Assignment-4.3—Supervised-Classification-Models.R

arpan

2023-07-02

```r
#Use the attached "titanic.csv" data and do as follows in R Studio with R script:

#1. Read the titanic.csv data with base R function and save it as "data" and
# remove the name column and save again as data
setwd("/Users/arpan/Desktop/MDS/01 MDS I-I/MDS 503 - Statistical Computing with R/Class Lab/Data")
data <- read.csv("Arpan Sapkota - titanic.csv")
data <- data[, -3]  # Remove the name column
head(data)
```

```
##   Survived Pclass    Sex Age Siblings.Spouses.Aboard Parents.Children.Aboard
## 1        0      3   male  22                       1                       0
## 2        1      1 female  38                       1                       0
## 3        1      3 female  26                       0                       0
## 4        1      1 female  35                       1                       0
## 5        0      3   male  35                       0                       0
## 6        0      3   male  27                       0                       0
##      Fare
## 1  7.2500
## 2 71.2833
## 3  7.9250
## 4 53.1000
## 5  8.0500
## 6  8.4583
```

```r
str(data)
```

```
## 'data.frame':    887 obs. of  7 variables:
##  $ Survived               : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass                 : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Sex                    : chr  "male" "female" "female" "female" ...
##  $ Age                    : num  22 38 26 35 35 27 54 2 27 14 ...
##  $ Siblings.Spouses.Aboard: int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parents.Children.Aboard: int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Fare                   : num  7.25 71.28 7.92 53.1 8.05 ...
```

```r
#2. Fit binary logistic regression model with "Survived" variable as dependent
# variable and rest of variables as independent variables using "data",
# get summary of the model, check VIF and interpret the results carefully

#Converting factor
data$Survived <- as.factor(data$Survived)
```

```
data$Pclass <- as.factor(data$Pclass)
str(data$Pclass)
```

```
##  Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
```

```
data$Sex <- as.factor(data$Sex)
str(data$Sex)
```

```
##  Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
```

```
model.full <- glm(Survived~., data= data, family = binomial)
summary(model.full)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7773  -0.5991  -0.3984   0.6131   2.4412
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)              4.109777   0.463602   8.865  < 2e-16 ***
## Pclass2                 -1.161491   0.300960  -3.859 0.000114 ***
## Pclass3                 -2.350022   0.304666  -7.713 1.22e-14 ***
## Sexmale                 -2.756710   0.200642 -13.739  < 2e-16 ***
## Age                     -0.043410   0.007790  -5.573 2.51e-08 ***
## Siblings.Spouses.Aboard -0.401572   0.110795  -3.624 0.000290 ***
## Parents.Children.Aboard -0.106884   0.118767  -0.900 0.368151
## Fare                     0.002823   0.002468   1.144 0.252771
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1182.77  on 886  degrees of freedom
## Residual deviance:  780.93  on 879  degrees of freedom
## AIC: 796.93
##
## Number of Fisher Scoring iterations: 5
```

```
library(car)
```

```
## Loading required package: carData
```

```
vif(model.full)
```

```
##                             GVIF Df GVIF^(1/(2*Df))
## Pclass                  2.041787  2        1.195371
```

```
## Sex                      1.201233  1        1.096008
## Age                      1.477422  1        1.215492
## Siblings.Spouses.Aboard 1.290358  1        1.135939
## Parents.Children.Aboard 1.267656  1        1.125902
## Fare                     1.578965  1        1.256569
```

```
# The VIF values indicate that there is no severe multicollinearity among the
# predictor variables in the logistic regression model. The variables Pclass, Sex,
# Age, Siblings.Spouses.Aboard, Parents.Children.Aboard, and Fare have VIF values
# ranging from 1.167931 to 2.115788 These values suggest that there is little to
# moderate correlation between the predictor variables, indicating that they can
# be included in model without significant issues related to multicollinearity.

#3. Randomly split the data into 70% and 30% with replacement of samples as
#"train" and "test" data
set.seed(07)
ind <- sample(2,nrow(data), replace = T, prob = c(0.7,0.3))
train <- data[ind==1,]
test <- data[ind==2,]

#4. Fit binary logistic regression classifier, knn classifier, ann classifier,
# naive bayes classifier, svm classifier, decision tree classifier, decision
# tree bagging classifier, random forest classifier, tuned random forest
# classifier and random forest boosting classifier models using the "train" data

# Binary logistic regression classifier
model.full <- glm(Survived~., data= train, family = binomial)

# KNN classifier
library('caret')
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
model.knn <- train(Survived~., data = train, method = "knn")

# ANN classifie
library(nnet)
nn_model <- nnet(Survived~., data = train, size = 5, linear.output= TRUE)
```

```
## # weights:  46
## initial  value 598.698341
## iter  10 value 360.198491
## iter  20 value 288.557036
## iter  30 value 262.672486
## iter  40 value 249.650329
## iter  50 value 244.914836
## iter  60 value 241.691526
## iter  70 value 237.200412
## iter  80 value 235.132564
## iter  90 value 234.250296
```

```
## iter 100 value 234.110433
## final  value 234.110433
## stopped after 100 iterations
```

```
# Naive bayes classifier
library(e1071)
model.nb <- naiveBayes(Survived~., data = train)

# SVM classifier
library('e1071')
model.svm <- svm(formula = Survived~., data= train,
                 type= 'C-classification',
                 kernel = 'linear')

# Decision tree classifier
library(party)
```

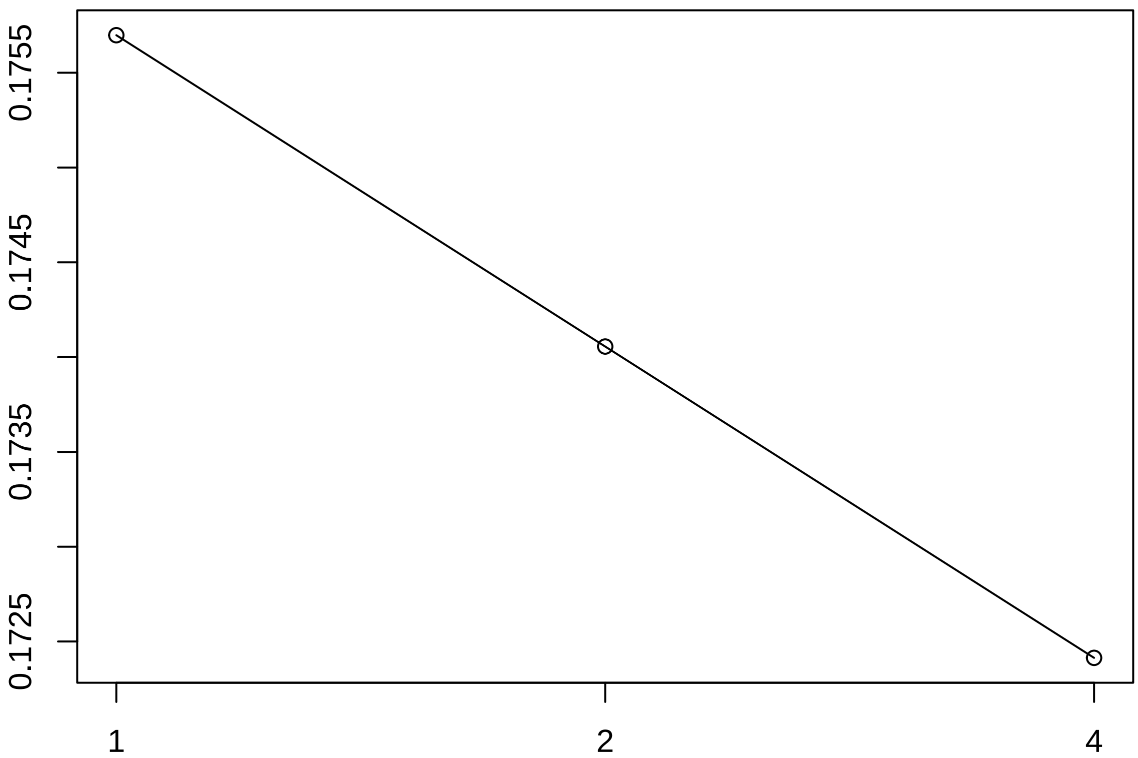
```
## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

##
## Attaching package: 'modeltools'

## The following object is masked from 'package:car':
##
##     Predict

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```
tree <- ctree(Survived~., data= train)

# Decision tree bagging classifier
library(ipred)
MBTree <- bagging(Survived~., data = train, coob= T)

# Random forest classifier
library(randomForest)
```

```
## randomForest 4.7-1.1

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

##
## Attaching package: 'randomForest'

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

```
set.seed(07)
rfm <- randomForest(Survived~., data= train)

par(mar = c(3, 3, 2, 2))

# Tuned random forest classifier
t <- tuneRF(train[,-1],train[,1],
            stepFactor = 0.5,
            plot = T,
            ntreeTry = 300,
            trace = T,
            improve = 0.05)
```

```
## mtry = 2  OOB error = 17.41%
## Searching left ...
## mtry = 4     OOB error = 17.24%
## 0.009433962 0.05
## Searching right ...
## mtry = 1     OOB error = 17.57%
## -0.009433962 0.05
```




```r
# Improve "rfm" model
rfm1 <- randomForest(Survived~., data= train, ntree= 300, mtry = 4,
                     improtance = T, proximity= T)

# Random forest boosting classifier models
library(caret)
mod.gbm <- train(Survived~., data= train, method = "gbm",verbose = F)


#5. Get confusion matrix and accuracy/misclassification error for all the
# classifier models and interpret them carefully

#Binary logistic regression classifier
predict.test <- predict(model.full,test, type= "response")
predicted.test <- factor(ifelse(predict.test>0.5,1,0))
reference.test <- factor(test$Survived)
(Bcm <- table(predicted.test, reference.test))
```

```
##                 reference.test
## predicted.test   0   1
##               0 155  30
##               1  29  64
```

```r
(Baccuracy <- sum(diag(Bcm))/sum(Bcm))
```

```
## [1] 0.7877698
```

```r
(Berror <- 1 -Baccuracy)
```

```
## [1] 0.2122302
```

```r
# KNN classifier
kpredict.test <- predict(model.knn, test)
(kcm <- table(kpredict.test, test$Survived))
```

```
##
## kpredict.test   0   1
##             0 143  46
##             1  41  48
```

```r
(kaccuracy <- sum(diag(kcm))/sum(kcm))
```

```
## [1] 0.6870504
```

```r
(kerror <- 1 -Baccuracy)
```

```
## [1] 0.2122302
```

```r
# ANN classifier
Apredict.test <- predict(nn_model, test)
Apredicted.test <- factor(ifelse(predict.test>0.5,1,0))
Areference.test <- factor(test$Survived)
(Acm <- table(Apredicted.test, Areference.test))
```

```
##                  Areference.test
## Apredicted.test   0   1
##               0 155  30
##               1  29  64
```

```r
(Aaccuracy <- sum(diag(Acm))/sum(Acm))
```

```
## [1] 0.7877698
```

```r
(Aerror <- 1 -Aaccuracy)
```

```
## [1] 0.2122302
```

```r
# Naive bayes classifier
Npredict.test <- predict(model.nb, test)
(Ncm <- table(Npredict.test, test$Survived))
```

```
##
## Npredict.test   0   1
##             0 160  33
##             1  24  61
```

```r
(Naccuracy <- sum(diag(Ncm))/sum(Ncm))
```

```
## [1] 0.794964
```

```r
(Nerror <- 1 -Naccuracy)
```

```
## [1] 0.205036
```

```r
# SVM classifier
Spredict.test <- predict(model.svm, test)
(Scm <- table(Spredict.test, test$Survived))
```

```
##
## Spredict.test   0   1
##             0 154  35
##             1  30  59
```

```r
(Saccuracy <- sum(diag(Scm))/sum(Scm))
```

```
## [1] 0.7661871
```

```r
(Serror <- 1 -Saccuracy)
```

```
## [1] 0.2338129
```

```r
# Decision tree
Dpredict.test <- predict(tree, test)
(Dcm <- table(Dpredict.test, test$Survived))
```

```
##
## Dpredict.test   0   1
##             0 153  32
##             1  31  62
```

```r
(Daccuracy <- sum(diag(Dcm))/sum(Dcm))
```

```
## [1] 0.7733813
```

```r
(Derror <- 1 -Daccuracy)
```

```
## [1] 0.2266187
```

```r
# Decision tree bagging classifier
Mpredict.test <- predict(MBTree, test)
(Mcm <- table(Mpredict.test, test$Survived))
```

```
##
## Mpredict.test   0   1
##             0 158  27
##             1  26  67
```

```r
(Maccuracy <- sum(diag(Mcm))/sum(Mcm))
```

```
## [1] 0.8093525
```

```r
(Merror <- 1 -Maccuracy)
```

```
## [1] 0.1906475
```

```r
# Random forest classifier
Rpredict.test <- predict(rfm, test)
(Rcm <- table(Rpredict.test, test$Survived))
```

```
##
## Rpredict.test   0   1
##             0 163  29
##             1  21  65
```

```r
(Raccuracy <- sum(diag(Rcm))/sum(Rcm))
```

```
## [1] 0.8201439
```

```r
(Rerror <- 1 - Raccuracy)
```

```
## [1] 0.1798561
```

```r
# Tuned random forest classifier
Rtpredict.test <- predict(rfm1, test)
(Rtcm <- table(Rtpredict.test, test$Survived))
```

```
##
## Rtpredict.test   0   1
##              0 154  27
##              1  30  67
```

```r
(Rtaccuracy <- sum(diag(Rtcm))/sum(Rtcm))
```

```
## [1] 0.794964
```

```r
(Rterror <- 1 - Rtaccuracy)
```

```
## [1] 0.205036
```

```r
# Random forest boosting classifier
Fpredict.test <- predict(mod.gbm, test)
(Fcm <- table(Fpredict.test, test$Survived))
```

```
##
## Fpredict.test   0   1
##             0 159  25
##             1  25  69
```

```r
(Faccuracy <- sum(diag(Fcm))/sum(Fcm))
```

```
## [1] 0.8201439
```

```r
(Ferror <- 1 - Faccuracy)
```

```
## [1] 0.1798561
```

```r
#Based on the accuracy values, the Random Forest model achieved the
# highest accuracy of approximately 0.8201439 However, comparing accuracy alone
# may not provide a comprehensive evaluation of the models.

#6. Get confusion matrix and accuracy/misclassification error for all the
# predicted models and interpret them carefully

# Binary logistic regression classifier
predict.test <- predict(model.full,test, type= "response")
predicted.test <- factor(ifelse(predict.test>0.5,1,0))
reference.test <- factor(test$Survived)
(Bcm <- table(predicted.test, reference.test))
```

```
##                 reference.test
## predicted.test   0   1
##              0 155  30
##              1  29  64
```

```r
confusionMatrix(Bcm)
```

```
## Confusion Matrix and Statistics
##
##                 reference.test
## predicted.test   0   1
##              0 155  30
##              1  29  64
##
##                Accuracy : 0.7878
##                  95% CI : (0.735, 0.8343)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 2.83e-06
##
##                   Kappa : 0.5246
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8424
##             Specificity : 0.6809
##          Pos Pred Value : 0.8378
##          Neg Pred Value : 0.6882
##              Prevalence : 0.6619
##          Detection Rate : 0.5576
##    Detection Prevalence : 0.6655
```

```
##       Balanced Accuracy : 0.7616
##
##        'Positive' Class : 0
##
```

```r
# KNN classifier
kpredict.test <- predict(model.knn, test)
(kcm <- table(kpredict.test, test$Survived))
```

```
##
## kpredict.test   0   1
##             0 143  45
##             1  41  49
```

```r
confusionMatrix(kcm)
```

```
## Confusion Matrix and Statistics
##
##
## kpredict.test   0   1
##             0 143  45
##             1  41  49
##
##                Accuracy : 0.6906
##                  95% CI : (0.6327, 0.7445)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 0.1710
##
##                   Kappa : 0.3016
##
##  Mcnemar's Test P-Value : 0.7463
##
##             Sensitivity : 0.7772
##             Specificity : 0.5213
##          Pos Pred Value : 0.7606
##          Neg Pred Value : 0.5444
##              Prevalence : 0.6619
##          Detection Rate : 0.5144
##    Detection Prevalence : 0.6763
##       Balanced Accuracy : 0.6492
##
##        'Positive' Class : 0
##
```

```r
# ANN classifier
Apredict.test <- predict(nn_model, test)
Apredicted.test <- factor(ifelse(predict.test>0.5,1,0))
Areference.test <- factor(test$Survived)
Acm <- table(Apredicted.test, Areference.test)
confusionMatrix(kcm)
```

```
## Confusion Matrix and Statistics
```

```
##
##
## kpredict.test   0   1
##             0 143  45
##             1  41  49
##
##                Accuracy : 0.6906
##                  95% CI : (0.6327, 0.7445)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 0.1710
##
##                   Kappa : 0.3016
##
##  Mcnemar's Test P-Value : 0.7463
##
##             Sensitivity : 0.7772
##             Specificity : 0.5213
##          Pos Pred Value : 0.7606
##          Neg Pred Value : 0.5444
##              Prevalence : 0.6619
##          Detection Rate : 0.5144
##    Detection Prevalence : 0.6763
##       Balanced Accuracy : 0.6492
##
##        'Positive' Class : 0
##
```

```
# Naive bayes classifier
Npredict.test <- predict(model.nb, test)
Ncm <- table(Npredict.test, test$Survived)
confusionMatrix(Ncm)
```

```
## Confusion Matrix and Statistics
##
##
## Npredict.test   0   1
##             0 160  33
##             1  24  61
##
##                Accuracy : 0.795
##                  95% CI : (0.7427, 0.8409)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 7.312e-07
##
##                   Kappa : 0.5309
##
##  Mcnemar's Test P-Value : 0.2893
##
##             Sensitivity : 0.8696
##             Specificity : 0.6489
##          Pos Pred Value : 0.8290
##          Neg Pred Value : 0.7176
##              Prevalence : 0.6619
##          Detection Rate : 0.5755
```

```
##    Detection Prevalence : 0.6942
##       Balanced Accuracy : 0.7593
##
##        'Positive' Class : 0
##
```

```r
# SVM classifier
Spredict.test <- predict(model.svm, test)
Scm <- table(Spredict.test, test$Survived)
confusionMatrix(Scm)
```

```
## Confusion Matrix and Statistics
##
##
## Spredict.test   0   1
##             0 154  35
##             1  30  59
##
##                Accuracy : 0.7662
##                  95% CI : (0.7119, 0.8147)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 0.0001007
##
##                   Kappa : 0.4707
##
##  Mcnemar's Test P-Value : 0.6197964
##
##             Sensitivity : 0.8370
##             Specificity : 0.6277
##          Pos Pred Value : 0.8148
##          Neg Pred Value : 0.6629
##              Prevalence : 0.6619
##          Detection Rate : 0.5540
##    Detection Prevalence : 0.6799
##       Balanced Accuracy : 0.7323
##
##        'Positive' Class : 0
##
```

```r
# Decision Tree
Dpredict.test <- predict(tree, test)
Dcm <- table(Dpredict.test, test$Survived)
confusionMatrix(Dcm)
```

```
## Confusion Matrix and Statistics
##
##
## Dpredict.test   0   1
##             0 153  32
##             1  31  62
##
##                Accuracy : 0.7734
##                  95% CI : (0.7196, 0.8212)
```

```
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 3.314e-05
##
##                   Kappa : 0.4924
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8315
##             Specificity : 0.6596
##          Pos Pred Value : 0.8270
##          Neg Pred Value : 0.6667
##              Prevalence : 0.6619
##          Detection Rate : 0.5504
##    Detection Prevalence : 0.6655
##       Balanced Accuracy : 0.7455
##
##        'Positive' Class : 0
##
```

```r
# Decision tree bagging classifier
Mpredict.test <- predict(MBTree, test)
Mcm <- table(Mpredict.test, test$Survived)
confusionMatrix(Mcm)
```

```
## Confusion Matrix and Statistics
##
##
## Mpredict.test   0   1
##             0 158  27
##             1  26  67
##
##                Accuracy : 0.8094
##                  95% CI : (0.7582, 0.8538)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 3.769e-08
##
##                   Kappa : 0.573
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8587
##             Specificity : 0.7128
##          Pos Pred Value : 0.8541
##          Neg Pred Value : 0.7204
##              Prevalence : 0.6619
##          Detection Rate : 0.5683
##    Detection Prevalence : 0.6655
##       Balanced Accuracy : 0.7857
##
##        'Positive' Class : 0
##
```

```r
# Random forest classifier
Rpredict.test <- predict(rfm, test)
Rcm <- table(Rpredict.test, test$Survived)
confusionMatrix(Rcm)
```

```
## Confusion Matrix and Statistics
##
##
## Rpredict.test   0   1
##             0 163  29
##             1  21  65
##
##                Accuracy : 0.8201
##                  95% CI : (0.7699, 0.8635)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 3.218e-09
##
##                   Kappa : 0.5896
##
##  Mcnemar's Test P-Value : 0.3222
##
##             Sensitivity : 0.8859
##             Specificity : 0.6915
##          Pos Pred Value : 0.8490
##          Neg Pred Value : 0.7558
##              Prevalence : 0.6619
##          Detection Rate : 0.5863
##    Detection Prevalence : 0.6906
##       Balanced Accuracy : 0.7887
##
##        'Positive' Class : 0
##
```

```r
# Tuned random forest classifier
Rtpredict.test <- predict(rfm1, test)
(Rtcm <- table(Rtpredict.test, test$Survived))
```

```
##
## Rtpredict.test   0   1
##              0 154  27
##              1  30  67
```

```r
confusionMatrix(Rtcm)
```

```
## Confusion Matrix and Statistics
##
##
## Rtpredict.test   0   1
##              0 154  27
##              1  30  67
##
##                Accuracy : 0.795
```

```
##                  95% CI : (0.7427, 0.8409)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 7.312e-07
##
##                   Kappa : 0.5455
##
##  Mcnemar's Test P-Value : 0.7911
##
##             Sensitivity : 0.8370
##             Specificity : 0.7128
##          Pos Pred Value : 0.8508
##          Neg Pred Value : 0.6907
##              Prevalence : 0.6619
##          Detection Rate : 0.5540
##    Detection Prevalence : 0.6511
##       Balanced Accuracy : 0.7749
##
##        'Positive' Class : 0
##
```

```
# Random forest boosting classifier
Fpredict.test <- predict(mod.gbm, test)
Fcm <- table(Fpredict.test, test$Survived)
confusionMatrix(Fcm)
```

```
## Confusion Matrix and Statistics
##
##
## Fpredict.test   0   1
##             0 159  25
##             1  25  69
##
##                Accuracy : 0.8201
##                  95% CI : (0.7699, 0.8635)
##     No Information Rate : 0.6619
##     P-Value [Acc > NIR] : 3.218e-09
##
##                   Kappa : 0.5982
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8641
##             Specificity : 0.7340
##          Pos Pred Value : 0.8641
##          Neg Pred Value : 0.7340
##              Prevalence : 0.6619
##          Detection Rate : 0.5719
##    Detection Prevalence : 0.6619
##       Balanced Accuracy : 0.7991
##
##        'Positive' Class : 0
##
```

```
#7. Compare accuracy and misclassification error of predicted models based on
# "test" data to decide the "best" model

# Comparing all the model, random forest boosting classifier have higher
# accuracy which is 0.8201 and less misclassification error

#8. Write a reflection on your own word focusing on "what did I learn from this assignment?"

# 1. Data Preparation: I learned the importance of handling missing values and
# converting variables into appropriate formats before building models.

# 2. Logistic Regression: I gained knowledge on fitting logistic regression models,
# interpreting coefficients, and checking multicollinearity using VIF.

# 3. Classifier Models: Explored various classifiers such as knn, ANN, Naive Bayes,
# SVM, Decision Trees, Bagging, Random Forests, Tuned RF, and RF Boosting.

# 4. Evaluation Metrics: Used confusion matrices and accuracy to assess the
# performance of classifier models.

# 5. Model Comparison: Compared the accuracy and misclassification error of
# different models to select the best one.

# 6. Iterative Process: Emphasized the iterative nature of model building,
# evaluation, and refinement.

# 7. Practical Skills: Acquired hands-on experience in data preparation, model
# fitting, interpretation, and evaluation.

# 8. Foundation for ML: Developed a strong foundation for further exploration in
# machine learning and predictive modeling.
```