Evaluate Model Performance

Mingcheng Hu

Table of contents


```r
library(tidyverse)
library(survival)
library(survcomp) # general way to calculate concordance index
library(glmnet)
library(randomForestSRC)
library(xgboost)
library(kableExtra) # include knitr automatically

source("/work/users/y/u/yuukias/BIOS-Material/BIOS992/utils/csv_utils.r")
# * Don't use setwd() for Quarto documents!
# setwd("/work/users/y/u/yuukias/BIOS-Material/BIOS992/data")

adjust_type <- ifelse(exists("params"), params$adjust_type, "minimal") #
↪ options: "minimal", "partial", "full"
impute_type <- ifelse(exists("params"), params$impute_type, "unimputed") #
↪ options: "unimputed", "imputed"
include_statin <- ifelse(exists("params"), params$include_statin, "no") #
↪ options: "yes", "no"
```

```r
n_folds <- 10
set.seed(1234)
```

```r
# string of parameters
adjust_type_str <- switch(adjust_type,
    minimal = "minimal",
    partial = "partial",
    full = "full"
)
print(paste0("Model Adjustment Type: ", adjust_type_str))
```

```
[1] "Model Adjustment Type: minimal"
```

```r
impute_type_str <- switch(impute_type,
    unimputed = "unimputed",
    imputed = "imputed"
)
print(paste0("Data Imputation Type: ", impute_type_str))
```

```
[1] "Data Imputation Type: unimputed"
```

Load Models

```r
# load(get_data_path("cox_model_univariate", adjust_type_str,
↪  impute_type_str, include_statin, model = "cox"))
load(get_data_path("cox_model_multivariate", adjust_type_str,
↪  impute_type_str, include_statin, model = "cox"))
load(get_data_path("cox_model_lasso", adjust_type_str, impute_type_str,
↪  include_statin, model = "cox"))
load(get_data_path("cox_model_step", adjust_type_str, impute_type_str,
↪  include_statin, model = "cox"))

# load(get_data_path("rsf_model", adjust_type_str, impute_type_str,
↪  include_statin, model = "rsf"))

load(get_data_path("xgb_var_select_name", adjust_type_str, impute_type_str,
↪  include_statin, model = "xgb"))
load(get_data_path("xgb_model", adjust_type_str, impute_type_str,
↪  include_statin, model = "xgb"))
```

Load Data

```r
if (include_statin == "yes") {
    data_test <-
 ↪  read.csv(paste0("/work/users/y/u/yuukias/BIOS-Material/BIOS992/data/test_data_",
 ↪  impute_type_str, "_statin.csv"),
        header = TRUE
    )
} else {
    data_test <-
 ↪  read.csv(paste0("/work/users/y/u/yuukias/BIOS-Material/BIOS992/data/test_data_",
 ↪  impute_type_str, ".csv"),
        header = TRUE
    )
}

data_test <- data_test[, -1] # the first column is the index generated by
 ↪  sklearn
(dim(data_test))
```

```
[1] 7032  100
```

```r
data <- select_subset(data_test, type = adjust_type)
(dim(data))
```

```
[1] 7032   48
```

```r
colnames(data)
```

```
 [1] "event"                          "time"
 [3] "HRV_SD1"                        "HRV_SD2"
 [5] "HRV_SD1SD2"                     "HRV_S"
 [7] "HRV_CSI"                        "HRV_CVI"
 [9] "HRV_CSI_Modified"               "HRV_PIP"
[11] "HRV_IALS"                       "HRV_PSS"
```

```
[13] "HRV_PAS"                       "HRV_GI"
[15] "HRV_SI"                        "HRV_AI"
[17] "HRV_PI"                        "HRV_C1d"
[19] "HRV_C1a"                       "HRV_SD1d"
[21] "HRV_SD1a"                      "HRV_C2d"
[23] "HRV_C2a"                       "HRV_SD2d"
[25] "HRV_SD2a"                      "HRV_Cd"
[27] "HRV_Ca"                        "HRV_SDNNd"
[29] "HRV_SDNNa"                     "HRV_ApEn"
[31] "HRV_ShanEn"                    "HRV_FuzzyEn"
[33] "HRV_MSEn"                      "HRV_CMSEn"
[35] "HRV_RCMSEn"                    "HRV_CD"
[37] "HRV_HFD"                       "HRV_KFD"
[39] "HRV_LZC"                       "HRV_DFA_alpha1"
[41] "HRV_MFDFA_alpha1_Width"        "HRV_MFDFA_alpha1_Peak"
[43] "HRV_MFDFA_alpha1_Mean"         "HRV_MFDFA_alpha1_Max"
[45] "HRV_MFDFA_alpha1_Delta"        "HRV_MFDFA_alpha1_Asymmetry"
[47] "HRV_MFDFA_alpha1_Fluctuation" "HRV_MFDFA_alpha1_Increment"
```

```r
data <- tibble::as_tibble(data)
```

```r
# * It is very hard to compare the HR as different predictors are on
 ↪ different magnitudes, so we need to normalize them.
time_col <- data$time
event_col <- data$event
data <- data %>%
    select(-c(time, event)) %>%
    mutate(across(where(is.numeric), scale)) %>%
    mutate(
        time = time_col,
        event = event_col
    )
```

Note now the interpretation of HR is different! For example, if HR=1.16 for the predictor in the univariate model fitted using scaled data, it means that each standard deviation increase is associated with 16% higher risk of event.

```r
# For Cox model:
data_complete <- na.omit(data)

# For RSF model: We don't need to exclude the missing values
```

```r
# For XGBoost model:
test_x <- as.matrix(data_complete %>% select(-c(time, event)))
test_y_lower_bound <- data_complete$time
test_y_upper_bound <- ifelse(data_complete$event == 1, data_complete$time,
↪  Inf)
dtest_selected <- xgb.DMatrix(
    data = test_x[, vars_selected],
    label_lower_bound = test_y_lower_bound,
    label_upper_bound = test_y_upper_bound
)
```

Model Performance Evaluation and Comparison

Cox Models

```r
# Multivariate Cox Model (only on complete data for fair comparison)
concord_full1 <- concordance(cox_model_full_complete, newdata =
↪  data_complete)
print(paste0("Concordance of Multivariate Cox Model: ",
↪  round(concord_full1$concordance, 3)))
```

```
[1] "Concordance of Multivariate Cox Model: 0.552"
```

```r
pred_full <- predict(cox_model_full_complete, newdata = data_complete, type =
↪  "risk")
concord_full2 <- concordance.index(pred_full, data_complete$time,
↪  data_complete$event)$c.index
lower_full <- concordance.index(pred_full, data_complete$time,
↪  data_complete$event)$lower
upper_full <- concordance.index(pred_full, data_complete$time,
↪  data_complete$event)$upper
print(paste0("Concordance of Multivariate Cox Model: ", round(concord_full2,
↪  3), " (", round(lower_full, 3), ", ", round(upper_full, 3), ")"))
```

```
[1] "Concordance of Multivariate Cox Model: 0.552 (0.518, 0.586)"
```

Both approaches give the same result.

```
# LASSO
# * Now we should use Cindex() instead of concordance()
x_test <- as.matrix(data_complete %>% select(-c(time, event)))
y_test <- cbind(time = data_complete$time, status = data_complete$event)
pred_lasso <- predict(cox_model_lasso, newx = x_test, s = "lambda.1se")
concord_lasso1 <- apply(pred_lasso, 2, Cindex, y = y_test)
print(paste0("Concordance of LASSO Cox Model: ", round(concord_lasso1, 3)))
```

```
[1] "Concordance of LASSO Cox Model: 0.54"
```

```
pred_lasso2 <- predict(cox_model_lasso, newx = x_test, s = "lambda.1se")
concord_lasso2 <- concordance.index(pred_lasso2, data_complete$time,
 ↪  data_complete$event)$c.index
lower_lasso <- concordance.index(pred_lasso2, data_complete$time,
 ↪  data_complete$event)$lower
upper_lasso <- concordance.index(pred_lasso2, data_complete$time,
 ↪  data_complete$event)$upper
print(paste0("Concordance of LASSO Cox Model: ", round(concord_lasso2, 3), "
 ↪  (", round(lower_lasso, 3), ", ", round(upper_lasso, 3), ")"))
```

```
[1] "Concordance of LASSO Cox Model: 0.54 (0.505, 0.574)"
```

Both approaches give the same result.

```
# Stepwise
concord_step1 <- concordance(cox_model_step, newdata = data_complete)
print(paste0("Concordance of Stepwise Cox Model: ",
 ↪  round(concord_step1$concordance, 3)))
```

```
[1] "Concordance of Stepwise Cox Model: 0.55"
```

```
pred_step <- predict(cox_model_step, newdata = data_complete, type = "risk")
concord_step2 <- concordance.index(pred_step, data_complete$time,
 ↪  data_complete$event)$c.index
lower_step <- concordance.index(pred_step, data_complete$time,
 ↪  data_complete$event)$lower
```

```
upper_step <- concordance.index(pred_step, data_complete$time,
  ↪  data_complete$event)$upper
print(paste0("Concordance of Stepwise Cox Model: ", round(concord_step2, 3),
  ↪  " (", round(lower_step, 3), ", ", round(upper_step, 3), ")"))
```

```
[1] "Concordance of Stepwise Cox Model: 0.55 (0.515, 0.584)"
```

Both approaches give the same result.

RSF Model

XGBoost Model

```
pred_xgb <- predict(xgb_model, dtest_selected) # Note this is the
  ↪  log(survival time), larger value means lower risk!
pred_xgb <- -pred_xgb # now larger value means higher risk
concord_xgb <- concordance.index(pred_xgb, data_complete$time,
  ↪  data_complete$event)$c.index
lower_xgb <- concordance.index(pred_xgb, data_complete$time,
  ↪  data_complete$event)$lower
upper_xgb <- concordance.index(pred_xgb, data_complete$time,
  ↪  data_complete$event)$upper
print(paste0("Concordance of XGBoost Model: ", round(concord_xgb, 3), " (",
  ↪  round(lower_xgb, 3), ", ", round(upper_xgb, 3), ")"))
```

```
[1] "Concordance of XGBoost Model: 0.539 (0.504, 0.573)"
```

Summary

```
concord_table <- data.frame(
    Model = c("Multivariate Cox Model", "LASSO Cox Model", "Stepwise Cox
      ↪  Model", "RSF Model", "XGBoost Model"),
    Concordance = c(round(concord_full1$concordance, 3), round(concord_full2,
      ↪  3), round(concord_step1$concordance, 3), NA, round(concord_xgb, 3)),
    Lower = c(round(lower_full, 3), round(lower_lasso, 3), round(lower_step,
      ↪  3), NA, round(lower_xgb, 3)),
```

```
    Upper = c(round(upper_full, 3), round(upper_lasso, 3), round(upper_step,
     ↪  3), NA, round(upper_xgb, 3))
)
```

```
concord_table %>%
    kbl(
        caption = "Concordance Index of Different Models",
        align = c("|l", "c", "c", "c|"),
        col.names = c("Model", "Concordance", "Lower", "Upper")
    ) %>%
    kable_styling(
        bootstrap_options = c("striped", "hover", "condensed", "responsive"),
        position = "center",
        latex_options = c("striped", "HOLD_position")
    )
```

Table 1: Concordance Index of Different Models

Model	Concordance	Lower	Upper
Multivariate Cox Model	0.552	0.518	0.586
LASSO Cox Model	0.552	0.505	0.574
Stepwise Cox Model	0.550	0.515	0.584
RSF Model	NA	NA	NA
XGBoost Model	0.539	0.504	0.573