Fairness-aware

Tianyi Xia, Danielle Solomon

2024-04-08

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library(caTools)
library(neuralnet)
library(DALEX)
```

```
## Welcome to DALEX (version: 2.4.3).
## Find examples and detailed introduction at: http://ema.drwhy.ai/
## Additional features will be available after installation of: ggpubr.
## Use 'install_dependencies()' to get all suggested dependencies
```

```
library(ggplot2)
library(keras)
library(tensorflow)
library(VGAM)
```

```
##      stats4
```

```
##      splines
```

```
library(tidyr)
library(iml)
library(dplyr)
```

```
##
##    'dplyr'
```

```
## The following object is masked from 'package:DALEX':
##
##     explain
```

```
## The following object is masked from 'package:neuralnet':
##
##     compute
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
##    'randomForest'

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

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(xgboost)
```

```
##
##    'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(shapr)
```

```
##
##    'shapr'

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

## The following objects are masked from 'package:DALEX':
##
##     explain, update_data
```

```r
da=read.csv('../data/compas-scores-two-years.csv',header=T)
da <- da[ grepl("Caucasian|African-American", da$race),]
da <- da[,c("id",'sex','age','c_charge_degree','priors_count',"race",'end',"two_year_recid")]
names(da)[names(da) == "c_charge_degree"] <- "charge_degree"
names(da)[names(da) == "end"] <- "length_of_stay"
Y <- da$two_year_recid
da$sex <- ifelse(da$sex == "Male", 1, 0)
da$charge_degree <- ifelse(da$charge_degree == "M", 1, 0)
da$race <- ifelse(da$race == "Caucasian",0, 1)

set.seed(07)
split <- sample.split(Y, SplitRatio = 0.8)

train <- da[split, ]
test <- da[!split, ]
x_train <- as.matrix(da[split,c('sex','age','charge_degree','priors_count',"race",'length_of_stay')])
y_train <- da[split,]$two_year_recid
A_train <- da[split,]$race  # Binary sensitive feature
x_test <- as.matrix(da[!split,c('sex','age','charge_degree','priors_count',"race",'length_of_stay')])
y_test <- da[!split,]$two_year_recid
A_test <- da[!split,]$race
#
cat("Training set dimensions:", dim(x_train), "\n")
```

```
## Training set dimensions: 4920 6
```

```r
cat("Test set dimensions:", dim(x_test), "\n")
```

```
## Test set dimensions: 1230 6
```

```r
calibration <- function(sensitive_var, y_pred, y_true) {
  protected_points <- which(sensitive_var == 1)
  y_pred_protected <- y_pred[protected_points]
  y_true_protected <- y_true[protected_points]
  p_protected <- sum(y_pred_protected == y_true_protected) / length(y_true_protected)

  not_protected_points <- which(sensitive_var == 0)
  y_pred_not_protected <- y_pred[not_protected_points]
  y_true_not_protected <- y_true[not_protected_points]
  p_not_protected <- sum(y_pred_not_protected == y_true_not_protected) / length(y_true_not_protected)

  calibration_value <- abs(p_protected - p_not_protected)

  return(calibration_value)
}
accuracy_race <- function(sensitive_var, y_pred, y_true) {
  # Find indices of the protected group
  protected_points <- which(sensitive_var == 1)
  y_pred_protected <- y_pred[protected_points]
  y_true_protected <- y_true[protected_points]
  # Calculate accuracy for the protected group
  p_protected <- sum(y_pred_protected == y_true_protected) / length(y_true_protected)
```

```r
  # Find indices of the not protected group
  not_protected_points <- which(sensitive_var == 0)
  y_pred_not_protected <- y_pred[not_protected_points]
  y_true_not_protected <- y_true[not_protected_points]
  # Calculate accuracy for the not protected group
  p_not_protected <- sum(y_pred_not_protected == y_true_not_protected) / length(y_true_not_protected)

  # Return both accuracies as a list
  return(list(protected = p_protected, not_protected = p_not_protected))
}
```

```r
# Convert 'two_year_recid' to a factor if it's not already
da$two_year_recid <- as.factor(da$two_year_recid)

# Train the Random Forest model as a classifier
model <- randomForest(two_year_recid ~ sex + age + charge_degree + priors_count + race + length_of_stay
                      data = da,
                      ntree = 100)
y_pred <- predict(model, x_test)
accuracy <- mean(y_pred == y_test)
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.926016260162602"
```

```r
print(accuracy_race(x_test[,'race'],y_pred,y_test))
```

```
## $protected
## [1] 0.9295213
##
## $not_protected
## [1] 0.9205021
```

```r
print(paste("Calibration:", calibration(x_test[,'race'],y_pred,y_test)))
```

```
## [1] "Calibration: 0.00901918454553552"
```

```r
results_rf <- data.frame(
  `Random Forest` = c(accuracy, accuracy_race(x_test[,'race'],y_pred,y_test)$protected, accuracy_race(x
  row.names = c("Model Accuracy", "Protected", "Not Protected", "Calibration")
)
```

```r
model <- xgboost(data = x_train, label = y_train, nround = 20, verbose = FALSE)

explainer <- shapr(x_train, model)
```

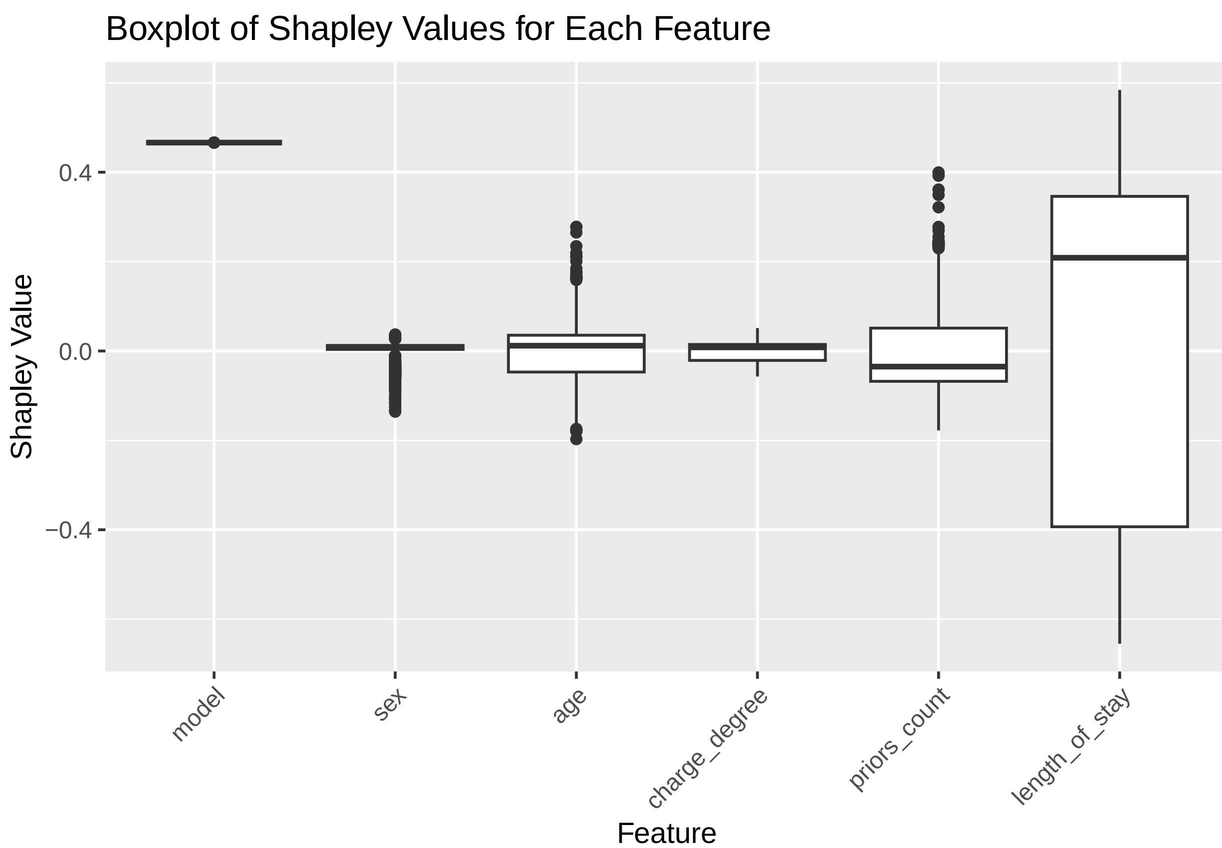
```
## The specified model provides feature classes that are NA. The classes of data are taken as the truth
```

```r
p <- mean(y_train)
explanation <- explain( x_test,
                        approach = "empirical",
                        explainer = explainer,
                        prediction_zero = p)

print(explanation$dt)
```

```
##            none          sex           age charge_degree priors_count
##    1: 0.4662602  0.013265721  0.0004083264   0.007324682  -0.11267897
##    2: 0.4662602  0.013164470  0.0343570813   0.014292850  -0.06231050
##    3: 0.4662601 -0.034374004 -0.0236521219  -0.033754027  -0.05991526
##    4: 0.4662602  0.011128832  0.0351585683   0.018770506  -0.07377799
##    5: 0.4662601  0.010681863 -0.0395533094   0.017015992  -0.07762538
##   ---
## 1226: 0.4662601  0.004434993 -0.1050729728   0.013895747  -0.02693263
## 1227: 0.4662602  0.012199113  0.0177911272   0.015949975  -0.06595766
## 1228: 0.4662602  0.006883470  0.0115652158  -0.009211248  -0.02160611
## 1229: 0.4662602  0.012761079  0.0342215346   0.018005475  -0.07023390
## 1230: 0.4662602  0.014025732  0.0339713956   0.015627717  -0.08159046
##              race length_of_stay
##    1:  0.02475243      0.4363923
##    2:  0.02141949     -0.4899228
##    3: -0.01929222     -0.2901120
##    4: -0.02051219     -0.4291922
##    5:  0.02286888     -0.3930537
##   ---
## 1226: -0.02653896     -0.3311264
## 1227:  0.01879926     -0.4593748
## 1228:  0.01114031      0.1157586
## 1229: -0.02649632     -0.4405813
## 1230:  0.02430835     -0.4730566
```

```r
explanation$dt <- explanation$dt %>%
  rename(model = none) %>%
  select(-race)
#plot(explanation)
```

```r
sharply_mean=colMeans(explanation$dt)
y_pred <- predict(model, x_test)
y_labels <- ifelse(y_pred >= 0.5, 1, 0)
accuracy <- mean(y_labels == y_test)


data_long <- pivot_longer(explanation$dt, cols = everything(), names_to = "Feature", values_to = "Shapl
data_long$Feature <- factor(data_long$Feature, levels = c("model", setdiff(data_long$Feature, "model"))
ggplot(data_long, aes(x = Feature, y = ShapleyValue)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Boxplot of Shapley Values for Each Feature", x = "Feature", y = "Shapley Value")
```




```
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.884552845528455"
```

```
print(accuracy_race(x_test[,'race'],y_labels,y_test))
```

```
## $protected
## [1] 0.900266
##
## $not_protected
## [1] 0.8598326
```

```
print(paste("Calibration:", calibration(x_test[,'race'],y_labels,y_test)))
```

```
## [1] "Calibration: 0.0404333214635448"
```

```
results_xg <- data.frame(
  `XGBoost` = c(accuracy, accuracy_race(x_test[,'race'],y_labels,y_test)$protected, accuracy_race(x_tes
  row.names = c("Model Accuracy", "Protected", "Not Protected", "Calibration")
)
print(sharply_mean)
```

```
##            model             sex             age   charge_degree    priors_count
```

```
##   4.662602e-01  -1.585255e-03  -3.031974e-05  -7.416862e-04  -1.945618e-03
## length_of_stay
##   7.487567e-03
```

```r
result_df <- cbind(results_rf, results_xg)
print(result_df)
```

```
##                 Random.Forest    XGBoost
## Model Accuracy    0.926016260 0.88455285
## Protected         0.929521277 0.90026596
## Not Protected     0.920502092 0.85983264
## Calibration       0.009019185 0.04043332
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.