Logistic regression

October 20, 2017

The linear regression model discussed in previous chapters assumes that the response variable Y is *quantitative*. But in many situations, the response variable is instead *qualitative*. For example, eye color is qualitative, taking on values blue, brown, or green. Often qualitative variables are referred to as categorical ; we will use these terms interchangeably. In this chapter, we study approaches for predicting qualitative responses, a process that is known as classification. Predicting a qualitative response for an observation can be referred to as classifying that observation, since it involves assigning the observation to a category, or class. On the other hand, often the methods used for classification first predict the probability of each of the categories of a qualitative variable, as the basis for making the classification.
Classification problems occur often, perhaps even more so than regression problems. Some examples include:

1. A person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of three medical conditions. Which of the three conditions does the individual have?

2. An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the users IP address, past transaction history, and so forth.

3. On the basis of DNA sequence data for a number of patients with and without a given disease, a biologist would like to figure out which DNA mutations are deleterious (disease-causing) and which are not.

Just as in the regression setting, in the classification setting we have a set of training observations $(x_1, y_1), ..., (x_n, y_n)$ that we can use to build a classifier. We want our classifier to perform well not only on the ***training data***, but also on ***test observations*** that were not used to train the classifier.
In this chapter, we learn how to build a model to predict $(Y\ )$ for any given value of (X_1) and (X_2), and so forth.

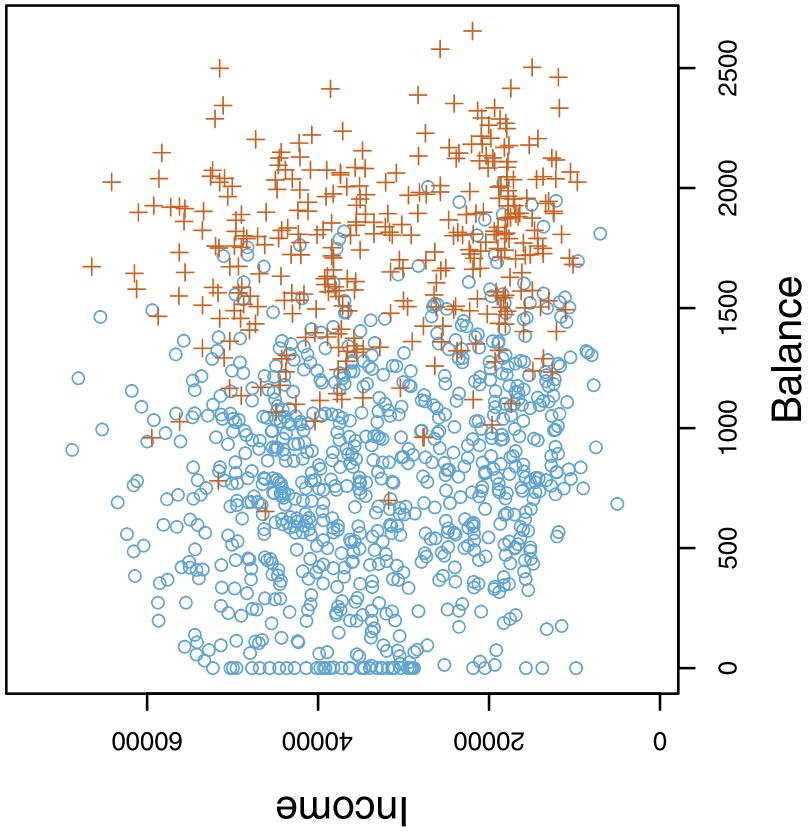

Figure 1: classification

Consider the *Default data* set, where the response *default* falls into one of two categories, Yes or No. logistic regression models the probability that Y belongs to a particular category.
For the Default data, logistic regression models the probability of default. For example, the probability of default given balance can be written as

$$Pr(default = Yes|balance).$$

The Logistic Model

How should we model the relationship between $p(X) = Pr(Y = 1|X)$ and X?
We are using a linear regression model to represent these probabilities: $p(X) = \beta_0 + \beta_1 X$. Any time a straight line is fit to a binary response that is coded as 0 or1, in principle we can always predict $p(X) < 0$ for some values of X and $p(X) > 1$ for others (unless the range of$X$ is limited).

To avoid this problem in logistic regression, we use the logistic function

$$p(X) = \frac{e^{(\beta_0+\beta_1 X)}}{1 + e^{(\beta_0+\beta_1 X)}}$$

After a bit of manipulation, we find that:
The quantity $p(X)/[1 - p(X)]$ is called the ***odds***
and can take on any value between. Values of the odds close to 0 and ∞ indicate very low and very high probabilities of default, respectively.
By taking the logarithm of both sides of the above equation, we arrive at
The left-hand side is called the ***log-odds*** or ***logit***.

Multiple Logistic Regression

$$\log(\frac{p(X)}{1 - p(X)}) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$$

where$X = (X_1, ..., X_p)$ are p predictors. Logistic regression can be performed in R with the glm (generalized linear model) function. This function uses a link function to determine which kind of model to use, such as logistic, probit, or poisson. These are indicated in the family and link options. See ?glm and ?family for more information. Multiple logistic regression can be determined by a stepwise procedure using the step function. This function selects models to minimize AIC, not according to p-values. Multiple correlation is one tool for investigating the relationship among potential independent variables. For example, if two independent variables are correlated to one another, likely both wont be needed in a final model, but there may be reasons why you would choose one variable over the other.

Assumptions

Generalized linear models have fewer assumptions than most common parametric tests. Observations still need to be independent, and the correct link function needs to be specified. So, for example you should understand when to use a poisson regression, and when to use a logistic regression. However, the normal distribution of data or residuals is not required.

Specifying the counts of successes and failures

Logistic regression has a dependent variable with two levels. In R, this can be specified in three ways.

1. The dependent variable can be a factor variable where the first level is interpreted as failure and the other levels are interpreted as success.

2. The dependent variable can be a vector of proportions of successes, with the caveat that the number of observations for each proportion is indicated in the weights option.

3. The dependent variable can be a matrix with two columns, with the first column being the number of successes and the second being the number of failures.

Not all proportions or counts are appropriate for logistic regression analysis

Note that in each of these specifications, both the number of successes and the number of failures is known. You should not perform logistic regression on proportion data where you dont know (or dont tell R) how many individuals went into those proportions. In statistics, 75% is different if it means 3 out of 4 rather than 150 out of 200. As another example where logistic regression doesnt apply, the weight people lose in a diet study expressed as a proportion of initial weight cannot be interpreted as a count of successes and failures. Here, you might be able to use common parametric methods, provided the model assumptions are met; log or arc-sine transformations may be appropriate. Likewise, if you count the number of people in front of you in line, you cant interpret this as a percentage of people since you dont know how many people are not in front of you in line. In this case with count data as the dependent variable, you might use poisson regression.

Overdispersion

One potential problem to be aware of when using generalized linear models is overdispersion. This occurs when the residual deviance of the model is high relative to the residual degrees of freedom. It is basically an indication that the model doesnt fit the data well.
That overdispersion is technically not a problem for a simple logistic regression, that is one with a binomial dependent and a single continuous independent variable.

Pseudo-R-squared

R does not produce r-squared values for generalized linear models (glm). These pseudo-R-squared values compare the maximum likelihood of the model to a nested null model fit with the same method. They should not be thought of as the same as the r-squared from an ordinary-least-squares linear (OLS) model, but instead as a relative measure among similar models. I have seen it mentioned that a McFadden pseudo-R-squared of 0.20.4 indicates a good fit. Whereas, I find that the Nagelkerke usually gives a reasonable indication of the goodness of fit for a model on a scale of 0 to 1. That being said, I have found the Cox and Snell and Nagelkerke to sometimes yield values I wouldnt expect for some glm. The function pR2 in the package pscl will also produce these pseudo-R-squared values.

Testing for p-values

Note that testing p-values for a logistic or poisson regression uses Chi-square tests. This is achieved through the test=Wald option in Anova to test the significance of each coefficient, and the test=Chisq option in anova for the significance of the overall model.

Example:The Stock Market Data

This data is part of the *ISLR library*
This data set consists of percentage returns for the S&P 500 stock index over 1, 250 days, from the beginning of 2001 until the end of 2005. For each date, we have recorded the percentage returns for each of the five previous trading days, *Lag1* through *Lag5*. We have also recorded *Volume* (the number of shares traded on the previous day, in billions), *Today* (the percentage return on the date in question) and *Direction* (whether the market was Up or Down on this date).

Examining correlations among variables

Note that we may use Spearman correlations here:

```
library(PerformanceAnalytics)

chart.Correlation(Smarket[,-c(1,9)],
method="spearman",
histogram=TRUE,
pch=16)
```

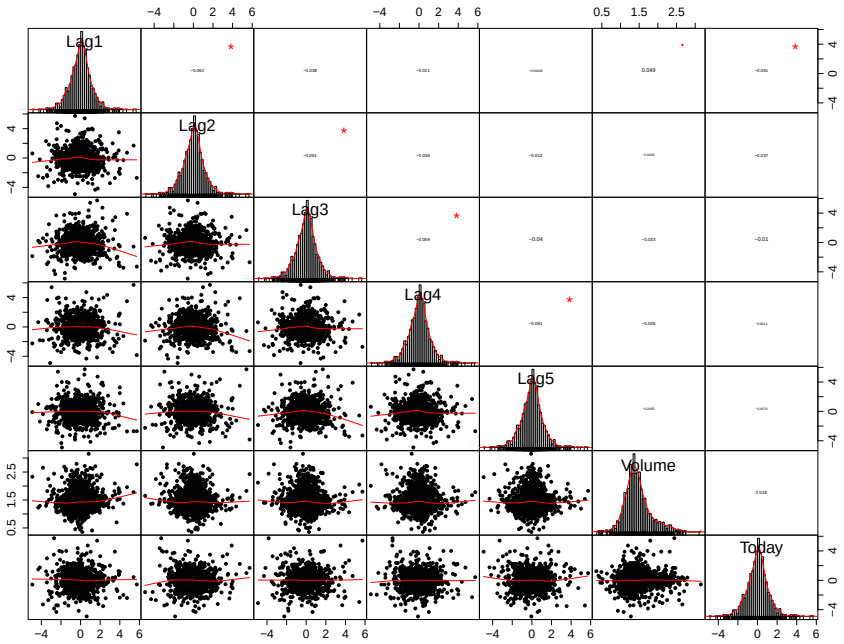

Figure 2: Correlation Matrix

Determining model with step procedure

Taking care of the missing values, by creating a new data set:

```
### Create new data frame with all missing values removed (NAs)
Data.omit = na.omit(Smarket)
```

Define full and null models and do step procedure

Fitting full model and null model and the we apply the step procedure in R:

```
model.null = glm(Direction ~ 1,
data=Data.omit,
family = binomial(link="logit")
)
```

```
 model.full = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,
data=Data.omit,
family = binomial(link="logit")
)

step(model.null,
scope = list(upper=model.full),
direction="both",
test="Chisq",
data=Data)
```

Final model

```
model.final = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,
data=Data.omit,
family = binomial(link="logit"),
na.action(na.omit)
)

summary(model.final)
Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
Volume, family = binomial(link = "logit"), data = Data.omit,
weights = na.action(na.omit))

Deviance Residuals:
Min      1Q  Median      3Q     Max
-1.446  -1.203   1.065   1.145   1.326

Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.126000   0.240736  -0.523    0.601
Lag1        -0.073074   0.050167  -1.457    0.145
Lag2        -0.042301   0.050086  -0.845    0.398
Lag3         0.011085   0.049939   0.222    0.824
Lag4         0.009359   0.049974   0.187    0.851
Lag5         0.010313   0.049511   0.208    0.835
Volume       0.135441   0.158360   0.855    0.392

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1731.2  on 1249  degrees of freedom
Residual deviance: 1727.6  on 1243  degrees of freedom
AIC: 1741.6

Number of Fisher Scoring iterations: 3
```

Analysis of variance for individual terms

```
Analysis of Deviance Table (Type II tests)

Response: Direction
Df  Chisq Pr(>Chisq)
Lag1         1 2.1217     0.1452
Lag2         1 0.7133     0.3983
Lag3         1 0.0493     0.8243
Lag4         1 0.0351     0.8514
Lag5         1 0.0434     0.8350
Volume       1 0.7315     0.3924
Residuals 1243
```

Pseudo-R-squared

```
library(pscl)
pR2(glm.fit)
```

Overall p-value for model

```
###################################Overall p-value for model

anova(glm.fit,
update(glm.fit, ~1),    # update here produces null model for comparison
test="Chisq")
library(lmtest)
lrtest(glm.fit)
```

Check for overdispersion

Overdispersion is a situation where the residual deviance of the glm is large relative to the residual degrees of freedom. These values are shown in the summary of the model. One guideline is that if the ratio of the residual deviance to the residual degrees of freedom exceeds 1.5, then the model is overdispersed. Overdispersion indicates that the model doesnt fit the data well: the explanatory variables may not well describe the dependent variable or the model may not be specified correctly for these data. If there is overdispersion, one potential solution is to use the quasibinomial family option in glm.

```
summary(glm.fit)
Dispersion parameter for binomial family taken to be 1)

Null deviance: 1731.2  on 1249  degrees of freedom
Residual deviance: 1727.6  on 1243  degrees of freedom

summary(glm.fit)$deviance / summary(glm.fit)$df.residual
1.38985
```

```
> library (ISLR)
> names(Smarket )
[1] "Year" "Lag1" "Lag2" "Lag3" "Lag4"
[6] "Lag5" "Volume " "Today" " Direction "
> dim(Smarket )
[1] 1250 9
> summary (Smarket )

glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume ,
data=Smarket ,family =binomial )
> summary (glm.fit )
Call:
glm (formula = Direction  Lag1 + Lag2 + Lag3 + Lag4 + Lag5
+ Volume , family = binomial , data = Smarket )
Deviance Residuals :
Min 1Q Median 3Q Max
-1.45 -1.20 1.07 1.15 1.33
Coefficients:
Estimate Std. Error z value Pr(>|z|)
(Intercept ) -0.12600 0.24074 -0.52 0.60
Lag1 -0.07307 0.05017 -1.46 0.15
Lag2 -0.04230 0.05009 -0.84 0.40
Lag3 0.01109 0.04994 0.22 0.82
Lag4 0.00936 0.04997 0.19 0.85
Lag5 0.01031 0.04951 0.21 0.83
Volume 0.13544 0.15836 0.86 0.39

(Dispersion parameter for binomial family taken to be 1)
Null deviance : 1731.2 on 1249 degrees of freedom
Residual deviance : 1727.6 on 1243 degrees of freedom
AIC: 1742
Number of Fisher Scoring iterations : 3
```

The smallest p-value here is associated with *Lag1*. The negative coefficient for this predictor suggests that if the market had a positive return yesterday, then it is less likely to go up today. We use the *coef()* function in order to access just the coefficients for this fitted model. We can also use the *summary()* function to access particular aspects of the fitted model, such as the p-values for the coefficients.

```
> coef(glm.fit)
(Intercept ) Lag1 Lag2 Lag3 Lag4
-0.12600 -0.07307 -0.04230 0.01109 0.00936
Lag5 Volume
0.01031 0.13544
> summary (glm.fit )$coef
```

```
summary (glm.fit )$coef [,4]
```

The *predict()* function can be used to predict the probability that the market will go up, given values of the predictors. The **type="response"** option tells R to output probabilities of the form $P(Y = 1|X)$, as opposed to other information such as the logit. If no data set is supplied to the predict() function, then the probabilities are computed for the training data that was used to fit the logistic regression model.

```
glm.probs =predict (glm .fit ,type =" response ")
> glm.probs [1:10]
1 2 3 4 5 6 7 8 9 10
0.507 0.481 0.481 0.515 0.511 0.507 0.493 0.509 0.518 0.489
> contrasts (Direction )
     Up
Down 0
Up   1
```

the *contrasts()* function indicates that R has created a dummy variable with a 1 for Up.

```
glm.pred=rep ("Down " ,1250)
glm.pred[glm.probs >.5]=" Up"
> table(glm .pred ,Direction )
Direction
glm .pred   Down Up
       Down 145  141
        Up   457 507
> (507+145) /1250
[1] 0.5216
> mean(glm.pred== Direction )
[1] 0.5216
```

R Code

```
library(ISLR)
?Smarket             ##########################Data in IsWR package
View(Smarket)
names(Smarket )
dim(Smarket )
summary (Smarket )
attach(Smarket)    #####In order to be able to refer directly to the variables
par(mex=0.5)
pairs(Smarket, gap=0, cex.labels=0.9)
cor(Smarket ) ######################## wrong because the Direction variable is qualitative
cor(Smarket [,-c(1,9)])
##############################################
### Note I used Spearman correlations here
```

```
library(PerformanceAnalytics)

chart.Correlation(Smarket[,-c(1,9)],
method="spearman",
histogram=TRUE,
pch=16)

#####################################Determining model with Step procedure
### Create new data frame with all missing values removed (NAs)
Data.omit = na.omit(Smarket)
model.null = glm(Direction ~ 1,
data=Data.omit,
family = binomial(link="logit")
)
model.full = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,
data=Data.omit,
family = binomial(link="logit")
)

step(model.null,
scope = list(upper=model.full),
direction="both",
test="Chisq",
data=Data)
##############################Final model
model.final = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,
data=Data.omit,
family = binomial(link="logit"),
na.action(na.omit)
)

summary(model.final)

#########################################fite the model using glm function
glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume ,
data=Smarket ,family =binomial(link = "logit") )



summary (glm.fit )
confint(glm.fit)
coef(glm.fit)
exp(model$coefficients)         # exponentiated coefficients
summary (glm.fit )$coef
summary (glm.fit )$coef [,4]

library(car) #Analysis of variance for individual terms
Anova(glm.fit, type="II", test="Wald")
```

```
#####################################################Pseudo-R-squared

#library(rcompanion)
#nagelkerke(glm.fit)
#require(rms)
#mod1b <- lrm(formoula=Direction ~ Lag1+Lag2+Lag3+Lag4+Lag5+Volume)
library(pscl)
pR2(glm.fit)

##################################Overall p-value for model

anova(glm.fit,
update(glm.fit, ~1),    # update here produces null model for comparison
test="Chisq")
library(lmtest)
lrtest(glm.fit)
#########################Check for overdispersion
summary(glm.fit)
summary(glm.fit)$deviance / summary(glm.fit)$df.residual


#################################################predict
glm.probs =predict (glm.fit ,type =" response ")
glm.probs [1:10]

#########################################################calculating missclassification
########################################of the model
glm.pred=rep ("Down " ,1250)
glm.pred[glm.probs >.5]=" Up"
table(glm.pred ,Direction )


######################################################## Trainin and Testing data sets
train =(Year <2005)
Smarket.2005= Smarket [! train ,]
dim(Smarket.2005)
Direction.2005= Direction [! train]
###################################### another method
##split the observations into a training set and a test  set
set.seed (2)
train=sample (1: nrow(Carseats ), 200)
Carseats.test=Carseats [-train ,]
High.test=High[-train ]

########################################################## Fit the model
glm.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume ,
data=Smarket ,family =binomial ,subset =train )
glm.probs =predict (glm.fit ,Smarket.2005 , type="response")
```

```
#########################################################################creating confusion matrix
glm.pred=rep ("Down",252)
glm.pred[glm.probs >.5]=" Up"
table(glm.pred ,Direction.2005)
################################################calculating misclassification rate
mean(glm.pred== Direction.2005)
mean(glm.pred!= Direction.2005)
###################################
confint(glm.fit)
anova(glm.fit,test="Chisq")
anova(notcourse,model4,test="Chisq")


length(fitted(glm.fit))
length(Lag1[!train])
length(glm.probs)
plot(Lag1[!train], glm.probs)
```