Práctica aprendizaje supervisado

Guillermo Bonafonte Criado

15/1/2017

1. Cargamos los datos

Cargamos los datos proporcionados por la librería mlbench

```
library(mlbench)
data(Sonar)
```

Vemos de un vistazo la estructura del dataset

Mostramos solo las 5 primeras columnas ya que el numero de columnas del dataset es muy amplio

```
library(knitr)
kable(head(Sonar[,1:5]))
```

V1	V2	V3	V4	V5
0.0200	0.0371	0.0428	0.0207	0.0954
0.0453	0.0523	0.0843	0.0689	0.1183
0.0262	0.0582	0.1099	0.1083	0.0974
0.0100	0.0171	0.0623	0.0205	0.0205
0.0762	0.0666	0.0481	0.0394	0.0590
0.0286	0.0453	0.0277	0.0174	0.0384

2. Preparacion de los datos

Creamos las particiones de entrenamiento y test para los datos Sonar

```
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

set.seed(998)
inTraining <- createDataPartition(Sonar$Class, p = .75, list = FALSE)
training <- Sonar[ inTraining,]
testing  <- Sonar[-inTraining,]
```

3. Clasificación

3.1. Para el clasificador k-NN

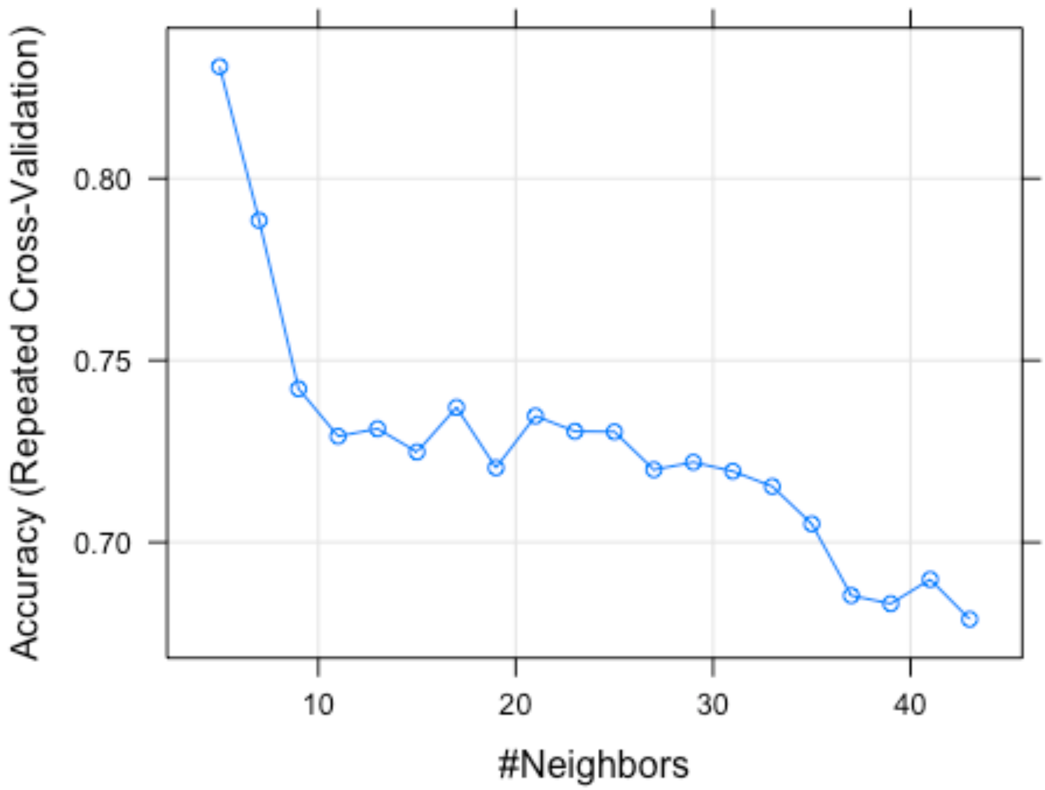
Train control y traning

```
library(class)
ctrl <- trainControl(method="repeatedcv",repeats = 3)
knnFit <- train(Class ~ ., data = training, method = "knn", trControl =
ctrl, preProcess = c("center","scale"), tuneLength = 20)

knnFit

## k-Nearest Neighbors
##
## 157 samples
##  60 predictor
##   2 classes: 'M', 'R'
##
## Pre-processing: centered (60), scaled (60)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 142, 142, 142, 141, 140, 142, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8307598  0.6549572
##    7  0.7884886  0.5685811
##    9  0.7422059  0.4739055
##   11  0.7291503  0.4471211
##   13  0.7312173  0.4486303
##   15  0.7248284  0.4358734
##   17  0.7370833  0.4601757
##   19  0.7204902  0.4270589
##   21  0.7347222  0.4563399
##   23  0.7305065  0.4487111
##   25  0.7303840  0.4491056
##   27  0.7199510  0.4277507
##   29  0.7220180  0.4321965
##   31  0.7195343  0.4288471
##   33  0.7153513  0.4205972
##   35  0.7050572  0.4008932
##   37  0.6853676  0.3621249
##   39  0.6831291  0.3574491
##   41  0.6897958  0.3705577
##   43  0.6788399  0.3499199
##
## Accuracy was used to select the optimal model using  the largest
value.
## The final value used for the model was k = 5.
```

Accuracy = 0,83 Kappa = 0.65

Plot

```r
plot(knnFit)
```



###Prediccion y matriz de confusion

```r
knnPredict <- predict(knnFit,newdata = testing )
#Obtenemos la matriz de confusion y vemos accuracy value y otros parámetros
confusionMatrix(knnPredict, testing$Class )

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M 24 11
##          R  3 13
##
##                Accuracy : 0.7255
##                  95% CI : (0.5826, 0.8411)
##     No Information Rate : 0.5294
##     P-Value [Acc > NIR] : 0.003347
##
```

```
##                  Kappa : 0.4387
##  Mcnemar's Test P-Value : 0.061369
##
##            Sensitivity : 0.8889
##            Specificity : 0.5417
##         Pos Pred Value : 0.6857
##         Neg Pred Value : 0.8125
##             Prevalence : 0.5294
##         Detection Rate : 0.4706
##   Detection Prevalence : 0.6863
##      Balanced Accuracy : 0.7153
##
##       'Positive' Class : M
##
```

3.2. Para el clasificador C-SVM kernel lineal

Train control y training

```
library(class)
ctrl <- trainControl(method="repeatedcv",repeats = 10)
svmFit <- train(Class ~ ., data = training, method = "svmLinear",
trControl = ctrl, preProcess = c("center","scale"), tuneLength = 20 )
```

```
## Loading required package: kernlab

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```
svmFit
```

```
## Support Vector Machines with Linear Kernel
##
## 157 samples
##  60 predictor
##   2 classes: 'M', 'R'
##
## Pre-processing: centered (60), scaled (60)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 141, 141, 142, 142, 141, 141, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7761471  0.5491175
##
## Tuning parameter 'C' was held constant at a value of 1
##
```

Accuracy = 0,77 Kappa = 0.59

Lo hacemos tambien con este modo.

Diferentes valores de C

```
ctrl <- trainControl(method="repeatedcv",   # 10fold cross validation
                     repeats=5,         # do 5 repititions of cv
                     summaryFunction=twoClassSummary,   # Use AUC to pick
the best model
                     classProbs=TRUE)


#Train and Tune the SVM
svm.tune <- train(Class ~ .,
                  training,
                  method = "svmRadial",   # Radial kernel
                  tuneLength = 9,                   # 9 values of the
cost function
                  preProc = c("center","scale"),  # Center and scale data
                  metric="ROC",
                  trControl=ctrl)

svm.tune
```

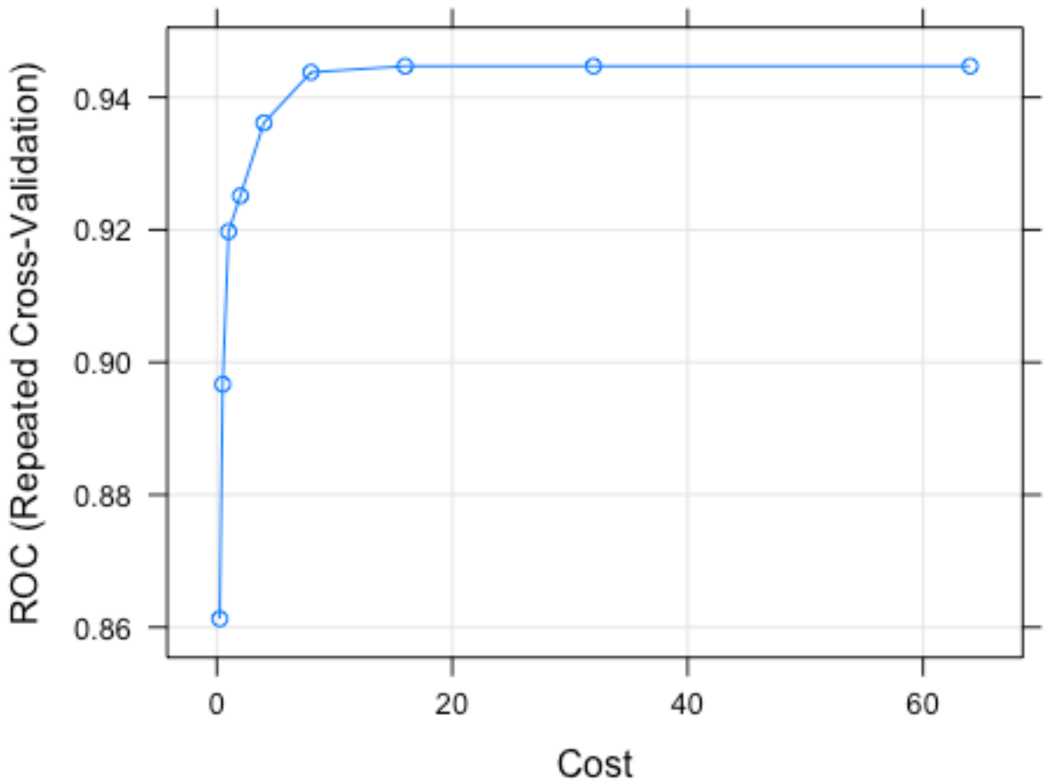
```
## Support Vector Machines with Radial Basis Function Kernel
##
## 157 samples
##  60 predictor
##   2 classes: 'M', 'R'
##
## Pre-processing: centered (60), scaled (60)
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 142, 142, 142, 142, 141, 141, ...
## Resampling results across tuning parameters:
##
##   C      ROC        Sens       Spec
##    0.25  0.8612798  0.7308333  0.7628571
##    0.50  0.8966766  0.8341667  0.7632143
##    1.00  0.9197321  0.8658333  0.7700000
##    2.00  0.9251339  0.8772222  0.7521429
##    4.00  0.9361359  0.8875000  0.8078571
##    8.00  0.9437748  0.8966667  0.8332143
##   16.00  0.9447073  0.8986111  0.8189286
##   32.00  0.9447073  0.9086111  0.8139286
##   64.00  0.9447073  0.9091667  0.8135714
##
## Tuning parameter 'sigma' was held constant at a value of 0.01129264
## ROC was used to select the optimal model using  the largest value.
```

```
## The final values used for the model were sigma = 0.01129264 and C =
16.
```

El valor óptimo de C es 0.25

Plot

```
plot(svm.tune)
```



Prediccion y matriz de confusion

```
svmPredict <- predict(svmFit,newdata = testing )
#Obtenemos la matriz de confusion y vemos accuracy value y otros
parámetros
confusionMatrix(svmPredict, testing$Class )

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M 23  9
##          R  4 15
##
##                Accuracy : 0.7451
##                  95% CI : (0.6037, 0.8567)
```

```
##     No Information Rate : 0.5294
##     P-Value [Acc > NIR] : 0.001311
##
##                   Kappa : 0.4824
##  Mcnemar's Test P-Value : 0.267257
##
##             Sensitivity : 0.8519
##             Specificity : 0.6250
##          Pos Pred Value : 0.7188
##          Neg Pred Value : 0.7895
##              Prevalence : 0.5294
##          Detection Rate : 0.4510
##    Detection Prevalence : 0.6275
##       Balanced Accuracy : 0.7384
##
##        'Positive' Class : M
##
```

```r
svmPredict <- predict(svm.tune,newdata = testing )
#Obtenemos la matriz de confusion y vemos accuracy value y otros parámetros
confusionMatrix(svmPredict, testing$Class )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M 24  7
##          R  3 17
##
##                Accuracy : 0.8039
##                  95% CI : (0.6688, 0.9018)
##     No Information Rate : 0.5294
##     P-Value [Acc > NIR] : 4.341e-05
##
##                   Kappa : 0.6028
##  Mcnemar's Test P-Value : 0.3428
##
##             Sensitivity : 0.8889
##             Specificity : 0.7083
##          Pos Pred Value : 0.7742
##          Neg Pred Value : 0.8500
##              Prevalence : 0.5294
##          Detection Rate : 0.4706
##    Detection Prevalence : 0.6078
##       Balanced Accuracy : 0.7986
##
##        'Positive' Class : M
##
```

3.3. Para el clasificador kernel no lineal RBF

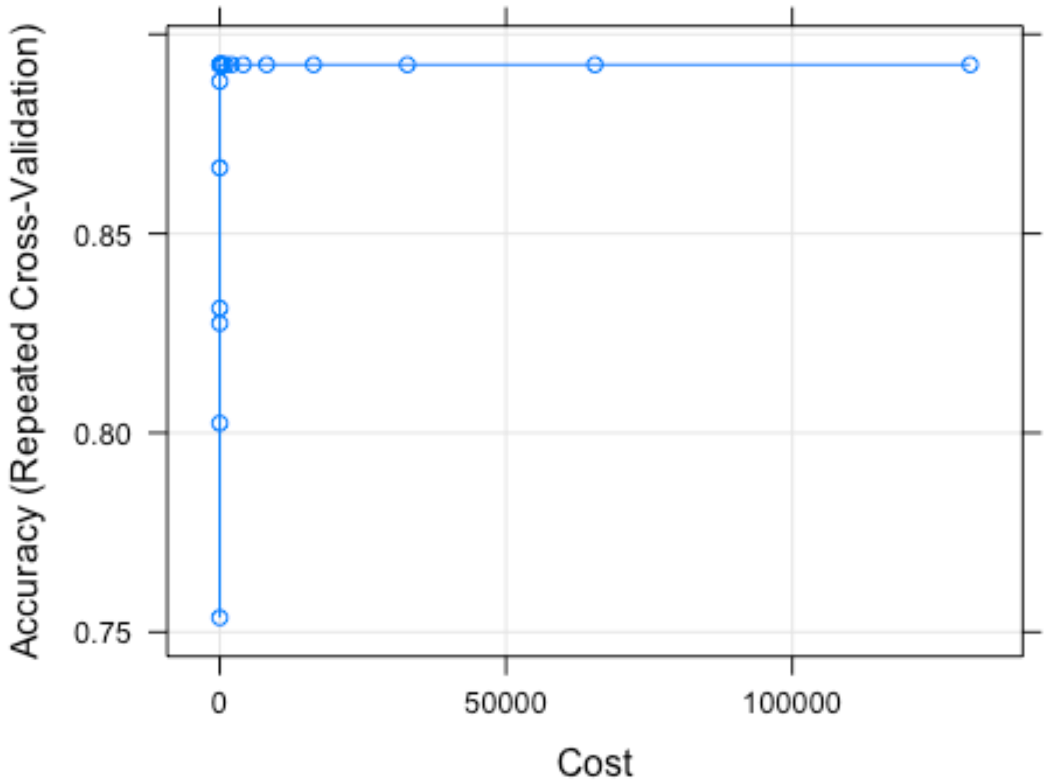
Train control y training

```
library(class)
ctrl <- trainControl(method="repeatedcv",repeats = 3)
svmRadialFit <- train(Class ~ ., data = training, method = 'svmRadial',
trControl = ctrl, preProcess = c("center","scale"), tuneLength = 20)

svmRadialFit

## Support Vector Machines with Radial Basis Function Kernel
##
## 157 samples
##  60 predictor
##   2 classes: 'M', 'R'
##
## Pre-processing: centered (60), scaled (60)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 142, 140, 141, 141, 142, 141, ...
## Resampling results across tuning parameters:
##
##   C          Accuracy   Kappa
##        0.25  0.7536193  0.4904887
##        0.50  0.8024183  0.5957495
##        1.00  0.8274673  0.6492538
##        2.00  0.8312663  0.6565980
##        4.00  0.8665033  0.7287658
##        8.00  0.8882026  0.7724565
##       16.00  0.8923856  0.7806451
##       32.00  0.8923856  0.7806451
##       64.00  0.8923856  0.7806451
##      128.00  0.8923856  0.7806451
##      256.00  0.8923856  0.7806451
##      512.00  0.8923856  0.7806451
##     1024.00  0.8923856  0.7806451
##     2048.00  0.8923856  0.7806451
##     4096.00  0.8923856  0.7806451
##     8192.00  0.8923856  0.7806451
##    16384.00  0.8923856  0.7806451
##    32768.00  0.8923856  0.7806451
##    65536.00  0.8923856  0.7806451
##   131072.00  0.8923856  0.7806451
##
## Tuning parameter 'sigma' was held constant at a value of 0.01200998
## Accuracy was used to select the optimal model using  the largest
value.
## The final values used for the model were sigma = 0.01200998 and C =
16.
```

Accuracy = 0,75 Kappa = 0.59 C = 0.25

Plot

```
plot(svmRadialFit)
```



###Prediccion y matriz de confusion

```
knnPredict <- predict(svmRadialFit,newdata = testing )
#Obtenemos la matriz de confusion y vemos accuracy value y otros parámetros
confusionMatrix(knnPredict, testing$Class )

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M 24  7
##          R  3 17
##
##                Accuracy : 0.8039
##                  95% CI : (0.6688, 0.9018)
##     No Information Rate : 0.5294
##     P-Value [Acc > NIR] : 4.341e-05
##
##                   Kappa : 0.6028
##  Mcnemar's Test P-Value : 0.3428
```

```
##
##             Sensitivity : 0.8889
##             Specificity : 0.7083
##          Pos Pred Value : 0.7742
##          Neg Pred Value : 0.8500
##              Prevalence : 0.5294
##          Detection Rate : 0.4706
##    Detection Prevalence : 0.6078
##       Balanced Accuracy : 0.7986
##
##        'Positive' Class : M
##
```

3.3. Para el clasificador C-SVM kernel lineal

Train control y training

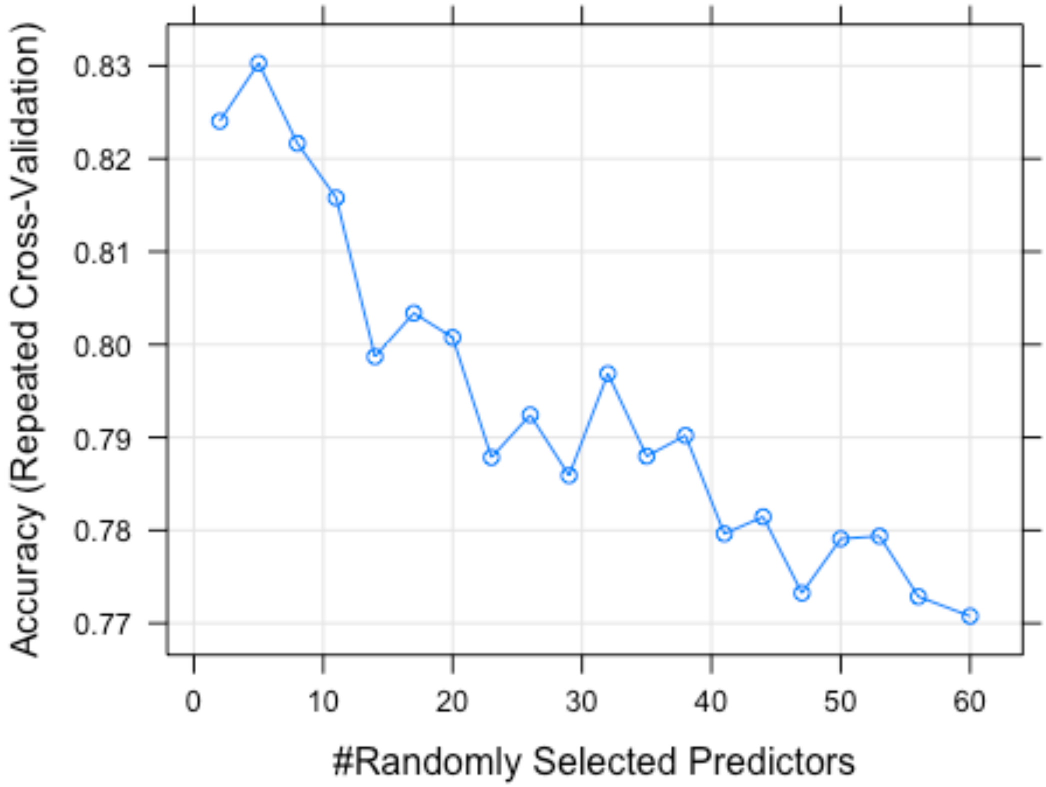
```r
library(class)
#install.packages("randomForest")
library(randomForest)

## randomForest 4.6-12

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

ctrl <- trainControl(method="repeatedcv",repeats = 3)
rfFit <- train(Class ~ ., data = training, method = 'rf', trControl =
ctrl, preProcess = c("center","scale"), tuneLength = 20)

rfFit

## Random Forest
##
## 157 samples
##  60 predictor
##   2 classes: 'M', 'R'
##
## Pre-processing: centered (60), scaled (60)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 141, 141, 141, 140, 142, 141, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8240359  0.6419663
##    5    0.8302859  0.6547825
```

```
##    8    0.8216585  0.6373165
##   11    0.8157925  0.6257464
##   14    0.7986928  0.5907531
##   17    0.8033824  0.5998898
##   20    0.8007598  0.5946896
##   23    0.7878431  0.5682022
##   26    0.7924265  0.5780824
##   29    0.7858987  0.5640818
##   32    0.7968709  0.5863526
##   35    0.7879820  0.5691455
##   38    0.7902042  0.5729258
##   41    0.7796324  0.5509802
##   44    0.7814542  0.5552291
##   47    0.7732435  0.5386946
##   50    0.7790931  0.5503694
##   53    0.7793709  0.5512255
##   56    0.7728595  0.5372546
##   60    0.7707598  0.5337022
##
## Accuracy was used to select the optimal model using  the largest
value.
## The final value used for the model was mtry = 5.
```

Accuracy = 0,82 Kappa = 0.64 Metry = 2

Plot

```
plot(rfFit)
```



###Prediccion y matriz de confusion

```
knnPredict <- predict(rfFit,newdata = testing )
#Obtenemos la matriz de confusion y vemos accuracy value y otros parámetros
confusionMatrix(knnPredict, testing$Class )

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  M  R
##          M 25  6
##          R  2 18
##
##                Accuracy : 0.8431
##                  95% CI : (0.7141, 0.9298)
##     No Information Rate : 0.5294
##     P-Value [Acc > NIR] : 2.534e-06
##
##                   Kappa : 0.6822
##  Mcnemar's Test P-Value : 0.2888
##
##             Sensitivity : 0.9259
##             Specificity : 0.7500
```

```
##          Pos Pred Value : 0.8065
##          Neg Pred Value : 0.9000
##              Prevalence : 0.5294
##          Detection Rate : 0.4902
##    Detection Prevalence : 0.6078
##       Balanced Accuracy : 0.8380
##
##        'Positive' Class : M
##
```