Logistic Regression

Toan

2016年4月24日

Problem:

A researcher is interested in how variables, such as GRE (Graduate Record Exam scores), GPA (grade point average) and prestige of the undergraduate institution, effect admission into graduate school. The response variable, admit/don’t admit, is a binary variable.

1. Description of the data

```
mydata <- read.csv("http://www.ats.ucla.edu/stat/data/binary.csv")
## view the first few rows of the data
head(mydata)
```

```
##   admit gre  gpa rank
## 1     0 380 3.61    3
## 2     1 660 3.67    3
## 3     1 800 4.00    1
## 4     1 640 3.19    4
## 5     0 520 2.93    4
## 6     1 760 3.00    2
```

```
str(mydata)
```

```
## 'data.frame':    400 obs. of  4 variables:
##  $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
##  $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
##  $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
##  $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
```

```
summary(mydata)
```

```
##      admit             gre             gpa             rank      
##  Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000  
##  Median :0.0000   Median :580.0   Median :3.395   Median :2.000  
##  Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :2.485  
##  3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000  
##  Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :4.000
```

```
sapply(mydata, sd)
```

```
##       admit         gre         gpa        rank 
##   0.4660867 115.5165364   0.3805668   0.9444602
```

```
## two-way contingency table of categorical outcome and predictors
## we want to make sure there are not 0 cells
xtabs(~ admit + rank, data = mydata)
```

```
##      rank
## admit  1  2  3  4
##     0 28 97 93 55
##     1 33 54 28 12
```

2. Building Logistic Regression.

```
mydata$rank <- factor(mydata$rank)
mylogit <- glm(admit ~ gre + gpa + rank, data = mydata, family = "binomial")
summary(mylogit)
```

```
## 
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = "binomial", 
##     data = mydata)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -1.6268  -0.8662  -0.6388   1.1490   2.0790  
## 
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)    
## (Intercept) -3.989979   1.139951  -3.500 0.000465 ***
## gre          0.002264   0.001094   2.070 0.038465 *  
## gpa          0.804038   0.331819   2.423 0.015388 *  
## rank2       -0.675443   0.316490  -2.134 0.032829 *  
## rank3       -1.340204   0.345306  -3.881 0.000104 ***
## rank4       -1.551464   0.417832  -3.713 0.000205 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 458.52  on 394  degrees of freedom
## AIC: 470.52
## 
## Number of Fisher Scoring iterations: 4
```

3. Interpretation of the model.

```
## CIs using profiled log-likelihood
cbind(coef(mylogit), confint(mylogit))
```

```
## Waiting for profiling to be done...
```

```
##                                 2.5 %       97.5 %
## (Intercept) -3.989979073 -6.2716202334 -1.792547080
## gre          0.002264426  0.0001375921  0.004435874
## gpa          0.804037549  0.1602959439  1.464142727
## rank2       -0.675442928 -1.3008888002 -0.056745722
## rank3       -1.340203916 -2.0276713127 -0.670372346
## rank4       -1.551463677 -2.4000265384 -0.753542605
```

```
## CIs using standard errors
cbind(coef(mylogit), confint.default(mylogit))
```

```
##                                 2.5 %       97.5 %
## (Intercept) -3.989979073 -6.2242418514 -1.755716295
## gre          0.002264426  0.0001202298  0.004408622
## gpa          0.804037549  0.1536836760  1.454391423
## rank2       -0.675442928 -1.2957512650 -0.055134591
## rank3       -1.340203916 -2.0169920597 -0.663415773
## rank4       -1.551463677 -2.3703986294 -0.732528724
```

Note: Interpretation for gpa as flow, if gpa increase 1 unit then the log of odd of get admission increase by 0.804. the interpretation for other coefficients are the same. ### 4. Interpretation by odd of event.

```
# Change log of odd value to odd value.
exp(cbind(coef(mylogit), confint(mylogit)))
```

```
## Waiting for profiling to be done...
```

```
##                            2.5 %    97.5 %
## (Intercept) 0.0185001 0.001889165 0.1665354
## gre         1.0022670 1.000137602 1.0044457
## gpa         2.2345448 1.173858216 4.3238349
## rank2       0.5089310 0.272289674 0.9448343
## rank3       0.2617923 0.131641717 0.5115181
## rank4       0.2119375 0.090715546 0.4706961
```

Note: If GPA increase one uinit the odd of admission increase by 2.2345. Other coefficeints are interpretated as same as GPA.

5. Assess model fit.

```
#find the different in the residual deviance between the model with
#predictors and the null model (intercept only)
with(mylogit, null.deviance - deviance)
```

```
## [1] 41.45903
```

```
#find the different in the degree of freedom.
with(mylogit, df.null - df.residual)
```

```
## [1] 5
```

```
#find the p-value of the test.
with(mylogit, pchisq(null.deviance - deviance, df.null - df.residual, lower.tail = FALS
E))
```

```
## [1] 7.578194e-08
```

```
# calculate the log of likelihood-ratio test
logLik(mylogit)
```

```
## 'log Lik.' -229.2587 (df=6)
```

Note : From this test we can reject the hypothesis that there is no different between the null model and the model with predictors. ### 6. Probability prediction.

We create a new data set for the prediction.

```
newdata1 <- with(mydata,
  data.frame(gre = mean(gre), gpa = mean(gpa), rank = factor(1:4)))

## view data frame
newdata1
```

```
##     gre    gpa rank
## 1 587.7 3.3899    1
## 2 587.7 3.3899    2
## 3 587.7 3.3899    3
## 4 587.7 3.3899    4
```

```
#that the type of prediction is a predicted probability (type="response")
```

Now, we predict the pobability of admission base on the newdata.

```
newdata1$rankP <- predict(mylogit, newdata = newdata1, type = "response")
newdata1
```

```
##     gre    gpa rank     rankP
## 1 587.7 3.3899    1 0.5166016
## 2 587.7 3.3899    2 0.3522846
## 3 587.7 3.3899    3 0.2186120
## 4 587.7 3.3899    4 0.1846684
```

Note: We can see that if student come frome prestige school (other variable be the same) the probability that he or she get admission will be higher.

7. Accuracy, lift.

a) Accuracty

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
mydata$rankP <- predict(mylogit, newdata = mydata, type = "response")
mydata$Prediction <- ifelse(mydata$rankP > 0.5, 1, 0)
head(arrange(mydata, desc(rankP)))
```

```
##   admit gre  gpa rank     rankP Prediction
## 1     1 800 4.00    1 0.7384082          1
## 2     0 800 3.97    1 0.7337223          1
## 3     1 760 4.00    1 0.7205386          1
## 4     1 800 3.74    1 0.6960719          1
## 5     0 800 3.73    1 0.6943683          1
## 6     1 700 4.00    1 0.6923799          1
```

```
tail(arrange(mydata, desc(rankP)))
```

```
##     admit gre  gpa rank      rankP Prediction
## 395     0 380 2.91    4 0.08776685          0
## 396     0 360 2.56    3 0.07895335          0
## 397     0 300 2.92    4 0.07486000          0
## 398     0 440 2.48    4 0.07235381          0
## 399     0 220 2.83    3 0.07198547          0
## 400     0 420 2.26    4 0.05878643          0
```

```
AccuracyTable <- table(mydata$admit, mydata$Prediction)
AccuracyTable
```

```
##
##       0   1
##   0 254  19
##   1  97  30
```

```
# calculate the accuracy within group
diag(prop.table(AccuracyTable,1))
```

```
##         0         1
## 0.9304029 0.2362205
```

```
#calculate the overal accuracy
sum(diag(prop.table(AccuracyTable)))
```

```
## [1] 0.71
```

b) Lift

You can also embed plots, for example:

```
##
##   0   1
## 273 127
```

```
##      1
## 0.3175
```

```
##   admit gre  gpa rank     rankP Prediction
## 1     1 800 4.00    1 0.7384082          1
## 2     0 800 3.97    1 0.7337223          1
## 3     1 760 4.00    1 0.7205386          1
## 4     1 800 3.74    1 0.6960719          1
## 5     0 800 3.73    1 0.6943683          1
## 6     1 700 4.00    1 0.6923799          1
```

```
##
##      0  1
##   0 18 19
##   1 13 30
```

```
##      1
## 2.1974
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.