How to: know if your model is good

Pseudo-R2

In linear models you can estimate the share of the variance explained by the model.

But as we already know, no variance can be observed in logistic models.

Nevertheless, logistic models have ***likelihood***.

So, you can compare *the likelihood of your model* to *the likelihood of the empty model* and that is pseudo-R2 (here's the formula for McFadden's R2):

$$R^2_{MF} = \frac{Log(L_m)}{Log(L_0)}$$

Values > 0.1 indicate acceptable fit; values 0.2<r<0.4 indicate excellent fit.

PCP and ePCP

You can also look at how good is your model at predicting real values

PCP (percent of correctly predicted) and ePCP (expected percent of correctly predicted) show if your model works well.

1. Set the threshold for the probability values (usually it's 0.5 but you can change it). Values above the threshold predict 1, values below the threshold predict 0.
2. Compare predicted values to the real ones. Calculate the percent of correctly predicted values. If it's below 0.5 – your model works worse than random prediction. PCP and ePCP > 0.7 indicate good fit.

How to: do it in R

Generalized linear models

For binomial regressions we are going to use *glm* function

Let’s have a look at this function using *Greene* data from package *car.*

This data frame contains the following columns:

- ***rater*** - judgment of independent rater; a factor with levels: no, case has no merit; yes, case has some merit (leave to appeal should be granted).
- ***decision*** - judge's decision; a factor with levels: no, leave to appeal not granted; yes, leave to appeal granted.
- ***language*** - language of case; a factor with levels: English, French.
- ***location*** - location of original refugee claim; afactor with levels: Montreal, other, Toronto.
- ***success*** - logit of success rate, for all cases from the applicant's nation.

Let’s predict the decision!

```
m1 = glm(decision~success + rater + language + location,
                                          data = Greene,
                              family = binomial(link = logit))
```

```
M1 = glm(decision ~ success + rater + language + location,
                                           data = Greene,
                              family = binomial(link = logit))
```

Specify the type of your model (don't forget, or you will get an OLS model instead)

Specify the link function (let's have logit)

```
Call:
glm(formula = decision ~ success + rater + language + location,
    family = binomial(link = logit), data = Greene)
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6754  -0.8228  -0.5337   1.0403   2.6341
Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)      -0.2635     0.5918  -0.445    0.656
success           1.3501     0.2695   5.010 5.45e-07 ***
rateryes          1.1889     0.2468   4.816 1.46e-06 ***
languageFrench   -0.3513     0.5670  -0.619    0.536
locationother     0.6408     0.6316   1.015    0.310
locationToronto   0.4237     0.5685   0.745    0.456
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for binomial family taken to be 1)
```

Let’s check for multicollinearity

```
> vif(m1)
             GVIF Df GVIF^(1/(2*Df))
success  1.108675  1        1.052936
rater    1.013855  1        1.006904
language 4.262896  1        2.064678
location 4.464368  2        1.453584
```

Choose between language and location

Let’s keep language, then:

```
Call:
glm(formula = decision ~ success + rater + language, family = binomial(link = logit),
    data = Greene)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6589  -0.8362  -0.5264   1.0393   2.5949

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)      0.1438     0.3045   0.472  0.63684
success          1.3005     0.2634   4.936 7.96e-07 ***
rateryes         1.1705     0.2456   4.765 1.89e-06 ***
languageFrench  -0.7566     0.2822  -2.681  0.00733 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> exp(coef(m2))

  (Intercept)        success      rateryes languageFrench
    1.1546335      3.6709978     3.2235083      0.4692407

>library(margins)
> margins(m2)

Average marginal effects
glm(formula = decision ~ success + rater + language, family =
binomial(link = logit),  data = Greene)

 success rateryes languageFrench
  0.2303   0.2275        -0.1268
```

Let’s make our output beautiful:

```
> library(stargazer)
> stargazer(m2, type = 'text', apply.coef = exp, report = "vc*", se = NULL)

=============================================
                      Dependent variable:    
                  ---------------------------
                           decision          
---------------------------------------------
success                    3.671***          
rateryes                   3.224***          
languageFrench              0.469*           
Constant                   1.155***          
---------------------------------------------
Observations                  384            
Log Likelihood             -203.272          
Akaike Inf. Crit.           414.544          
=============================================
Note:             *p<0.1; **p<0.05; ***p<0.01
```

```
> library(BaylorEdPsych)
> PseudoR2(m2)

       McFadden     Adj.McFadden        Cox.Snell       Nagelkerke
      0.1296293        0.1082202        0.1458763        0.2072987
McKelvey.Zavoina           Effron            Count        Adj.Count
      0.2380312        0.1597444        0.7473958        0.1491228
            AIC    Corrected.AIC
    414.5435294      414.6490703
```

Let’s look at predictions

Find predicted values

```
fitted = predict(m2, type = 'response')
Greene$y[fitted >=0.5] = 2
Greene$y[fitted < 0.5] = 1
```

Specify the threshold

```
pcp = length(Greene$y[Greene$y==as.numeric(Greene$decision)])/384
> pcp
[1] 0.7473958 #Looks fine

> library(OOmisc)
> ePCP(Greene$y, as.numeric(Greene$decision), alpha = 0.05)

      ePCP     lower     upper
 0.7604167 0.7177256 0.8031077 #looks fine
```

Let's try probit!

```
m3 = glm(decision ~ success + rater + language,
                                  data = Greene,
            family = binomial(link = probit))
```

New link function

```
Call:
glm(formula = decision ~ success + rater + language, family =
binomial(link = probit), data = Greene)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6241  -0.8521  -0.5361   1.0550   2.6791

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)     0.04424    0.17819   0.248    0.804
success         0.73758    0.14855   4.965 6.86e-07 ***
rateryes        0.69626    0.14617   4.763 1.90e-06 ***
languageFrench -0.41900    0.16040  -2.612    0.009 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> margins(m3)
Average marginal effects
glm(formula = decision ~ success + rater + language, family = binomial(link = probit), data = Greene)

 success rateryes languageFrench
  0.2221   0.2276        -0.1212

Comparing to the logit model:
> margins(m2)
Average marginal effects
glm(formula = decision ~ success + rater + language, family = binomial(link = logit),  data = Greene)

 success rateryes languageFrench
  0.2303   0.2275        -0.1268

#the average effects are similar
```

Let's have another practice

Get data on university admission:

```
mydata <- read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")
```

It should go as follows:

```
> 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
```

1. Predict *admission* by the results of *GPA* and *GRE* tests and the *rank* of the university.
2. Interpret the coefficients. Which are significant? Which are positive/negative?
3. Test how the model fits the data. Find pseudo R2 and PCP (ePCP).