Machine Learning on CVD Data

Ted Laderas

8/8/2017

Optional: Running multiple machine learning methods on the `cvd_patient` dataset

The following section is optional. You can use a variety of other machine learning methods conveniently wrapped in the 'caret' package. Here we show how to use the following methods:

- `lda` - linear discriminant analysis,
- `rpart` - Classification and regression trees

A full list of machine learning methods in `caret` is available here: http://topepo.github.io/caret/available-models.html Note that you will also have to install the corresponding packages listed for that method.

Predicting CVD Risk

We will attempt to predict cardiovascular risk using a patient dataset called `cvd_patient`, which we will load from a Dropbox folder. This dataset is completely synthetic, so don't worry about patient confidentiality.

```
library(tidyverse)
```

```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr

## Conflicts with tidy packages ----------------------------------------------

## filter(): dplyr, stats
## lag():    dplyr, stats
```

```
library(broom)
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

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

```
set.seed(111)
cvd_patient <- read.csv("https://www.dropbox.com/s/ve6a3at7h2r9zkg/patient_cvd.csv?raw=1")

summary(cvd_patient)
```

```
##   cvd            age                 race        gender          bmi       
##   N:28566   Min.   :41.00   AmInd     :  170   F:21034   Min.   :15.00  
##   Y: 7034   1st Qu.:48.00   Asian/PI  : 6389   M:14566   1st Qu.:19.00  
##             Median :57.00   Black/AfAm: 2046             Median :21.00  
##             Mean   :58.64   White     :26995             Mean   :21.91  
##             3rd Qu.:67.00                                3rd Qu.:24.00  
##             Max.   :90.00                                Max.   :36.00  
##       sbp          htn             tchol        smoking    
##   Min.   : 81.0   N:21965   Min.   :155.0   N:31273  
##   1st Qu.:117.0   Y:13635   1st Qu.:160.0   Y: 4327  
##   Median :128.0             Median :181.0            
##   Mean   :142.1             Mean   :188.1            
##   3rd Qu.:175.0             3rd Qu.:207.0            
##   Max.   :217.0             Max.   :245.0            
```

Separating Our Data

One of the things we might like to check is the predictive power of the model. For this reason, we want to save a little bit of our data that the model doesn't "see" for testing the predicitve power of the model.

We hold out 20 percent of the data by using the `createPartitionData()` function in `caret`. `createPartitionData()` returns a number of rows that we can use to subset the data into two sets: 1) our `test` dataset (20% of our data), which we'll use to test our model's predictive value, and 2) our `training` dataset (80% of our data), which we'll use to actually build (or train) our model.

```
#grab indices of the dataset that represent 80% of the data
trainingIndices <- createDataPartition(y = cvd_patient$cvd, p=.80,
                                       list=FALSE)

#show the first few training indices
trainingIndices[1:10]
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
#select the rows
trainData <- cvd_patient[trainingIndices,]
#confirm the number of rows (should be 80)
nrow(trainData)
```

```
## [1] 28481
```

```
#build our test set using the R-indexing
#using the "-" operator
testData <- cvd_patient[-trainingIndices,]

#confirm the number of rows
nrow(testData)
```

```
## [1] 7119
```

Build the Models using caret

The `caret` package gives us a standard function to train our learners using the `train()` function. Notice that it uses a similar format to the `glm()` function, which we used for logistic regression.

```r
#train linear discriminant analysis method
ldaCVD <- train(cvd ~ age + gender, method= "lda", data=trainData)

#train classification and regression tree
cartCVD <- train(cvd ~ age + gender, method= "rpart", data=trainData)
```

Assessing the models on the Test Set

Now that we have our models trained, we can evaluate them on our test dataset. To do this, we use the `predict` function, and pass both our trained learner `ldaCVD` and our `testData` into `predict`.

```r
#Predict cvd on test data
classPredLDA <- predict(ldaCVD, newdata=testData)

#Compare predictions directly with the truth
data.frame(classPredLDA, truth=testData$cvd)[1:10,]
```

```
##    classPredLDA truth
## 1             N     N
## 2             N     N
## 3             N     N
## 4             N     N
## 5             N     N
## 6             N     Y
## 7             N     N
## 8             N     N
## 9             N     Y
## 10            N     N
```

Here we evaluate our LDA model based on how accurately it classifies test set samples as the correct cvd.

```r
truthPredict_lda <-  table(testData$cvd, classPredLDA)

#number of cases
totalCases_lda <-  sum(truthPredict_lda)
totalCases_lda
```

```
## [1] 7119
```

```r
#number of misclassified samples
misclassified_lda <- truthPredict_lda[1,2] + truthPredict_lda[2,1]
misclassified_lda
```

```
## [1] 1336
```

```r
accuracy_lda <- (totalCases_lda - misclassified_lda) / totalCases_lda
accuracy_lda
```

```
## [1] 0.8123332
```

We can also use the `caret` package to make the confusion matrices more quickly! Luckily when we compare the accuracy measures compute by our method and `caret` they are the same.

```r
#calculate confusion Matrix and other measures of accuracy
confMatLDA <- confusionMatrix(testData$cvd, classPredLDA)
```

```r
#Show everything from `confusionMatrix`
confMatLDA
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    N    Y
##          N 5562  151
##          Y 1185  221
##
##                Accuracy : 0.8123
##                  95% CI : (0.8031, 0.8213)
##     No Information Rate : 0.9477
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1809
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8244
##             Specificity : 0.5941
##          Pos Pred Value : 0.9736
##          Neg Pred Value : 0.1572
##              Prevalence : 0.9477
##          Detection Rate : 0.7813
##    Detection Prevalence : 0.8025
##       Balanced Accuracy : 0.7092
##
##        'Positive' Class : N
##
```

```r
#access confusion matrix directly
confMatLDA$table
```

```
##           Reference
## Prediction    N    Y
##          N 5562  151
##          Y 1185  221
```

```r
#Show accuracy values
confMatLDA$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   8.123332e-01   1.809009e-01   8.030659e-01   8.213440e-01   9.477455e-01
## AccuracyPValue  McnemarPValue
##   1.000000e+00  1.024486e-175
```

```r
#Show class agreement values
confMatLDA$byClass
```

```
##          Sensitivity          Specificity       Pos Pred Value
##            0.8243664            0.5940860            0.9735691
##       Neg Pred Value            Precision               Recall
##            0.1571835            0.9735691            0.8243664
##                   F1           Prevalence       Detection Rate
##            0.8927769            0.9477455            0.7812895
## Detection Prevalence    Balanced Accuracy
##            0.8025004            0.7092262
```

So which algorithm did best?

Let's run our predictions on the other learners as well, and compare accuracies:

```
classPredCart <- predict(cartCVD, newdata = testData)

#compare all the predictions directly
#were there any rows where the predictions didn't match?
data.frame(truth=testData$cvd, LDA=classPredLDA, CART=classPredCart)[1:10,]
```

```
##    truth LDA CART
## 1      N   N    N
## 2      N   N    N
## 3      N   N    N
## 4      N   N    N
## 5      N   N    N
## 6      Y   N    N
## 7      N   N    N
## 8      N   N    N
## 9      Y   N    N
## 10     N   N    N
```

Comparing Accuracies of our models

Here we compare the accuracies of our models.

```
confMatCart <- confusionMatrix(classPredCart, testData$cvd)

accuracyComparison = rbind(
                       LDA = confMatLDA$overall,
                       CART = confMatCart$overall
                    )

accuracyComparison
```

```
##       Accuracy     Kappa AccuracyLower AccuracyUpper AccuracyNull
## LDA  0.8123332 0.1809009     0.8030659     0.8213440    0.9477455
## CART 0.8089619 0.1446112     0.7996340     0.8180362    0.8025004
##      AccuracyPValue McnemarPValue
## LDA      1.00000000 1.024486e-175
## CART     0.08737198 7.547992e-196
```