Binary Logistic DrPH (Epidemiology)

Kamarul Imran M

1 February 2016

Contents

Modeling Binomial Data

1. Describe data
2. Explore data - Exploratory Data Analysis
3. Estimate parameters
4. Make Inference
5. Make Prediction
6. Interpretation

Locate files

- Browse your folders.
- Look for the files.
- Check the path to the folder containing the files

Set the folder

Set our working directory. REMEMBER! your working directory (working folder) is different from my working directory

```
# this is my working directory. You have to specify yours
setwd("E:/Epi_Stat_Matters/LectureNotes2015/binary-logistic/binary-logistic-DrPH-2015/BinaryLogisticDrP
```

Read data

- Read our data in the working folder
- Then, save as a csv file in our working directory

```
mydata <- read.csv("http://www.ats.ucla.edu/stat/data/binary.csv")
write.csv2(mydata,'logistic.csv')
```

Describe data

```
#observe data
head(mydata,10)
```

Rank is taken as numerical variable which does not make sense. We need to convert it to a categorical (factor) variable

```
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  
```

```
mydata$rank<-factor(mydata$rank)
summary(mydata$rank)
```

```
##   1   2   3   4
##  61 151 121  67
```

More fancy, we can use *psych::describe* function

```
library(psych)
describe(mydata)
```

```
##        vars   n   mean     sd median trimmed    mad    min max  range  skew
## admit     1 400   0.32   0.47    0.0    0.27   0.00   0.00   1   1.00  0.78
## gre       2 400 587.70 115.52  580.0  589.06 118.61 220.00 800 580.00 -0.14
## gpa       3 400   3.39   0.38    3.4    3.40   0.40   2.26   4   1.74 -0.21
## rank*     4 400   2.48   0.94    2.0    2.48   1.48   1.00   4   3.00  0.10
##        kurtosis   se
## admit     -1.39 0.02
## gre       -0.36 5.78
## gpa       -0.60 0.02
## rank*     -0.91 0.05
```

Explore data

Use plots like * Histogram for numerical variables * and barplot for categorical variables, at least.

Estimate parameters

- we estimate the logit or the log odds.
- We used **summary** to see the results stored as **mylogit**
- We used **coefficients** to examine the regression coefficients

```
mylogit <- glm(admit~gre+gpa+rank,family = 'binomial'(link = logit),data=mydata)
summary(mylogit)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = binomial(link = logit),
##     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
```

```
coefficients(mylogit)
```

```
##  (Intercept)          gre          gpa        rank2        rank3
## -3.989979073  0.002264426  0.804037549 -0.675442928 -1.340203916
##        rank4
## -1.551463677
```

To obtain the odds ratios and their 95% CI, we need to exponentiate using **exp** the regression coefficients or the betas

```
exp(coefficients(mylogit))
```

```
## (Intercept)         gre         gpa       rank2       rank3       rank4
##   0.0185001   1.0022670   2.2345448   0.5089310   0.2617923   0.2119375
```

```
exp(confint(mylogit))
```

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

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

Make inference

Here, we examine the p-values (hypothesis testing) and the confidence intervals.

- First, using the method of maximum likelihood
- Next, using the SE method (function **confint.default**)

```
confint(mylogit)
```

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

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

```r
confint.default(mylogit)
```

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

Calculate the fitted values

The fitted values are the expected values of the model. These expected values are the predicted probability for each observation (each patient) in the dataset

```r
fitted(mylogit)
```

```
##          1          2          3          4          5          6 
## 0.17262654 0.29217496 0.73840825 0.17838461 0.11835391 0.36996994 
##          7          8          9         10         11         12 
## 0.41924616 0.21700328 0.20073518 0.51786820 0.37431440 0.40020025 
##         13         14         15         16         17         18 
## 0.72053858 0.35345462 0.69237989 0.18582508 0.33993917 0.07895335 
##         19         20         21         22         23         24 
## 0.54022772 0.57351182 0.16122101 0.43727108 0.12837525 0.19204860 
##         25         26         27         28         29         30 
## 0.43759396 0.68229503 0.57848091 0.20475422 0.42307349 0.45829857 
##         31         32         33         34         35         36 
## 0.21765393 0.28583616 0.22481919 0.42494837 0.34296523 0.21293277 
##         37         38         39         40         41         42 
## 0.48413281 0.13931720 0.26569575 0.11942769 0.18975965 0.33567002 
##         43         44         45         46         47         48 
## 0.31560404 0.17702923 0.32817441 0.18025548 0.36121718 0.11699101 
##         49         50         51         52         53         54 
## 0.07235381 0.15047417 0.31488795 0.11624726 0.23936553 0.37838478 
##         55         56         57         58         59         60 
## 0.24045684 0.39213236 0.18283980 0.10853139 0.30472142 0.12837525 
##         61         62         63         64         65         66 
## 0.33078459 0.16742893 0.28289780 0.33295972 0.30988311 0.39645173 
##         67         68         69         70         71         72 
## 0.27784995 0.51681586 0.57206626 0.69436828 0.33966212 0.07486000 
##         73         74         75         76         77         78 
## 0.15073716 0.46607599 0.24284830 0.38139149 0.20415281 0.42494837 
##         79         80         81         82         83         84 
## 0.43570986 0.65251556 0.16456653 0.31150713 0.20517359 0.08776685 
##         85         86         87         88         89         90 
## 0.21358749 0.25126279 0.34584314 0.37549461 0.55783057 0.51131037 
##         91         92         93         94         95         96 
## 0.49978497 0.63809471 0.57000341 0.26968427 0.40010880 0.37907977 
##         97         98         99        100        101        102 
## 0.22063013 0.33002244 0.31762762 0.14640896 0.11633954 0.24114689 
##        103        104        105        106        107        108 
## 0.11883427 0.28100436 0.50126183 0.35394219 0.61241920 0.25695415
##        109        110        111        112        113        114 
## 0.11218813 0.30904921 0.17869743 0.13603549 0.10881750 0.48942091 
##        115        116        117        118        119        120 
## 0.35153649 0.32780508 0.29004920 0.47768876 0.68922540 0.09863460 
##        121        122        123        124        125        126 
## 0.38205848 0.19283124 0.13456621 0.14161529 0.35890251 0.16784107 
##        127        128        129        130        131        132 
## 0.55353632 0.29761787 0.29364378 0.12270194 0.32900715 0.27429792 
##        133        134        135        136        137        138 
## 0.35016196 0.15167362 0.26397051 0.20956391 0.16855273 0.37076538 
##        139        140        141        142        143        144 
## 0.37104174 0.56147017 0.48592324 0.24487554 0.27496207 0.21702497 
##        145        146        147        148        149        150 
## 0.18326999 0.15292361 0.30053113 0.13202601 0.36278299 0.58590453 
##        151        152        153        154        155        156 
## 0.69607194 0.26076336 0.48793196 0.22533437 0.27701027 0.12691355 
##        157        158        159        160        161        162 
## 0.20243105 0.49385024 0.40979572 0.33767745 0.31214097 0.40081797 
##        163        164        165        166        167        168 
## 0.44572710 0.21536268 0.33209361 0.69237989 0.12564635 0.33881603 
##        169        170        171        172        173        174 
## 0.27253083 0.25713529 0.16766865 0.13610230 0.27045353 0.47601029 
##        175        176        177        178        179        180 
## 0.17207711 0.36543032 0.20079352 0.20929210 0.22290898 0.09702710 
##        181        182        183        184        185        186 
## 0.29173405 0.21592659 0.53390445 0.41213948 0.10284874 0.51016205 
##        187        188        189        190        191        192 
## 0.23875288 0.26184001 0.28313813 0.30160149 0.29894660 0.33797096 
##        193        194        195        196        197        198 
## 0.29780561 0.14252603 0.37361105 0.37499458 0.20306181 0.11520619 
##        199        200        201        202        203        204 
## 0.25867413 0.23203530 0.29790835 0.31450637 0.69237989 0.19176895 
##        205        206        207        208        209        210 
## 0.62160882 0.37552455 0.62994688 0.59336886 0.17269671 0.36867073 
##        211        212        213        214        215        216 
## 0.23500145 0.28417171 0.21145148 0.23806753 0.39069474 0.18303592 
##        217        218        219        220        221        222 
## 0.29144726 0.49458858 0.36532833 0.37499458 0.18691983 0.35841190 
##        223        224        225        226        227        228 
## 0.38346629 0.32549498 0.37234438 0.29200523 0.40539785 0.13119209 
##        229        230        231        232        233        234 
## 0.30562595 0.42917277 0.17040039 0.20845157 0.25212831 0.09688336 
##        235        236        237        238        239        240 
## 0.65921863 0.30806878 0.40979572 0.41039144 0.10815929 0.27465027 
##        241        242        243        244        245        246 
## 0.19001218 0.56239934 0.19616746 0.33794240 0.41996550 0.40736827 
##        247        248        249        250        251        252 
## 0.39171070 0.24596016 0.29657173 0.29278619 0.20011793 0.17414395 
##        253        254        255        256        257        258 
## 0.43247252 0.18780755 0.26200847 0.23371984 0.30267400 0.32075797 
##        259        260        261        262        263        264 
## 0.33944941 0.46187255 0.34863249 0.24298996 0.16969339 0.32075797 
##        265        266        267        268        269        270 
## 0.26562483 0.14378335 0.15865328 0.26021896 0.41492493 0.12579904 
##        271        272        273        274        275        276
## 0.48994106 0.19310678 0.45641226 0.54337733 0.27302605 0.28684953
##        277        278        279        280        281        282
## 0.22143462 0.55028996 0.16945136 0.34384116 0.49925174 0.13172559
##        283        284        285        286        287        288
## 0.21874547 0.13337693 0.28021662 0.17925207 0.60122274 0.25502619
##        289        290        291        292        293        294
## 0.23197657 0.05878643 0.38047126 0.35008696 0.46240272 0.73372225
##        295        296        297        298        299        300
## 0.29885443 0.17659931 0.45483793 0.23950580 0.34785059 0.27566478
##        301        302        303        304        305        306
## 0.36288468 0.28067279 0.22671860 0.51860565 0.07198547 0.19060160
##        307        308        309        310        311        312
## 0.44561844 0.37054412 0.28373804 0.12588934 0.30028221 0.44520022
##        313        314        315        316        317        318
## 0.30907647 0.19322270 0.17701800 0.15412239 0.18491373 0.29806393
##        319        320        321        322        323        324
## 0.18670880 0.46755914 0.14630641 0.32183935 0.12035456 0.17486941
##        325        326        327        328        329        330
## 0.12112920 0.66498227 0.38597852 0.35450549 0.33926538 0.11370930
##        331        332        333        334        335        336
## 0.39213236 0.27905234 0.34097123 0.21344965 0.20393972 0.59795326
##        337        338        339        340        341        342
## 0.16520993 0.16070084 0.45158492 0.26006097 0.14037382 0.12659514
##        343        344        345        346        347        348
## 0.22560760 0.29075910 0.18859648 0.14657301 0.35132030 0.42636137
##        349        350        351        352        353        354
## 0.25767548 0.27488628 0.57858815 0.23714608 0.18120291 0.43779599
##        355        356        357        358        359        360
## 0.40050290 0.49758253 0.38909423 0.57487559 0.25063922 0.37007654
##        361        362        363        364        365        366
## 0.59956970 0.50972425 0.35412991 0.29777892 0.49491656 0.11836196
##        367        368        369        370        371        372
## 0.12645014 0.26745319 0.63170496 0.56803162 0.39857395 0.31708679
##        373        374        375        376        377        378
## 0.37650752 0.53085361 0.41142403 0.18735742 0.41512421 0.58958954
##        379        380        381        382        383        384
## 0.20223990 0.21896113 0.46366743 0.34602886 0.34967678 0.67275941
##        385        386        387        388        389        390
## 0.18665107 0.35189341 0.52842881 0.34287938 0.33908140 0.40275050
##        391        392        393        394        395        396
## 0.40093595 0.48719398 0.22202911 0.43872524 0.25342327 0.48866999
##        397        398        399        400
## 0.16550430 0.18106222 0.46366743 0.30073055
```

Make prediction

Similarly, one of the important objectives in modelling is to perform prediction based on the model using new data.

We can perform these predictions:

1. Predict the log odds for having the outcome

2. Predict the conditional probability for having the outcome

Let us say we have these data

gre = 380 gpa = 3.61 rank = 3

First, we create a data frame

```
new_datal <- data.frame( gre = 380, gpa = 3.61, rank = '3')
```

Now, we can perform the prediction

```
pred.logit<-predict(mylogit,newdata = new_datal, type='link')
pred.logit
```

```
##          1
## -1.567126
```

We can confirm this by calculate this

```
-3.99+0.00226*380+0.8041*3.61-1.34
```

```
## [1] -1.568399
```

Notice, that similarity between **predict(x, type='response'** and **fitted** Remember, we can calculate the conditional probability of having the outcome

```
pred.prob<-predict(mylogit, newdata = new_datal, type='response')
pred.prob
```

```
##         1
## 0.1726265
```

We can verify this by two ways

```
head(fitted(mylogit))
```

```
##         1         2         3         4         5         6
## 0.1726265 0.2921750 0.7384082 0.1783846 0.1183539 0.3699699
```

```
# calculate the logistic probability
exp(-1.567)/(1+exp(-1.567))
```

```
## [1] 0.1726445
```

Compare models

We compare a model with **vs** and without **gre**. This is done using the deviance

```
summary(mylogit)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = binomial(link = logit),
##     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
```

```
mylogit2 <- glm(admit~gpa+rank,family = 'binomial'(link = logit),data=mydata)
summary(mylogit2)
```

```
##
## Call:
## glm(formula = admit ~ gpa + rank, family = binomial(link = logit),
##     data = mydata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5055  -0.8663  -0.6590   1.1505   2.0913
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.4636     1.1003  -3.148 0.001645 **
## gpa           1.0521     0.3102   3.392 0.000694 ***
## rank2        -0.6810     0.3141  -2.168 0.030181 *
## rank3        -1.3919     0.3419  -4.071 4.68e-05 ***
## rank4        -1.5943     0.4152  -3.840 0.000123 ***
## ---
## 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: 462.88  on 395  degrees of freedom
## AIC: 472.88
##
## Number of Fisher Scoring iterations: 4
```

```
anova(mylogit,mylogit2,test = 'Chisq')
```

```
## Analysis of Deviance Table
##
## Model 1: admit ~ gre + gpa + rank
## Model 2: admit ~ gpa + rank
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       394     458.52
## 2       395     462.88 -1  -4.3578  0.03684 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Linearity in logits

gre is tested for linearity in logit. **gre** is linear but it is rescaled to produce less decimals

The linearity of logits is tested using library **mfp**

```r
library(mfp)
```

```
## Loading required package: survival
```

```r
mylogit3 <- mfp(admit~fp(gre)+gpa+rank,family = 'binomial'(link = logit),data=mydata,verbose=T)
```

```
##
##  Variable    Deviance    Power(s)
## ------------------------------------------------
## Cycle 1
##  rank2
##               463.096
##               458.517     1
##
##
##
##  rank3
##               474.043
##               458.517     1
##
##
##
##  rank4
##               473.551
##               458.517     1
##
##
##
##  gpa
##               464.532
##               458.517     1
##
##
##
##  gre
##               462.875
##               458.517     1
##               458.415     0
##               458.366     -2 -2
##
##
## Tansformation
##       shift scale
## rank2     0     1
## rank3     0     1
## rank4     0     1
## gpa       0     1
## gre       0  1000
##
## Fractional polynomials
##       df.initial select alpha df.final power1 power2
## rank2          1      1  0.05        1      1      .
## rank3          1      1  0.05        1      1      .
## rank4          1      1  0.05        1      1      .
## gpa            1      1  0.05        1      1      .
## gre            4      1  0.05        1      1      .
##
##
## Transformations of covariates:
##              formula
## gre  I((gre/1000)^1)
## gpa              gpa
## rank            rank
##
##
## Deviance table:
##           Resid. Dev
## Null model    499.9765
## Linear model  458.5175
## Final model   458.5175
```

```
summary(mylogit3)
```

```
##
## Call:
## glm(formula = admit ~ rank + gpa + I((gre/1000)^1), family = binomial(link = logit),
##     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)      -3.9900     1.1400  -3.500 0.000465 ***
## rank2            -0.6754     0.3165  -2.134 0.032829 *
## rank3            -1.3402     0.3453  -3.881 0.000104 ***
## rank4            -1.5515     0.4178  -3.713 0.000205 ***
## gpa               0.8040     0.3318   2.423 0.015388 *
## I((gre/1000)^1)   2.2644     1.0940   2.070 0.038465 *
## ---
## 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
```

```
mylogit3$fptable
```

```
##       df.initial select alpha df.final power1 power2
## rank2          1      1  0.05        1      1      .
## rank3          1      1  0.05        1      1      .
## rank4          1      1  0.05        1      1      .
## gpa            1      1  0.05        1      1      .
## gre            4      1  0.05        1      1      .
```

Diagnostics for a model with a binomial response

To do these diagnostics, you need to load **library('LogisticDx')**.

First, we produce the diagnostic measures for a binary regression model by covariate pattern

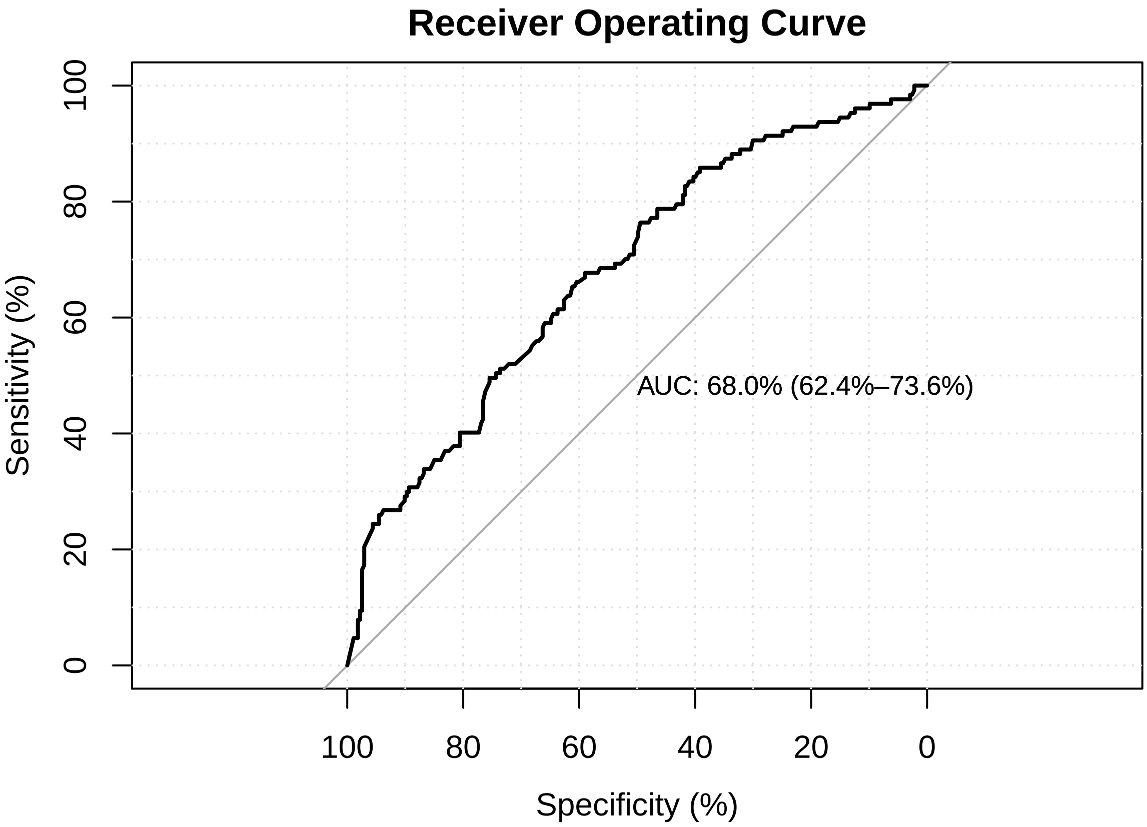
Next, we produce the Goodness-of-fit for binomial regression. Usually, the number of groups (quantiles) equal 10 to perform the Hosmer-Lemeshow test. At the same time, we plot the ROC curve

Similarly, we can check the **auc** value

```
library('LogisticDx')
dx(mylogit2,byCov=T)
```

```
##      (Intercept)  gpa rank2 rank3 rank4 y         P n      yhat
##   1:           1 3.94     1     0     0 1 0.5001242 2 1.0002483
##   2:           1 3.30     1     0     0 1 0.3378663 3 1.0135990
##   3:           1 3.99     0     1     0 1 0.3412259 3 1.0236777
##   4:           1 3.35     1     0     0 1 0.3497327 3 1.0491980
##   5:           1 3.17     1     0     0 1 0.3079792 3 0.9239375
##  ---
## 256:           1 3.52     0     0     1 2 0.2051122 3 0.6153366
## 257:           1 2.68     0     1     0 1 0.1154721 1 0.1154721
## 258:           1 3.00     0     0     1 1 0.1299149 1 0.1299149
## 259:           1 2.42     0     0     0 1 0.2854368 1 0.2854368
## 260:           1 2.65     0     1     0 1 0.1122874 1 0.1122874
##                 Pr            dr           h           sPr           sdr
##   1: -0.0003511985 -0.0003511985 0.014377039 -0.0003537507 -0.0003537506
##   2: -0.0165997514 -0.0166181074 0.006902268 -0.0166573376 -0.0166757573
##   3: -0.0288329487 -0.0288874508 0.015618469 -0.0290607843 -0.0291157171
##   4: -0.0595625368 -0.0597856465 0.006844794 -0.0597674360 -0.0599913131
##   5:  0.0951239167  0.0944359041 0.007474218  0.0954814103  0.0947908121
##  ---
## 256:  1.9798618855  1.7253451406 0.017098521  1.9970084425  1.7402874601
## 257:  2.7676886108  2.0778480687 0.011720502  2.7840519420  2.0901328741
## 258:  2.5879228606  2.0203343433 0.013448875  2.6055027084  2.0340585431
## 259:  1.5822145401  1.5834991249 0.034641074  1.6103525417  1.6116599714
## 260:  2.8117113691  2.0912646219 0.011927131  2.8286306943  2.1038486967
##             dChisq         dDev        dBhat
##   1: 1.251395e-07 1.251395e-07 1.825379e-09
##   2: 2.774669e-04 2.780809e-04 1.928462e-06
##   3: 8.445292e-04 8.477250e-04 1.339953e-05
##   4: 3.572146e-03 3.598958e-03 2.461912e-05
##   5: 9.116700e-03 8.985298e-03 6.865333e-05
##  ---
## 256: 3.988043e+00 3.028600e+00 6.937586e-02
## 257: 7.750945e+00 4.368655e+00 9.192235e-02
## 258: 6.788644e+00 4.137394e+00 9.254425e-02
## 259: 2.593235e+00 2.597448e+00 9.305602e-02
## 260: 8.001152e+00 4.426179e+00 9.658274e-02
```

```
fit.mylogit2<-gof(mylogit2,g=10,plotROC = T)
```



```
fit.mylogit2
```

```
##       chiSq  df     pVal
## PrI  400.78 395 0.409645
## drI  462.88 395 0.010379 *
## PrG  255.36 255 0.481870
## drG  294.60 255 0.044609 *
## PrCT 255.36 255 0.481870
## drCT 294.60 255 0.044609 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##                       val df     pVal
## HL chiSq         8.190458  8 0.415090
## mHL F            1.484491  9 0.153918
## OsRo Z           0.027903 NA 0.977740
## SstPgeq0.5 Z     0.217564 NA 0.827769
## SstPl0.5 Z       0.422859 NA 0.672398
## SstBoth chiSq    0.226144  2 0.893086
## SllPgeq0.5 chiSq 0.047739  1 0.827045
## SllPl0.5 chiSq   0.176745  1 0.674186
## SllBoth chiSq    0.177398  2 0.915121
```

```r
#area under curve
fit.mylogit2$auc
```

```
##          auc lower 95% CI upper 95% CI
##     68.01073     62.39737     73.62409
## attr(,"interpret")
## [1] "auc = 0.5       --> useless"   "0.7 < auc < 0.8 --> good"
## [3] "0.8 < auc < 0.9 --> excellent"
```

```r
#chi square test for gof
fit.mylogit2$chiSq
```

```
##    test    chiSq  df       pVal
## 1:  PrI 400.7833 395 0.40964543
## 2:  drI 462.8753 395 0.01037867
## 3:  PrG 255.3601 255 0.48187033
## 4:  drG 294.6050 255 0.04460896
## 5: PrCT 255.3601 255 0.48187033
## 6: drCT 294.6050 255 0.04460896
```

```r
#contigency table for HL test
fit.mylogit2$ctHL
```

```
##         P y1     y1hat y0    y0hat  n      Pbar
##  1: 0.149  6  5.037833 34 34.96217 40 0.1259458
##  2: 0.192  5  6.824095 35 33.17591 40 0.1706024
##  3: 0.219  7  8.260683 33 31.73932 40 0.2065171
##  4: 0.258 11  9.327547 28 29.67245 39 0.2391679
##  5: 0.293 12 11.338710 29 29.66129 41 0.2765539
##  6:  0.34 15 12.556219 25 27.44378 40 0.3139055
##  7: 0.376 14 13.796927 25 25.20307 39 0.3537674
##  8: 0.437 14 16.561867 27 24.43813 41 0.4039480
##  9: 0.511 13 18.253216 25 19.74678 38 0.4803478
## 10: 0.678 30 25.042903 12 16.95710 42 0.5962596
```

```r
#GOF test
fit.mylogit2$gof
```

```
##          test  stat        val df      pVal
## 1:         HL chiSq 8.19045835  8 0.4150903
## 2:        mHL     F 1.48449073  9 0.1539182
## 3:       OsRo     Z 0.02790263 NA 0.9777398
## 4: SstPgeq0.5     Z 0.21756392 NA 0.8277689
## 5:   SstPl0.5     Z 0.42285933 NA 0.6723979
## 6:    SstBoth chiSq 0.22614408  2 0.8930863
## 7: SllPgeq0.5 chiSq 0.04773921  1 0.8270450
## 8:   SllPl0.5 chiSq 0.17674490  1 0.6741857
## 9:    SllBoth chiSq 0.17739848  2 0.9151208
```

We can also perform model fitness by using *ROCR* package

```r
library(ROCR)
```

```
## Loading required package: gplots
```

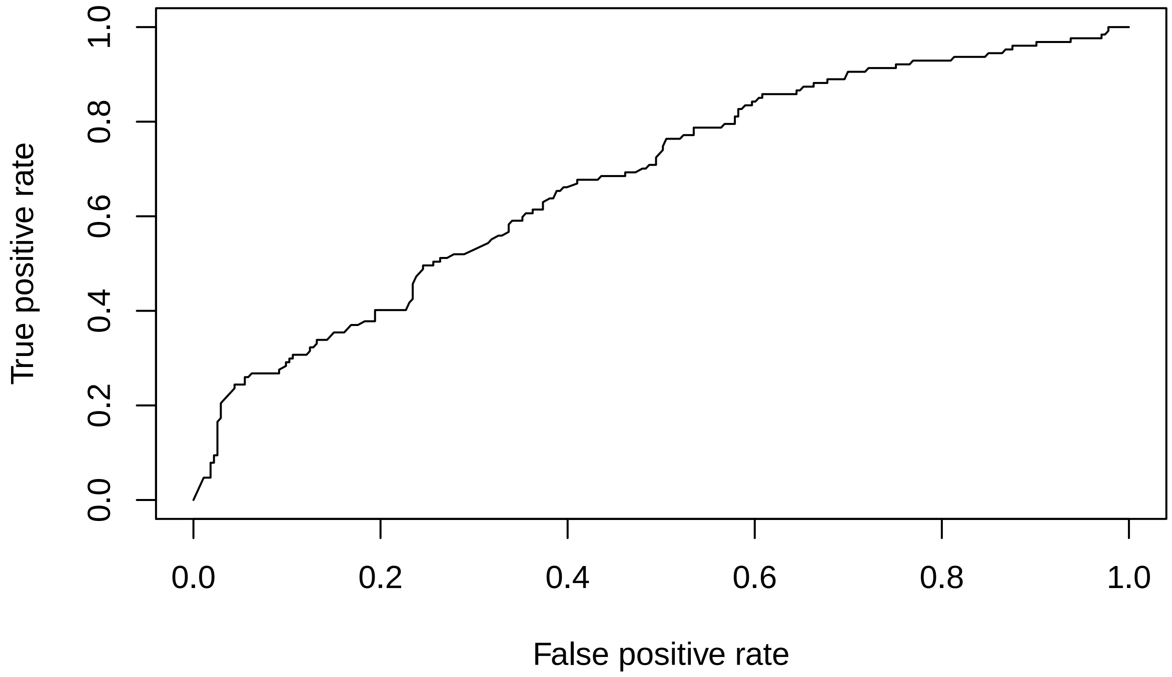
```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
pred.prob2<-predict(mylogit2, type='response')
head(pred.prob2)
```

```
##         1         2         3         4         5         6
## 0.2577653 0.2700256 0.6779993 0.1542276 0.1218149 0.2712217
```

```
pred.prob22<-prediction(pred.prob2, mydata$admit )
pred.prob22f<-performance(pred.prob22, measure='tpr', x.measure='fpr')
plot(pred.prob22f)
```



Using *ROCR* package, we can also calculate the *AUC*

```
auc2<-performance(pred.prob22, measure='auc')
auc2@y.values[[1]]
```

```
## [1] 0.6801073
```

Another package is *MKmisc* package, to perform the Hosmer-Lemeshow test of GOF

```
library(MKmisc)
```

```
##
```

```
## Attaching package: 'MKmisc'

## The following object is masked from 'package:psych':
##
##     corPlot
```

```
HLgof.test(fit = fitted(mylogit2), obs = mydata$admit)
```

```
## $C
##
##  Hosmer-Lemeshow C statistic
##
## data:  fitted(mylogit2) and mydata$admit
## X-squared = 9.8564, df = 8, p-value = 0.2752
##
##
## $H
##
##  Hosmer-Lemeshow H statistic
##
## data:  fitted(mylogit2) and mydata$admit
## X-squared = 7.6802, df = 8, p-value = 0.4653
```

Diagnostic plot

Will return many diagnostic plot

```
plot(mylogit2)
```

Using K-fold validation. Will not discuss here.

Interaction

Let use see how we deal an interaction. First, read data from this text file.

Columns (variables) no 2, and from 5 to 10 need to be converted to categorical (factor) variables

```
data.l<-read.table("LOWBWT.txt",header=T)
data.l[,c(2,5:10)]<-lapply(data.l[,c(2,5)],factor)
```

To simulate a binary predictor variable, we now recode LWT to LWD (LWT<110 vs >=110)

```
data.l$LWD<-findInterval(data.l$LWT,110)
data.l$LWD<-factor(data.l$LWD,labels = c("less 110",">=110"))
head(data.l$LWD,10)
```

```
##  [1] >=110    >=110    >=110    less 110 less 110 >=110    less 110
##  [8] >=110    >=110    >=110
## Levels: less 110 >=110
```

```
head(data.l$LWT,10)
```

```
##  [1] 120 130 187 105  85 150  97 128 132 165
```

```
str(data.l$LWD)
```

```
##  Factor w/ 2 levels "less 110",">=110": 2 2 2 1 1 2 1 2 2 2 ...
```

Model the relationship; outcome (LOW=0,1) with predictors of LWD and AGE interact with each other. You may try with using #

```
mod.lwd.age<-glm(LOW~LWD*AGE,family = binomial(link =logit ),data=data.l)
summary(mod.lwd.age)
```

```
##
## Call:
## glm(formula = LOW ~ LWD * AGE, family = binomial(link = logit),
##     data = data.l)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4257  -0.8554  -0.6960   1.1602   2.0329
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -1.16959    1.46515  -0.798   0.4247
## LWD>=110        1.94409    1.72481   1.127   0.2597
## AGE             0.05262    0.06449   0.816   0.4145
## LWD>=110:AGE -0.13220    0.07570  -1.746   0.0807 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 234.67  on 188  degrees of freedom
## Residual deviance: 221.14  on 185  degrees of freedom
## AIC: 229.14
##
## Number of Fisher Scoring iterations: 4
```

Predict our model using new data. Before doing so, we need to create a dataset containing new data

```
newdata.2<-data.frame(AGE=c(15,15,20,20),LWD=rep(c("less 110",">=110"),2))
```

Now let us predict the log odds

```
predict(mod.lwd.age,newdata=newdata.2)
```

```
##          1          2          3          4
## -0.3802252 -0.4190874 -0.1171023 -0.8169483
```

```
newdata.2
```

```
##   AGE      LWD
## 1  15 less 110
## 2  15    >=110
## 3  20 less 110
## 4  20    >=110
```

Can you prove these?

```
-1.1696+0+0.0526*15+0
```

```
## [1] -0.3806
```

```
-1.1696+1.944*1+0.0526*15-0.1322*1*15
```

```
## [1] -0.4196
```

```
-1.1696+0+0.0526*20+0
```

```
## [1] -0.1176
```

```
-1.1696+1.944*1+0.0526*20-0.1322*1*20
```

```
## [1] -0.8176
```

Resources

1. http://www.ats.ucla.edu/stat/r/dae/logit.htm
2. https://cran.r-project.org/web/packages/HSAUR/vignettes/Ch_logistic_regression_glm.pdf