[DATA-03] Logistic regression

Miguel-Angel Canela, IESE Business School

September 22, 2016

Classification methods

Classification is data mining 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**.

In the business context, two classics are:

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

Classification models can be obtained with various methods. Let me assume, to simplify, a case of binary classification:

- **Models based on a regression equation**. By coding the classes to be predicted with a dummy, we can develop a classification model based on a regression equation. The equation is used to calculate a **predictive score** which is later transformed into a predicted class, using a **cutoff** value. This method is discussed in this lecture.
- **Neural networks**. In the approach explained in the above paragraph, the regression equation can be replaced by neural network, whioch is a model gthat combines several equations.
- **Decision trees**. The classification is based on a tree. We enter the tree by the **root** and proceed along the branches until arriving to the **leaves**. Each leaf is assigned to one of the categories of the class. At each **node**, the branching is based on the value of one variable. There are many methods for developing decision tree models. I use the R package `rpart` for this purpose in the next lecture of this course.
- **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. The R package `randomForest` is a favourite.

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. The labels positive/negative are assigned so that they favour intuition, but, if you leave this to the computer, it may call positive what you call negative.

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 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.

Another approach to the evaluation of the model uses a **cost/benefit analysis**. This is appropriate in applications where a positive case is taken as harmful, as in churn modelling. The analysis is based on a **cost matrix**, in which a cost is assigned to each of the four situations of the confusion matrix.

How to use a regression equation in binary classification

Regression equations can be used for classification purposes. To use a regression equation in binary classification, we code the classes to be predicted with a dummy (positive = 1, negative = 0). There are two main approaches:

- **Linear discriminant analysis** uses a linear regression equation.
- **Logistic regression** combines a linear expression and a mathematical function that forces the predicted values into the 0–1 range.

Although these two methods are presented in the textbooks as supported by different theoretical ideas, their performance in binary classification is similar. In both methods, the equation is fitted to a data set that contains the dummy coding the two classes (Y) and a collection of numeric variables ($X_1, \ldots, X_k$) which enter the right side of the equation as a linear combination $a + b_1\, X_1 + \cdots + b_k\, X_k$. Nominal variables can also be used, but they must be transformed previously into dummies.

The value given by the equation is taken as a predictive score. For classification purposes, the scores are turned into zeros and ones using a cutoff. The instances whose scores exceed the cutoff are classified as positive and the rest as negative. A difference between the two methods mentioned above is that, in linear discriminant analysis, some of the scores can fall out the 0–1 range. This does not hapen in logistic regression.

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. Although it is not equally evident in the various methods and implementations, all classifiers produce scores and use cutoffs. When the classifier is based on a regression equation, this is obvious to the user.

Logistic regression

The logistic regression equation is

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

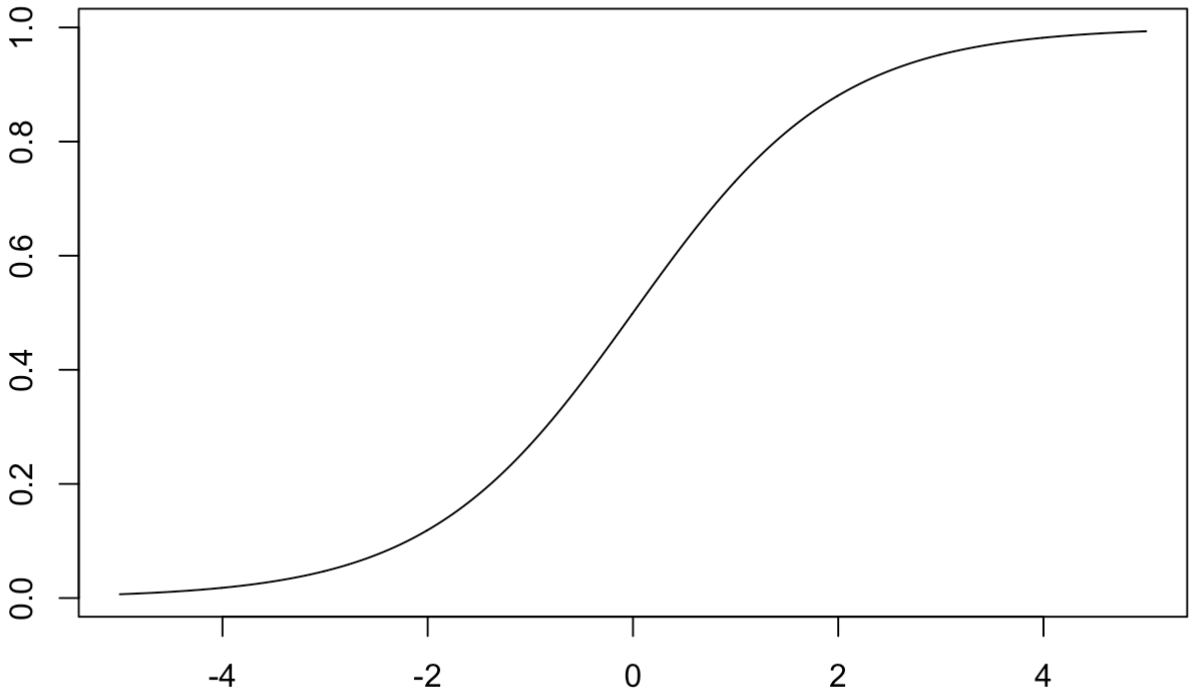
p is the prediction score, and F is the **logistic function**,

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

whose graph has the inverted S shape shown in Figure 1. Inverting the logistic function, the equation can be rewritten as

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

Figure 1. Logistic function



Since it takes values within the unit interval, the score can be regarded as the probability, or propensity, of an instance to have positive class. This is an advantage of logistic regression, with respect to linear regression, and facilitates to keep the score for management purposes. So, in credit scoring, the risk manager can use two cutoffs to classify the instances as either high, medium or low risk.

A word of caution about the predictions of a logistic regression equation. Since the equation involves a transformation, data mining software usually offers us a choice of the scale of the predicted values. More specifically, in logistic regression, we can predict eikther p or $\log[p/(1-p)]$. 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.

The data set used in this first analysis, which is ready at the end of the year, shows a **churning rate** of 19.4%. The variables included in the data set are:

- A customer ID, in this case the phone number.
- The number of days account has been active at the beginning of the period monitored.
- A dummy for having an international plan.
- A dummy for having a data plan.
- The gigabytes available according to the data plan.
- The total minutes call to any Omicron mobile phone number, voicemail or national landline.
- The total number of calls to any Omicron mobile& phone number, voicemail or national landline.
- The total minutes call to other mobile networks.
- The total number of calls to other networks.
- 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. They are available in many European countries.
- The total number of calls to nongeographic numbers
- The total minutes in international calls
- The total international calls
- The number of calls to customer service.
- A dummy for churning.

All the data correspond to the third quarter except the last variable. The data come in csv file which I import with the default of the `read.csv` function.

```
churn <- read.csv("churn.csv")
str(churn)
```

```
## 'data.frame':    5000 obs. of  15 variables:
##  $ id      : Factor w/ 5000 levels "350-1149","350-1404",..: 2915 4687 2525 2883 3989 1281
 2466 1968 3481 4789 ...
##  $ 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  : Factor w/ 7 levels "0","1.5G","100M",..: 1 1 2 1 2 7 1 6 7 1 ...
##  $ 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 ...
```

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, like `datagb`, are probably redundant. Note that the default of `read.csv` imports character fields as factors (this can be changed with the argument `stringsAsFactors=FALSE`).

```
fm1 <- 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 a technicality, so I skip the details. The `summary` function produces a report similar to that of `lm`.

```
mod1 <- glm(fm1, data=churn, family="binomial")
summary(mod1)
```

```
##
## Call:
## glm(formula = fm1, 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 the factor `datagb` , excluding the level which comes first in alphabetical order (`datagb0`). But one of the dummies is dropped by R due to colinearity. Indeed, `datagb` takes value `"0"` when `dataplan` takes value `0` , so we have two superfluous dummies here. But this does not harm your model, so I do not worry.

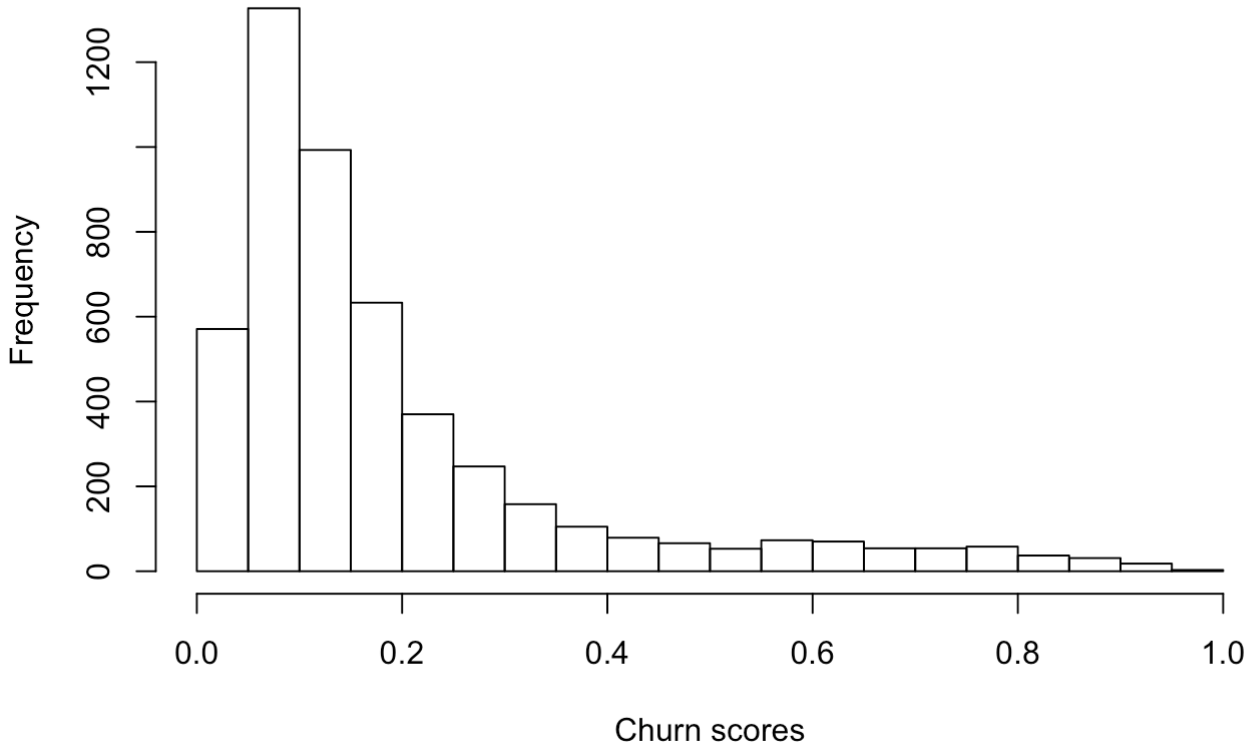
The predicted values are obtained as for the linear regression model, but, in order to obtain the scores, we have to specify the argument `type="response"` .

```
pred1 <- predict(mod1, newdata=churn, type="response")
```

The predictive score, which is in the 0-1 range, can be interpreted as the propensity to churn. If you do not have experience with predictive scores, plotting them may help you to guess how to transform them into real predictions of either churn or no churn. The histogram of Figure 2 illustrates this.

```
hist(pred1, main="Figure 2. Churn scores", xlab="Churn scores")
```

Figure 2. Churn scores



Reading the scores as probabilities, it would be natural to set the cutoff at 0.5.

```
cut1 <- 0.5
```

The confusion matrix is obtained with the function `table`, which can be used to tabulate one vector or to cross-tabulate two vectors, as I do here. 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. Note that I rewrite the name of the object calculated, to have it printed.

```
conf1 <- table(pred1 > cut1, churn$churn==1); conf1
```

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

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

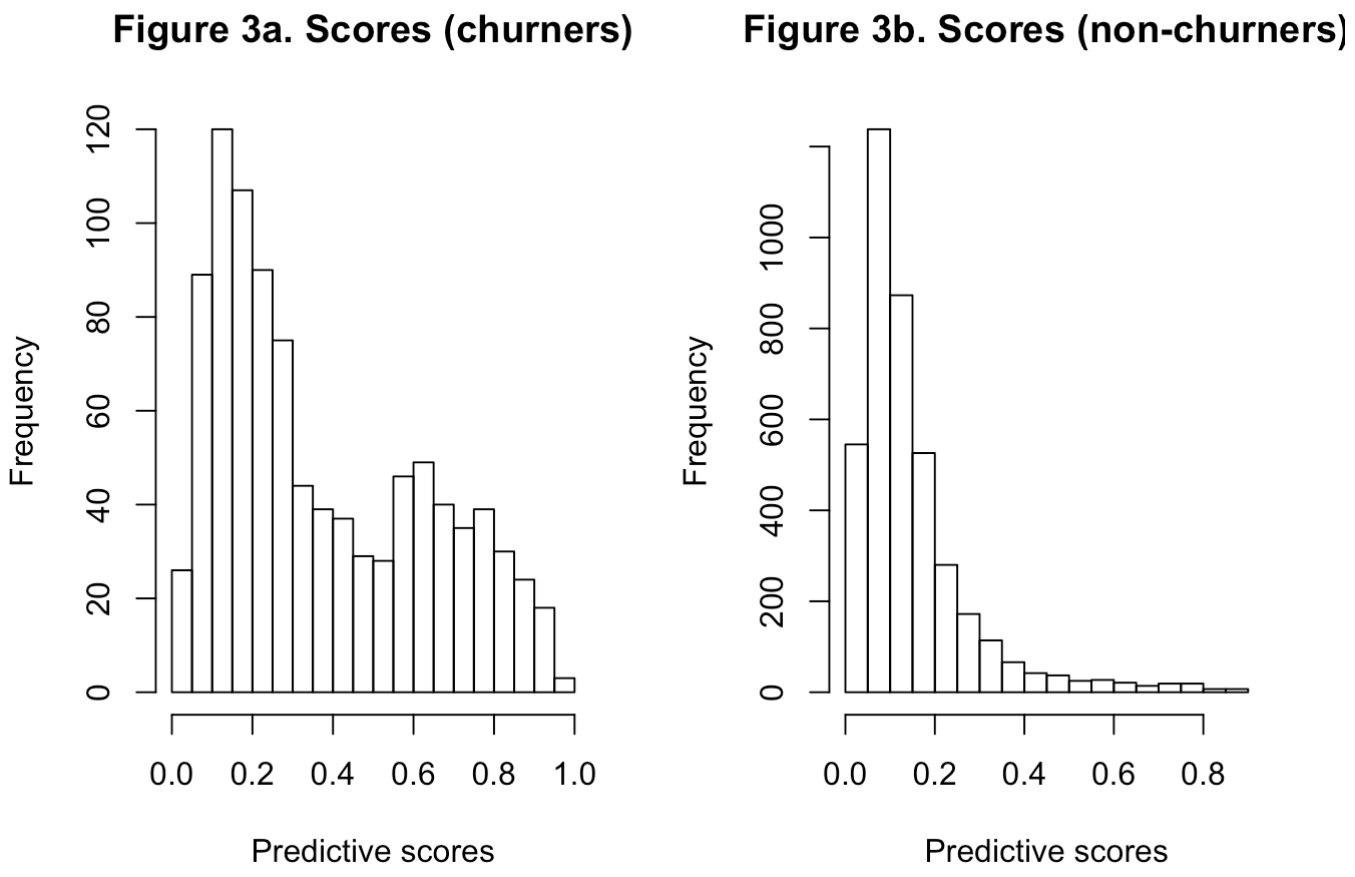
```
## [1] 0.322314
```

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

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

The FP rate is great, but the TP points to a problem which coud be detected in the histogram: a cutoff so high does not capture enough customers as potential churners. This is more easily seen splitting the histogram in two parts, one for churners and the other for non-churners.

```
par(mfrow=c(1,2))
hist(pred1[churn$churn==1], breaks=20, main="Figure 3a. Scores (churners)",
  xlab="Predictive scores")
hist(pred1[churn$churn==0], breaks=20, main="Figure 3b. Scores (non-churners)",
  xlab="Predictive scores")
```



We see in Figure 3 that we have to decrease the cutoff about 0.2 to capture enough churners, but then we start getting a high number of false positives. 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$churn)
conf2 <- table(pred1 > cut2, churn$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 reasonable. Nevertheless, in a business application, a cost/benefit model would be the right way to decide how useful this model could be. As an illustration, I use a simple example. Let me 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.

I set the objective function as the benefit that my model produces compared to having no predictive model. In that case, all the churners quit. I calculate the benefit using the customer as a unit:

- With no model, the true positives quit, but with the churn model they stay, paying 80%, so Omicron get a benefit of 0.8 for every instance.
- The false positives do not quit anyway, but with the model they get a 20% discount, so they get a benefit -0.2 for every instance.
- The false negatives and the true negatives do the same, model or no model, and they do not get discount, so the benefit is zero.

I set this as the benefit matrix.

```
B <- matrix(c(0, 0, -0.2, 0.8), nrow=2, byrow=TRUE); B
```

```
##      [,1] [,2]
## [1,]  0.0  0.0
## [2,] -0.2  0.8
```

Now, multiplying the benefit matrix by the confusion matrix and summing, I get the benefit of my two models.

```
benefit1 <- sum(conf1*B); benefit1
```

```
## [1] 221.8
```

```
benefit2 <- sum(conf2*B); benefit2
```

```
## [1] 327
```

So, with this retention policy, both models are beneficial, and the second one would be preferred. Of course, you can change the discount, or introduce sophistication in the retention policy in various ways.

Question

Define an R function that gives the benefit in terms of the cutoff and find an optimal cutoff for this retention policy.