Logistic Regression

XDASI Fall 2021

Answer KEY

Introduction

Logistic regression is a form of machine learning where a generalized linear model can be created using a training set and then tested using another dataset. Logistic regression method is very similar to linear regression but the output variable is binary instead of continuous and the variance does not have to be equal. In this example we will use a breast cancer dataset containing 15,000 observations (patients) to create a model which can be used to predict the output variable, in this case whether the patient will die. The output variable is alivestatus and the input variables are nodespos (number of positive nodes, size size of the tumor, and grade tumor grade. Tumor cells are classified into 4 different grades:

- Grade 1) low-grade: cells are well differentiated.
- Grade 2) intermediate-grade: moderately well differentiated,
- Grade 3) poorly differentiated and
- Grade 4) undifferentiated.

Grades 3 and 4 are generally associated with poor prognosis.

Example data and scripts in this exercise are taken from [1].

Analysis

Loading the data

First make sure working directory is set correctly. Then read the file into the variable called my_data.

```
my_data <- read.table("Data_all.15k.patients.txt")

head(my_data)
```

```
##          dateofbirth maritalstatus race ageatdiagnosis alivestatus survivaltime
## pid00001  28/01/1932             5    2             61           0          110
## pid00002  07/05/1934             2    3             60           0          100
## pid00003  14/04/1921             5    3             76           0           70
## pid00004  08/11/1931             3    3             61           1           31
## pid00005  08/01/1922             2    3             74           1           47
## pid00006  03/12/1939             2    3             55           0          100
##          grade nodesexam nodespos extent nodalstatus size pgr er
## pid00001     3        32        3     10           6   60   2  2
## pid00002     2        13        1     10           6   15   1  1
## pid00003     3         8        0     10           0    8   1  1
## pid00004     3        20        0     10           0   10   2  2
```

```
## pid00005     2        16        8     10           6   15   2  1
## pid00006     3        19        0     10           0   48   1  1
```

```
str(my_data)
```

```
## 'data.frame':    15000 obs. of  14 variables:
##  $ dateofbirth   : chr  "28/01/1932" "07/05/1934" "14/04/1921" "08/11/1931" ...
##  $ maritalstatus : int  5 2 5 3 2 2 2 4 4 2 ...
##  $ race          : int  2 3 3 3 3 3 1 2 2 3 ...
##  $ ageatdiagnosis: int  61 60 76 61 74 55 84 81 70 78 ...
##  $ alivestatus   : int  0 0 0 1 1 0 0 0 0 0 ...
##  $ survivaltime  : int  110 100 70 31 47 100 87 86 138 86 ...
##  $ grade         : int  3 2 3 3 2 3 3 2 3 2 ...
##  $ nodesexam     : int  32 13 8 20 16 19 3 13 21 15 ...
##  $ nodespos      : int  3 1 0 0 8 0 0 0 0 0 ...
##  $ extent        : int  10 10 10 10 10 10 10 10 10 10 ...
##  $ nodalstatus   : int  6 6 0 0 6 0 0 0 0 0 ...
##  $ size          : int  60 15 8 10 15 48 32 15 22 6 ...
##  $ pgr           : int  2 1 1 2 2 1 2 1 1 2 ...
##  $ er            : int  2 1 1 2 1 1 2 1 1 1 ...
```

Performing Logistic Regression

Creating a logistic regression model is very similar to creating a linear regression model. Setting the parameter "family" to "binomial" tells the **glm()** function to create a logistic regression model.

```
my_lr <- glm(alivestatus ~ size + nodespos + grade,
             data = my_data,
             family = "binomial")

summary(my_lr)
```

```
##
## Call:
## glm(formula = alivestatus ~ size + nodespos + grade, family = "binomial",
##     data = my_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.0612  -0.6022  -0.4451  -0.3164   2.5375
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.899557   0.095006  -41.05   <2e-16 ***
## size         0.023213   0.001391   16.69   <2e-16 ***
## nodespos     0.149229   0.005960   25.04   <2e-16 ***
## grade        0.628004   0.035757   17.56   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 14307  on 14999  degrees of freedom
## Residual deviance: 12094  on 14996  degrees of freedom
## AIC: 12102
##
## Number of Fisher Scoring iterations: 5
```

Note that the summary output of logistic regression model is very similar to that of linear regression.

The **predict()** function can be used to determine the probability of getting the value 1. The parameter "type" specifies which results to return. In order to retrieve the predicted probability of the outcome "1" (1 in the output variable alivestatus means death.) use "response".

```
predicted_alivestatus <- predict(my_lr,
                                 newdata = my_data,
                                 type = "response")

my_data$predicted_alivestatus <- predicted_alivestatus

head(my_data)
```

```
##          dateofbirth maritalstatus race ageatdiagnosis alivestatus survivaltime
## pid00001  28/01/1932             5    2             61           0          110
## pid00002  07/05/1934             2    3             60           0          100
## pid00003  14/04/1921             5    3             76           0           70
## pid00004  08/11/1931             3    3             61           1           31
## pid00005  08/01/1922             2    3             74           1           47
## pid00006  03/12/1939             2    3             55           0          100
##          grade nodesexam nodespos extent nodalstatus size pgr er
## pid00001     3        32        3     10           6   60   2  2
## pid00002     2        13        1     10           6   15   1  1
## pid00003     3         8        0     10           0    8   1  1
## pid00004     3        20        0     10           0   10   2  2
## pid00005     2        16        8     10           6   15   2  1
## pid00006     3        19        0     10           0   48   1  1
##          predicted_alivestatus
## pid00001             0.4563383
## pid00002             0.1046935
## pid00003             0.1382568
## pid00004             0.1438815
## pid00005             0.2494546
## pid00006             0.2887764
```

From the prediction results, a probability greater than 0.5 will most likely be dead (have a value of 1). Using this cutoff, a predicted result vector can be created which can be compared to the actual result. In order to compare the predicted with actual results there are certain terminologies that are important to understand:

- TP: True positive: Predictions of TRUE event and it is actually TRUE
- TN: True negative: Prediction of FALSE event and it is actually FALSE.
- FP: False positive: Prediction of TRUE even, but it is actually FALSE.
- FN: False negative: Prediction of FALSE even and it is TRUE

These numbers in combination can provide useful statistics to judge the quality of the model.

- accuracy = (TP + TN) / (TP + TN + FP + FN)
- precision = TP / (TP + FP)
- sensitivity = recall = true positive rate = TP / (TP + FN)
- specificity = true negative rate = TN / (TN + FP)

```
my_data$predicted_alivestatus_binary <- ifelse(test = predicted_alivestatus >= 0.5,
                                               yes = 1,
                                               no = 0)

confmat <- table(my_data[,c("alivestatus","predicted_alivestatus_binary")])
confmat
```

```
##            predicted_alivestatus_binary
## alivestatus     0     1
##           0 11928   317
##           1  2203   552
```

```
TP <- confmat[2,2]
TN <- confmat[1,1]
FP <- confmat[1,2]
FN <- confmat[2,1]

accuracy <- (TP + TN) / (TP + TN + FP + FN)
accuracy
```

```
## [1] 0.832
```

```
precision <- TP / (TP + FP)
precision
```

```
## [1] 0.6352129
```

```
sensitivity <- TP / (TP + FN)
sensitivity
```

```
## [1] 0.200363
```

```
specificity <- TN / (TN + FP)
specificity
```

```
## [1] 0.9741119
```

Recall is also often referred to as the measurement of how specific the method is and Precision is the measurement of sensitivity. We can use the AUC package to see how the sensitive and specity change as the cutoff is changed. Several performance characteristics plotted against each other is called receiver operating characteristic curve, or ROC curve.

ROC curve

```
my_roc <- roc(data = my_data,
              response = alivestatus,
              predictor = predicted_alivestatus)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(x = my_roc$thresholds,
     y = my_roc$sensitivities,
     col = "blue",
     type = "l")

points(x = my_roc$thresholds,
       y = my_roc$specificities,
       col = "red",
       type = "l")
```