CART, KNN, Naive Bayes

Laura Cosgrove

5/8/2019

```
knitr::opts_chunk$set(eval = FALSE)

library(tidyverse)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## Registered S3 method overwritten by 'rvest':
##   method            from
##   read_xml.response xml2
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.1.1       v purrr   0.3.2
## v tibble  2.1.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------ 
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(RANN)
library(rpart)
library(rpart.plot)
```

Data loading and train

```
# Using caret
ctrl1 <- trainControl(method = "repeatedcv",
                      repeats = 5,
                      summaryFunction = twoClassSummary, #because we're in the two-class setting
                      classProbs = TRUE) #because need predicted class probabilities to get ROC curve

#Read RDS
cog_train <- readRDS("./data/cog_train_preproc.RDS")
cog_test <- readRDS("./data/cog_test_preproc.RDS")
```

KNN

```
set.seed(13)
knn_fit <- train(x = cog_train[3:10],
                 y = cog_train$cdr,
                  method = "knn",
                  tuneGrid = data.frame(k = seq(1, 100, by = 1)),
                  metric = "ROC",
                 trControl = ctrl1)
ggplot(knn_fit, highlight = TRUE)
knn_fit$finalModel

knn_fit$results[(which.max(knn_fit$results$ROC)),]
```

Resampled AUC: 0.764509

Performance on test data:

```
test_pred_knn  <- predict(knn_fit, newdata = cog_test, type = "raw")

confusionMatrix(data = test_pred_knn,
                reference = cog_test$cdr,
                positive = "Dementia")

test_pred_prob_knn  <- predict(knn_fit, newdata = cog_test, type = "prob")

roc_knn_test <- roc(cog_test$cdr, test_pred_prob_knn$Dementia)

plot(roc_knn_test, legacy.axes = TRUE, print.auc = TRUE)
plot(smooth(roc_knn_test), col = 4, add = TRUE)
```

Classification Tree

```
set.seed(14)
```

```
rpart_fit <- train(x = cog_train[3:10],
                   y = cog_train$cdr,
                   method = "rpart",
                   tuneGrid = data.frame(cp = exp(seq(-12,-3, length = 100))),
                   trControl = ctrl1)

ggplot(rpart_fit, highlight = TRUE) + theme_minimal()


cp_optimal <- rpart_fit$results %>% as.tibble() %>%
  mutate(ROC_onesd = ROC - ROCSD) %>%
  select(cp, ROC, ROC_onesd) %>%
  filter(cp >= cp[which.max(ROC)],
         ROC >= ROC_onesd[which.max(ROC)]) %>%
  summarize(cp_maxROC = min(cp),
            cp_1se = max(cp))

tree_min <- prune(rpart_fit$finalModel, cp = cp_optimal$cp_maxROC)
tree_1se <- prune(rpart_fit$finalModel, cp = cp_optimal$cp_1se)


rpart.plot(tree_1se, box.palette = "GnPu", branch.lty = 3, shadow.col = "gray", nn = TRUE)
```

Naive Bayes

```
set.seed(1)

# 3 tuning parameters
nbGrid <- expand.grid(usekernel = c(FALSE, TRUE), # FALSE means use Gaussian; TRUE means non-parametric
                      fL = 1, # rLaplace smoother. Handles density of points that fall lutside of the o
                      adjust = seq(0,10,by = 1)) # constrols smoothness of density estimate; large mean

model.nb <- train(x = cog_train[3:10],
                  y = cog_train$cdr,
                  method = "nb", # specifies bandwidth of kernel estimate
                  tuneGrid = nbGrid,
                  metric = "ROC",
                  trControl = ctrl1)

plot(model.nb)
model.nb$results[which.max(model.nb$results$ROC), ]
```

```
saveRDS(knn_fit, "./data/knn.RDS")
saveRDS(model.nb, "./data/nb.RDS")
saveRDS(rpart_fit, "./data/cart.RDS")
```