Thoracic surgery survival

Yograj Karki

5/19/2021

Loading libraries

```
library(caTools)
```

Loading the dataset

Here I created csv file by copying the contents from the arff file, and then fixed the attributes of certain columns.

```
setwd("~/MSDS/DSC520/dsc520/thoracic_surgery")

data <- read.csv("thoracic.csv")
str(data)
```

```
## 'data.frame':    470 obs. of  17 variables:
##  $ DGN    : Factor w/ 7 levels "DGN1","DGN2",..: 2 3 3 3 3 3 3 2 3 3 ...
##  $ PRE4   : num  2.88 3.4 2.76 3.68 2.44 2.48 4.36 3.19 3.16 2.32 ...
##  $ PRE5   : num  2.16 1.88 2.08 3.04 0.96 1.88 3.28 2.5 2.64 2.16 ...
##  $ PRE6   : Factor w/ 3 levels "PRZ0","PRZ1",..: 2 1 2 1 3 2 2 2 3 2 ...
##  $ PRE7   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ PRE8   : logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
##  $ PRE9   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ PRE10  : logi  TRUE FALSE TRUE FALSE TRUE TRUE ...
##  $ PRE11  : logi  TRUE FALSE FALSE FALSE TRUE FALSE ...
##  $ PRE14  : Factor w/ 4 levels "OC11","OC12",..: 4 2 1 1 1 1 2 1 1 1 ...
##  $ PRE17  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ PRE19  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ PRE25  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ PRE30  : logi  TRUE TRUE TRUE FALSE TRUE FALSE ...
##  $ PRE32  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ AGE    : int  60 51 59 54 73 51 59 66 68 54 ...
##  $ Risk1Yr: logi  FALSE FALSE FALSE FALSE TRUE FALSE ...
```

Splitting the data into train and test datasets

```r
# Splitting the data into train and test with the 80:20 ratio
dt = sort(sample(nrow(data), nrow(data)*0.8))

# Training dataset
train<-data[dt,]

# Testing dataset
test<-data[-dt,]
```

Fitting the data to a logistic regression model

Here I used the glm() function to fit the model and the method was binomial logistic regression. Note: I used . notation to select all other independent variables in the glm function parameters.

```r
# fitting the model
logistic <- glm(Risk1Yr ~ ., data = train, family="binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
# summary of the model
summary(logistic)
```

```
##
## Call:
## glm(formula = Risk1Yr ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7971  -0.5265  -0.3682  -0.2157   2.5431
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -15.53069 2399.54544  -0.006  0.99484
## DGNDGN2       14.61422 2399.54479   0.006  0.99514
## DGNDGN3       13.74090 2399.54477   0.006  0.99543
## DGNDGN4       13.50021 2399.54486   0.006  0.99551
## DGNDGN5       16.59008 2399.54486   0.007  0.99448
## DGNDGN6        0.39980 2674.18769   0.000  0.99988
## DGNDGN8       18.09075 2399.54522   0.008  0.99398
## PRE4           0.02278    0.40093   0.057  0.95469
## PRE5          -0.52599    0.47142  -1.116  0.26453
## PRE6PRZ1      -0.41810    0.59905  -0.698  0.48522
## PRE6PRZ2      -0.51018    0.95233  -0.536  0.59216
## PRE7TRUE       1.78807    0.62250   2.872  0.00407 **
## PRE8TRUE       0.09182    0.45170   0.203  0.83892
## PRE9TRUE       1.49928    0.56951   2.633  0.00847 **
## PRE10TRUE      0.88668    0.55739   1.591  0.11166
## PRE11TRUE      0.85763    0.45960   1.866  0.06203 .
## PRE14OC12      0.59216    0.38513   1.538  0.12416
## PRE14OC13      0.53003    0.74346   0.713  0.47589
```

```
## PRE14OC14      1.68220    0.83133   2.024  0.04302 *
## PRE17TRUE      0.94114    0.51947   1.812  0.07003 .
## PRE19TRUE    -15.05295 1610.46089  -0.009  0.99254
## PRE25TRUE     -0.05192    1.09896  -0.047  0.96232
## PRE30TRUE      0.96935    0.55836   1.736  0.08255 .
## PRE32TRUE    -13.94421 1667.38928  -0.008  0.99333
## AGE           -0.01953    0.02127  -0.918  0.35858
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 305.84  on 375  degrees of freedom
## Residual deviance: 247.17  on 351  degrees of freedom
## AIC: 297.17
##
## Number of Fisher Scoring iterations: 15
```

Checking the accuracy of the model

Running the test data throught the model

```
##            1            5           18           20           33           43
## 6.940514e-01 2.027267e-01 2.047146e-01 3.089701e-02 8.302683e-01 7.600318e-02
##           45           48           51           53           61           73
## 3.841896e-01 9.791539e-02 3.749891e-02 7.205021e-01 8.276480e-02 2.913206e-02
##           74           76           79           82           85           88
## 3.511760e-03 2.961729e-01 1.792458e-01 3.405665e-01 5.458278e-02 1.815994e-01
##           89           96           97           99          110          112
## 7.322236e-01 3.931625e-02 8.641507e-02 2.220446e-16 2.946415e-01 2.266053e-01
##          124          125          127          129          132          133
## 2.556065e-02 1.440121e-01 5.374246e-02 2.402224e-01 1.116033e-01 2.220446e-16
##          138          144          145          147          152          155
## 4.064629e-01 6.328683e-02 2.139251e-01 1.450202e-02 4.953588e-02 6.176448e-02
##          161          164          175          178          181          184
## 2.649280e-02 9.470652e-02 2.280887e-01 1.127615e-01 1.787039e-01 7.197388e-02
##          188          199          207          219          228          231
## 6.971125e-02 1.807488e-02 3.363010e-02 3.671339e-02 9.857330e-02 1.852688e-01
##          242          247          249          250          251          253
## 5.973706e-02 1.903169e-02 4.552778e-02 1.116857e-01 7.696867e-02 7.947362e-02
##          259          264          266          278          279          282
## 6.960343e-02 7.821830e-03 6.425960e-02 1.857301e-01 9.059207e-03 1.369522e-02
##          284          285          286          289          305          311
## 2.941928e-01 5.071016e-02 7.891187e-02 5.237099e-01 3.060561e-02 8.843997e-03
##          318          319          320          322          324          325
## 3.010862e-01 7.057708e-02 2.220446e-16 4.126512e-02 4.842370e-01 6.528293e-03
##          332          338          340          341          363          364
## 2.580416e-02 6.651329e-02 1.305229e-01 2.721861e-02 5.179824e-01 1.663883e-01
##          372          390          393          407          413          420
## 4.130803e-02 4.871355e-01 6.435389e-02 5.201030e-02 1.577893e-02 1.571920e-01
##          421          422          429          439          445          452
## 3.124935e-01 3.668936e-01 2.392542e-01 2.220446e-16 2.220446e-16 8.910825e-02
```

```
##          453          454          461          469
## 3.740027e-01 1.230385e-01 1.934002e-02 2.169172e-01
```

Confusion matrix

```
confmatrix <- table(Actual_value = test$Risk1Yr, Predicted_value = res >0.5)
confmatrix
```

```
##             Predicted_value
## Actual_value FALSE TRUE
##        FALSE    72    5
##        TRUE     16    1
```

Accuracy percentage

```
#calculating accuracy percentage
(confmatrix[[1]]+confmatrix[[2]])/sum(confmatrix)
```

```
## [1] 0.9361702
```

Our model turned out to be 95.7% accurate which is an excellent model.