lungs and smoking

Luke Yee

8/8/2020

```
library(tidyverse)
```

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

```
## v ggplot2 3.3.1     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------- tidyverse_co
## 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
```

```
load(file = "~/misc_data/lung.RData")
```

I used an 800 observation dataset called "lung" with three columns, "biopsy", "smoke_years", and "second_hand_years".Biopsy is either a 1 or 0 indicating if a person had lung cancer or not, the other two are continous variables measuring exposure to cigarettes. I will divide the dataset into groups using different approaches to demonstrate a range of methods. The desired result is to have a training group and a test group, with each group divided between lung cancer observations and no lung cancer observations, for a total of four groups. In this case my train:test ratio will simply be 1:1, but I could make it 1:5 or 1:10 as needed.

```
#APPROACH 1: (use the caret library and createDataPartition)
sampling_vector <- createDataPartition(lung$biopsy, p = 0.5, list = FALSE) #use as index
train_biopsy0 <- lung[sampling_vector,,] %>% filter(biopsy == 0)
train_biopsy1 <- lung[sampling_vector,,] %>% filter(biopsy == 1)
test_biopsy0 <- lung[-sampling_vector,,] %>% filter(biopsy == 0)
test_biopsy1 <- lung[-sampling_vector,,] %>% filter(biopsy == 1)

#alternatively, you can use the createFolds function to create your sampling vector index
# sampling_vector2 <- createFolds(lung$biopsy, k = 2, list = FALSE)
# train_biopsy0 <- lung[sampling_vector2==1,,] %>% filter(biopsy == 0)
# and so on

#APPROACH 2: (if you aren't familiar with the caret library) seperate into two sets one with lung cance
```

```
# biopsy0 <- lung %>% filter(biopsy == 0)
# biopsy1 <- lung %>% filter(biopsy == 1)

#divide each set into half, training data and testing data.

# training_set0 <- biopsy0[sample(nrow(biopsy0), round(nrow(biopsy0)/2), replace = FALSE),,]
# test_set0 <- anti_join(biopsy0, training_set0)
#
# training_set1 <- biopsy1[sample(nrow(biopsy1), round(nrow(biopsy1)/2), replace = FALSE),,]
# test_set1 <- anti_join(biopsy1, training_set1)
```

From here, you could start looking into the data, based on the divided groups. For example, we can find the average number of smoking years for those who did not have lung cancer and for those who did have lung cancer

```
train_biopsy0 %>% pull(smoke_years) %>% mean()
```

```
## [1] 3.883412
```

```
train_biopsy1 %>% pull(smoke_years) %>% mean()
```

```
## [1] 11.01451
```

As expected, the group without lung cancer averaged less smoking years than the group with lung cancer. However, R can do much more than this. Let's try a typical classification problem. For each observation, based on the amount of smoke years and second hand years, the model will predict if a person had lung cancer or not

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

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

```
#combine to create a training and testing set
traindata <- rbind(train_biopsy0, train_biopsy1)
testdata <- rbind(test_biopsy0, test_biopsy1)

train_lda <- lda(biopsy ~ smoke_years + second_hand_years, traindata) #train the model
test_lda <- predict(train_lda, testdata)
test_matrix1 <- table(true = testdata$biopsy, predicted = test_lda$class)
test_matrix1
```

```
##     predicted
## true   0   1
##    0 385   0
##    1  13  19
```

```
(error_rate1 <- (test_matrix1[1,2] + test_matrix1[2,1]) / (test_matrix1[1,1] + test_matrix1[1,2] + test
```

```
## [1] 0.03117506
```

In this case, the error rate of our model is roughly 3.1%. We can also look further into information provided by the discriminant analysis. For example, back when we were looking at average smoking years, a single command could have given us the answer

```
train_lda$means   #answer to earlier question
```

```
##   smoke_years second_hand_years
## 0    3.883412          6.503502
## 1   11.014509          4.964803
```

We can also look at the coefficients that the model used to determine if an observation belonged to the cancer or no-cancer group

```
train_lda$scaling
```

```
##                         LD1
## smoke_years        0.4302795
## second_hand_years -0.1451589
```

Let's try using a generalized linear model now

```
train_glm <- glm(biopsy ~ smoke_years + second_hand_years, traindata, family = binomial)
test_glm <- predict(train_glm, testdata)
test_glm <- ifelse(test_glm > 0, 1, 0)
test_matrix2 <- table(true = testdata$biopsy, predicted = test_glm)
test_matrix2
```

```
##     predicted
## true   0   1
##    0 385   0
##    1  11  21
```

```
(error_rate2 <- (test_matrix2[1,2] + test_matrix2[2,1]) / (test_matrix2[1,1] + test_matrix2[1,2] + test
```

```
## [1] 0.0263789
```

The model performs slightly better than LDA in this case, with an error rate of about 2.6%. We can learn more about this model using a summary call. For instance, we can determine the quality of our variables with the glm by looking at the p-values in the summary. In this case, we can see that smoke years is a far more significant predictor than second hand years.

```
summary(train_glm)
```

```
##
## Call:
## glm(formula = biopsy ~ smoke_years + second_hand_years, family = binomial,
##     data = traindata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0634  -0.0371  -0.0028  -0.0002   3.5261
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -17.6008     5.7530  -3.059  0.00222 **
## smoke_years         2.3898     0.7282   3.282  0.00103 **
## second_hand_years  -0.5315     0.2789  -1.906  0.05665 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 172.38  on 417  degrees of freedom
## Residual deviance:  28.14  on 415  degrees of freedom
## AIC: 34.14
##
## Number of Fisher Scoring iterations: 10
```

We will also try using k-nearest-neighbors to classify this dataset

```
library(class)
```

```
knn1 <- knn(train = traindata[c("smoke_years", "second_hand_years")], test = testdata[c("smoke_years",
cbind(testdata, knn1)
test_matrix3 <- table(true = testdata$biopsy, predicted = knn1)
```

```
test_matrix3
```

```
##     predicted
## true   0   1
##    0 385   0
##    1  11  21
```

```
(error_rate3 <- (test_matrix3[1,2] + test_matrix3[2,1]) / (test_matrix3[1,1] + test_matrix3[1,2] + test
```

```
## [1] 0.0263789
```

With k = 3, the error rate is roughly 2.3%. Lets try increasing k to see its effect

```
knn2 <- knn(train = traindata[c("smoke_years", "second_hand_years")], test = testdata[c("smoke_years",
cbind(testdata, knn2)
```

```
test_matrix4 <- table(true = testdata$biopsy, predicted = knn2)
test_matrix4
```

```
##     predicted
## true   0   1
##    0 385   0
##    1  15  17
```

```
(error_rate4 <- (test_matrix4[1,2] + test_matrix4[2,1]) / (test_matrix4[1,1] + test_matrix4[1,2] + test
```

```
## [1] 0.03597122
```

With k = 7, the error rate increases to roughly 3.8%. It seems as error rate increases with K, and a likely explanation would be the overabundance of 0 values in the true set. As k increases, on-the-fence observations that truly belong to the 1 class will also consider the impact of the 0 values that are nearby, turning some true 1 values into predicted 0 values. As k decreases and less neighbors are considered, values that are truly 1 are less swayed by the nearby 0 values. For example, if k = # of all observations in the set, every single prediction would consider nearly 400 "0" values and only 20 "1" values, and the majority class would be absolutely dominant in predictions.

```
knn3 <- knn(train = traindata[c("smoke_years", "second_hand_years")], test = testdata[c("smoke_years",
cbind(testdata, knn3)
```

```
test_matrix5 <- table(true = testdata$biopsy, predicted = knn3)
test_matrix5
```

```
##     predicted
## true   0   1
##    0 385   0
##    1  32   0
```

```
(error_rate5 <- (test_matrix5[1,2] + test_matrix5[2,1]) / (test_matrix5[1,1] + test_matrix5[1,2] + test
```

```
## [1] 0.07673861
```

As expected, the model does not predict a single “1” case when all observations are used