Math574M_Hw4

Saheed Adisa, Ganiyu

2023-10-15

loading required libraries

```r
library(MASS)  # for LDA
library(class)  # for k-NN
library(ggplot2)  # for ggplot2  plotting
```

5. Download the zip code data set from the book homepage. For both training and test sets, you need to unzip the files first and then load them into R.

```r
# Loading the dataset
train_data <- read.table("zip.train", sep = "")
test_data <- read.table("zip.test", sep = "")

# taking the subsets of the data with class Y in {0, 3, 5, 6, 9} for part (a)
class_labels_a <- c(0, 3, 5, 6, 9)
train_data_a <- train_data[train_data$V1 %in% class_labels_a, ]
test_data_a <- test_data[test_data$V1 %in% class_labels_a, ]
```

```r
# Function to calculate the error rate
calculate_error <- function(true_labels, predicted_labels) {
  return (mean(true_labels != predicted_labels))
}

# fitting LDA model
  lda_model <- lda(V1 ~ ., data = train_data_a)
  lda_train_pred <- predict(lda_model, train_data_a)$class
  lda_test_pred <- predict(lda_model, test_data_a)$class
  lda_train_error <- calculate_error(train_data_a$V1, lda_train_pred)
  lda_test_error <- calculate_error(test_data_a$V1, lda_test_pred)
  cat("=============Training and testing errors=============\n")
```

(a) Consider the five-class classification problems with $Y \in \{0, 3, 5, 6, 9\}$. Apply LDA and the knn with $k = 1, 3, 5, 7, 15$, respectively. Report the training and test errors for LDA and knn with each choice of k. Summarize your findings.

```
## =============Training and testing errors=============
```

```r
  cat(paste("LDA Train Error:", lda_train_error, "\t LDA Test Error:", lda_test_error, "\n"))
```

```
## LDA Train Error: 0.0223358449946179   LDA Test Error: 0.062015503875969
```

```r
# fitting k-NN classification for part (a)
k_values <- c(1, 3, 5, 7, 15)
for (k in k_values) {
  knn_train_pred <- knn(train_data_a[, -1], train_data_a[, -1], train_data_a$V1, k)
  knn_train_error <- calculate_error(train_data_a$V1, knn_train_pred)
  knn_test_pred <- knn(train_data_a[, -1], test_data_a[, -1], train_data_a$V1, k)
  knn_test_error <- calculate_error(test_data_a$V1, knn_test_pred)
  cat(paste("\n k =", k, "\t k-NN Train Error:", knn_train_error, "\t k-NN Test Error:", knn_test_error
}
```

```
##
##  k = 1    k-NN Train Error: 0     k-NN Test Error: 0.0300387596899225
##
##  k = 3    k-NN Train Error: 0.00780409041980624   k-NN Test Error: 0.0329457364341085
##
##  k = 5    k-NN Train Error: 0.0115715823466093    k-NN Test Error: 0.0329457364341085
##
##  k = 7    k-NN Train Error: 0.0134553283100108    k-NN Test Error: 0.0339147286821705
##
##  k = 15   k-NN Train Error: 0.0185683530678149    k-NN Test Error: 0.0406976744186047
```

Comment: The LDA model performs very well on both training and testing data with 2% and 6% error respectively and no overfitting involved, while KNN model also perform relatively well but, there was overfiting when $k = 1$ and the best k in this our result is 7 due to consideration in respect of model complexity, flexibility, and overfitting.

(b) Consider the 10-class classification problems with $Y \in \{0, 1, \cdots, 9\}$. Apply LDA and the knn with $k = 1, 3, 5, 7, 15$, respectively. Report the training and test errors for LDA and knn with each choice of k. Summarize your findings.

```r
# fitting LDA model
  lda_model <- lda(V1 ~ ., data = train_data)
  lda_train_pred <- predict(lda_model, train_data)$class
  lda_test_pred <- predict(lda_model, test_data)$class
  lda_train_error <- calculate_error(train_data$V1, lda_train_pred)
  lda_test_error <- calculate_error(test_data$V1, lda_test_pred)
  cat("================== Training and testing errors ==================\n")
```

```
## ================== Training and testing errors ==================
```

```r
  cat(paste("LDA Train Error:", lda_train_error, "\t LDA Test Error:", lda_test_error, "\n"))
```

```
## LDA Train Error: 0.0619942394733233   LDA Test Error: 0.114598903836572
```

```r
# fitting k-NN classification for part (a)
k_values <- c(1, 3, 5, 7, 15)
for (k in k_values) {
  knn_train_pred <- knn(train_data[, -1], train_data[, -1], train_data$V1, k)
  knn_train_error <- calculate_error(train_data$V1, knn_train_pred)
  knn_test_pred <- knn(train_data[, -1], test_data[, -1], train_data$V1, k)
  knn_test_error <- calculate_error(test_data$V1, knn_test_pred)
  cat(paste("\n k =", k, "\t k-NN Train Error:", knn_train_error, "\t k-NN Test Error:", knn_test_error
}
```

```
##
##  k = 1    k-NN Train Error: 0     k-NN Test Error: 0.0563029397110115
```

```
##
## k = 3    k-NN Train Error: 0.0128926073240982    k-NN Test Error: 0.0558046836073742
##
## k = 5    k-NN Train Error: 0.0211219311479907    k-NN Test Error: 0.0563029397110115
##
## k = 7    k-NN Train Error: 0.0257852146481964    k-NN Test Error: 0.0592924763328351
##
## k = 15   k-NN Train Error: 0.0370319572075161    k-NN Test Error: 0.0722471350274041
```

Comment: The LDA model performs very well on both training and testing data with 6% and 11% error respectively and no overfitting involved, while KNN model also perform relatively well but, there was overfiting when $k = 1$ and the best k in this our result is 7 due to consideration in respect of model complexity, flexibility, and overfitting.

6 Cross validation can be used to evaluate a model's generalization performance. Consider a binary classification problem. We will train LDA and compute its 5-fold CV error.

(a) In R, write a function LDA5cv (), which takes the training data D as the input, randomly partitions D into five folds of (roughly) equal size, implements 5 -fold CV for LDA, and calculate 5 -fold CV error as the output.

```r
LDA5cv <- function(data, num_folds = 5, seed = 2024) {
  set.seed(seed)

  # Shuffle the data randomly
  data <- data[sample(nrow(data)), ]

  # Split the data into 5 roughly equal folds
  fold_size <- floor(nrow(data) / num_folds)
  folds <- split(data, rep(1:num_folds, each = fold_size, length.out = nrow(data)))

  cv_errors <- numeric(num_folds)

  for (i in 1:num_folds) {
    # Create training and validation sets
    validation_data <- folds[[i]]
    training_data <- do.call(rbind, folds[-i])

    # Fit LDA model on training data
    lda_model <- lda(Class ~ ., data = training_data)

    # Make predictions on validation data
    lda_pred <- predict(lda_model, validation_data)$class

    # Calculate classification error on validation set
    cv_errors[i] <- mean(lda_pred != validation_data$Class)
  }

  # Calculate the mean cross-validation error
  mean_cv_error <- mean(cv_errors)
```

```
  return(mean_cv_error)
}
```

(b) Consider the training data set generated under Scenario 1 in Problem 6 of HW2.

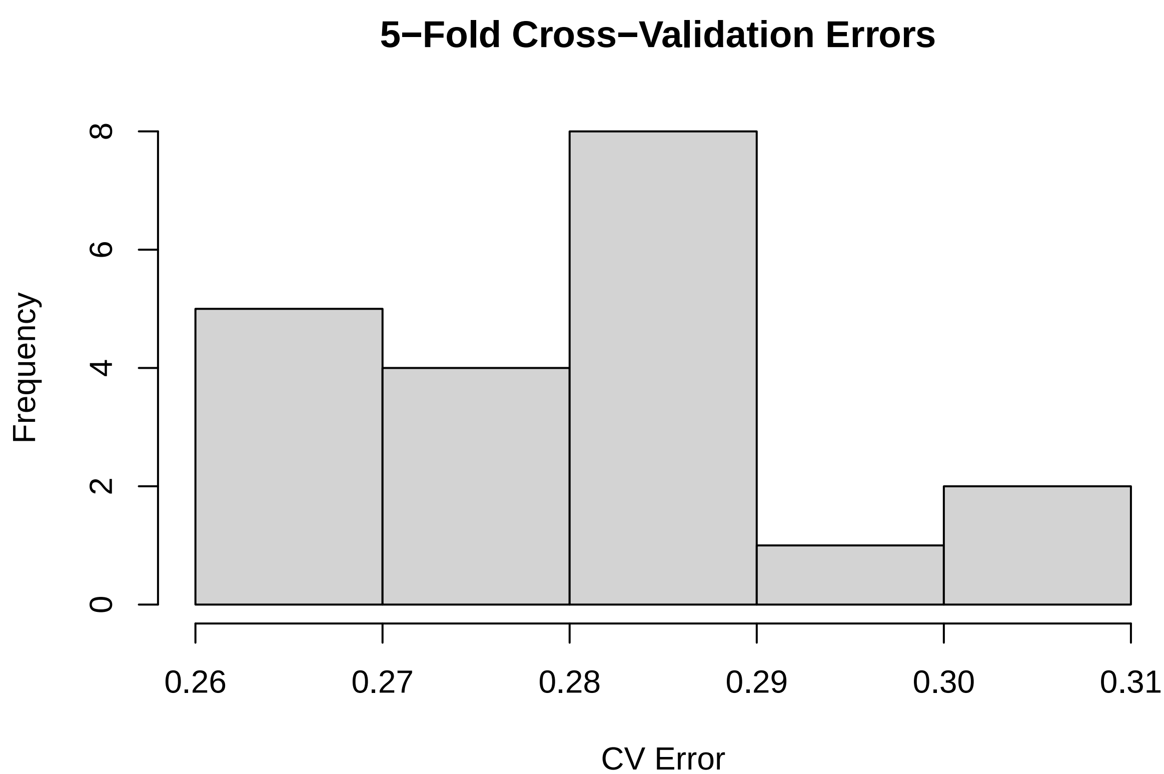
Specify a random seed. Call your function LDA5cv () and report the 5-fold CV error on the training data.

```
#______________Loading scenario 1 data from problem 6 of HW2___________________
### (a) Write $\mathrm{R}$ code to generate the training data.
### Set the seed with set.seed(2023) before calling the random number generation function.
set.seed(2023)
# setting mean and covariance
n1 = 100
mean1 <- c(2,1)
mean2 <- c(1,2)
cov1 = matrix(c(1,0,0,1),nrow=2)
cov2 = matrix(c(1,0,0,1),nrow=2)
green_train = mvrnorm(100,mean1,cov1)     # generating a random sample of 50 data points from a multiva
class1_labels <- rep("Green", n1)
red_train = mvrnorm(n1,mean2,cov2)
class2_labels <- rep("Red", n1)

# Combining both green and red data, then add class labels
training <- rbind(data.frame(green_train, Class = class1_labels),
                       data.frame(red_train, Class = class2_labels))

# Generate a test set, with 500 observations from each class, using set.seed(2024). Save the data set f
set.seed(2024)
# Generate test data
n2 <- 500  # Number of observations per class
green_test <- mvrnorm(n2, mean1, cov1)
class1_test_labels <- rep("Green", n2)
red_test <- mvrnorm(n2, mean2, cov2)
class2_test_labels <- rep("Red", n2)

# Combine data and labels for the test set
testing <- rbind(data.frame(green_test, Class = class1_test_labels),
                    data.frame(red_test, Class = class2_test_labels))

data <- training
data$Class <- as.factor(data$Class)

# Perform 5-fold cross-validation for LDA
cv_error <- LDA5cv(data)
cat("5-Fold CV Error:", cv_error, "\n")
```

```
## 5-Fold CV Error: 0.28
```

(c) Repeat part (b) 20 times using different random seeds and the same dataset. Draw a histogram of these 5 -fold CV errors.

```r
num_repetitions <- 20
cv_errors <- numeric(num_repetitions)

for (i in 1:num_repetitions) {
  seed <- sample(1:1000, 1)
  cv_errors[i] <- LDA5cv(data, seed = seed)
}

# Draw a histogram of the CV errors
hist(cv_errors, main = "5-Fold Cross-Validation Errors ", xlab = "CV Error")
```



(d) Compute the test error for LDA. Compare it with the results in (c).

```r
# Fit  LDA model, and make predictions on testing data
test_data <- testing
lda_model <- lda(Class ~ X1 + X2, data = data)
testing_pred_lda <- predict(lda_model, test_data)$class
lda_testing_error <- mean(testing_pred_lda != test_data$Class)     # Calculate testing error
cat("LDA Testing Error:", lda_testing_error, "\n")
```

```
## LDA Testing Error: 0.238
```

Comment: We notice that the CV error is ranging between 26% and 31% with high frequency within range 28% and 29% CV Error, while the testing error is 23.8% and falls outside the CV error range.

7. Cross validation can be used to compare two models' prediction performance. Consider a binary classification problem. We will train LDA and logistic regression and compare them using 5-fold CV errors.

(a) In R, write a function logLDA5cv(), which takes the training data D as the input, randomly partitions D into five folds of (roughly) equal size, implements 5 -fold CV for LDA and Logistic Regression, and calculate their 5-fold CV errors as the output.

```r
logLDA5cv <- function(data, num_folds = 5, seed = 2024) {
  set.seed(seed)

  # Shuffle the data randomly
  data <- data[sample(nrow(data)), ]

  # Split the data into 5 roughly equal folds
  fold_size <- floor(nrow(data) / num_folds)
  folds <- split(data, rep(1:num_folds, each = fold_size, length.out = nrow(data)))

  cv_errors_lda <- numeric(num_folds)
  cv_errors_logistic <- numeric(num_folds)

  for (i in 1:num_folds) {
    # create training and validation sets
    validation_data <- folds[[i]]
    training_data <- do.call(rbind, folds[-i])

    # fit LDA model and Logistic Regression on training data
    lda_model <- lda(Class ~ ., data = training_data)
    logistic_model <- glm(Class ~ ., data = training_data, family = binomial(link = "logit"))

    # predictions  and classification errors on validation data for both models
    lda_pred <- predict(lda_model, validation_data)$class
    logistic_training_probabilities <- predict(logistic_model, type = "response", newdata = validation_
    logistic_pred <- ifelse(logistic_training_probabilities > 0.5, "Red", "Green")
    cv_errors_lda[i] <- mean(lda_pred != validation_data$Class)
    cv_errors_logistic[i] <- mean(logistic_pred != validation_data$Class)

  }

  # computing the mean cross-validation errors for both models
  mean_cv_error_lda <- mean(cv_errors_lda)
  mean_cv_error_logistic <- mean(cv_errors_logistic)

  return(list(LDA = mean_cv_error_lda, Logistic = mean_cv_error_logistic))
}
```

(b) Consider the training data set generated under Scenario 1 in Problem 6 of HW2.

Specify a random seed. Call your function logLDA5cv() **and**

report two 5-fold CV errors. Comment on your findings.

```
# Perform 5-fold cross-validation for both LDA and Logistic Regression
cv_errors <- logLDA5cv(data)

cat("5-Fold CV Error for LDA:", cv_errors$LDA, "\n")
```

```
## 5-Fold CV Error for LDA: 0.28
```

```
cat("5-Fold CV Error for Logistic Regression:", cv_errors$Logistic, "\n")
```

```
## 5-Fold CV Error for Logistic Regression: 0.27
```

Comment: We observed that Logistic regression slightly perform better than LDA with 27% where LDA gives 28%

8. Cross validation can be used to select the optimal tuning parameter for a learner.

Consider a binary classification problem with a training data D of size n.

We will select the best k for the nearest neighbor classifier knn using 5-fold CV.

(a) In R, write a function KNN 5cv()**, which takes**

the training data D and a candidate set of $k = \{1, 2, \cdots, n\}$ as the input,

randomly partitions the training data into five folds, implements 5-fold CV for each k,

and report the 5-fold CV errors and their standard errors as the output.

```
KNN5cv <- function(data, k_values, num_folds = 5, seed = 123) {
  set.seed(seed)
  data <- data[sample(nrow(data)), ]  # Shuffle the data randomly

  # Split the data into 5 roughly equal folds
  fold_size <- floor(nrow(data) / num_folds)
  folds <- split(data, rep(1:num_folds, each = fold_size, length.out = nrow(data)))
  cv_errors <- matrix(0, nrow = length(k_values), ncol = num_folds)

  for (i in 1:num_folds) {
    # Create training and validation sets
    validation_data <- folds[[i]]
    training_data <- do.call(rbind, folds[-i])

    for (j in 1:length(k_values)) {
      k <- k_values[j]

      # Fit k-NN model on training data for the current k
      knn_pred <- knn(training_data[, -3], validation_data[, -3], training_data[, "Class"], k)
      cv_errors[j, i] <- mean(knn_pred != validation_data$Class)
    }
  }
```

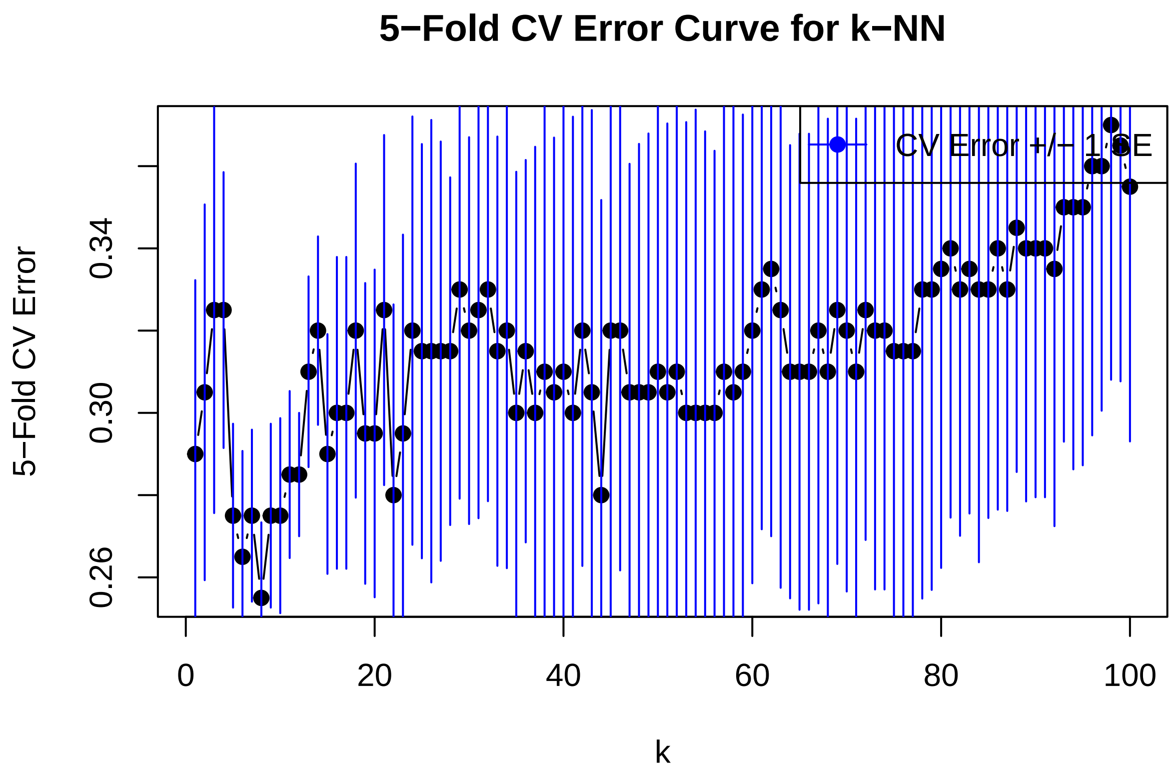
```r
  # Calculate the mean cross-validation errors and their standard errors for each k
  mean_cv_errors <- rowMeans(cv_errors)
  std_errors <- apply(cv_errors, 1, sd) / sqrt(num_folds)

  return(list(k_values = k_values, mean_cv_errors = mean_cv_errors, std_errors = std_errors))
}
```

(b) Consider the traing data generated under Scenario 2 in Problem 6 of HW3. Specify a random seed. Call your function KNN5cv (). Plot the 5 -fold CV error curve, where x-axis is $\{1, 2, \cdots, n\}$, and add one standard-error bar.

generating dataset under Scenario 2 in Problem 6 of HW3.

```r
# generate ten centers, which are treated as fixed parameters
Sig <- matrix(c(1,0,0,1),nrow=2)
seed_center <- 16
set.seed(seed_center)
center_green <- mvrnorm(n=10,c(1,0),Sig)
center_red <- mvrnorm(n=10,c(0,1),Sig)

# define a function "gendata2" first
gendata2 <-function(n,mu1,mu2,Sig1,Sig2,myseed)
{
set.seed(myseed)
mean1 <- mu1[sample(1:10,n,replace=T),]
mean2 <- mu2[sample(1:10,n,replace=T),]
green <- matrix(0,ncol=2,nrow=n)
red <- matrix(0,ncol=2,nrow=n)
for(i in 1:n){
green[i,] <- mvrnorm(1,mean1[i,],Sig1)
red[i,] <- mvrnorm(1,mean2[i,],Sig2)
}
x <- rbind(green,red)
return(x)
}
set.seed(2013)
# generate the training set
seed_train <- 2000
ntrain <- 100
train2 <- gendata2(ntrain,center_green,center_red,Sig/5,Sig/5,seed_train)
ytrain <- c(rep(1,ntrain),rep(0,ntrain))
training2 <- data.frame(X1 = train2[, 1], X2 = train2[, 2], Class = factor(ytrain))

set.seed(2014)
# Generate the test set
seed_test <- 2014
ntest <- 500
test2 <- gendata2(ntest, center_green, center_red, Sig / 5, Sig / 5, seed_test)
ytest <- c(rep(1, ntest), rep(0, ntest))
testing2 <- data.frame(X1 = test2[, 1], X2 = test2[, 2], Class = factor(ytest) )

data2 <- training2
```

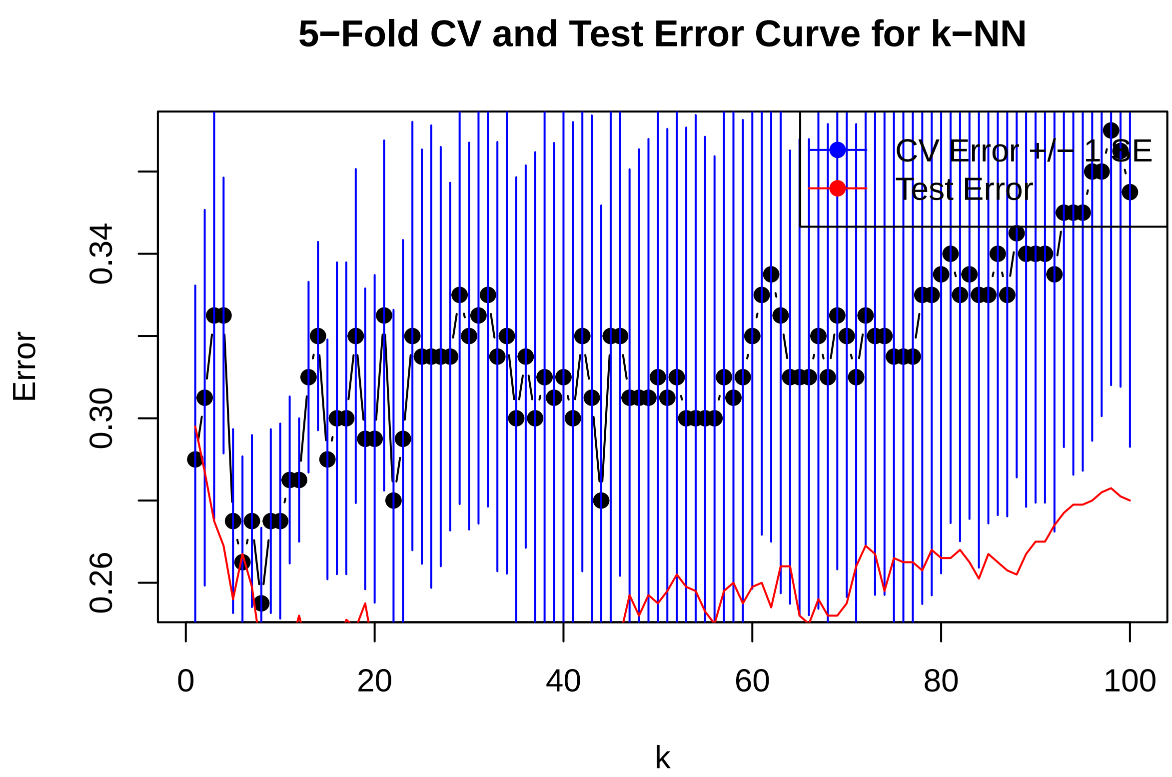
```r
# Define a range of k values to test
k_values <- 1:100

# Perform 5-fold cross-validation for different k values
cv_results <- KNN5cv(data2, k_values)

# Plot the 5-fold CV error curve with standard error bars
plot(cv_results$k_values, cv_results$mean_cv_errors, type = "b", pch = 19, xlab = "k", ylab = "5-Fold C
     main = "5-Fold CV Error Curve for k-NN")
segments(cv_results$k_values, cv_results$mean_cv_errors - cv_results$std_errors,
         cv_results$k_values, cv_results$mean_cv_errors + cv_results$std_errors, col = "blue")
legend("topright", legend = "CV Error +/- 1 SE", col = "blue", lty = 1, pch = 19)
```



(c) Compute the test error for each k, and add the test error curve on the plot (b). Use different color and add the legend. Summarize your observation.

```r
KNN5cv <- function(train_data, k_values, test_data, num_folds = 5, seed = 2024) {
  test_errors <- numeric(length(k_values))
  for (j in 1:length(k_values)) {
    k <- k_values[j]

    # Fit k-NN model
    knn_model <- knn(train_data[, -3], test_data[, -3], train_data[, 3], k)
    test_errors[j] <- mean(knn_model != test_data$Class)
  }
```

```
  return(list(k_values = k_values, mean_cv_errors = cv_results$mean_cv_errors, std_errors = cv_results$s
}

test_data <- testing2
cv_and_test_results <- KNN5cv(data2, k_values, test_data)

# Plot the 5-fold CV error curve with standard error bars and add the test error curve
plot(cv_and_test_results$k_values, cv_and_test_results$mean_cv_errors, type = "b", pch = 19, xlab = "k"
     main = "5-Fold CV and Test Error Curve for k-NN")
segments(cv_and_test_results$k_values, cv_and_test_results$mean_cv_errors - cv_and_test_results$std_erro
         cv_and_test_results$k_values, cv_and_test_results$mean_cv_errors + cv_and_test_results$std_erro
lines(cv_and_test_results$k_values, cv_and_test_results$test_errors, col = "red")
legend("topright", legend = c("CV Error +/- 1 SE", "Test Error"), col = c("blue", "red"), lty = c(1, 1)
```



(d) Report the best k chosen by the 5 -fold CV.

```
# Find the best k based on the minimum cross-validation error
best_k_index <- which.min(cv_and_test_results$mean_cv_errors)
best_k <- cv_and_test_results$k_values[best_k_index]
cat("Best k chosen by 5-Fold CV:", best_k, "\n")
```

```
## Best k chosen by 5-Fold CV: 8
```

Comment: From the above result, we observed that the lowest error is 25.5% at $k = 8$ which is reasonably good in respect to model complexity, and overfitting. Also , the test error curve gives the expected U-shape.