Clinical Prediction Pipeline

JAS

## Exercise: Comparison between Random Forest SVC and Logistic Regression for Clinical Risk Scores

This exercise uses the same data as Assignment 6. You will use three different algorithms (random forest, SVC and logistic regression) to generate a clincal risk score for diabetes. We will then compare the three models.

For this exercise, please attempt the following before class:

1. Run the code chunk dataprep below to process the data.
2. Partition data into a 70/30 training/testing split.
3. Construct three models in the training set using each of the three algorithms to predict diabetes. For the random forest, you should try 3 different values of mtry and 3 different numbers of trees and compare them to each other before selecting a final model to compare to the other two. For the SVC, vary the cost parameter using 10-fold cross validation and choose the optimal tuning value.
4. Compare accuracy across the three models in the training set.

The calculation of the clinical risk scores will be demonstrated within class.As a group, we will do the following:

1. Output predicted probabilities from each of the three models applied within the testing set.
2. Plot and compare calibration curves across the three algorithms.
3. Calibrate the predicted probabilites from SVC and Random Forest using two common methods.
4. Plot and compare the new calibration curves across the three algorithms.

The code below will load the data and process it.

1. Subsetting the data to only include the relevent features
2. Removing observations with missing values
3. Taking a random sample of the non-diabetes so the data will be balanced

## Step 1: Load and Import Data

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────────────── tidyverse 1.2.1 ──

## ✔ ggplot2 3.2.1 ✔ purrr 0.3.2  
## ✔ tibble 2.1.3 ✔ dplyr 0.8.3  
## ✔ tidyr 1.0.0 ✔ stringr 1.4.0  
## ✔ readr 1.3.1 ✔ forcats 0.4.0

## ── Conflicts ────────────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(lattice)  
library(NHANES)  
library(dplyr)  
library(e1071)  
library(rpart)  
library(caret)

##   
## Attaching package: 'caret'

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

library(rpart.plot)  
library(pROC)

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

##   
## Attaching package: 'pROC'

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

set.seed(100)  
  
nhanes\_data = NHANES %>%   
 janitor::clean\_names() %>%   
 select(age, gender, race1, education, hh\_income, weight, height, pulse, diabetes, bmi, phys\_active, smoke100, bp\_sys\_ave, bp\_dia\_ave, tot\_chol) %>%   
 mutate(  
 id = row\_number(),   
 id = as.character(id)) %>%   
 drop\_na()  
  
diabetes = nhanes\_data %>%   
 filter(diabetes == "Yes")  
  
no\_diabetes = nhanes\_data %>%   
 filter(diabetes == "No") %>%   
 sample\_n(760)  
  
data\_final = rbind(diabetes, no\_diabetes)

## Step 2: Training/Testing Data Sets

train\_nhanes = data\_final %>% sample\_frac(.7)  
test\_nhanes = anti\_join(data\_final, train\_nhanes, by = 'id') %>%   
 select(-id, -diabetes, diabetes)  
  
train\_nhanes = train\_nhanes %>%   
 select(-id, -diabetes, diabetes)

## Step 3: Try 3 Models to Predict Diabetes

#1: Logistic Regression  
nhanes\_reg = glm(  
 diabetes ~.,   
 family = binomial(link = 'logit'),  
 data = train\_nhanes)  
  
prob\_reg\_interim = predict(nhanes\_reg,   
 train\_nhanes,   
 type = 'response')  
  
results\_reg1 = ifelse(prob\_reg\_interim > 0.5,1,0)  
  
outcome\_reg = (as.numeric(train\_nhanes$diabetes) - 1)  
  
table(results\_reg1, outcome\_reg)

## outcome\_reg  
## results\_reg1 0 1  
## 0 422 114  
## 1 111 310

accuracy\_reg = mean(results\_reg1 == outcome\_reg)  
  
round(accuracy\_reg, digits = 3)

## [1] 0.765

#2: SVC  
svm\_tune = tune(svm,   
 diabetes~.,   
 data = train\_nhanes,   
 kernel="linear",   
 range = list(cost = 10^(-1:1)))  
  
print(svm\_tune)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost  
## 0.1  
##   
## - best performance: 0.2529057

## best cost = 0.1  
  
nhanes\_svm = svm(diabetes ~ .,   
 data = train\_nhanes,   
 kernel = "linear",   
 cost = 0.1,  
 scale = TRUE)  
  
pred\_svm = predict(nhanes\_svm,   
 newdata = train\_nhanes[,1:14])  
  
table(pred\_svm, train\_nhanes$diabetes)

##   
## pred\_svm No Yes  
## No 422 115  
## Yes 111 309

accuracy\_svm = mean(pred\_svm == train\_nhanes$diabetes)  
  
round(accuracy\_svm, digits = 3)

## [1] 0.764

#3: Random Forests