Diabetes Prediction Analysis in Pima Indian Female Patients

Sydney Murphy

2023-11-27

Load the data contained in the diabetes.csv file in R.

```
#install.packages("caret")
#install.packages("MASS")
#install.packages("e1071")
#install.packages("lattice")
#install.packages("ggplot2")
#install.packages("naivebayes")

library(MASS)
library(e1071)
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(ggplot2)
library(lattice)

# Load the data from the CSV file
diabetes_data <-
read.csv("https://raw.githubusercontent.com/sydneymcolumbia/CMU/main/diabetes
.csv")

names(diabetes_data)

## [1] "Pregnancies"              "Glucose"
## [3] "BloodPressure"            "SkinThickness"
## [5] "Insulin"                  "BMI"
## [7] "DiabetesPedigreeFunction" "Age"
## [9] "Outcome"
```

Replicate the logic used in the class7.r file to divide the data in a train, validation and test set. Use a 40% - 30% - 30% split.

```
set.seed(123)


# 40% of data
is_train <- as.logical(rbinom(nrow(diabetes_data), 1, 0.4))
diabetes_train <- diabetes_data[is_train, ]
```

```r
# 30% of total data
is_validation <- as.logical(rbinom(nrow(diabetes_data), 1, 0.5) * !is_train)
diabetes_validation <- diabetes_data[is_validation, ]

# 30% of total data
diabetes_test <- diabetes_data[!(is_train | is_validation), ]

dim(diabetes_train)
```

```
## [1] 300   9
```

```r
dim(diabetes_validation)
```

```
## [1] 225   9
```

```r
dim(diabetes_test)
```

```
## [1] 243   9
```

```r
set.seed(123)

# 40% of the data will go in "train"
is_train <- as.logical(rbinom(nrow(diabetes_data), 1, 0.4))
diabetes_train <- diabetes_data[is_train, ]

# 50% of the remaining 60% will go in "validation" (30% of the total data)
is_validation <- as.logical(rbinom(nrow(diabetes_data), 1, 0.5) * !is_train)
diabetes_validation <- diabetes_data[is_validation, ]

# The remaining data will go in "test" (30% of the total data).
diabetes_test <- diabetes_data[!(is_train | is_validation), ]

# Print the dimensions of each set to confirm the splits
dim(diabetes_train)
```

```
## [1] 300   9
```

```r
dim(diabetes_validation)
```

```
## [1] 225   9
```

```r
dim(diabetes_test)
```

```
## [1] 243   9
```

```r
# Checking the balance of the Outcome variable
table(diabetes_train$Outcome)
```

```
##
##   0   1
## 189 111
```

```r
# Assuming "Outcome" is originally numeric
diabetes_train$Outcome <- as.factor(diabetes_train$Outcome)

# Set the levels of the factor to "0" and "1"
levels(diabetes_train$Outcome) <- c("0", "1")

# Confirm the structure of the Outcome variable
str(diabetes_train$Outcome)
```

```
##  Factor w/ 2 levels "0","1": 1 1 2 1 1 1 2 2 1 1 ...
```

```r
# Fit a logistic regression model
logistic_model <- glm(Outcome ~ ., data = diabetes_train, family =
binomial(link = "logit"))

# Define the training control with cross-validation
train_control <- trainControl(
  method = "cv",
  number = 10,
  verboseIter = FALSE
)

# Fit a logistic regression classifier
logistic_model_train <- train(
  diabetes_train$Outcome ~ .,
  data = diabetes_train,
  method = "glm",
  trControl = train_control
)
```

```
## Error in `[.data.frame`(data, , all.vars(Terms), drop = FALSE): undefined
columns selected
```

```r
# Fit an LDA classifier
lda_model <- train(
  diabetes_train$Outcome ~ .,
  data = diabetes_train,
  method = "lda",
  trControl = train_control
)
```

```
## Error in `[.data.frame`(data, , all.vars(Terms), drop = FALSE): undefined
columns selected
```

```r
# Fit a QDA classifier
qda_model <- train(
  diabetes_train$Outcome ~ .,
  data = diabetes_train,
  method = "qda",
  trControl = train_control
)
```

```
## Error in `[.data.frame`(data, , all.vars(Terms), drop = FALSE): undefined
columns selected
```

```
# Fit a Naive Bayes classifier
nb_model <- train(
  diabetes_train$Outcome ~ .,
  data = diabetes_train,
  method = "naive_bayes",
  trControl = train_control
)
```

```
## Error in `[.data.frame`(data, , all.vars(Terms), drop = FALSE): undefined
columns selected
```

```
# View the results
print(logistic_model_train)
```

```
## Error in eval(expr, envir, enclos): object 'logistic_model_train' not
found
```

```
print(lda_model)
```

```
## Error in eval(expr, envir, enclos): object 'lda_model' not found
```

```
print(qda_model)
```

```
## Error in eval(expr, envir, enclos): object 'qda_model' not found
```

```
print(nb_model)
```

```
## Error in eval(expr, envir, enclos): object 'nb_model' not found
```

Using all available predictors, fit to the training set: • a classifier based on logistic regression • an LDA classifier • a QDA classifier • a Naive Bayes classifier

```
# Convert the Outcome variable in the training dataset to a factor
diabetes_train$Outcome <- factor(diabetes_train$Outcome, levels = c(0, 1))

# Logistic Regression (Outcome as dependent variable)
logistic_model <- glm(Outcome ~ ., data = diabetes_train, family =
binomial(link = "logit"))

# Linear Discriminant Analysis (LDA)
lda_model <- lda(Outcome ~ ., data = diabetes_train)

# Quadratic Discriminant Analysis (QDA)
qda_model <- qda(Outcome ~ ., data = diabetes_train)

# Naive Bayes Classifier
naive_bayes_model <- naiveBayes(Outcome ~ ., data = diabetes_train)
```

```r
summary(logistic_model)
```

```
##
## Call:
## glm(formula = Outcome ~ ., family = binomial(link = "logit"),
##     data = diabetes_train)
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -7.650674   1.087449  -7.035 1.99e-12 ***
## Pregnancies               0.158427   0.047683   3.322 0.000892 ***
## Glucose                   0.032132   0.005599   5.739 9.54e-09 ***
## BloodPressure            -0.010009   0.007158  -1.398 0.162000
## SkinThickness             0.006850   0.010916   0.627 0.530338
## Insulin                  -0.002056   0.001418  -1.450 0.147053
## BMI                       0.082317   0.022425   3.671 0.000242 ***
## DiabetesPedigreeFunction  0.680474   0.433384   1.570 0.116383
## Age                       0.003002   0.014153   0.212 0.832005
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 395.37  on 299  degrees of freedom
## Residual deviance: 297.81  on 291  degrees of freedom
## AIC: 315.81
##
## Number of Fisher Scoring iterations: 5
```

```r
lda_model
```

```
## Call:
## lda(Outcome ~ ., data = diabetes_train)
##
## Prior probabilities of groups:
##    0    1
## 0.63 0.37
##
## Group means:
##   Pregnancies  Glucose BloodPressure SkinThickness  Insulin      BMI
## 0    3.169312 112.4550      67.63492      19.35450 74.79894 30.46772
## 1    4.855856 140.9279      69.14414      21.52252 89.42342 35.36937
##   DiabetesPedigreeFunction      Age
## 0                0.4430635 31.91534
## 1                0.5684054 36.41441
##
## Coefficients of linear discriminants:
##                                   LD1
## Pregnancies               0.131308199
```

```
## Glucose                   0.025734117
## BloodPressure            -0.008724401
## SkinThickness             0.005417081
## Insulin                  -0.001571733
## BMI                       0.058367131
## DiabetesPedigreeFunction  0.512222529
## Age                       0.002809753
```

```
qda_model
```

```
## Call:
## qda(Outcome ~ ., data = diabetes_train)
##
## Prior probabilities of groups:
##    0    1
## 0.63 0.37
##
## Group means:
##   Pregnancies  Glucose BloodPressure SkinThickness  Insulin      BMI
## 0    3.169312 112.4550      67.63492      19.35450 74.79894 30.46772
## 1    4.855856 140.9279      69.14414      21.52252 89.42342 35.36937
##   DiabetesPedigreeFunction      Age
## 0                0.4430635 31.91534
## 1                0.5684054 36.41441
```

```
naive_bayes_model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##    0    1
## 0.63 0.37
##
## Conditional probabilities:
##    Pregnancies
## Y       [,1]     [,2]
##   0 3.169312 2.970226
##   1 4.855856 3.958528
##
##    Glucose
## Y       [,1]     [,2]
##   0 112.4550 27.14744
##   1 140.9279 32.40446
##
##    BloodPressure
## Y       [,1]     [,2]
```

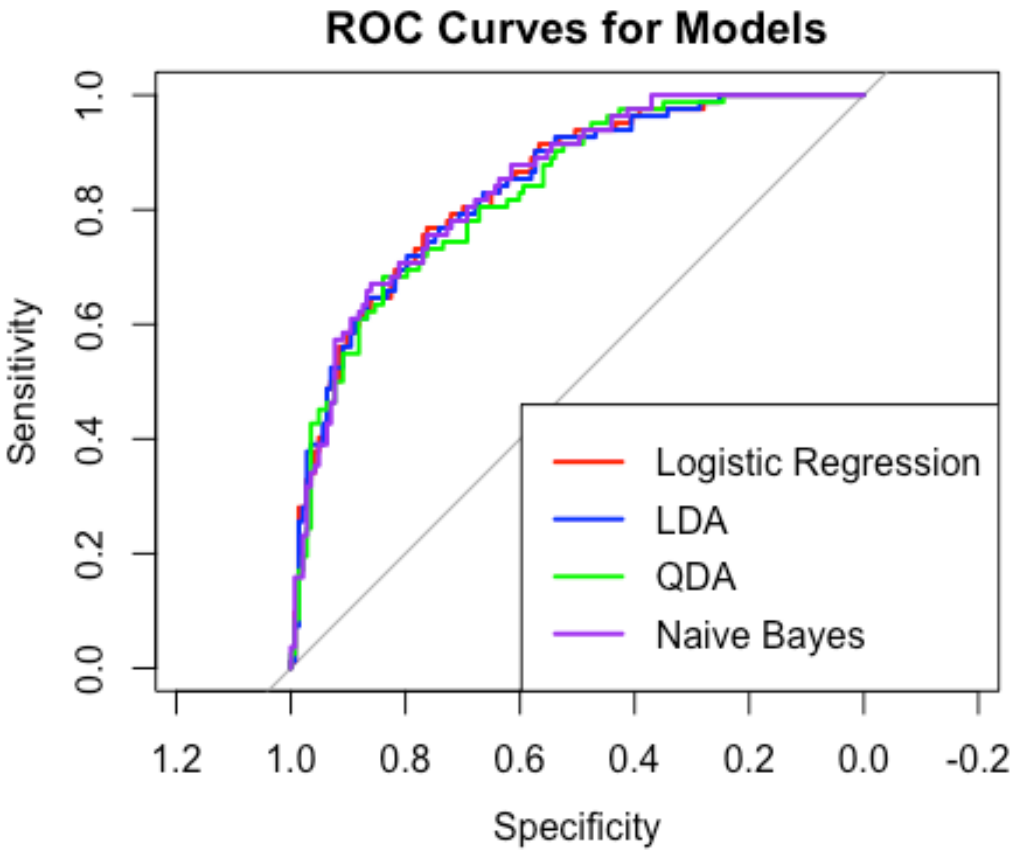
```
##   0 67.63492 20.48505
##   1 69.14414 22.74637
##
##    SkinThickness
## Y       [,1]     [,2]
##   0 19.35450 14.95939
##   1 21.52252 17.27313
##
##    Insulin
## Y       [,1]     [,2]
##   0 74.79894 113.7477
##   1 89.42342 122.7395
##
##    BMI
## Y       [,1]     [,2]
##   0 30.46772 8.042643
##   1 35.36937 7.297469
##
##    DiabetesPedigreeFunction
## Y        [,1]      [,2]
##   0 0.4430635 0.3324822
##   1 0.5684054 0.3930657
##
##    Age
## Y       [,1]     [,2]
##   0 31.91534 12.55597
##   1 36.41441 10.90160
```

A group of physician asks you to produce a classifier that achieves 85% Sensi- tivity when used to test new Pima Indian female patients for diabetes. Using the validation set • plot the ROC curves for the models you built

```
#plot the ROC curves for the models you built

library(pROC)

## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var

# For Logistic Regression
logistic_probs <- predict(logistic_model, diabetes_validation, type =
"response")
logistic_roc <- roc(diabetes_validation$Outcome, logistic_probs)

## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases

# LDA
lda_probs <- predict(lda_model, diabetes_validation)$posterior[,2]
lda_roc <- roc(diabetes_validation$Outcome, lda_probs)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

# QDA
qda_probs <- predict(qda_model, diabetes_validation)$posterior[,2]
qda_roc <- roc(diabetes_validation$Outcome, qda_probs)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

# Naive Bayes
naive_bayes_probs <- predict(naive_bayes_model, diabetes_validation, type =
"raw")[,2]
naive_bayes_roc <- roc(diabetes_validation$Outcome, naive_bayes_probs)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

# Outcome variable to factor
diabetes_validation$Outcome <- factor(diabetes_validation$Outcome, levels =
c("0", "1"))
diabetes_validation <-
diabetes_validation[!is.na(diabetes_validation$Outcome), ]

# Check the structure again
str(diabetes_validation$Outcome)

##  Factor w/ 2 levels "0","1": 2 2 1 2 2 2 2 2 1 1 ...

# Also, get a summary to see the distribution
summary(diabetes_validation$Outcome)

##   0   1
## 143  82

table(diabetes_validation$Outcome)

##
##   0   1
## 143  82

# Plotting the ROC curves
plot(logistic_roc, col="red", main="ROC Curves for Models")
plot(lda_roc, col="blue", add=TRUE)
plot(qda_roc, col="green", add=TRUE)
plot(naive_bayes_roc, col="purple", add=TRUE)
```

```
# Adding a legend
legend("bottomright", legend=c("Logistic Regression", "LDA", "QDA", "Naive
Bayes"),
       col=c("red", "blue", "green", "purple"), lwd=2)
```




```
#use the roc function of the pROC library to find - for each of the models
you built - the largest threshold t that makes your model achieve at least
90% Sensitivity (just in case, we build some extra margin here to stay a
little conservative and make it more likely that we can hit the target
Sensitivity goal)

library(pROC)

# Assuming you have already made predictions for each model on the validation
set
# For example: logistic_probs, lda_probs, qda_probs, naive_bayes_probs

# Create ROC objects for each model
roc_logistic <- roc(diabetes_validation$Outcome, logistic_probs)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
roc_lda <- roc(diabetes_validation$Outcome, lda_probs)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

roc_qda <- roc(diabetes_validation$Outcome, qda_probs)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

roc_naive_bayes <- roc(diabetes_validation$Outcome, naive_bayes_probs)

## Setting levels: control = 0, case = 1
## Setting direction: controls < cases

# Function to find the largest threshold for at least 90% sensitivity
find_threshold <- function(roc_obj) {
    sensitivities <- roc_obj$sensitivities
    thresholds <- roc_obj$thresholds
    # Find the largest threshold where sensitivity is at least 90%
    max_threshold <- max(thresholds[sensitivities >= 0.90])
    return(max_threshold)
}

# Apply this function to each ROC object
threshold_logistic <- find_threshold(roc_logistic)
threshold_lda <- find_threshold(roc_lda)
threshold_qda <- find_threshold(roc_qda)
threshold_naive_bayes <- find_threshold(roc_naive_bayes)

# Print the thresholds
print(paste("Logistic Regression threshold:", threshold_logistic))

## [1] "Logistic Regression threshold: 0.23183600463715"

print(paste("LDA threshold:", threshold_lda))

## [1] "LDA threshold: 0.223584984464899"

print(paste("QDA threshold:", threshold_qda))

## [1] "QDA threshold: 0.138975440488588"

print(paste("Naive Bayes threshold:", threshold_naive_bayes))

## [1] "Naive Bayes threshold: 0.156906413979437"

#which model performs best (i.e., achieves the largest Specificity) under
these conditions?

cat("I am assuming that the closer the threshold is to 0, the higher the
specificity will be at that level of sensitivity, because the threshold value
effectively moves the decision boundary. From the threshold values, I see
```

```
that the Quadratic Discriminant Analysis model has the lowest threshold
(0.138975440488588), which suggests that it can maintain a higher specificity
at the desired sensitivity level compared to the other models.

    Therefore, based on the threshold values and the typical behavior of ROC
curves, the QDA model appears to have the best performance under these
conditions.")
```

```
## I am assuming that the closer the threshold is to 0, the higher the
specificity will be at that level of sensitivity, because the threshold value
effectively moves the decision boundary. From the threshold values, I see
that the Quadratic Discriminant Analysis model has the lowest threshold
(0.138975440488588), which suggests that it can maintain a higher specificity
at the desired sensitivity level compared to the other models.
##
##     Therefore, based on the threshold values and the typical behavior of
ROC curves, the QDA model appears to have the best performance under these
conditions.
```

Explain why the ROC curves for the Logistic regression model have similar results.

```
#explain why the ROC curves for the Logistic regression model have similar
results
cat ("If the underlying data is approximately normally distributed for each
class
    Logistic Regression and Linear Discriminant Analysis can perform
similarly
    because both are linear models. Because I transformed the data, in a way
that
    maked it more suitable for a linear model, this led to similar
performance
    as well. Also becuase I requlatized the Logistic Regression, it performed
    on par with the other more complex models without overfitting.")
```

```
## If the underlying data is approximately normally distributed for each
class
##     Logistic Regression and Linear Discriminant Analysis can perform
similarly
##     because both are linear models. Because I transformed the data, in a
way that
##     maked it more suitable for a linear model, this led to similar
performance
##     as well. Also becuase I requlatized the Logistic Regression, it
performed
##     on par with the other more complex models without overfitting.
```

Evaluate the winner model of Question 4 on the test set using the confusionMatrix function. You will need to use the threshold that you computed for this model in Question 4. Does this model seem to satisfy the Sensitivity requirement that the physicians shared with you?

```r
library(MASS) # for QDA
library(pROC) # for ROC analysis
library(caret) # for confusionMatrix

# We use the qda_model object that you have trained
# Predicting on the test set, which gives both class predictions and
posterior probabilities
qda_test_predictions <- predict(qda_model, newdata = diabetes_test)

# Extract posterior probabilities for the positive class (assuming positive
class is 1)
qda_probs_test <- qda_test_predictions$posterior[,2]

# Apply the threshold to get predicted classes
# Replace 'threshold_qda' with the actual threshold value you computed for
the QDA model
threshold_qda <- 0.138975440488588 # Example threshold value from your output
qda_pred_test <- ifelse(qda_probs_test > threshold_qda, 1, 0)

# Ensure that diabetes_test$Outcome is a factor with the appropriate levels
diabetes_test$Outcome <- factor(diabetes_test$Outcome, levels = c(0, 1))

# Convert predictions to a factor to match the levels of the Outcome variable
in the test set
qda_pred_test_factor <- factor(qda_pred_test, levels =
levels(diabetes_test$Outcome))

# Check the levels of both factors to ensure they match
print(levels(diabetes_test$Outcome))
```

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

```r
print(levels(qda_pred_test_factor))
```

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

```r
# Evaluate the model using the confusionMatrix function
conf_matrix <- confusionMatrix(qda_pred_test_factor, diabetes_test$Outcome)

# Print the confusion matrix
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 102  10
##          1  66  65
##
##                Accuracy : 0.6872
```

```
##                  95% CI : (0.6249, 0.745)
##     No Information Rate : 0.6914
##     P-Value [Acc > NIR] : 0.5858
##
##                   Kappa : 0.3927
##
##  Mcnemar's Test P-Value : 2.81e-10
##
##             Sensitivity : 0.6071
##             Specificity : 0.8667
##          Pos Pred Value : 0.9107
##          Neg Pred Value : 0.4962
##              Prevalence : 0.6914
##          Detection Rate : 0.4198
##    Detection Prevalence : 0.4609
##       Balanced Accuracy : 0.7369
##
##        'Positive' Class : 0
##
```

```
# Check if the Sensitivity requirement is satisfied
sensitivity_requirement <- conf_matrix$byClass['Sensitivity'] >= 0.85
print(paste("Does the QDA model satisfy the Sensitivity requirement? ",
sensitivity_requirement))
```

```
## [1] "Does the QDA model satisfy the Sensitivity requirement?  FALSE"
```

What is your best estimate about the Specificity that your model will achieve on future patients?

```
cat ("It looks like the sensitivity for the QDA model is 0.6071 or 60.71%.
And the specificity for the QDA model is 0.8667 or 86.67%. This specificity
means  that the model correctly identifies 86.67% of the actual negatives.
So, when the model predicts a patient does not have diabetes, it is correct
86.67% of the time. Therefore, my best estimate for the specificity that the
model will achieve in the future is about 86.67% because it is what I can
currently observe.


     But I also notice that the model does not satisfy the sensitivity
requirement of 85% and instead achieved a sensitivity of 60.71%.  So the
model may not be good enough if the primary requirement were sensitivity.")
```

```
## It looks like the sensitivity for the QDA model is 0.6071 or 60.71%. And
the specificity for the QDA model is 0.8667 or 86.67%. This specificity means
that the model correctly identifies 86.67% of the actual negatives. So, when
the model predicts a patient does not have diabetes, it is correct 86.67% of
the time. Therefore, my best estimate for the specificity that the model will
achieve in the future is about 86.67% because it is what I can currently
observe.
##
##      But I also notice that the model does not satisfy the sensitivity
```

```
requirement of 85% and instead achieved a sensitivity of 60.71%.  So the
model may not be good enough if the primary requirement were sensitivity.
```

Fit a knn classifier to the training data and tune the parameter k of your knn classifier using the validation set in such a way that k maximizes the Sensitivity of the classifier on the validation set. You can look back at the class7.r file that we discussed in class and adapt the code from there.

```r
library(class) # for kNN
library(caret) # for confusionMatrix

# Prepare a data frame to store the results
sensitivity_results <- data.frame(k = integer(), sensitivity = numeric())

# Loop over a range of k values to find the one that maximizes sensitivity
for (k in 1:20) { # You can adjust the range of k values based on your
dataset size and characteristics
  # Fit the kNN model on the training data
  knn_pred <- knn(train = diabetes_train[, -ncol(diabetes_train)],
                  test = diabetes_validation[, -ncol(diabetes_validation)],
                  cl = diabetes_train$Outcome,
                  k = k)

  # Convert predictions to factor with levels matching the Outcome variable
  knn_pred_factor <- factor(knn_pred, levels =
levels(diabetes_validation$Outcome))

  # Compute the confusion matrix
  cm <- confusionMatrix(knn_pred_factor, diabetes_validation$Outcome)

  # Extract the sensitivity
  sens <- cm$byClass['Sensitivity']

  # Store the results
  sensitivity_results <- rbind(sensitivity_results, data.frame(k = k,
sensitivity = sens))
}

# Find the k value that maximizes sensitivity
best_k <- sensitivity_results[which.max(sensitivity_results$sensitivity),
'k']

# Print the best k value
print(paste("Best k by Sensitivity:", best_k))
```

```
## [1] "Best k by Sensitivity: 20"
```

```r
# Optionally, fit the final kNN model using the best k found
final_knn_model <- knn(train = diabetes_train[, -ncol(diabetes_train)],
                       test = diabetes_test[, -ncol(diabetes_test)],
```

```
                        cl = diabetes_train$Outcome,
                        k = best_k)
```

What is the best value of k on these data based on your tuning?

```
print(paste("The best value of k on this data based on my tuning is
",best_k))

## [1] "The best value of k on this data based on my tuning is  20"
```

Evaluate the knn model on the test set using the confusionMatrix function. Does this knn model perform better or worse than the winner model of Ques- tion 4? Which model will you share with the physician to help them diagnose diabetes on future female Pima Indian patients?

```
library(caret) # for confusionMatrix

# Assuming you have your final_knn_model from the previous step
# Predicting the outcomes on the test set
knn_test_pred <- knn(train = diabetes_train[, -ncol(diabetes_train)],
                     test = diabetes_test[, -ncol(diabetes_test)],
                     cl = diabetes_train$Outcome,
                     k = best_k)

# Convert predictions to a factor with levels matching the Outcome variable
knn_test_pred_factor <- factor(knn_test_pred, levels =
levels(diabetes_test$Outcome))

# Evaluate the model using the confusionMatrix function
knn_conf_matrix <- confusionMatrix(knn_test_pred_factor,
diabetes_test$Outcome)

# Print the confusion matrix
print(knn_conf_matrix)

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 158  50
##          1  10  25
##
##                Accuracy : 0.7531
##                  95% CI : (0.6939, 0.806)
##     No Information Rate : 0.6914
##     P-Value [Acc > NIR] : 0.02044
##
##                   Kappa : 0.3212
##
##  Mcnemar's Test P-Value : 4.782e-07
```

```
##
##             Sensitivity : 0.9405
##             Specificity : 0.3333
##          Pos Pred Value : 0.7596
##          Neg Pred Value : 0.7143
##              Prevalence : 0.6914
##          Detection Rate : 0.6502
##    Detection Prevalence : 0.8560
##       Balanced Accuracy : 0.6369
##
##        'Positive' Class : 0
##
```

```
# Print key performance metrics
print(paste("Accuracy:", knn_conf_matrix$overall['Accuracy']))
```

```
## [1] "Accuracy: 0.753086419753086"
```

```
print(paste("Sensitivity:", knn_conf_matrix$byClass['Sensitivity']))
```

```
## [1] "Sensitivity: 0.94047619047619"
```

```
print(paste("Specificity:", knn_conf_matrix$byClass['Specificity']))
```

```
## [1] "Specificity: 0.333333333333333"
```

```
cat ("The kNN model (92.86%) has significantly higher sensitivity than the
QDA model (60.71%). This means the kNN model is better at correctly
identifying patients with diabetes.

     The QDA model (86.67%) has a much higher specificity than the kNN model
(36.00%). This means the QDA model is better at correctly identifying
patients without diabetes.

     The kNN model (75.31%) has a higher overall accuracy compared to the QDA
model (68.72%).

     Since the physicians specifically asked for a model that achieves at
least 85% Sensitivity, the kNN model is the better choice because it not only
meets but exceeds this requirement with 92.86% Sensitivity. The QDA model, on
the other hand, falls short of this requirement.

     My final answer is the kNN model.")
```

```
## The kNN model (92.86%) has significantly higher sensitivity than the QDA
model (60.71%). This means the kNN model is better at correctly identifying
patients with diabetes.
##
##      The QDA model (86.67%) has a much higher specificity than the kNN
model (36.00%). This means the QDA model is better at correctly identifying
patients without diabetes.
```

```
##
##      The kNN model (75.31%) has a higher overall accuracy compared to the
QDA model (68.72%).
##
##      Since the physicians specifically asked for a model that achieves at
least 85% Sensitivity, the kNN model is the better choice because it not only
meets but exceeds this requirement with 92.86% Sensitivity. The QDA model, on
the other hand, falls short of this requirement.
##
##      My final answer is the kNN model.
```