Logistic regression

Miguel-Angel Canela, IESE Business School

October 31, 2017

Contents

Classification

Classification is Machine Learning most frequent job. A classification model allocates instances to two or more prespecified groups or classes. In **binary classification**, the most frequent case by far, there are only two classes, which are usually called **positive** and **negative**. The labels positive/negative should be assigned so that they favour our intuition. If we leave this to the computer, it may call positive what we call negative.

In the business context, two classics are:

- **Churn modelling**. A telephone company classifies its customers as either churners or non-churners (see the example). The class has two values, "churn" and "no churn".
- **Credit scoring**. A bank classifies credit applications as either "good" or "bad".

Let me assume, to simplify, a case of binary classification, and that the classes to be predicted are coded with a dummy (1/0). Although it is not equally evident in the various methods and implementations, a binary classification method produces a **predictive score** for every instance. The score is a number in the 0-1 range. The score is later transformed into a **predicted class**, based on a **cutoff** value. The instances whose score exceed the cutoff are classified as positive and the rest as negative.

The simplest approach would be to take 0.5 as the cutoff. Nevertheless, it can be replaced by another value with a better performance. In a business application, the choice of the cutoff may be based on a cost/benefit analysis. Specialized software can find the **optimal cutoff** for a user-specified cost matrix.

Classification methods

Classification models can be obtained with various methods:

- **Logistic regression**. The score is calculated by means of a (nonlinear) regression equation. This method is discussed in this lecture.
- **Neural networks**. The regression equation of the above paragraph is replaced by a neural network, which is a model which combines several equations.
- **Decision trees**. The scores are derived from a tree. We enter the tree by the **root** and proceed along the branches until arriving to the **leaves**. At each **node**, the branching is based on the value of one

variable. The same score is assigned to all the instances of the same leaf. The score is equal to the positive rate in that leaf. This method is discussed in the next lecture.

- **Random forests**. There are many methods based on combining several decision trees. The random forest models are very popular, due to their performance on big data sets.

How to evaluate a classification model

The evaluation of a classification model is usually based on a **confusion matrix**, obtained by crosstabulation of the actual class and the predicted class given by the model. Although there is not a universal consensus, in the confusion matrix the predicted class comes usually in the rows and the actual class in the columns. Table 1 is an example, calculated for a churn model. The four cells of the table are referred to as **true positive** (TP = 114), **false positive** (FN = 91), **false negative** (FN = 369) and **true negative** (TN = 2,759), respectively.

Table 1: TABLE 1. Confusion matrix

	Actual positive	Actual negative
Predicted positive	114	91
Predicted negative	369	2759

The proportion of instances classified in the right way, frequently called the **accuracy**, would be

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} = \frac{114 + 2759}{114 + 91 + 369 + 2759} = 86.2\%.$$

The proportion of right classification is not always the main criterion in the evaluation. Other measures which can be prioritized, depending on the application, are:

- The **true positive rate**, or proportion of right classification among the actual positives,

$$\text{TP rate} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{114}{114 + 369} = 23.6\%.$$

- The **false positive rate**, or proportion of wrong classification among the actual negatives,

$$\text{FP rate} = \frac{\text{FP}}{\text{FP} + \text{TN}} = \frac{91}{91 + 2759} = 3.2\%.$$

Not everybody agrees on this terminology. I use these terms as in Witten *et al.* (2011). In a good model, the TP rate should be high and the FP rate low. Nevertheless, the relative importance given to these statistics depends on the actual application.

Logistic regression

Let me suppose that the two classes to be predicted are coded with a dummy (positive = 1, negative = 0) and that there is a collection of numeric variables $X_1, \ldots, X_k$ available for prediction (some of them may come from coding categorical variables). The logistic regression equation is

$$p = F\big(a + b_1\, X_1 + \cdots + b_k\, X_k\big).$$

Here, p is the predictive score, and F is the **logistic function**,

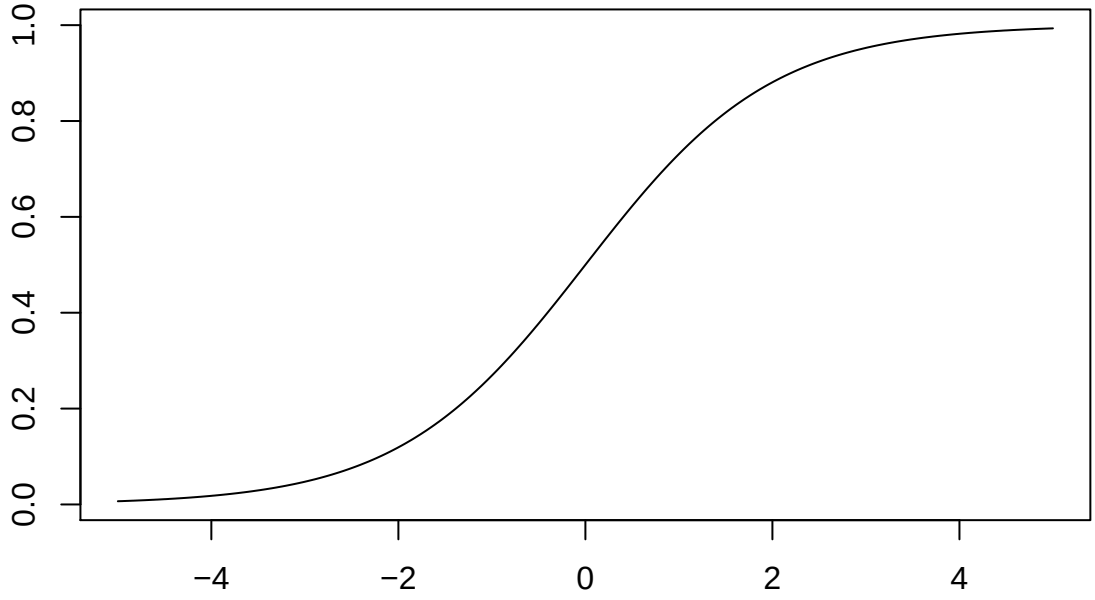
$$F(x) = \frac{1}{1 + \exp(-x)}\,.$$

Inverting the logistic function, the equation can be rewritten as

$$\log\left(\frac{p}{1-p}\right) = a + b_1\, X_1 + \cdots + b_k\, X_k.$$

The graph of the logistic function has an inverted S shape, as shown in Figure 1.

Figure 1. Logistic function



As given by the logistic function, the scores fall within the unit interval ($0 < p < 1$). So, the score can be regarded as the probability, or propensity, of an instance to have positive class. In logistic regression, this is obvious to the user.

A word of caution about the predictions of a logistic regression equation. Since the equation involves a transformation, Data Mining software can offer us a choice of the scale of the predicted values. More specifically, in logistic regression, we can predict either p or $\log\left[p/(1-p)\right]$. I will come back to this point in the example that follows.

Example: Churn modelling

The term **churn** is used in marketing to refer to a customer leaving the company in favour of a competitor. Churning is a common concern of **Customer Relationship Management** (CRM). A key step in proactive churn management is to predict whether a customer is likely to churn, since an early detection of the potential churners helps to plan the retention campaigns.

In this example, I develop a churn model, based on a logistic regression equation, for a company called *Omicron mobile*, which provides mobile phone services. The data set is a random sample from the customers database of the third quarter, whose accounts were still alive by September 30, and have been monitored during the fourth quarter. The sample size is 5,000, with a **churning rate** of 19.4%.

The variables included in the data set are:

- `id`, a customer ID (the phone number).
- `aclentgh`, the number of days the account has been active at the beginning of the period monitored.
- `intplan`, a dummy for having an international plan.
- `dataplan`, a dummy for having a data plan.
- `datagb`, the gigabytes available according to the data plan.
- `ommin`, the total minutes call to any Omicron mobile phone number, voicemail or national landline.
- `omcall`, the total number of calls to any Omicron mobile phone number, voicemail or national landline.
- `otmin`, the total minutes call to other mobile networks.
- `otcall`, the total number of calls to other networks.
- `ngmin`, the total minutes call to nongeographic numbers. Nongeographic numbers, such as UK 0844 or 0871 numbers, are often helplines for organizations like banks, insurance companies and utilities and charities.
- `ngcall`, the total number of calls to nongeographic numbers.
- `imin`, the total minutes in international calls.
- `icall`, the total international calls.
- `cuscall`, the number of calls to customer service.
- `churn`, a dummy for churning.

All the data are from the third quarter except the last variable. The data come in a csv file which I import with the function `read.csv`. I am using `stringsAsFactors=F` now. This means that string columns (`id` and `datagb`) are imported as character vectors, not as factors.

```
churn <- read.csv("churn.csv", stringsAsFactors=F)
str(churn)
```

```
## 'data.frame':    5000 obs. of  15 variables:
##  $ id      : chr  "409-8978" "444-7077" "401-9132" "409-2971" ...
##  $ aclength: int  77 105 121 115 133 95 50 157 35 96 ...
##  $ intplan : int  0 0 0 0 0 0 1 0 0 0 ...
##  $ dataplan: int  0 0 1 0 1 1 0 1 1 0 ...
##  $ datagb  : chr  "0" "0" "1.5G" "0" ...
##  $ ommin   : num  80.8 131.8 212.1 186.1 166.5 ...
##  $ omcall  : int  70 66 57 64 61 85 96 73 56 99 ...
##  $ otmin   : num  166 132 195 231 176 ...
##  $ otcall  : int  67 105 140 125 74 98 73 71 77 99 ...
##  $ ngmin   : num  18.6 5.1 14.9 26.5 36.1 11.1 34.5 15.3 21.6 12.4 ...
##  $ ngcall  : int  6 6 14 16 11 2 10 8 7 2 ...
##  $ imin    : num  9.5 6.7 28.6 9.9 5.3 0 18.4 11.3 0 5.2 ...
##  $ icall   : int  4 2 8 4 2 0 7 3 0 2 ...
##  $ cuscall : int  1 0 1 1 1 1 1 3 0 0 ...
##  $ churn   : int  0 0 0 0 0 1 1 0 1 0 ...
```

Prediction with a logistic regression model

I set the formula as in linear regression. I omit the customer ID (it does not make sense in a regression equation). I include the rest of the variables, although some of them may be redundant. I leave to R the job

of creating the dummies for the values of `datagb`.

```
fm <- churn ~ aclength + intplan + dataplan + datagb + ommin + omcall +
  otmin + otcall + ngmin + ngcall + imin + icall + cuscall
```

In `R`, logistic regression is a particular case of **generalized linear modelling**, performed with the function `glm`. This function has an extra argument, `family`, used for choosing among GLM methods. In the context of this course, this is not relevant, so I skip the details. The `summary` function works as for `lm`.

```
mod <- glm(formula=fm, data=churn, family="binomial")
summary(mod)
```

```
##
## Call:
## glm(formula = fm, family = "binomial", data = churn)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1194  -0.5904  -0.4306  -0.2832   2.8124
##
## Coefficients: (1 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.2055915  0.3084099 -16.879  < 2e-16 ***
## aclength     0.0004580  0.0010206   0.449  0.65363
## intplan      2.0718784  0.1342263  15.436  < 2e-16 ***
## dataplan     0.1802750  0.1676287   1.075  0.28218
## datagb1.5G  -0.3700765  0.2059367  -1.797  0.07233 .
## datagb100M  -0.1880175  0.3678689  -0.511  0.60928
## datagb1G    -0.6830531  0.2298969  -2.971  0.00297 **
## datagb250M  -0.4123199  0.2810901  -1.467  0.14241
## datagb2G    -0.2442181  0.3484156  -0.701  0.48334
## datagb500M          NA         NA      NA       NA
## ommin        0.0084588  0.0009783   8.646  < 2e-16 ***
## omcall      -0.0036002  0.0027493  -1.310  0.19036
## otmin        0.0039567  0.0009969   3.969 7.22e-05 ***
## otcall      -0.0005723  0.0020500  -0.279  0.78010
## ngmin        0.0064098  0.0035096   1.826  0.06779 .
## ngcall      -0.0092453  0.0108768  -0.850  0.39532
## imin         0.0423446  0.0101769   4.161 3.17e-05 ***
## icall        0.0569669  0.0314886   1.809  0.07043 .
## cuscall      0.4089702  0.0315578  12.959  < 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: 4914.0  on 4999  degrees of freedom
## Residual deviance: 3946.9  on 4982  degrees of freedom
## AIC: 3982.9
##
## Number of Fisher Scoring iterations: 5
```

Note that `R` creates the dummies associated to `datagb`, excluding the level which comes first in alphabetical order (`datagb = 0`). But one of the dummies (`datagb500M`) is dropped by `R` due to colinearity. Indeed, `datagb` takes value `"0"` when `dataplan` takes value `0`, so we have two superfluous dummies here. This is illustrated by the table below.

```
table(churn$dataplan, churn$datagb)
```

```
##
##        0 1.5G 100M   1G 250M   2G 500M
##   0 3449    0    0    0    0    0    0
##   1    0  522   74  410  168   86  291
```

But this does not harm our model, so we leave it as it is. The function `predict` is used as for the linear regression model, but, in order to obtain the scores, we have to specify the argument `type="response"`.

```
score <- predict(mod, newdata=churn, type="response")
```

The predictive score can be interpreted as the propensity to churn. Reading it so, it would be natural to set the cutoff at 0.5.

```
cut1 <- 0.5
```

Confusion matrix

The confusion matrix is obtained with the function `table`. Note that the vector first specified comes in the rows. In this example, the two vectors are logical, but, in general, they do not need to be of the same type.

```
conf1 <- table(score > cut1, churn$churn == 1)
conf1
```

```
##
##         FALSE TRUE
##   FALSE  3893  656
##   TRUE    139  312
```

The FP rate is pretty low, but the TP rate points to a problem which coud be detected in the histogram: a cutoff so high does not capture enough potential churners.

```
tp1 <- conf1["TRUE", "TRUE"]/sum(conf1[, "TRUE"])
tp1
```

```
## [1] 0.322314
```

```
fp1 <- conf1["TRUE", "FALSE"]/sum(conf1[, "FALSE"])
fp1
```

```
## [1] 0.03447421
```

The inadequacy of the cutoff can be easily seen plotting separate histograms of the scores, one for churners and the other for non-churners.

```
par(mfrow=c(1,2))
hist(score[churn$churn==1], breaks=20, main="Figure 2a. Scores (churners)",
  xlab="Churn score")
hist(score[churn$churn==0], breaks=20, main="Figure 2b. Scores (non-churners)",
  xlab="Churn score")
```

Figure 2a. Scores (churners)

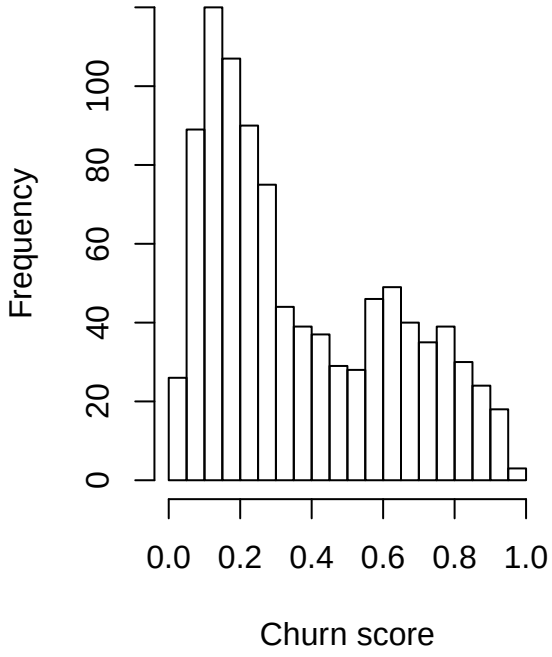
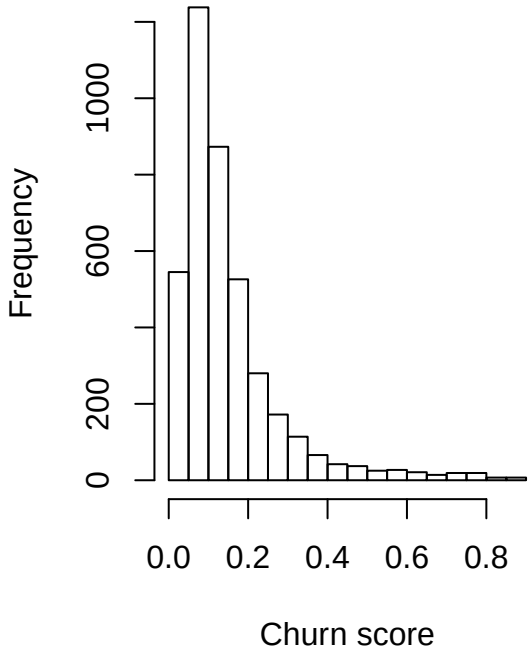


Figure 2b. Scores (non−churners



We see this in Figure 2. As a rule of thumb, setting the cutoff as the actual churn rate gives a good balance for TP and FP rates. So, I try this next.

```
cut2 <- mean(churn$churn)
conf2 <- table(score > cut2, churn$churn == 1)
conf2
```

```
##
##         FALSE TRUE
##   FALSE  3123  332
##   TRUE    909  636
```

```
tp2 <- conf2["TRUE", "TRUE"]/sum(conf2[, "TRUE"])
tp2
```

```
## [1] 0.6570248
```

```
fp2 <- conf2["TRUE", "FALSE"]/sum(conf2[, "FALSE"])
fp2
```

```
## [1] 0.2254464
```

Now the results look more acceptable. Nevertheless, in a business application, a cost/benefit model would be the right way to decide how useful this model could be.

Homework

Assume that the Omicron management plans to offer a 20% discount to the customers that the model classifies as potential churners, and that this offer is going to have a 100% success, so the company will retain all the churners.

- Evaluate the benefit produced by this retention policy with the two models already developed.
- Define an `R` function that gives the benefit in terms of the cutoff and find an optimal cutoff for this retention policy.

References

1. F Provost & T Fawcett (2013), *Data Science for Business — What You Need to Know About Data Mining and Data-Analytic Thinking*, O'Reilly.
2. IH Witten, E Frank & MA Hall (2011), *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann.