Experiences with the super learner

Thomas Alexander Gerds

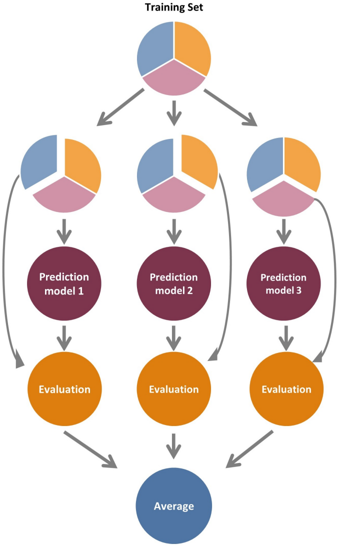
Cross-validation

Prediction performance describes how well a model will do on future patients.

Unfortunately, this is never knowable.

The best we can do is to simulate the model being applied to future patients by repeatedly splitting the data set into training and validation part.

3-fold cross-validation



Outline

Frequently asked questions about cross-validation:

Q1: 5-fold, 10-fold, 20-fold or bootstrap?

Q2: Bootstrap with or without replacement?

Q4: Model hyper parameter selection using all data?

Q5: In a survival setting where IPCW is used to estimate performance/loss: estimate censoring distribution in each training set separately?

Q6: Use prediction performance to create a new learner?

Super learner

The super learner[1] (aka stacked regression[2]) uses cross-validation data (aka level-1 data) to combine multiple prediction models into a super model.

From that perspective, the super learner is just another machine learning algorithm which takes in the data and spits out a predicted risk.

The idea is to pre-specify all steps of modelling and to deal with misspecified regression models by controlling all data-dependent modeling steps with cross-validation.

[1] van der Laan 2007

[2] Wolpert 1992, Breiman 1996

Two semi-parametric problems – in survival analysis

Prediction of an event between time 0 and time t based on predictors X:

$$\mathrm{P}(T \leq t | X = x)$$

Average treatment effect:

$$\int \Big\{\mathrm{P}(T \leq t | X = x, A = 1) - P(T \leq t | X = x, A = 0)\Big\}\mathrm{P}(\mathrm{d}x)$$

Two semi-parametric problems – in survival analysis

Prediction of an event between time 0 and time t based on predictors X:

$$\mathrm{P}(T \leq t | X = x)$$

Average treatment effect:

$$\int \Big\{\mathrm{P}(T \leq t | X = x, A = 1) - P(T \leq t | X = x, A = 0)\Big\}\mathrm{P}(\mathrm{d}x)$$

Both problems require a risk prediction model and hence a super learner could be applied ...

Modeling strategy

Training data $D_n = \{\tilde{T}_i, \Delta_i, X_i\}_{i=1}^{n}$

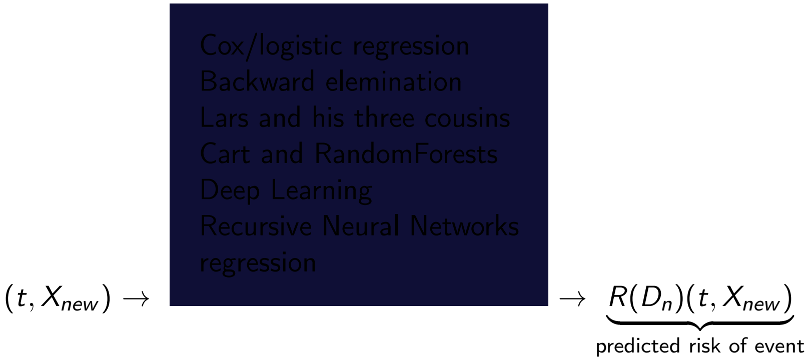
Cox/logistic regression
Backward elemination
Lars and his three cousins
Cart and RandomForests
Support vector machines
Recursive Neural Networks
Super learner

$D_n \rightarrow$ $\rightarrow$ $\underbrace{R(D_n)}_{\text{Risk prediction model}}$

Risk prediction model

Black box

$$R(D_n)(t, X_{new}) \approx \mathrm{P}(T \leq t | X_{new})$$



Cox/logistic regression
Backward elemination
Lars and his three cousins
Cart and RandomForests
Deep Learning
Recursive Neural Networks
regression

$$(t, X_{new}) \rightarrow \boxed{} \rightarrow \underbrace{R(D_n)(t, X_{new})}_{\text{predicted risk of event}}$$

Candidate modelling strategies

Flexible Cox regression with penalty:

$$R_n(t, X) = \exp(-\hat{\Lambda}_0(t) \exp(\hat{\beta}_1 \hat{f}_1(X_1) + \cdots + \hat{\beta}_p \hat{f}_p(X_p)))$$

where restricted cubic splines with k-knots are applied to the continuous predictor variables

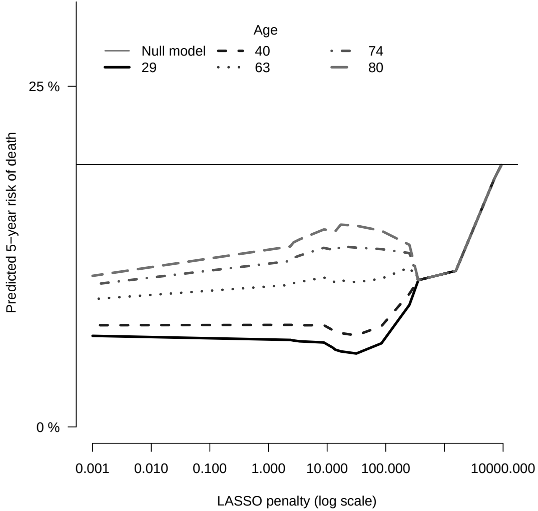
$$\hat{f}(x) = \hat{\alpha}_0 + \hat{\alpha}_1 x + \hat{\alpha}_2 \Big\{ (x - t_j)_+^3 - (x - t_{k-1})^3 + (t_k - t_j)/(t_k - t_{k-}$$

where the partial likelihood estimate of the hazard ratios

$$\hat{\beta} = \hat{\beta}(\hat{\lambda})$$

depends on the shrinkage parameter λ.

Penalized Cox regression

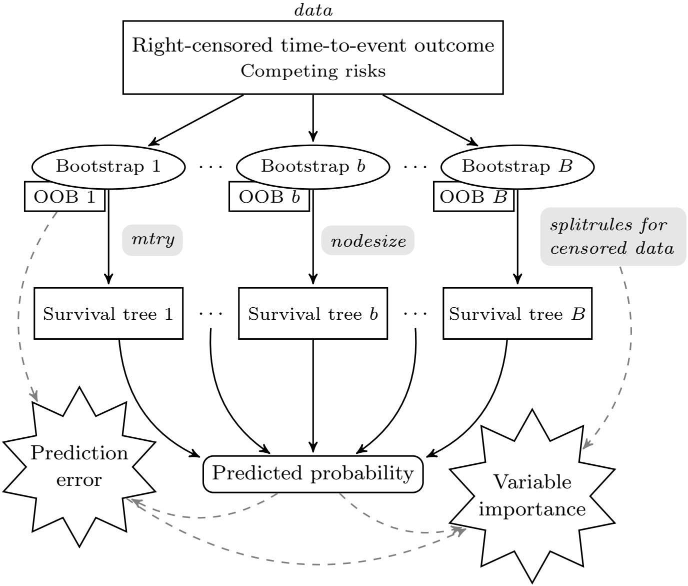


FAQ

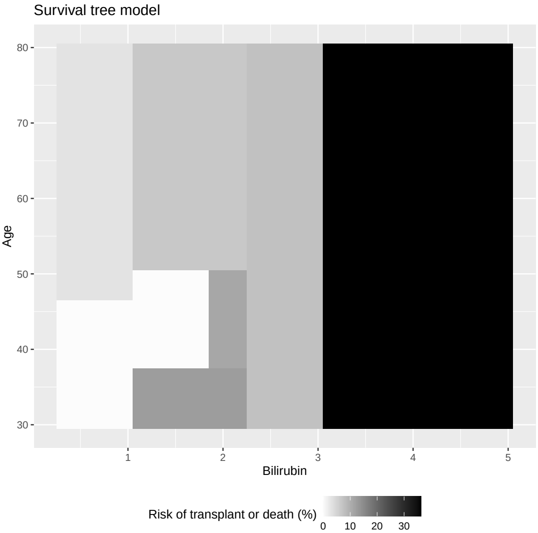
Q4: Model hyper parameter selection using all data?

A4: No, do not use all the data to optimize the penalty parameter $\hat{\lambda}$. Instead, use some form of cross-validation.

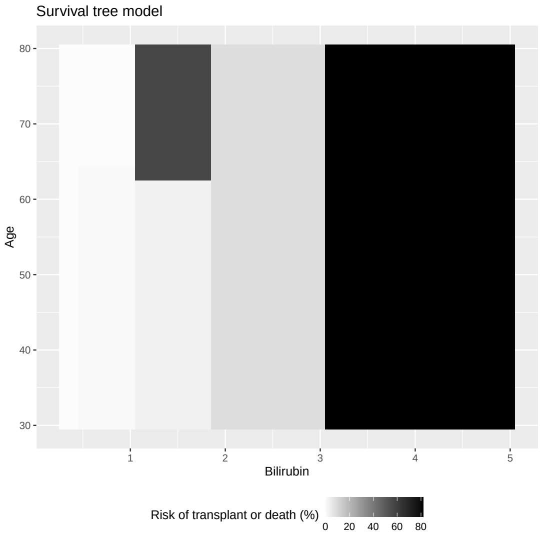
Candidate modelling strategies: random survival forest



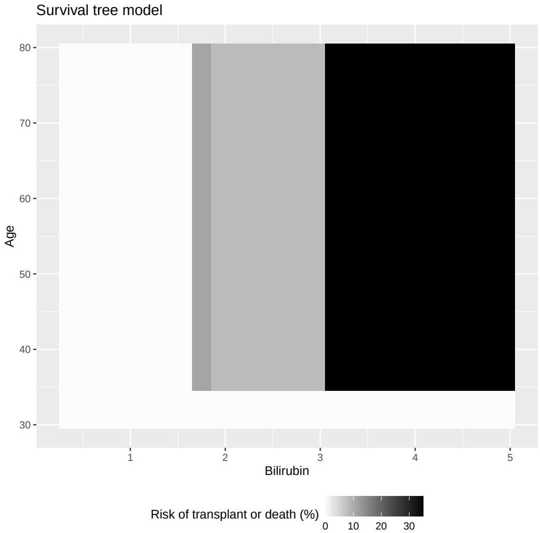
Predicted 3-year risk: survival tree in bootstrap sample



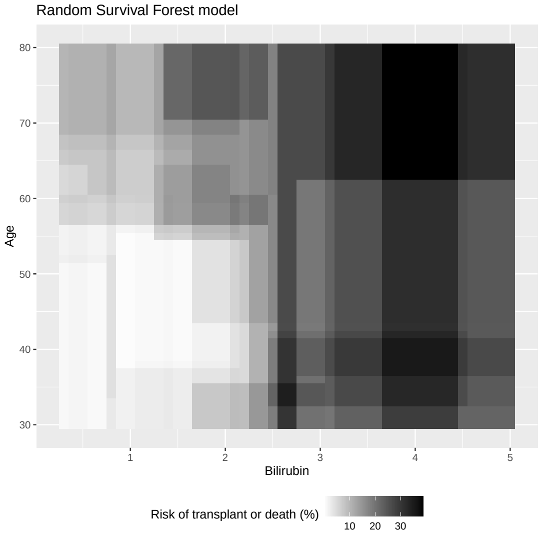
Predicted 3-year risk: survival tree in another bootstrap sample



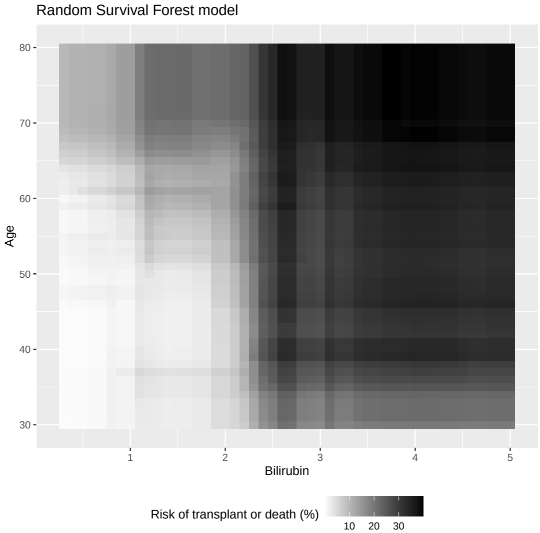
Predicted 3-year risk: survival tree in another bootstrap



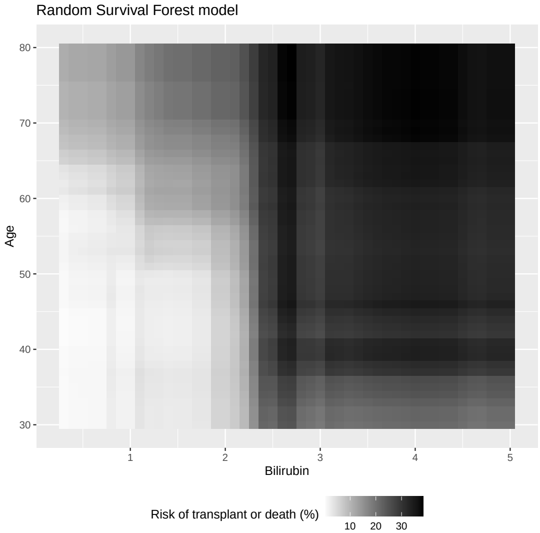
Predicted 3-year risk: random forest with 10 survival trees



Predicted 3-year risk: random forest with 100 survival trees



Predicted 3-year risk: random forest with 1000 survival trees



FAQ

Q2: Bootstrap with or without replacement?

A2: Both candidate modelling strategies internally use cross-validation or bootstrap to build prediction models. Hence, any outer cross-validation procedure should use bootstrap without replacement.

Current situation

We have developed two prediction models (Cox and random survival forest) both estimate the risk of an event until a given time t in new patients.

Which one is better?

The aim is to assess the prediction performance of the model in new patients. But: there are no data of new patients.

Fundamental idea

Data splitting is very intuitive:

we hide one part of the data, learn on the rest, and then check our *knowledge* on what was hidden.

There is a hidden parameter here: how much we hide and how much we show.

Right censored data

Notation:

n	sample size
m<n	bootstrap subsample size
$\tilde{T} = \min(T, C)$	time to event
$\Delta = 1\{T \leq C\}$	event indicator
X	p-dimensional covariate
t	prediction horizon
$\mathrm{D_n}=\{(\tilde{T}_i, \Delta_i, X_i)\}_{i=1}^n$	Dataset
$\mathrm{D^*}_{\mathrm{b,m}}=\{(\tilde{T}^*_{i,b}, \Delta^*_{i,b}, X^*_{i,b})\}_{i=1}^m$	Bootstrap dataset $m < n$
$\mathrm{R_t(D_n)}$	trained prediction model (size n)
$\mathrm{R_t(D^*{}_{b,m})}$	trained prediction model (size m)
$\mathrm{R_t(D_n)(X_{new})}$	predicted risk of event for $\mathrm{X_{new}}$
$\mathrm{R_t(D_{b,m}{}^*)(X_{oob})}$	predicted risk for subject oob

Estimate of Brier Score in right censored survival data

Inverse Probability of Censoring Weighted estimate

$$\left(1_{\{T_{new}\leq t\}} - R_t(X_{new})\right)^2 W_t(X_{new}, \hat{G})$$

where $\hat{G}$ is an estimate of the conditional censoring distribution $G(t|X) = \mathrm{P}(C > t|X)$ as obtained with the reverse Kaplan-Meier, an undersmoothed HAL, or a super learner for the censoring survival distribution.

The weights are given by:

$$W_t(X_{new}, \hat{G}) = \frac{1_{\{T_{new}\leq t\}}\Delta_{new}}{\hat{G}(T_{new} - |X_{new})} + \frac{1_{\{T_{new}>t\}}}{\hat{G}(t|X_{new})}$$

Leave-one-out-bootstrap [3]

Algorithm:

1. Draw a bootstrap sample $D^*_{b,m}$
2. Train all the prediction models: $R^1_t(D^*_{b,m}), R^2_t(D^*_{b,m}), \ldots$
3. For all out of bag subjects $i \notin D^*_{b,m}$ calculate residuals $\left(1_{\{T_i \leq t\}} - R_t(D^*_{b,m})(X_i)\right)^2$ for each prediction model
4. Repeat B times

[3] Efron & Tibshirani (1997)

Leave-one-out-bootstrap [3]

Algorithm:

1. Draw a bootstrap sample $D^*_{b,m}$
2. Train all the prediction models: $R^1_t(D^*_{b,m}), R^2_t(D^*_{b,m}), \ldots$
3. For all out of bag subjects $i \notin D^*_{b,m}$ calculate residuals $\left(1_{\{T_i \le t\}} - R_t(D^*_{b,m})(X_i)\right)^2$ for each prediction model
4. Repeat B times

Leave-one-out bootstrap estimate

$$\hat{\mu}_{t,n} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\eta_{i,B}} \sum_{b=1}^{B} \left(1_{\{T_i \le t\}} - R_t(D^*_{b,m})(X_i)\right)^2 1_{\{X_i \notin D^*_{b,m}\}} W_t(X_i, \hat{G})$$

where $\eta_{i,B}$ = number of bootstrap samples in which subject i is not included.

[3] Efron & Tibshirani (1997)

FAQ

Q5: In a survival setting where IPCW is used to estimate performance/loss: estimate censoring distribution in each training set separately?

A5: No, the formula for the leave-one-out bootstrap estimator clearly indicates to estimate the censoring distribution with all data.

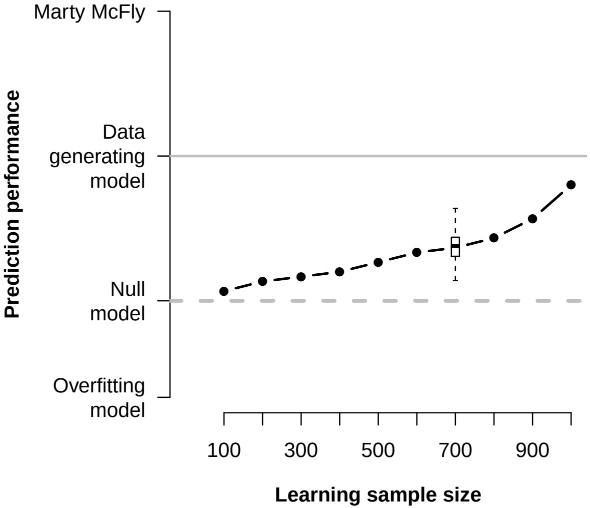
The estimated parameter

The bootstrap cross-validated estimate is an estimate of the expected value of the average residual:

$$\mathrm{E}_{new}\left[\mathrm{E}_{D_m}\left[\left(1_{\{\tilde{T}_{new}\leq t\}} - R_t(D_m)(X_{new})\right)^2 | (\tilde{T}_{new}, \Delta_{new}, X_{new})\right]\right]$$

- we are really interested in the performance of $R_t(D_n)$
- we get an estimate of the performance of the average model across all possible training data sets of size $m < n$:

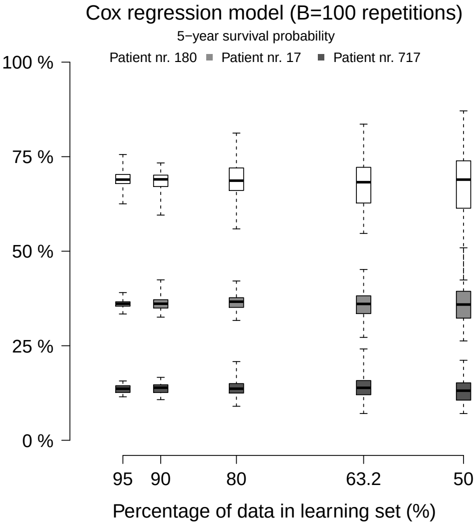
this is a bit unfortunate


Marty McFly
Data generating model
Null model
Overfitting model
Prediction performance
100
300
500
700
900
Learning sample size

FAQ

Q1: 5-fold, 10-fold, 20-fold or bootstrap?

A1: The target parameter depends on this choice. There is a bias-variance tradeoff . . .


Cox regression model (B=100 repetitions)
5−year survival probability
Patient nr. 180
Patient nr. 17
Patient nr. 717
100 %
75 %
50 %
25 %
0 %
95
90
80
63.2
50
Percentage of data in learning set (%)

Discrete super learner

```
ff <- Surv(time,event)~ age + sex+ bili + protime +
    albumin + chol +copper +ast +platelet+spiders+trig+
    hepato+ascites
# Cox
fit.cox <- list(
  lasso=penalizedS3(ff,data=pbc,model="cox",trace=0,
    lambda2=0,lambda1=4),
 elastic=penalizedS3(ff,data=pbc,model="cox",trace=0,
    lambda2=5,lambda1=18))
# random forest
fit.forest=list(
  Forest5=rfsrc(ff,data=pbc,seed=8,nodesize=5,ntree=100,
    mtry=13),
  Forest43=rfsrc(ff,data=pbc,seed=8,nodesize=43,ntree=100)
    )
# leave-one-out estimate of performance
set.seed(13)
x=Score(c(fit.cox,fit.forest),
        data=pbc, B=100, M=.632*NROW(pbc),
        split.method="loob",
        formula=Surv(time,status!=1)~1,
        times=1000, summary="risk")
summary(x,what="score")
```

Discrete super learner

times	Model	AUC (%)	Brier (%)
1000	Null model	50.0	15.3 [12.4;18.3]
1000	Null model	50.0 [50.0;50.0]	15.3 [12.4;18.3]
1000	lasso	86.0 [80.7;91.4]	11.2 [8.5;13.8]
1000	elastic	84.7 [79.2;90.1]	11.6 [8.9;14.3]
1000	Forest5	84.4 [79.0;89.8]	11.8 [9.4;14.2]
1000	Forest43	86.4 [81.3;91.6]	11.6 [9.4;13.8]

The winner is the lasso.

But,

- the result is quite sensitive to the seed
- the tuning parameters were not optimized in the training phase within the folds (to save computation time).

Ensemble super learner: level-1 data

ID	time	status	Null model	lasso	elastic	Forest5	Forest43
234	3255	1	17.9	2.7	4.1	4.1	6.5
169	2241	0	18.5	3.0	3.9	0.8	6.7
75	1230	1	18.4	3.2	4.4	0.4	6.6
158	2055	1	19.0	3.2	5.6	0.1	7.0
188	2449	1	18.2	3.2	6.6	23.2	8.0
11	198	1	18.0	3.4	6.1	0.4	7.5
261	4039	1	19.3	3.4	5.6	5.1	8.1
79	1271	1	18.3	3.5	5.3	0.2	6.7
128	1701	1	18.5	3.5	5.2	0.4	6.7
72	1212	1	18.3	3.6	6.7	7.2	8.1
163	2176	1	18.3	3.6	6.1	0.3	6.8
103	1434	1	18.4	3.8	4.7	0.4	6.7
...							
...							

Ensemble super learner

Breiman (1996): *Wolpert's idea is that the level one data has more information in it, and can be used to construct "good" combinations of learners.*

But he also remarks that just how to use level one data to form accurate combinations is "black art".

Breiman proposed ridge regression

```
penalized(Surv(time,status)∼NullModel+lasso+elastic+
    forest5+forest43,
          data=level1,
          lambda2=7)
```

FAQ

Q6: Use prediction performance to create a new learner?

A6: Yes, but . . .

- . . . the ensemble super learner has a tuning parameter (ridge L_2 penalty)
- . . . an outer cross-validation loop is required to assess the performance of the super learner.
- . . . it is possible to create a hyper learner by combining super learners and so on and so forth
- . . . garbage in garbage out