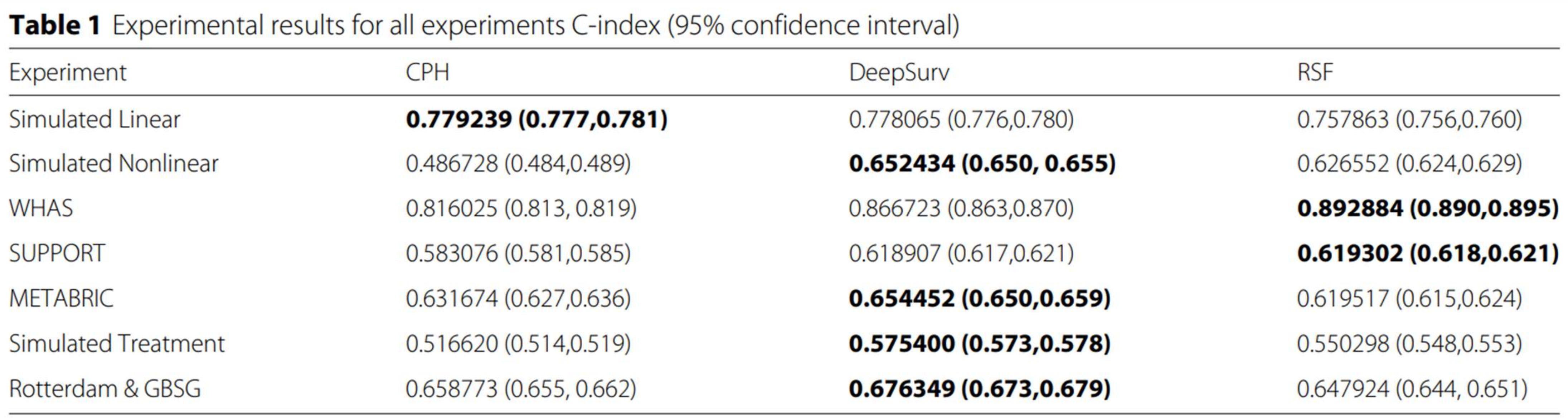


Table 1 Experimental results for all experiments C-index (95% confidence interval)

Experiment	CPH	DeepSurv	RSF
Simulated Linear	**0.779239 (0.777,0.781)**	0.778065 (0.776,0.780)	0.757863 (0.756,0.760)
Simulated Nonlinear	0.486728 (0.484,0.489)	**0.652434 (0.650, 0.655)**	0.626552 (0.624,0.629)
WHAS	0.816025 (0.813, 0.819)	0.866723 (0.863,0.870)	**0.892884 (0.890,0.895)**
SUPPORT	0.583076 (0.581,0.585)	0.618907 (0.617,0.621)	**0.619302 (0.618,0.621)**
METABRIC	0.631674 (0.627,0.636)	**0.654452 (0.650,0.659)**	0.619517 (0.615,0.624)
Simulated Treatment	0.516620 (0.514,0.519)	**0.575400 (0.573,0.578)**	0.550298 (0.548,0.553)
Rotterdam & GBSG	0.658773 (0.655, 0.662)	**0.676349 (0.673,0.679)**	0.647924 (0.644, 0.651)

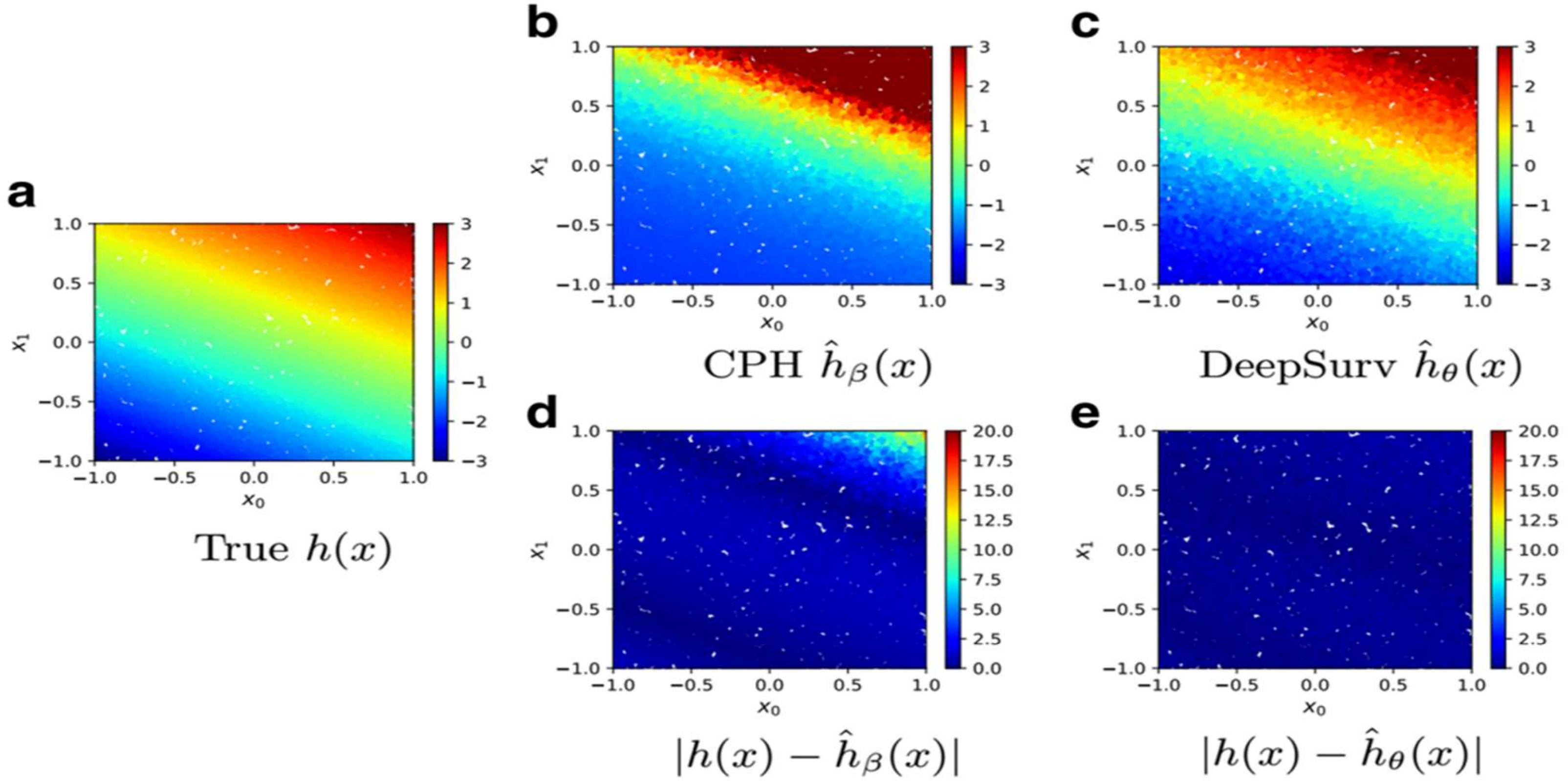


Fig. 2 Simulated Linear Experimental Log-Risk Surfaces. Predicted log-risk surfaces and errors for the simulated survival data with linear log-risk function with respect to a patient's covariates x_0 and x_1. **a** The true log-risk $h(x) = x_0 + 2x_1$ for each patient. **b** The predicted log-risk surface of $\hat{h}_\beta(x)$ from the linear CPH model parameterized by β. **c** The output of DeepSurv $\hat{h}_\theta(x)$ predicts a patient's log-risk. **d** The absolute error between true log-risk $h(x)$ and CPH's predicted log-risk $\hat{h}_\beta(x)$. **e** The absolute error between true log-risk $h(x)$ and DeepSurv's predicted log-risk $\hat{h}_\theta(x)$

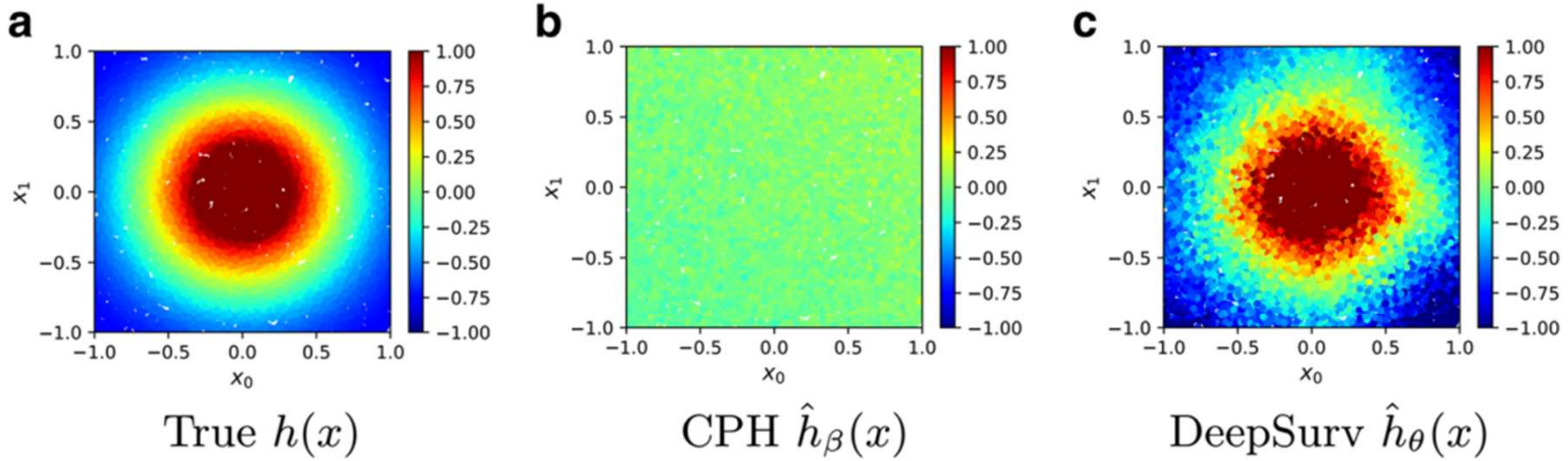


Fig. 3 Simulated Nonlinear Experimental Log-Risk Surfaces. Log-risk surfaces of the nonlinear test set with respect to patient's covariates x_0 and x_1. **a** The calculated true log-risk $h(x)$ (Eq. 9) for each patient. **b** The predicted log-risk surface of $\hat{h}_\beta(x)$ from the linear CPH model parameterized on β. The linear CPH predicts a constant log-risk. **c** The output of DeepSurv $\hat{h}_\theta(x)$ is the estimated log-risk function

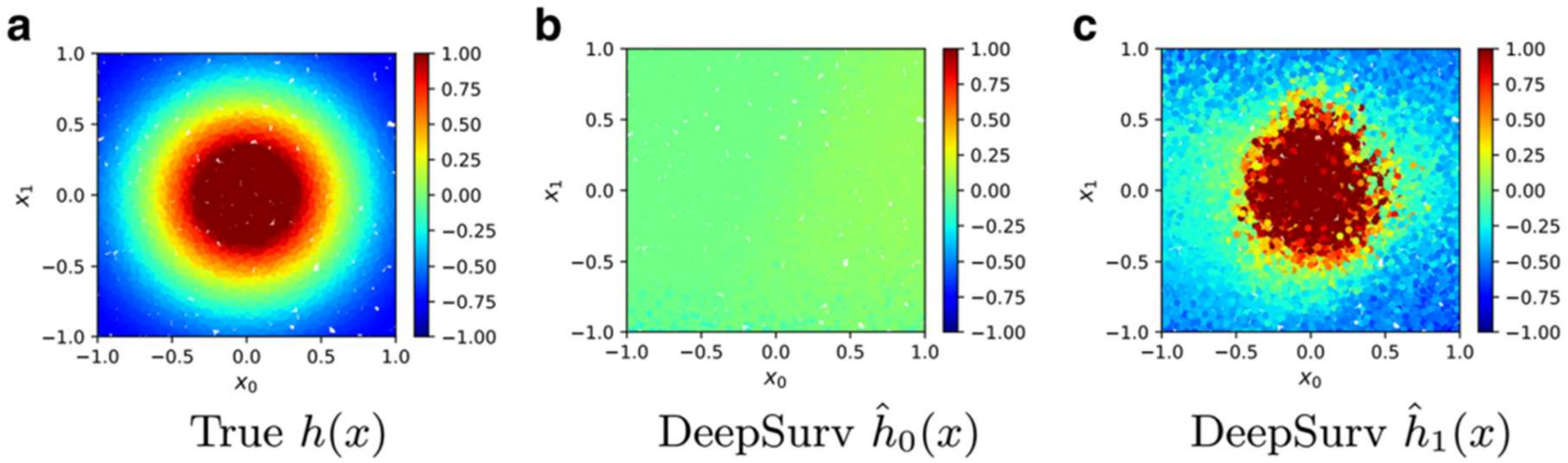
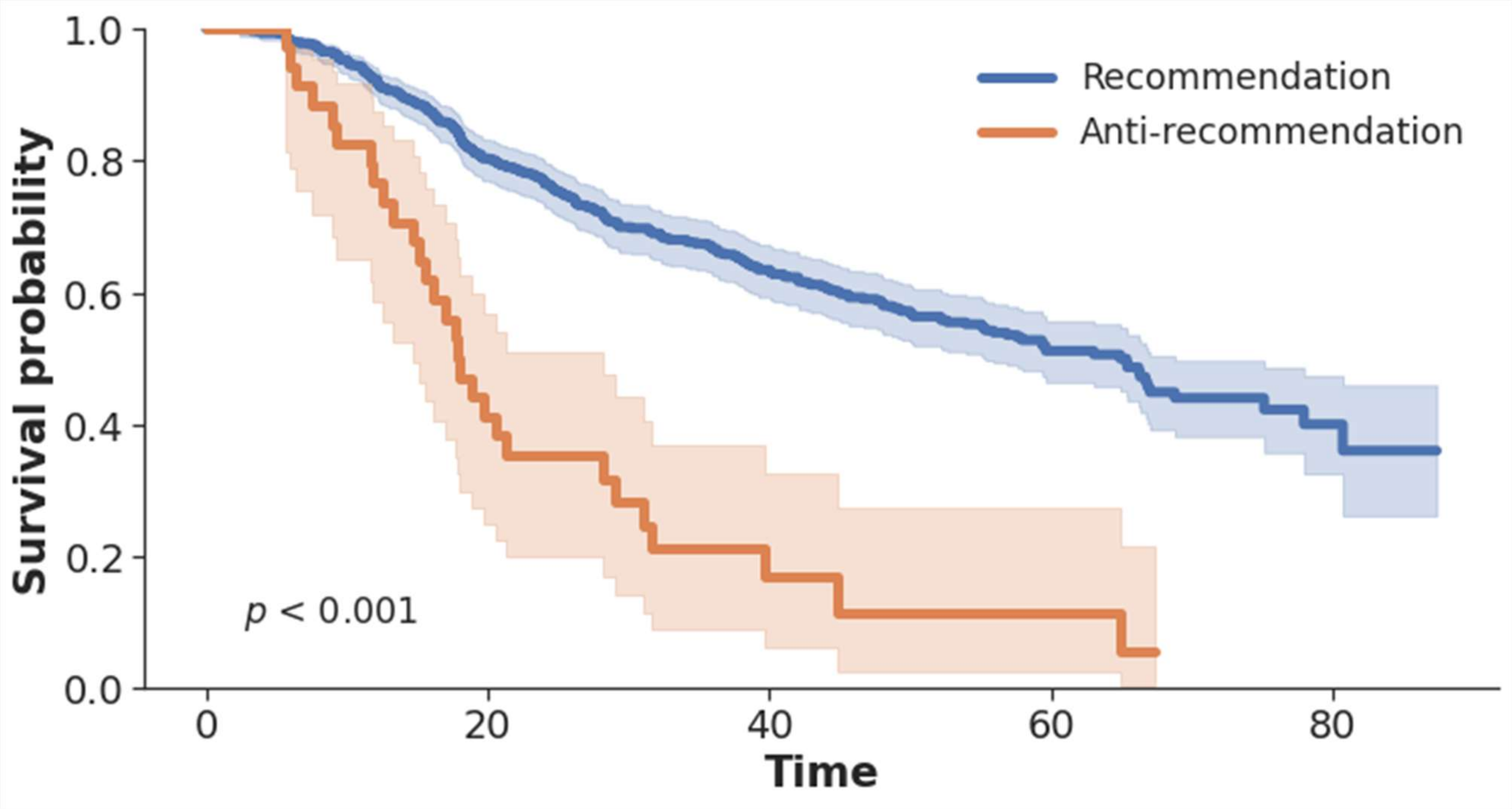
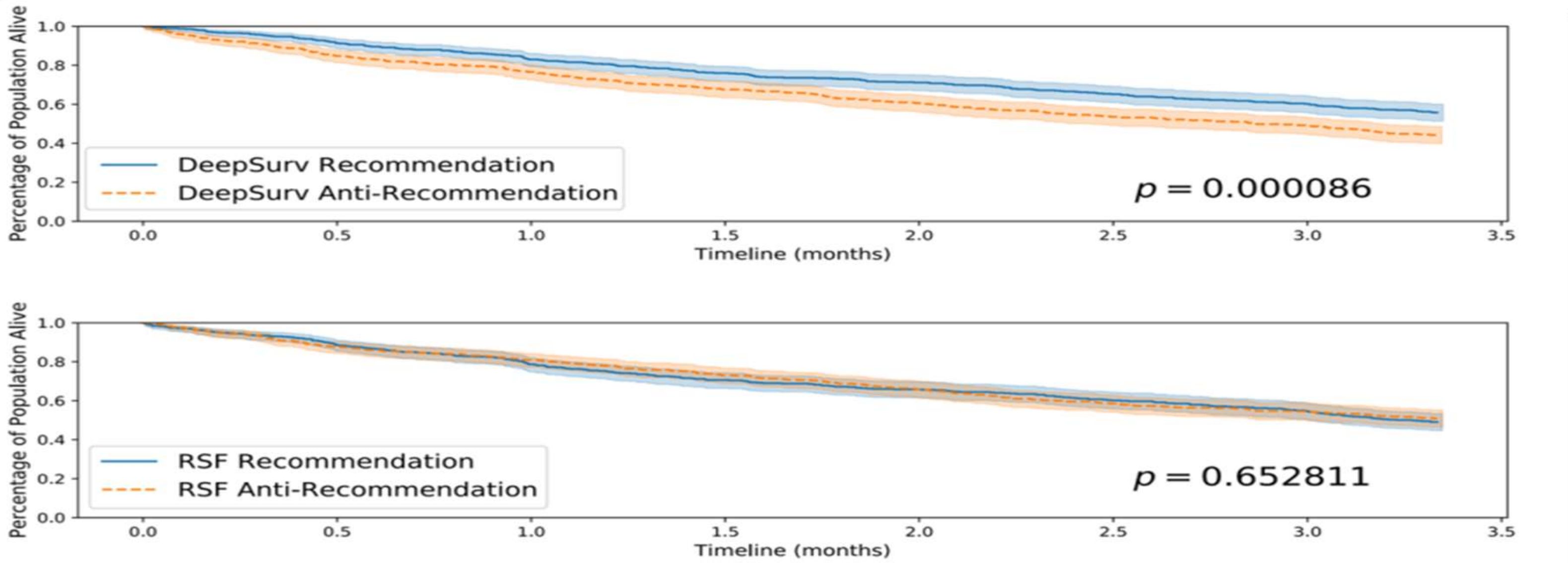


Fig. 4 Simulated Treatment Log-Risk Surface. Treatment Log-Risk Surfaces as a function of a patient's relevant covariates x_0 and x_1. **a** The true log-risk $h_1(x)$ if all patients in the test set were given treatment $\tau = 1$. We then manually set all treatment groups to either $\tau = 0$ or $\tau = 1$. **b** The predicted log-risk $\hat{h}_0(x)$ for patients with treatment group $\tau = 0$. **c** The network's predicted log-risk $\hat{h}_1(x)$ for patients in treatment group $\tau = 1$

DEEP SURV TREATMENT RECOMMENDATION SURVIVAL CURVE ON RGBSG DATA



SIMULATED TREATMENT SURVIVAL CURVES



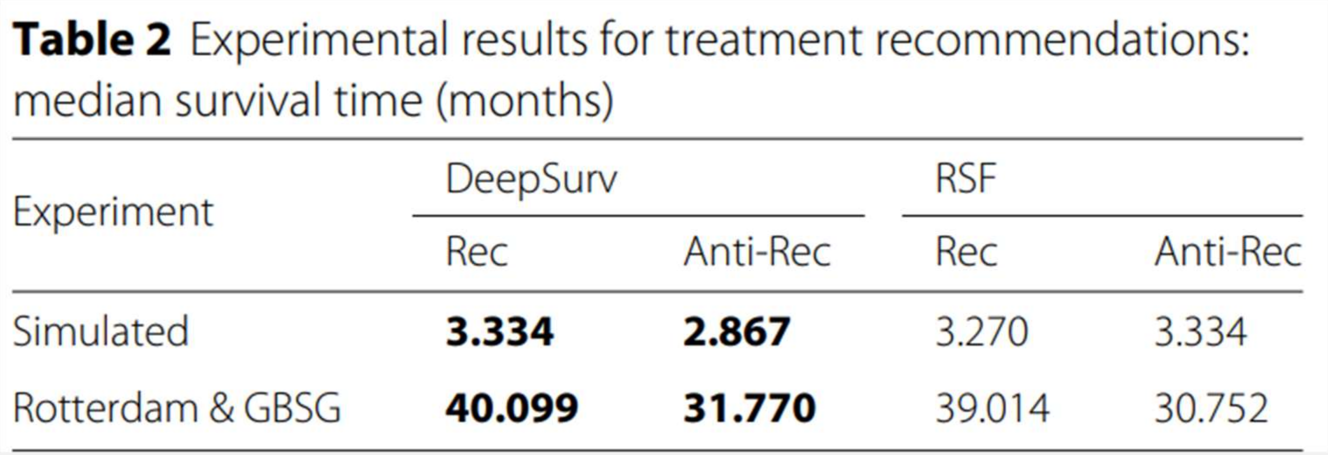


Table 2 Experimental results for treatment recommendations: median survival time (months)

Experiment	DeepSurv		RSF	
	Rec	Anti-Rec	Rec	Anti-Rec
Simulated	**3.334**	**2.867**	3.270	3.334
Rotterdam & GBSG	**40.099**	**31.770**	39.014	30.752

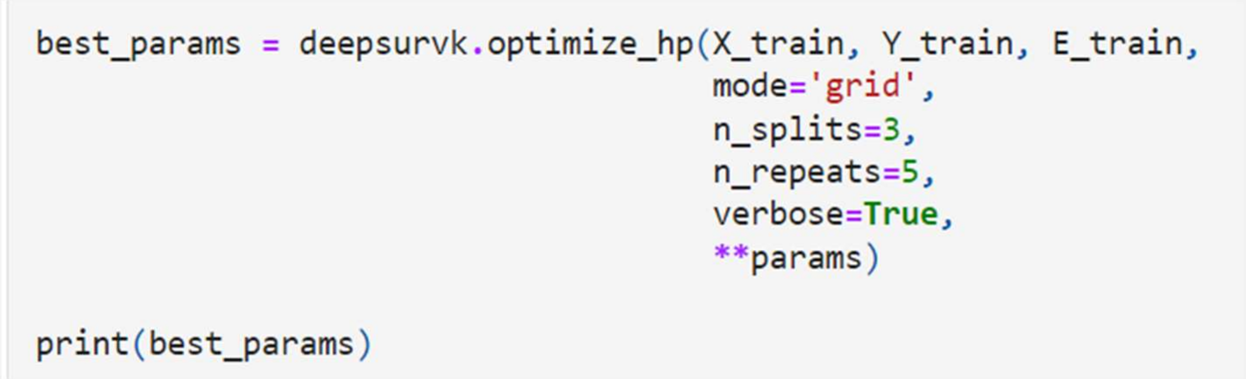
PYTHON PACKAGES

- Python package published with the paper: https://github.com/jaredleekatzman/DeepSurv
- But the package only can be install in python 3.4 or python 2.8 as it needs lasagne==0.2.dev1
- And lasagne==0.2.dev1 is not available in python newer versions

- So instead of DeepSurv package , we can use deepsurvk deepsurvk · PyPI
- Deepsurvk is also based on the above paper and can be installed and used in newer version of python easily.

HYPERPARAMETER OPTIMIZATION

- Deepsurvk package also provide hyperparameter optimization features.
- It uses Grid Search method for this optimization
- So, depending on the dataset , it my take time for optimization.



```
best_params = deepsurvk.optimize_hp(X_train, Y_train, E_train,
                                    mode='grid',
                                    n_splits=3,
                                    n_repeats=5,
                                    verbose=True,
                                    **params)

print(best_params)
```

PACKAGE LIMITATION

- Deepsurvk package only can use two class of treatments at this moment.
- For hyper parameter optimization, it use grid search method which takes lots of time.

Thank You