Stat 292 - Recitation 10

Logistic Regression

Orçun Oltulu

2 / 6 / 2021

Logistic regression in R

The Logistic Regression is a regression model in which the response variable (dependent variable) has categorical values such as True/False or 0/1. It actually measures the probability of a binary response as the value of response variable based on the mathematical equation relating it with the predictor variables.

Logistic regression analysis belongs to the class of generalized linear models. In R generalized linear models are handled by the glm() function. The function is written as glm(response ~ predictor, family = binomial(link = "logit"), data). Please note that logit is the default for binomial; thus, we do not have to type it explicitly.

The glm() function returns a model object, therefore we may apply extractor functions, such as summary(), fitted() or predict, among others, on it. However, please note that the output numbers are on the logit scale. To actually predict probabilities we need to provide the predict() function an additional argument type = "response".

We will use a data on containing health-related measurements on women and whether they can be (or will be at a future point?) classified as diabetic. The data was collected by the US National Institute of Diabetes and is contained in the MASS package.

Diabetes in Pima Indian Women data set

A population of women who were at least 21 years old, of Pima Indian heritage and living near Phoenix, Arizona, was tested for diabetes according to World Health Organization criteria. The data were collected by the US National Institute of Diabetes and Digestive and Kidney Diseases. We used the 532 complete records after dropping the (mainly missing) data on serum insulin.

Format:

These data frames contains the following columns:

npreg: number of pregnancies. **glu:** plasma glucose concentration in an oral glucose tolerance test. **bp:** diastolic blood pressure (mm Hg). **skin:** triceps skin fold thickness

(mm). **bmi:** body mass index (weight in kg/(height in m)2). **ped:** diabetes pedigree function. **age:** age in years. **type:** Yes or No, for diabetic according to WHO criteria.

The training set Pima.tr contains a randomly selected set of 200 subjects, and Pima.te contains the remaining 332 subjects.

Load Required Packages

```
inst_pack_func <- function(list.of.packages){
  new.packages <- list.of.packages[!(list.of.packages %in%
                                       installed.packages()[,"Package"])]
  if(length(new.packages)) install.packages(new.packages)
  lapply(list.of.packages,function(x){library(x,character.only=TRUE)})
}

list.of.packages <- c("MASS","tidyverse","magrittr",
                      "ISLR","psych","dplyr",
                      "ggplot2","knitr","e1071","caret")

inst_pack_func(list.of.packages)
```

Exercise 1:

Part A:

Save Pima.tr as train.dataset and save Pima.te as test.dataset.

```
data("Pima.tr");data("Pima.te")

train.dataset <- Pima.tr
test.dataset <- Pima.te
```

Part B:

Check the dimensions of the data sets, get some summary statistics for each data set.

```
str(train.dataset)
```

```
## 'data.frame':    200 obs. of  8 variables:
##  $ npreg: int  5 7 5 0 0 5 3 1 3 2 ...
##  $ glu  : int  86 195 77 165 107 97 83 193 142 128 ...
##  $ bp   : int  68 70 82 76 60 76 58 50 80 78 ...
##  $ skin : int  28 33 41 43 25 27 31 16 15 37 ...
##  $ bmi  : num  30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...
##  $ ped  : num  0.364 0.163 0.156 0.259 0.133 ...
##  $ age  : int  24 55 35 26 23 52 25 24 63 31 ...
##  $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
```

```
summary(train.dataset)
```

```
##      npreg            glu              bp              skin      
##  Min.   : 0.00   Min.   : 56.0   Min.   : 38.00   Min.   : 7.00  
##  1st Qu.: 1.00   1st Qu.:100.0   1st Qu.: 64.00   1st Qu.:20.75  
##  Median : 2.00   Median :120.5   Median : 70.00   Median :29.00  
##  Mean   : 3.57   Mean   :124.0   Mean   : 71.26   Mean   :29.21  
##  3rd Qu.: 6.00   3rd Qu.:144.0   3rd Qu.: 78.00   3rd Qu.:36.00  
##  Max.   :14.00   Max.   :199.0   Max.   :110.00   Max.   :99.00  
##       bmi             ped              age         type    
##  Min.   :18.20   Min.   :0.0850   Min.   :21.00   No :132  
##  1st Qu.:27.57   1st Qu.:0.2535   1st Qu.:23.00   Yes: 68  
##  Median :32.80   Median :0.3725   Median :28.00            
##  Mean   :32.31   Mean   :0.4608   Mean   :32.11            
##  3rd Qu.:36.50   3rd Qu.:0.6160   3rd Qu.:39.25            
##  Max.   :47.90   Max.   :2.2880   Max.   :63.00
```

```
str(test.dataset)
```

```
## 'data.frame':    332 obs. of  8 variables:
##  $ npreg: int  6 1 1 3 2 5 0 1 3 9 ...
##  $ glu  : int  148 85 89 78 197 166 118 103 126 119 ...
##  $ bp   : int  72 66 66 50 70 72 84 30 88 80 ...
##  $ skin : int  35 29 23 32 45 19 47 38 41 35 ...
##  $ bmi  : num  33.6 26.6 28.1 31 30.5 25.8 45.8 43.3 39.3 29 ...
##  $ ped  : num  0.627 0.351 0.167 0.248 0.158 0.587 0.551 0.183 0.704 0.263 ...
##  $ age  : int  50 31 21 26 53 51 31 33 27 29 ...
##  $ type : Factor w/ 2 levels "No","Yes": 2 1 1 2 2 2 2 1 1 2 ...
```

```
summary(test.dataset)
```

```
##      npreg            glu              bp              skin      
##  Min.   : 0.000   Min.   : 65.0   Min.   : 24.00   Min.   : 7.00  
##  1st Qu.: 1.000   1st Qu.: 96.0   1st Qu.: 64.00   1st Qu.:22.00  
##  Median : 2.000   Median :112.0   Median : 72.00   Median :29.00  
##  Mean   : 3.485   Mean   :119.3   Mean   : 71.65   Mean   :29.16  
##  3rd Qu.: 5.000   3rd Qu.:136.2   3rd Qu.: 80.00   3rd Qu.:36.00  
##  Max.   :17.000   Max.   :197.0   Max.   :110.00   Max.   :63.00  
##       bmi             ped              age         type    
##  Min.   :19.40   Min.   :0.0850   Min.   :21.00   No :223  
##  1st Qu.:28.18   1st Qu.:0.2660   1st Qu.:23.00   Yes:109  
##  Median :32.90   Median :0.4400   Median :27.00            
##  Mean   :33.24   Mean   :0.5284   Mean   :31.32            
##  3rd Qu.:37.20   3rd Qu.:0.6793   3rd Qu.:37.00            
##  Max.   :67.10   Max.   :2.4200   Max.   :81.00
```

Part C:

Change the coding of our variable of interest (type) to a numeric variable; 0 (non-diabetic) and 1 (diabetic).

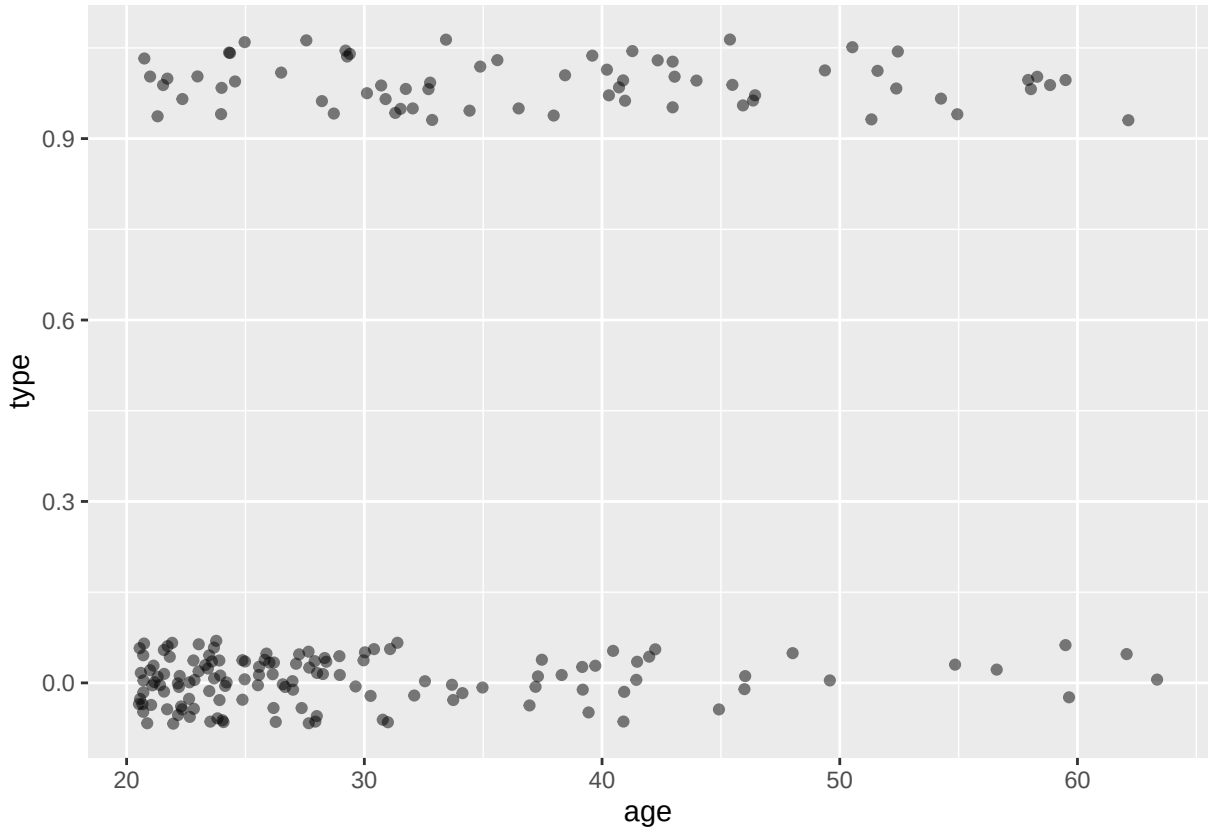
```
train.dataset %<>% mutate(type = as.numeric(type)-1)

test.dataset %<>% mutate(type = as.numeric(type)-1)
```

Part D:

Try to plot type as a function of age (Age vs Type). Use geom_jitter() to make your graph more informative.

```
p1 <- ggplot(train.dataset, aes(x = age, y = type)) +
 geom_jitter(width = 0.5, height = 0.07, alpha = .5)
p1
```

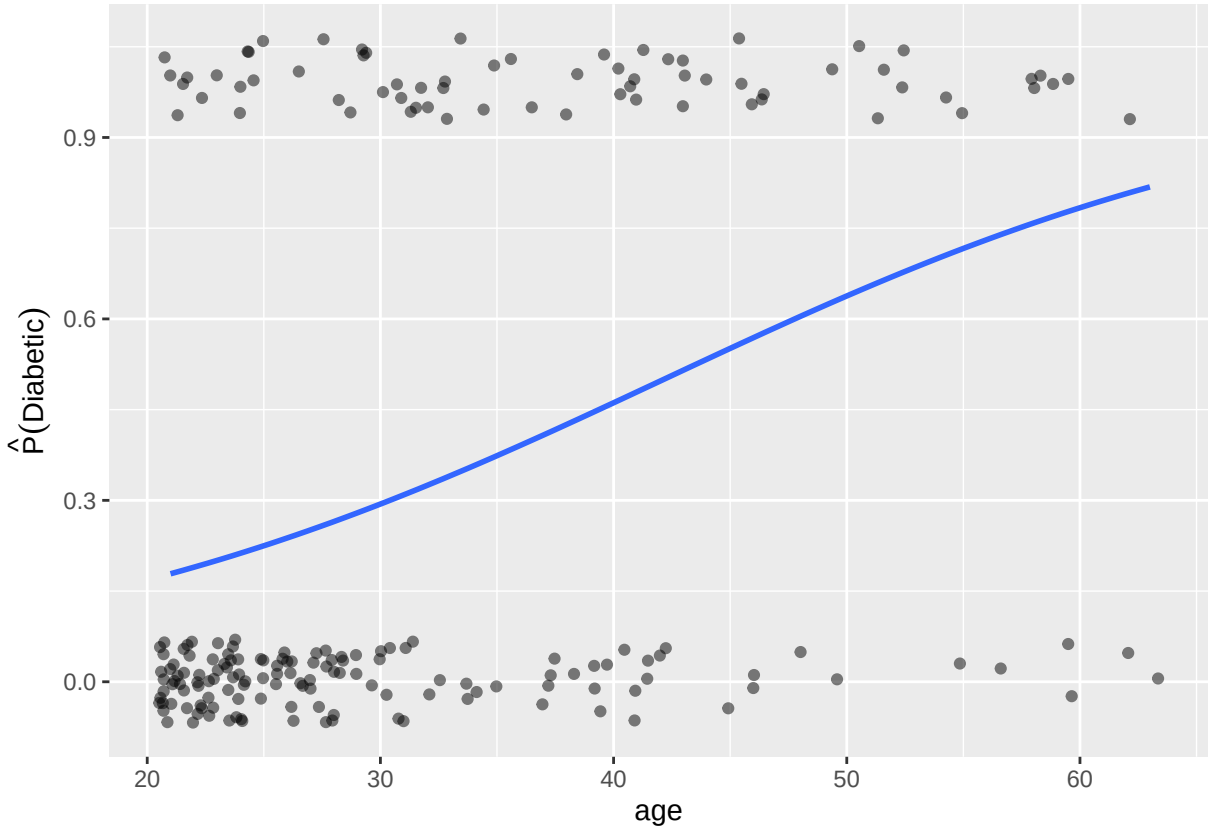


Part E:

Add a logistic regression line to the previous plot using geom_smooth() function.

```
p2 <- p1 + geom_smooth(method = "glm", se = FALSE,
                       method.args = list(family = "binomial")) +
 labs(y = expression(hat(P)(Diabetic)))
p2
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Part F:

Using the glm() and the train.dataset data fit a logistic model of type on age and bmi. Print out the results and comment on them.

```
lg1 <- glm(type ~ age + bmi, data = train.dataset, family = binomial)
summary(lg1)
```

```
##
## Call:
## glm(formula = type ~ age + bmi, family = binomial, data = train.dataset)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7935  -0.8368  -0.5033   1.0211   2.2531
```

```
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.49870    1.17459  -5.533 3.15e-08 ***
## age          0.07104    0.01538   4.620 3.84e-06 ***
## bmi          0.10519    0.02956   3.558 0.000373 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 256.41  on 199  degrees of freedom
## Residual deviance: 215.93  on 197  degrees of freedom
## AIC: 221.93
##
## Number of Fisher Scoring iterations: 4
```

Part G:

What does the model fitted in part G predict in terms of probability for someone with age 35 with bmi of 32, what about bmi of 22?

$$p(x) = P(Y = 1|X) = \frac{e^{\beta_0+\beta_i X_i}}{1 + e^{\beta_0+\beta_i X_i}}$$

```
new_obs <- data.frame(bmi = c(32, 22), age = 35)

## with predict command
predict(lg1, type = "response", newdata = new_obs)
```

```
##         1         2
## 0.3438203 0.1546990
```

```
## manually
lgs_fun1 <- function(par, x){ exp(par%*%x)/(1 + exp(par %*% x))}

sapply(list(c(1, 35, 32),
            c(1, 35, 22)),
       lgs_fun1,
       par = lg1$coefficients)
```

```
## [1] 0.3438203 0.1546990
```

Part H:

According to our model what are the odds that a woman in our sample is diabetic given age 55 and a bmi 37? Remember that odds in this context have a very precise definition which is different from probability.

$$\text{Odds} = \frac{p(x)}{1-p(x)} = e^{\beta_0+\beta_i X_i}$$

```
px <- lgs_fun1(lg1$coefficients, c(1, 55, 37))
odds <- px/(1-px)
odds
```

```
##          [,1]
## [1,] 3.670704
```

```
## alternatively
exp(c(1, 55, 37) %*% lg1$coefficients)
```

```
##          [,1]
## [1,] 3.670704
```

Part I:

Build the confusion matrix, a table of actual diabetic classification against model prediction. Use a cutoff value of 0.5, meaning that women who the model estimates to have at least 0.5 chance of being diabetic are predicted to be diabetic. What is the prediction accuracy?

```
## Get the predicted values from model
predicted.values <- as.numeric(predict(lg1, type = "response") >= 0.5)
predicted.values %>% head(20)
```

```
##  [1] 0 1 0 1 0 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0
```

```
## Confusion Matrix
cm1 <- table(train.dataset$type, predicted.values)
cm1
```

```
##    predicted.values
##       0   1
##   0 114  18
##   1  39  29
```

```
## Prediction accuracy
sum(diag(cm1)) / sum(cm1)
```

```
## [1] 0.715
```

```r
## Alternative Way
predicted.values <- as.numeric(predict(lg1, type = "response") >= 0.5)
caret::confusionMatrix(factor(predicted.values),factor(train.dataset$type))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 114  39
##          1  18  29
##
##                Accuracy : 0.715
##                  95% CI : (0.6471, 0.7764)
##     No Information Rate : 0.66
##     P-Value [Acc > NIR] : 0.056996
##
##                   Kappa : 0.3136
##
##  Mcnemar's Test P-Value : 0.008071
##
##             Sensitivity : 0.8636
##             Specificity : 0.4265
##          Pos Pred Value : 0.7451
##          Neg Pred Value : 0.6170
##              Prevalence : 0.6600
##          Detection Rate : 0.5700
##    Detection Prevalence : 0.7650
##       Balanced Accuracy : 0.6451
##
##        'Positive' Class : 0
##
```

Part J:

Apply the fitted model to the test set. Print the confusion matrix and prediction accuracy.

```r
## Get the predicted probability values from model
test_pred <- predict(lg1, type = "response",
                     newdata = test.dataset)
test_pred %>% head(20)
```

```
##         1         2         3         4         5         6         7
## 0.64280304 0.18265090 0.11394324 0.19927322 0.61646622 0.45961554 0.62741692
##         8         9        10        11        12        13        14
## 0.59874826 0.39013815 0.19970972 0.07617620 0.82496883 0.08889771 0.40160683
```

```
##         15         16         17         18         19         20
## 0.16384302 0.55719065 0.36298305 0.79930239 0.25440139 0.33157039
```

```
predicted.values_test <- as.numeric( test_pred >= 0.5)
predicted.values_test %>% head(20)
```

```
##  [1] 1 0 0 0 1 0 1 1 0 0 0 1 0 0 0 1 0 1 0 0
```

```
## Confusion Matrix
cm1_test <- table(test.dataset$type,
                  predicted.values_test)
cm1_test
```

```
##    predicted.values_test
##       0   1
##   0 194  29
##   1  67  42
```

```
## Prediction accuracy
sum(diag(cm1_test)) / sum(cm1_test)
```

```
## [1] 0.7108434
```

```
## Alternative Way
predicted.values <- as.numeric(
  predict(lg1,
          type = "response",
          newdata = test.dataset)
  >= 0.5)

caret::confusionMatrix(factor(predicted.values),
                       factor(test.dataset$type))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 194  67
##          1  29  42
##
##                Accuracy : 0.7108
##                  95% CI : (0.6588, 0.759)
##     No Information Rate : 0.6717
##     P-Value [Acc > NIR] : 0.0708896
##
##                   Kappa : 0.2802
##
```

```
##  Mcnemar's Test P-Value : 0.0001592
##
##             Sensitivity : 0.8700
##             Specificity : 0.3853
##          Pos Pred Value : 0.7433
##          Neg Pred Value : 0.5915
##              Prevalence : 0.6717
##          Detection Rate : 0.5843
##    Detection Prevalence : 0.7861
##       Balanced Accuracy : 0.6276
##
##        'Positive' Class : 0
##
```

Part K:

Fit another model with a different set of the variables and see if you can improve the prediction accuracy.

```
lg2 <- glm(type ~ .,
           data = train.dataset,
           family = binomial)
summary(lg2)
```

```
##
## Call:
## glm(formula = type ~ ., family = binomial, data = train.dataset)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9830  -0.6773  -0.3681   0.6439   2.3154
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.773062   1.770386  -5.520 3.38e-08 ***
## npreg        0.103183   0.064694   1.595  0.11073
## glu          0.032117   0.006787   4.732 2.22e-06 ***
## bp          -0.004768   0.018541  -0.257  0.79707
## skin        -0.001917   0.022500  -0.085  0.93211
## bmi          0.083624   0.042827   1.953  0.05087 .
## ped          1.820410   0.665514   2.735  0.00623 **
## age          0.041184   0.022091   1.864  0.06228 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 256.41  on 199  degrees of freedom
## Residual deviance: 178.39  on 192  degrees of freedom
## AIC: 194.39
##
## Number of Fisher Scoring iterations: 5
```

```
predicted.values <- as.numeric(
  predict(lg2,
          type = "response",
          newdata = test.dataset)
  >= 0.5)

caret::confusionMatrix(factor(predicted.values),
                       factor(test.dataset$type))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 200  43
##          1  23  66
##
##                Accuracy : 0.8012
##                  95% CI : (0.7542, 0.8428)
##     No Information Rate : 0.6717
##     P-Value [Acc > NIR] : 1.116e-07
##
##                   Kappa : 0.5271
##
##  Mcnemar's Test P-Value : 0.01935
##
##             Sensitivity : 0.8969
##             Specificity : 0.6055
##          Pos Pred Value : 0.8230
##          Neg Pred Value : 0.7416
##              Prevalence : 0.6717
##          Detection Rate : 0.6024
##    Detection Prevalence : 0.7319
##       Balanced Accuracy : 0.7512
##
##        'Positive' Class : 0
##
```

```r
lg3 <- glm(type ~ glu + bmi + ped + age,
           data = train.dataset,
           family = binomial)
summary(lg3)
```

```
##
## Call:
## glm(formula = type ~ glu + bmi + ped + age, family = binomial,
##     data = train.dataset)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0863  -0.6727  -0.3689   0.6823   2.2092
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -9.971388   1.527587  -6.528 6.69e-11 ***
## glu          0.031255   0.006627   4.716 2.40e-06 ***
## bmi          0.077030   0.032251   2.388 0.016921 *
## ped          1.719794   0.656088   2.621 0.008760 **
## age          0.058603   0.017574   3.335 0.000854 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 256.41  on 199  degrees of freedom
## Residual deviance: 181.08  on 195  degrees of freedom
## AIC: 191.08
##
## Number of Fisher Scoring iterations: 5
```

```r
predicted.values <- as.numeric(
  predict(lg3,
          type = "response",
          newdata = test.dataset)
  >= 0.5)

caret::confusionMatrix(factor(predicted.values),
                       factor(test.dataset$type))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
```

```
##          0 196  42
##          1  27  67
##
##                Accuracy : 0.7922
##                  95% CI : (0.7445, 0.8345)
##     No Information Rate : 0.6717
##     P-Value [Acc > NIR] : 8.166e-07
##
##                   Kappa : 0.5116
##
##  Mcnemar's Test P-Value : 0.09191
##
##             Sensitivity : 0.8789
##             Specificity : 0.6147
##          Pos Pred Value : 0.8235
##          Neg Pred Value : 0.7128
##              Prevalence : 0.6717
##          Detection Rate : 0.5904
##    Detection Prevalence : 0.7169
##       Balanced Accuracy : 0.7468
##
##        'Positive' Class : 0
##
```