Homework2

Jeong Seok Gyu

2022-11-02

```
Student information

- Student name : 정석규
- Major : 전기공학과
- Student id : 201724570
```

Table of Contents

Introduction

Open the data set OJ in the R package ISLR. THe data information is available with ?OJ. Let us begin with the following R commands

```
data(OJ)
y <- OJ[, 1]
x <- scale(OJ[, -c(1, 11:14, 17)])
```

A matrix x consists of n = 1,070 and p = 12, and a binary response y has either CH or MM. Let us regard CH as “negative” and MM as “positive”. Next randomly generate training, validation and tests samples using the following R commands.

```
set.seed(1111)
M <- sample(rep(c(-1, 0, 1), c(600, 370, 100)))
```

The vector M constists of 600 training samples (-1), 370 validation sample(0) and 100 test samples (1). In order to asses classification performance, consider 3 different scores which are accuracy(ACC), F1 score and Matthews correlation coefficieint(MCC). They are

$ACC = \frac{TP+TN}{TP+FP+TN+FN}$, $F_1 = \frac{2TP}{2TP+FP+FN}$

and $MCC = \frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$

respectively. Note that MCC = 0 if the denominator is equal to 0.

```r
# ==================== Introduction ====================
library(dplyr)
library(ISLR)
data(OJ)
y <- OJ[, 1]
x <- scale(OJ[, -c(1, 11:14, 17)])

set.seed(1111)
M <- sample(rep(c(-1, 0, 1), c(600, 370, 100)))

# Train-Validation-Test sets split
train <- M==-1; valid <- M==0; test <- M==1
# Factor matching of "CH", "MM" : 1, 2
# Our positive target of this problem is "MM"

# Functions of evaluation metrics : ACC, F1, MCC
calc.tp <- function(preds, actual) {
  res <- sum(preds[actual=='MM'] == 'MM')
  return(res)
}

calc.tn <- function(preds, actual) {
  res <- sum(preds[actual=='CH'] == 'CH')
  return(res)
}

calc.fp <- function(preds, actual) {
  res <- sum(preds[actual=='CH'] == 'MM')
  return(res)
}

calc.fn <- function(preds, actual) {
  res <- sum(preds[actual=='MM'] == 'CH')
  return(res)
}

cfx.res <- function(preds, actual) {
  tp <- calc.tp(preds, actual)
  tn <- calc.tn(preds, actual)
  fp <- calc.fp(preds, actual)
  fn <- calc.fn(preds, actual)
  return(list(tp, tn, fp, fn))
}

score.acc <- function(preds, actual) {
  cfx <- cfx.res(preds, actual)
  tp <- cfx[[1]]; tn <- cfx[[2]]; fp <- cfx[[3]]; fn <- cfx[[4]]
  res <- (tp + tn) / (tp + fp + tn + fn)
  return(res)
}

score.f1 <- function(preds, actual) {
  cfx <- cfx.res(preds, actual)
  tp <- cfx[[1]]; tn <- cfx[[2]]; fp <- cfx[[3]]; fn <- cfx[[4]]
  res <- (2 * tp) / ((2 * tp) + fp + fn)
  return(res)
}

score.mcc <- function(preds, actual) {
  cfx <- cfx.res(preds, actual)
  tp <- cfx[[1]]; tn <- cfx[[2]]; fp <- cfx[[3]]; fn <- cfx[[4]]
  if (sqrt((tp + fp)*(tp + fn)*(tn + fp)*(tn + fn)) == 0) {
    res <- 0
  } else {
    res <- ((tp * tn) - (fp * fn)) / sqrt((tp + fp)*(tp + fn)*(tn + fp)*(tn + fn))
  }
  return(res)
}
```

Problem 1

Apply a logistic regression(LR) for the training samples and then predict the class labels of validation samples, where the prediction probability of ‘y=MM’ $P(y = MM|x) > c$

indicates $\hat{y} = MM$; otherwise $\hat{y} = CH$. The threshold c starts from 0 to 1 increased by 0.001. Based on the validation samples, find 3 optimal thresholds $\hat{c_1}, \hat{c_2}, \hat{c_3}$ that maximize ACC, F_1, and MCC, respectively. If multiple thresholds have the same largest score, the optimal threshold should be the average of the multiple thresholds. Provide a single plot with 3 line respresenting ACC, F_1, and MCC, respectively. In the plot, the thresholds are on the x-axis and the scores are on the y-axis. Also, include the numerical values of $\hat{c_1}, \hat{c_2}, \hat{c_3}$.

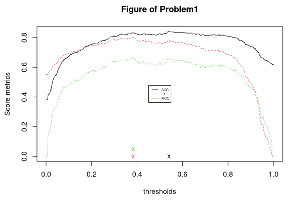
```r
# ==================== Problem 1 ====================
# The things need to consider : Logistic regression(LR), threshold <- (0, 1, 0.001)
# Required outputs :
#   1. Finding 3 optimal thresholds c1, c2, c3 that maximize ACC, F1, MCC
#   2. Provide a single plot with 3 lines representing ACC, F1 and MCC(including the numerical values of c1, c2,
 c3)
# Workflows of problem 1 :
#   1. Initialize thresholds and performance result matrix of length(thresholds) x 4
#       1.1 thresholds <- seq(0, 1, 0.001)
#       1.2 res <- matrix(NA, length(thresholds), 4)
#   2. Fitting model Logistic regression with training set.
#   3. Predict probability of validation set.
#   4. (Iterating in threshold) Predicting the class labels of the validation samples based on thresholds c.
#     4.1 Initialize yhat storing CH as negative
#     4.2 Convert CH into MM which value is higher than thresholds[i]
#     4.3 Calculate ACC, F1, MCC and store into res matrix
#   5. Extract 3 optimal thresholds c1, c2, c3 that maximize ACC, F1, MCC respectively.
#     5.1 Extract thresholds of maximum scores from metrics
#     5.2 Calculate the average of the multiple thresholds
#

# Make grids of thresholds and evaluation metrics
thresholds <- seq(0, 1, 0.001)
res <- matrix(NA, length(thresholds), 4)
res[, 1] <- thresholds
colnames(res) <- c('thresholds', 'ACC', 'F1', 'MCC')

# Training model Logistic Regressions
g1 <- glm(y ~ x, family="binomial", subset=train)
pred <- predict(g1, data.frame(x), type="response")[valid]

for (i in 1:length(thresholds)) {
  yhat <- rep("CH", length(pred))
  yhat[pred > thresholds[i]] <- "MM"
  res[i, 2] <- score.acc(yhat, y[valid])
  res[i, 3] <- score.f1(yhat, y[valid])
  res[i, 4] <- score.mcc(yhat, y[valid])
}

# Result matrix
res %>% head(3)
```

```
##      thresholds       ACC        F1 MCC
## [1,]      0.000 0.3810811 0.5518591   0
## [2,]      0.001 0.3810811 0.5518591   0
## [3,]      0.002 0.3810811 0.5518591   0
```

```r
# Find 3 optimal thresholds c1, c2, c3 that maximize ACC, F1, MCC respectively
c1 <- mean(res[which(res[, 2] == max(res[, 2])), 1])
c2 <- mean(res[which(res[, 3] == max(res[, 3])), 1])
c3 <- mean(res[which(res[, 4] == max(res[, 4])), 1])

# numerical values of c1, c2, c3
cbind(c1, c2, c3)
```

```
##          c1     c2     c3
## [1,] 0.5405 0.3835 0.3835
```

```r
# Visualization of problem1
matplot(x=res[, 1], y=res[, c(2:4)], type='l', pch=0.3, col=c(1:3),
        xlab="thresholds", ylab='Score metrics', main="Figure of Problem1")
legend("center", legend=c("ACC", "F1", "MCC"), col=c(1:3), lty=1:2, cex=0.5)
points(x=c1, y=0, pch="x", col=1)
points(x=c2, y=0, pch="x", col=2)
points(x=c3, y=0.05, pch="x", col=3)
```



Problem 2

With $\hat{c_1}, \hat{c_2}, \hat{c_3}$ obtained by Q1, find ACC, F_1 and MCC of the test samples, LR should be applied to compute ACC of the test samples with $\hat{c_1}$, F_1 score of the test samples with $\hat{c_2}$, and MCC of the test samples with $\hat{c_3}$.

```r
# ==================== Problem 2 ====================
# The things need to consider : c1, c2, c3 obtained by Q1, Logistic Regression should be applied.
# Required outputs :
#   1. Finding ACC, F1, MCC of the test samples
#   2. ACC of the test samples with c1, F1 score of the test samples with c2, MCC of the test samples with c3.
# Workflows of Problem2 :
#   1. Predict probability of test set.
#   2. Calculate ACC by c1 threshold, F1 by c2 threshold, MCC by c3 threshold.
#   3. Return result

pred <- predict(g1, data.frame(x), type="response")[test]

# ACC by c1 threshold
yhat <- rep("CH", length(pred))
yhat[pred > c1] <- "MM"
acc <- score.acc(yhat, y[test])

# F1 by c2 threshold
yhat <- rep("CH", length(pred))
yhat[pred > c2] <- "MM"
f1 <- score.f1(yhat, y[test])

# MCC by c3 threshold
yhat <- rep("CH", length(pred))
yhat[pred > c3] <- "MM"
mcc <- score.mcc(yhat, y[test])

cbind(acc, f1, mcc)
```

```
##       acc        f1       mcc
## [1,] 0.87 0.7901235 0.6688529
```

Problem 3

Repeat Q1 and Q2 with linear discriminant analysis(LDA), quadratic discriminant analysis(QDA), and naive Bayes(NB) classification methods. Note that the prediction probability is equivalent of the posterior probability of 3 methods. You don’t need to provide a line plot and the optimal thresholds here. For each classification method, just find the ACC, F_1 score and MCC of the test samples.

```r
# ==================== Problem 3 ====================
# The things need to consider : Repeat Q1 and Q2 with LDA, QDA, NB. Find the ACC, F1, MCC of test samples.
# Required outputs :
#   1. Repeat Q1 and Q2 with LDA, QDA, NB.
#   2. Find the ACC, F1, MCC of test samples.
# Workflows of Problem3 :
#   1. Importing library
#   2. Training model with training set applying LDA, QDA, NB.
#   3. Initialize 3 x 3 matrix, rows : (ACC, F1, MCC) and cols : (LDA, QDA, NB)
#   4. For each model, repeat Q1 and Q2.
#   5. Store result of (ACC, F1, MCC) in result matrix

# Importing library
library(MASS)
library(e1071)

# Training model with training set applying LDA, QDA, NB
g1 <- lda(y ~ x, subset=train)
g2 <- qda(y ~ x, subset=train)
g3 <- naiveBayes(x, y, subset=train)

thresholds <- seq(0, 1, 0.001)
# Initialize 3 x 3 matrix
model.err <- matrix(0, 3, 3)
rownames(model.err) <- c('ACC', 'F1', 'MCC'); colnames(model.err) <- c('LDA', 'QDA', 'NB')

# Repeat Q1 and Q2 for each model
for (k in 1:3) {
  # Part : Question1
  # Call model LDA, QDA, NB
  g <- get(paste("g", k, sep=""))

  # Evaluation matrix by thresholds by each model
  res <- matrix(NA, length(thresholds), 4)
  res[, 1] <- thresholds
  colnames(res) <- c('thresholds', 'ACC', 'F1', 'MCC')

  # Make prediction of validation set
  if (k==1 || k==2) {
    valid.pred <- predict(g, data.frame(x))$posterior[valid, 2]
  } else {
    valid.pred <- predict(g, data.frame(x), type="raw")[valid, 2]
  }
  for (i in 1:length(thresholds)) {
    yhat <- rep("CH", length(valid.pred))
    yhat[valid.pred > thresholds[i]] <- "MM"
    res[i, 2] <- score.acc(yhat, y[valid])
    res[i, 3] <- score.f1(yhat, y[valid])
    res[i, 4] <- score.mcc(yhat, y[valid])
  }

  # Find 3 optimal thresholds c1, c2, c3 that maximize ACC, F1, MCC respectively
  c1 <- mean(res[which(res[, 2] == max(res[, 2])), 1])
  c2 <- mean(res[which(res[, 3] == max(res[, 3])), 1])
  c3 <- mean(res[which(res[, 4] == max(res[, 4])), 1])

  # Part : Question2
  # Make prediction of test set
  if (k==1 || k==2) {
    test.pred <- predict(g, data.frame(x))$posterior[test, 2]
  } else {
    test.pred <- predict(g, data.frame(x), type="raw")[test, 2]
  }

  # ACC by c1 threshold
  yhat <- rep("CH", length(test.pred))
  yhat[test.pred > c1] <- "MM"
  model.err[1, k] <- score.acc(yhat, y[test])

  # F1 by c2 threshold
  yhat <- rep("CH", length(test.pred))
  yhat[test.pred > c2] <- "MM"
  model.err[2, k] <- score.f1(yhat, y[test])

  # MCC by c3 threshold
  yhat <- rep("CH", length(test.pred))
  yhat[test.pred > c3] <- "MM"
  model.err[3, k] <- score.mcc(yhat, y[test])
}
model.err
```

```
##           LDA       QDA        NB
## ACC 0.8500000 0.7400000 0.7700000
## F1  0.7804878 0.7042254 0.7073171
## MCC 0.6378677 0.5416138 0.4803845
```

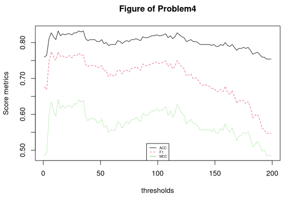
Problem 4

Repeat Q1 and Q2 with a K-nearest neighbor (KNN) classification methods, where K = 1, 3, 6, …, 197, 199. First, find the optimal K values that maximizes ACC, F_1 score and MCC of the validation samples respectively. If multiple K values have the same largest score, the optimal K should be the smallest one among them. Provide a single plot with 3 lines representing ACC, F_1 and MCC, respectively. In the plot, the values of K are on the x-axis and the scores are on the y-axis. Finally, find ACC, F_1 and MCC of the test samples, using the corresponding optimal thresholds.

```
# ===================== Problem 4 =====================
# The things need to consider :
#   1. Repeat Q1 and Q2 with KNN classification method.
#   2. Find the optimal K values that maximize metrics.
#   3. If multiple K values have the same largest score, the optimal K should be smallest one among them.
# Required outputs :
#   1. Provide a single plot with 3 lines
#   2. In the plot, the value of K are on the x-axis and the scores are on the y-axis.
#   3. find ACC, F1, MCC of the test samples, using the optimal thresholds
# Workflows of Problem4 :
#   1. Import library class
#   2. Set Hyper parameter grids : thresholds, K
#   3. Initialize Error matrix length(K) x 3.
#   4. Find the optimal K values that maximize ACC, F1 score and MCC of the validation samples.
#   5.

# Importing library for KNN
library(class)

# Hyper-parameter grids : thresholds, K
thresholds <- seq(0, 1, 0.001)
K <- seq(1, 199, 2)

# Initialize Error matrix 3 x length(K) matrix
model.err <- matrix(0, length(K), 3)
colnames(model.err) <- c('ACC', 'F1', 'MCC'); rownames(model.err) <- K

for (i in 1:length(K)) {
  # First, find the optimal K values that maximize ACC, F1 score and MCC of the validation samples, respectively.
  # Make prediction of validation set
  valid.preds <- knn(x[train,], x[valid,], y[train], k=K[i])
  model.err[i, 1] <- score.acc(valid.preds, y[valid])
  model.err[i, 2] <- score.f1(valid.preds, y[valid])
  model.err[i, 3] <- score.mcc(valid.preds, y[valid])
}

# Optimal K values that maximize ACC, F1 score, MCC, respectively.
wm.acc <- which.min(model.err[, 1])
wm.f1 <- which.min(model.err[, 2])
wm.mcc <- which.min(model.err[, 3])
cbind(wm.acc, wm.f1, wm.mcc)
```

```
##     wm.acc wm.f1 wm.mcc
## 195     98    98     98
```

```
# Visualization of problem1
matplot(x=K, y=model.err, type='l', pch=0.3, col=c(1:3),
        xlab="thresholds", ylab='Score metrics', main="Figure of Problem4")
legend("bottom", legend=c("ACC", "F1", "MCC"), col=c(1:3), lty=1:2, cex=0.5)
```



```
# Find ACC, F1, and MCC of the test samples
test.preds <- knn(x[train,], x[test,], y[train], k=98)
p4.res <- cbind(score.acc(test.preds, y[test]), score.f1(test.preds, y[test]), score.mcc(test.preds, y[test]))
colnames(p4.res) <- c('ACC', 'F1', 'MCC')
p4.res
```

```
##      ACC        F1       MCC
## [1,] 0.8 0.6969697 0.5507826
```

Problem 5

Next, randomly generate training, validation and test samples 100 times, using the following R commands.

```
set.seed(1234)
M <- rep(c(-1, 0, 1), c(600, 370, 100))
M <- apply(matrix(M, length(M), 100), 2, sample)
```

For each column of the matrix M, 1,070 samples consist of 600 training samples (-1), 370 validation samples(0) and 100 test samples(1). Since we have 100 different training, validation and test samples, you need to compute ACC, F_1 score and MCC of test sets 100 times. That is to say, you have to repeat Q1-Q4 for each set, where 5 classification methods such as LR, LDA, QDA, NB and KNN should be applied. Note that the optimal threshold or K can be determined using the validation set. For each method, find the average ACC, average F_1 score and average MCC of the test samples over 100 different sets. Summarize your answer using the following table.

(test)	LR	LDA	QDA	NB	KNN
ACC					
F_1					
MCC					

Which method is a winner?

```
# ===================== Problem 5 =====================
# The things need to consider : Repeat Q1-Q4 for each set, where 5 classification methods such as LR, LDA, QDA, N
B, and KNN.
# Required outputs : Find the average ACC, average F1 score and average MCC of the test samples over 100 differen
t sets.
# Workflows of Problem5 :
#   1. Initialize result matrix with 3 x 5. (rows for test metrics, cols for models)
#   2. Initialize five result matrix with 100 x 3 (This will store test score of LR, LDA, QDA, NB, KNN)
#   2. Repeat the steps 100 times
#     2.1 Indexes of training/valid/test sets is stored in M[, i]==-1, M[, i]==0, M[, i]==1
#     2.2 Repeat Q1 - Q4

# Randomly generate training, validation and test sample 100 times
set.seed(1234)
M <- rep(c(-1, 0, 1), c(600, 370, 100))
M <- apply(matrix(M, length(M), 100), 2, sample)

# Initialize 100 times test score matrix of each model
lr.score <- matrix(0, 100, 3)
lda.score <- matrix(0, 100, 3)
qda.score <- matrix(0, 100, 3)
nb.score <- matrix(0, 100, 3)
knn.score <- matrix(0, 100, 3)

# Initialize problem5 result matrix 3 x 5
p5.res <- matrix(0, 3, 5)
rownames(p5.res) <- c('ACC', 'F1', 'MCC'); colnames(p5.res) <- c('LR', 'LDA', 'QDA', 'NB', 'KNN')

for (i in 1:100){
  train <- M[,i]==-1; valid <- M[,i]==0; test <- M[,i]==1

  # Problem1 - 2
  p1.res <- matrix(NA, length(thresholds), 4)
  p1.res[, 1] <- thresholds
  colnames(res) <- c('thresholds', 'ACC', 'F1', 'MCC')

  g1 <- glm(y ~ x, family="binomial", subset=train)
  pred <- predict(g1, data.frame(x), type="response")[valid]

  for (j in 1:length(thresholds)) {
    yhat <- rep("CH", length(pred))
    yhat[pred > thresholds[j]] <- "MM"
    p1.res[j, 2] <- score.acc(yhat, y[valid])
    p1.res[j, 3] <- score.f1(yhat, y[valid])
    p1.res[j, 4] <- score.mcc(yhat, y[valid])
  }

  c1 <- mean(p1.res[which(p1.res[, 2] == max(p1.res[, 2])), 1])
  c2 <- mean(p1.res[which(p1.res[, 3] == max(p1.res[, 3])), 1])
  c3 <- mean(p1.res[which(p1.res[, 4] == max(p1.res[, 4])), 1])

  pred <- predict(g1, data.frame(x), type="response")[test]

  # ACC by c1 threshold
  yhat <- rep("CH", length(pred))
  yhat[pred > c1] <- "MM"
  lr.score[i, 1] <- score.acc(yhat, y[test])

  # F1 by c2 threshold
  yhat <- rep("CH", length(pred))
  yhat[pred > c2] <- "MM"
  lr.score[i, 2] <- score.f1(yhat, y[test])

  # MCC by c3 threshold
  yhat <- rep("CH", length(pred))
  yhat[pred > c3] <- "MM"
  lr.score[i, 3] <- score.mcc(yhat, y[test])

  # Problem3 : LDA, QDA, NB
  # Training model with training set applying LDA, QDA, NB
  g1 <- lda(y ~ x, subset=train)
  g2 <- qda(y ~ x, subset=train)
  g3 <- naiveBayes(x, y, subset=train)

  thresholds <- seq(0, 1, 0.001)
  # Initialize 3 x 3 matrix
  model.err <- matrix(0, 3, 3)
  rownames(model.err) <- c('ACC', 'F1', 'MCC'); colnames(model.err) <- c('LDA', 'QDA', 'NB')

  # Repeat Q1 and Q2 for each model
  for (k in 1:3) {
    # Part : Question1
    # Call model LDA, QDA, NB
    g <- get(paste("g", k, sep=""))

    # Evaluation matrix by thresholds by each model
    res <- matrix(NA, length(thresholds), 4)
    res[, 1] <- thresholds
    colnames(res) <- c('thresholds', 'ACC', 'F1', 'MCC')

    # Make prediction of validation set
    if (k==1 || k==2) {
      valid.pred <- predict(g, data.frame(x))$posterior[valid, 2]
    } else {
      valid.pred <- predict(g, data.frame(x), type="raw")[valid, 2]
    }
    for (l in 1:length(thresholds)) {
      yhat <- rep("CH", length(valid.pred))
      yhat[valid.pred > thresholds[l]] <- "MM"
      res[l, 2] <- score.acc(yhat, y[valid])
      res[l, 3] <- score.f1(yhat, y[valid])
      res[l, 4] <- score.mcc(yhat, y[valid])
    }

    # Find 3 optimal thresholds c1, c2, c3 that maximize ACC, F1, MCC respectively
    c1 <- mean(res[which(res[, 2] == max(res[, 2])), 1])
    c2 <- mean(res[which(res[, 3] == max(res[, 3])), 1])
    c3 <- mean(res[which(res[, 4] == max(res[, 4])), 1])

    # Part : Question2
    # Make prediction of test set
    if (k==1 || k==2) {
      test.pred <- predict(g, data.frame(x))$posterior[test, 2]
    } else {
      test.pred <- predict(g, data.frame(x), type="raw")[test, 2]
    }

    # ACC by c1 threshold
    yhat <- rep("CH", length(test.pred))
    yhat[test.pred > c1] <- "MM"
    model.err[1, k] <- score.acc(yhat, y[test])

    # F1 by c2 threshold
    yhat <- rep("CH", length(test.pred))
    yhat[test.pred > c2] <- "MM"
    model.err[2, k] <- score.f1(yhat, y[test])

    # MCC by c3 threshold
    yhat <- rep("CH", length(test.pred))
    yhat[test.pred > c3] <- "MM"
    model.err[3, k] <- score.mcc(yhat, y[test])
  }

  lda.score[i, ] <- model.err[, 1]
  qda.score[i, ] <- model.err[, 2]
  nb.score[i, ] <- model.err[, 3]

  # Problem4 : KNN
  # Initialize Error matrix 3 x length(K) matrix
  model.err <- matrix(0, length(K), 3)
  colnames(model.err) <- c('ACC', 'F1', 'MCC'); rownames(model.err) <- K

  for (m in 1:length(K)) {
    # First, find the optimal K values that maximize ACC, F1 score and MCC of the validation samples, respectivel
y.
    # Make prediction of validation set
    valid.preds <- knn(x[train,], x[valid,], y[train], k=K[m])
    model.err[m, 1] <- score.acc(valid.preds, y[valid])
    model.err[m, 2] <- score.f1(valid.preds, y[valid])
    model.err[m, 3] <- score.mcc(valid.preds, y[valid])
  }

  # Optimal K values that maximize ACC, F1 score, MCC, respectively.
  wm.acc <- which.min(model.err[, 1])
  wm.f1 <- which.min(model.err[, 2])
  wm.mcc <- which.min(model.err[, 3])

  test.preds1 <- knn(x[train,], x[test,], y[train], k=wm.acc)
  test.preds2 <- knn(x[train,], x[test,], y[train], k=wm.f1)
  test.preds3 <- knn(x[train,], x[test,], y[train], k=wm.mcc)

  knn.score[i, 1] <- score.acc(test.preds1, y[test])
  knn.score[i, 2] <- score.f1(test.preds2, y[test])
  knn.score[i, 3] <- score.mcc(test.preds3, y[test])
}

# Calculate average value of 100 times iteration and store into p5.res matrix
p5.res[, 1] <- apply(lr.score, 2, mean)
p5.res[, 2] <- apply(lda.score, 2, mean)
p5.res[, 3] <- apply(qda.score, 2, mean)
p5.res[, 4] <- apply(nb.score, 2, mean)
p5.res[, 5] <- apply(knn.score, 2, mean)

# View problem5 result matrix
p5.res
```

```
##            LR       LDA       QDA        NB       KNN
## ACC 0.8222000 0.8176000 0.7885000 0.7836000 0.7762000
## F1  0.7719514 0.7721626 0.7383222 0.7139288 0.6958412
## MCC 0.6263166 0.6204818 0.5544242 0.5327633 0.5301608
```