Evaluate Formula Performance: Sensitivity Analysis

Mingcheng Hu

Table of contents


```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.2     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.0
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(survival)
library(survcomp) # general way to calculate concordance index
```

```
Loading required package: prodlim
```

```r
library(glmnet)
```

```
Loading required package: Matrix

Attaching package: 'Matrix'

The following objects are masked from 'package:tidyr':

    expand, pack, unpack

Loaded glmnet 4.1-8
```

```r
library(randomForestSRC)
```

```
 randomForestSRC 3.3.3

 Type rfsrc.news() to see new features, changes, and bug fixes.


Attaching package: 'randomForestSRC'

The following object is masked from 'package:purrr':

    partial
```

```r
library(xgboost)
```

```
Attaching package: 'xgboost'

The following object is masked from 'package:dplyr':

    slice
```

```r
library("SHAPforxgboost")
library(kableExtra) # include knitr automatically
```

```
Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")
```

```
Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")

Warning: 'xfun::attr()' is deprecated.
Use 'xfun::attr2()' instead.
See help("Deprecated")


Attaching package: 'kableExtra'

The following object is masked from 'package:dplyr':

    group_rows
```

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

set.seed(1234)
```

```
# 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: full"
```

```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 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   89
```

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

```r
# * There are some imputed ethnicity set to "e". We will exclude them at this
 ↪  time.
data <- data %>%
    filter(ethnicity != "e")

# * We also need to manually relevel the categorical variables
data <- data %>%
    mutate(
        # Set "Never" (0) as baseline for smoking
        smoking = factor(smoking,
            levels = c("0", "1", "2", "-3"),
            labels = c("Never", "Previous", "Current", "Prefer not to
             ↪  answer")
        ),

        # Set "No" (0) as baseline for diabetes
        diabetes = factor(diabetes,
            levels = c("0", "1", "-1", "-3"),
            labels = c("No", "Yes", "Do not know", "Prefer not to answer")
        ),

        # Ensure other categorical variables are properly factored
        ethnicity = factor(ethnicity,
            levels = c("1", "2", "3", "4", "5", "6"),
            labels = c("White", "Mixed", "Asian/Asian British", "Black/Black
             ↪  British", "Chinese", "Other")
        ),
        education = factor(education,
            levels = c("1", "2", "3", "4", "5", "6", "-7", "-3"),
            labels = c(
                "College/University degree", "A levels/AS levels",
                "O levels/GCSEs", "CSEs", "NVQ/HND/HNC",
                "Other professional", "None of the above",
                "Prefer not to answer"
            )
        ),
        activity = factor(activity,
            levels = c("0", "1", "2"),
            labels = c("Low", "Moderate", "High")
```

```
        ),
        sex = factor(sex,
            levels = c("0", "1"),
            labels = c("Female", "Male")
        ),
        hypertension_treatment = factor(hypertension_treatment,
            levels = c("0", "1"),
            labels = c("No", "Yes")
        )
    )
```

```
# * 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.

Model Performance Evaluation and Comparison

Cox Model

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

```
load(get_data_path("cox_model_univariate", adjust_type_str, impute_type_str,
 ↪  include_statin, model = "sensitivity"))
load(get_data_path("cox_model_multivariate", adjust_type_str,
 ↪  impute_type_str, include_statin, model = "sensitivity"))
```

```r
load(get_data_path("cox_model_lasso", adjust_type_str, impute_type_str,
 ↪  include_statin, model = "sensitivity"))
load(get_data_path("cox_model_step", adjust_type_str, impute_type_str,
 ↪  include_statin, model = "sensitivity"))
```

```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: 0.71 (0.672, 0.746)"
```

```r
x_test <- model.matrix(~ . - 1 - time - event, data = data_complete)
y_test <- Surv(data_complete$time, data_complete$event)
pred_lasso <- predict(cox_model_lasso, newx = x_test, s = "lambda.1se")
concord_lasso <- concordance.index(pred_lasso, data_complete$time,
 ↪  data_complete$event)$c.index
lower_lasso <- concordance.index(pred_lasso, data_complete$time,
 ↪  data_complete$event)$lower
upper_lasso <- concordance.index(pred_lasso, data_complete$time,
 ↪  data_complete$event)$upper
print(paste0("Concordance of LASSO Cox Model: ", round(concord_lasso, 3), "
 ↪  (", round(lower_lasso, 3), ", ", round(upper_lasso, 3), ")"))
```

```
[1] "Concordance of LASSO Cox Model: 0.705 (0.666, 0.741)"
```

```r
pred_step <- predict(cox_model_step, newdata = data_complete, type = "risk")
concord_step <- 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_step, 3), "
 ↪  (", round(lower_step, 3), ", ", round(upper_step, 3), ")"))
```

```
[1] "Concordance of Stepwise Cox Model: 0.71 (0.671, 0.746)"
```

RSF Model

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

```
load(get_data_path("rsf_var_select_name", adjust_type_str, impute_type_str,
 ↪  include_statin, model = "sensitivity"))
load(get_data_path("rsf_model", adjust_type_str, impute_type_str,
 ↪  include_statin, model = "sensitivity"))
```

```
data_selected <- data_complete[, c("time", "event", vars_selected)]
data_selected <- model.frame(~ . - 1, data = data_selected, na.action =
 ↪  na.pass)
data_selected <- model.matrix(~ . - 1, data = data_selected)
data_selected <- as.data.frame(data_selected)
```

```
pred_rsf <- predict(rsf_model, newdata = data_selected)$predicted
concord_rsf <- concordance.index(pred_rsf, data_complete$time,
 ↪  data_complete$event)$c.index
lower_rsf <- concordance.index(pred_rsf, data_complete$time,
 ↪  data_complete$event)$lower
upper_rsf <- concordance.index(pred_rsf, data_complete$time,
 ↪  data_complete$event)$upper
print(paste0("Concordance of RSF Model: ", round(concord_rsf, 3), " (",
 ↪  round(lower_rsf, 3), ", ", round(upper_rsf, 3), ")"))
```

```
[1] "Concordance of RSF Model: 0.715 (0.677, 0.751)"
```

XGBoost Model

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

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

```r
# Note this is the survival time, larger value means lower risk!
pred_xgb <- predict(xgb_model, dtest_selected)
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: 0.713 (0.674, 0.749)"
```