STAT 6340 Mini Project 3

Lakshmipriya Narayanan

SECTION 1: Observations and Answers

1

(a): A reasonably good logistic regression model for this data is the response, Outcome with predictors Pregnancies, Glucose, BloodPressure, BMI, DiabetesPedigreeFunction, Insulin and Age. We drop the variable SkinThickness because it is not significant. It has a high p-value of 0.928. Whereas the rest of the predictors have very less p-values. Insulin and Age have slightly larger p-values compared to the rest of the predictors but still lower than that of SkinThickness.

(b): The intercept is -0.871 which is the log odds of the response (Outcome) and this happens when all predictor variables are = 0. In reality, this is not very insightful because we can't predict if a person has diabetes or not without any predictor variables. The coefficient of predictor BloodPressure is -0.255, so with each unit increase in blood pressure, the odds of having diabetes decrease by $1 - e^{-0.255}$ = 1 - 0.774 = 0.226 = 22.6 %.

(Intercept)	Pregnancies	Glucose	BloodPressure	BMI	DiabetesPedigreeFunction
-0.8712054	0.4150392	1.1226306	-0.2557594	0.7102729	0.3139654

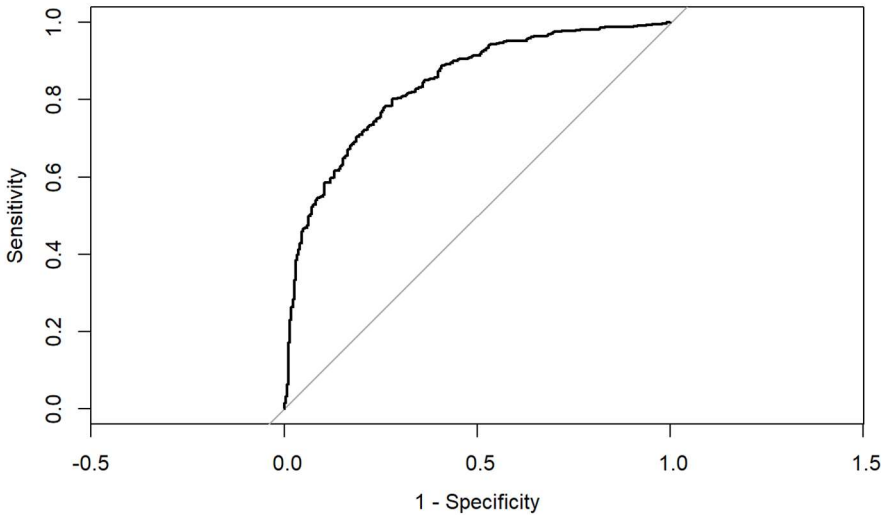
Insulin	Age
-0.1333414	0.1739202

The training error rate for this reasonably good model = 21.61 %

2

(a): Error rate = 0.217 Sensitivity = 0.582 Specificity = 0.890

Here, sensitivity < specificity which implies that the full model is in itself a good fit.



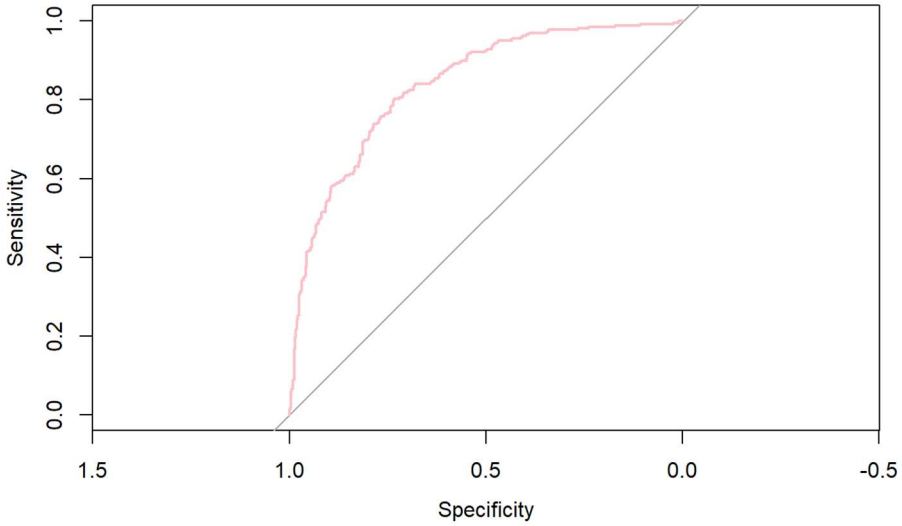
Area under the curve, AUC = 0.8394

(b): See code in section 2, LOOCV = 0.2226

(c): Error rate using LOOCV = 0.2226. This matches (b) above where we implemented the for loop discussed in lecture.

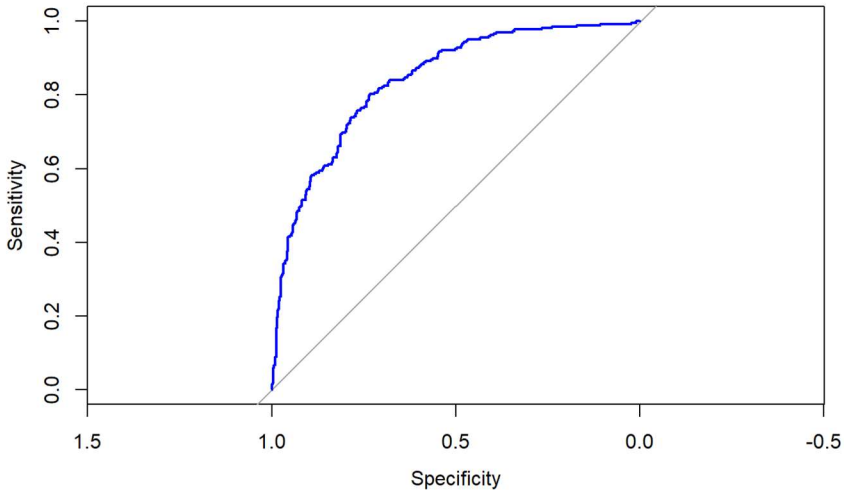
(d): Error rate = 0.2161 Sensitivity = 0.892 Specificity = 0.582 Test rate (LOOCV) = 0.225

From the above results and the AUC below, we can observe that these results match/ are very close to our results obtained in 2(a) above for normal logistic regression. The ROC curves also look very similar.



Area under the curve, AUC = 0.8393

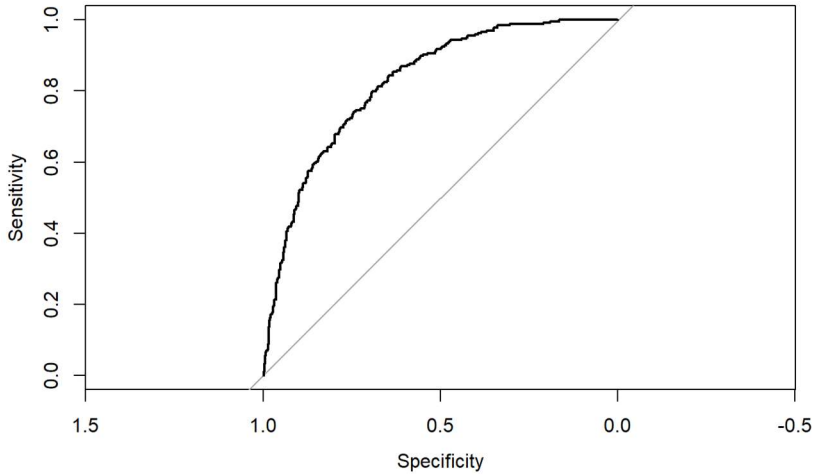
(e): Error rate = 0.2356 Sensitivity = 0.864 Specificity = 0.578 Test rate (LOOCV) = 0.261



Area under the curve, AUC = 0.8393
(same as LDA)

AUC is the same as LDA and this makes sense because the plots look like they are the exact same.

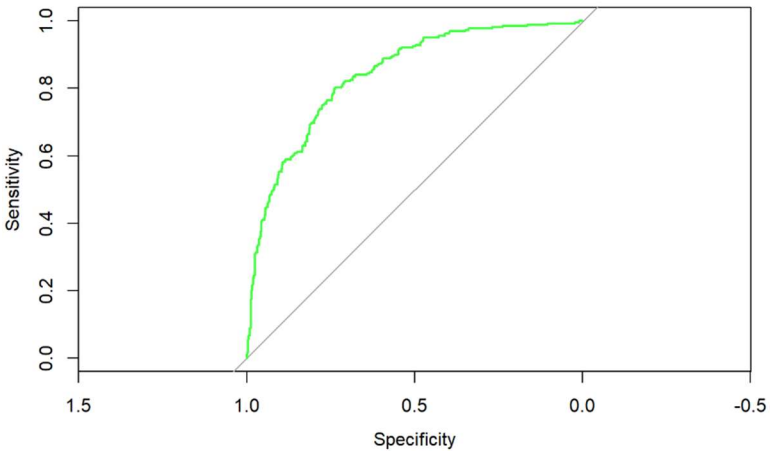
(f): Error rate = 0.2382 Sensitivity = 0.842 Specificity = 0.611 Test rate (LOOCV) = 0.247, 0.25



Area under the curve, AUC = 0.8243

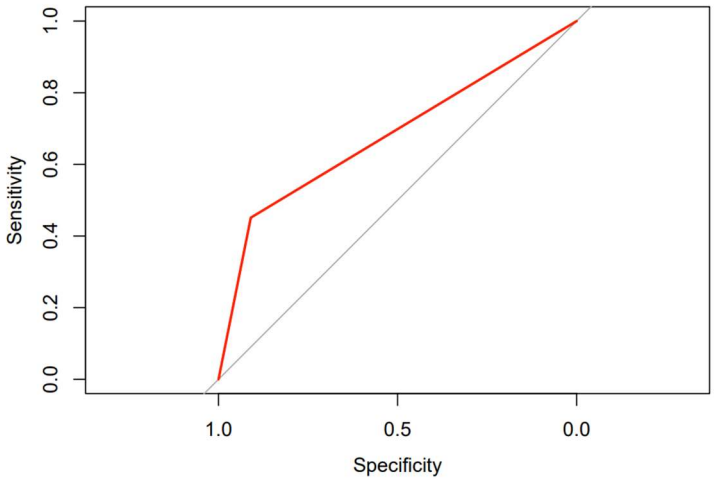
(g): Error rate = 0.2161 Sensitivity = 0.892 Specificity = 0.582 Test rate (LOOCV) = 0.225

From the above results we can see that clearly all of them match 2(d) where we did LDA on the full model. Since the results are the exact same we can conclude that SkinThickness is indeed a insignificant predictor.



Area under the curve, AUC = 0.8394

(h): Optimal k = 23 and the corresponding test error rate using LOOCV = 0.2501. For this optimal K,
Error rate = 0.2501 Sensitivity = 0.909 Specificity = 0.451 Test rate = 0.2501 (matches LOOCV)



Area under the curve, AUC = 0.6805

(i):

Classification Method	Misclassification error rate (Using confusion matrix)	Sensitivity	Specificity	Test error rate using LOOCV	Area Under the curve
Logistic Regression	0.217	0.582	0.890	0.222	0.839
LDA	0.216	0.892	0.582	0.225	0.839
QDA	0.235	0.864	0.578	0.261	0.839
Naïve Bayes	0.238	0.842	0.611	0.247	0.824
Logistic Regression (good model in 1a)	0.216	0.892	0.582	0.225	0.839
KNN	0.250	0.909	0.451	0.250	0.680

The above table summarizes the results we need to make a conclusion on which classification method can be chosen.

- Logistic regression and LDA have the lowest misclassification rates and test error rates using LOOCV.
- Also, sensitivity and specificity for both these classification methods are balanced as discussed in 2(a) and 2(d)
- AUC for both these classification methods are high which implies that they are closer to 1 which is our goal.

Therefore, using **logistic regression is better** because it has low error rates (computed by both confusion matrix and LOOCV) and it also has less sensitivity compared to its corresponding specificity.

Section 2: Code

```
#Load required libraries
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(pROC)
```

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

```
##
## Attaching package: 'pROC'
```

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

```
library(boot)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```r
library(class)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```r
library(e1071)
```

Problem 1

```r
# Read the data
diab_data <- read.csv("diabetes.csv")
summary(diab_data)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

```r
# Check for missing values
#sum(is.na(diab_data))

#Convert outcome to a factor variable because it is qualitative
diab_data$Outcome <- as.factor(diab_data$Outcome)
table(diab_data$Outcome)
```

```
##
##   0   1
## 500 268
```

```r
#------------------STANDARDIZING ALL PREDICTORS--------------------------------
# Function to standardize the training variable
standardize <- function(x) {
  (x - mean(x)) / sd(x)
}

# Standardize all columns except "Outcome"
standardize_Diab <- diab_data %>%
  mutate(across(-Outcome, standardize))

# View the standardized data
#standardize_Diab
```

(a):

```r
#Logistic regression model with all predictors
diab_log_reg_good <- glm(Outcome ~ Pregnancies + Glucose + BloodPressure + BMI + DiabetesPedigreeFunction + In
#summary(diab_log_reg_good)

# Since skin thickness has a high p-value, drop that predictor and this is
# our reasonably good model
diab_log_reg_all <- glm(Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
                          Insulin + BMI + DiabetesPedigreeFunction + Age,
                        family = binomial, data = standardize_Diab)
#summary(diab_log_reg_all)

# Model with only significant predictors. Drop skin thickness, age and insulin
diab_log_reg_drop3 <- glm(Outcome ~ Pregnancies + Glucose + BloodPressure + BMI +
                            DiabetesPedigreeFunction , family = binomial,
                          data = standardize_Diab)
#summary(diab_log_reg_drop3)

# p-value is 0.92 > 0.05 which is high. Therefore, dropping skinthickness
#does not affect our model significantly.
anova(diab_log_reg_all, diab_log_reg_good, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
##     Insulin + BMI + DiabetesPedigreeFunction + Age
## Model 2: Outcome ~ Pregnancies + Glucose + BloodPressure + BMI + DiabetesPedigreeFunction +
##     Insulin + Age
##   Resid. Df Resid. Dev Df   Deviance Pr(>Chi)
## 1       759     723.45
```

```
## 2       760     723.45 -1 -0.0080518   0.9285
```

```
# p-value = 0.077 > 0.05 but still 0.077 < 0.1. So, we retain age and insulin
#because these might help in improving out model and predictions.
anova(diab_log_reg_drop3, diab_log_reg_good, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: Outcome ~ Pregnancies + Glucose + BloodPressure + BMI + DiabetesPedigreeFunction
## Model 2: Outcome ~ Pregnancies + Glucose + BloodPressure + BMI + DiabetesPedigreeFunction +
##     Insulin + Age
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       762     728.56
## 2       760     723.45  2   5.1062  0.07784 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# p-value = 0.16 > 0.05 => predictors skin thickness, insulin and age are not
#very significant. But we still retain age and insulin from what we observed
#in the above anova test
anova(diab_log_reg_drop3, diab_log_reg_all, test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: Outcome ~ Pregnancies + Glucose + BloodPressure + BMI + DiabetesPedigreeFunction
## Model 2: Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
##     Insulin + BMI + DiabetesPedigreeFunction + Age
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       762     728.56
## 2       759     723.45  3   5.1142   0.1636
```

(b):

```
#Use the coef function to obtain coefficient estimates of the reasonably good model.
print(coef(diab_log_reg_good))
```

```
##               (Intercept)               Pregnancies                   Glucose
##                -0.8712054                 0.4150392                 1.1226306
##             BloodPressure                       BMI DiabetesPedigreeFunction
##                -0.2557594                 0.7102729                 0.3139654
##                   Insulin                       Age
##                -0.1333414                 0.1739202
```

```
#Get the probability of the predicted class to help calculate training error rate
log_reg_prob <- predict(diab_log_reg_good, standardize_Diab, type = "response")

# Predicted classes (using 0.5 cutoff)
```

```r
log_reg_pred <- ifelse(log_reg_prob >= 0.5, "1", "0")

# Training error rate
1 - mean(log_reg_pred == standardize_Diab[, "Outcome"])
```

```
## [1] 0.2161458
```

Problem 2

(a):

```r
#Logistic regression model with all predictors
diab_log_reg_all <- glm(Outcome ~ Pregnancies + Glucose + BloodPressure +
                    SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
                        family = binomial, data = standardize_Diab)
#summary(diab_log_reg_all)

# ---------------------------------Error rate calculation-----------------------
log_reg_all_prob <- predict(diab_log_reg_all, standardize_Diab, type = "response")
# Predicted classes (using 0.5 cutoff)
log_reg_all_pred <- ifelse(log_reg_all_prob >= 0.5, "1", "0")

# Training error rate
1 - mean(log_reg_all_pred == standardize_Diab[, "Outcome"])
```

```
## [1] 0.2174479
```

```r
#-------------------------Confusion matrix--------------------------------------
table(log_reg_all_pred, standardize_Diab[, "Outcome"])
```

```
##
## log_reg_all_pred   0   1
##                0 445 112
##                1  55 156
```

```r
# Calculate sensitivity and specificity from above obtained confusion matrix
c(156/(112 + 156), 445/(445 + 55))
```

```
## [1] 0.5820896 0.8900000
```

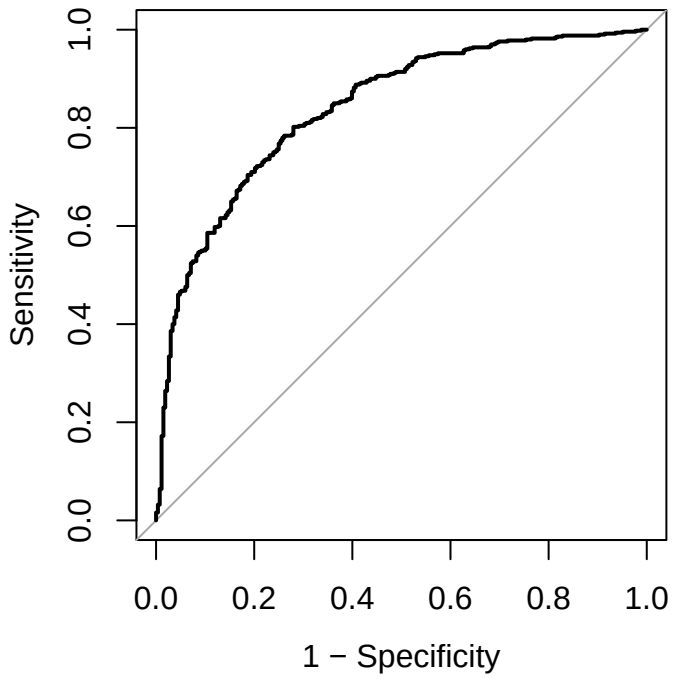
```r
#--------------------------------ROC curve-------------------------------------
# Plot the ROC curve to get the area under the curve, AUC
roc_log_reg_all <- roc(standardize_Diab[, "Outcome"], log_reg_all_prob,
                       levels = c("1", "0"))
```

```
## Setting direction: controls > cases
```

```
roc_log_reg_all
```

```
##
## Call:
## roc.default(response = standardize_Diab[, "Outcome"], predictor = log_reg_all_prob,     levels = c("1", "0"
##
## Data: log_reg_all_prob in 268 controls (standardize_Diab[, "Outcome"] 1) > 500 cases (standardize_Diab[, "O
## Area under the curve: 0.8394
```

```
#AUC : 0.8394

plot(roc_log_reg_all, legacy.axes = T)
```



(b):

```
#Number of predictions, n = number of rows
num_pred <- nrow(standardize_Diab)

# Perform LOOCV mentioned in lecture notes for chapter 5
for (i in 1:num_pred) {
  #Split standardized data to test and training set and leave out the ith observation (xi, yi)
  train_data <- standardize_Diab[-i, ]
  #Use the ith observation to predict yi_hat
  test_data <- standardize_Diab[i, , drop = FALSE]

  #Fit model as in 2(a)
  model_new <- glm(Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
                   Insulin + BMI + DiabetesPedigreeFunction + Age ,
                   family = binomial, data = train_data)
```

```r
  #Predict yi_hat
  y_hat <- predict(model_new, newdata = test_data, type = "response")
  num_pred[i] <- ifelse(y_hat > 0.5, 1, 0)

} #end forloop

#Return the final error rate by calculating true value of y, y_i
yi <- as.numeric(standardize_Diab$Outcome) - 1
#Compute MSE for the ith prediction, MSE_i to obtain approximately unbiased
#estimate of the test MSE
test_error_rate <- mean(num_pred != yi)
paste("Test Error Rate using LOOCV:", test_error_rate)
```

```
## [1] "Test Error Rate using LOOCV: 0.22265625"
```

(c):

```r
# Using the 'boot' package we calculate error rate using LOOCV method with the cv.glm function
cv_error <- cv.glm(standardize_Diab, diab_log_reg_all)
cv_error$delta
```

```
## [1] 0.1572645 0.1572614
```

```r
# test error rate of model using LOOCV is 0.157 which does not match 2(b) so we
#use a cost function to calculate LOOCV test error because this function helps
#to evaluate how well our model performs more accuartely.

# Define a cost function for error rate
cost_loocv <- function(r, pi = 0) mean(abs(r - pi) > 0.5)

# Perform LOOCV using the boot package
cv_error_new <- cv.glm(standardize_Diab, diab_log_reg_all, cost = cost_loocv,
                       K = nrow(standardize_Diab))

# Calculate the misclassification error rate
test_error_rate <- cv_error_new$delta[1]
cat("Test Error Rate using LOOCV:", test_error_rate, "\n")
```

```
## Test Error Rate using LOOCV: 0.2226563
```

```r
#Now the test error rate is 0.222 which matched our 2(b) implementation from scratch
```

(d):

```r
#LDA on data for all predictors
diab_lda <- lda(Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
            Insulin + BMI + DiabetesPedigreeFunction + Age, data = standardize_Diab)

# Get predictions for data
lda_pred <- predict(diab_lda, standardize_Diab)$class

# Confusion matrix for training and test data using the caret package and
#using confusionMatrix function
cm_lda <- confusionMatrix(lda_pred, standardize_Diab$Outcome)
cm_lda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 446 112
##          1  54 156
##
##                Accuracy : 0.7839
##                  95% CI : (0.753, 0.8125)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 7.051e-16
##
##                   Kappa : 0.4992
##
##  Mcnemar's Test P-Value : 9.686e-06
##
##             Sensitivity : 0.8920
##             Specificity : 0.5821
##          Pos Pred Value : 0.7993
##          Neg Pred Value : 0.7429
##              Prevalence : 0.6510
##          Detection Rate : 0.5807
##    Detection Prevalence : 0.7266
##       Balanced Accuracy : 0.7370
##
##        'Positive' Class : 0
##
```

```r
# Misclassification/error rate
mcr_lda <- 1 - cm_lda$overall['Accuracy']
mcr_lda
```

```
##  Accuracy
## 0.2161458
```

```r
lda_lr_prob <- predict(diab_lda, standardize_Diab)$posterior[,2]

#ROC
roc_lda_diab <- roc(standardize_Diab$Outcome, lda_lr_prob)
```

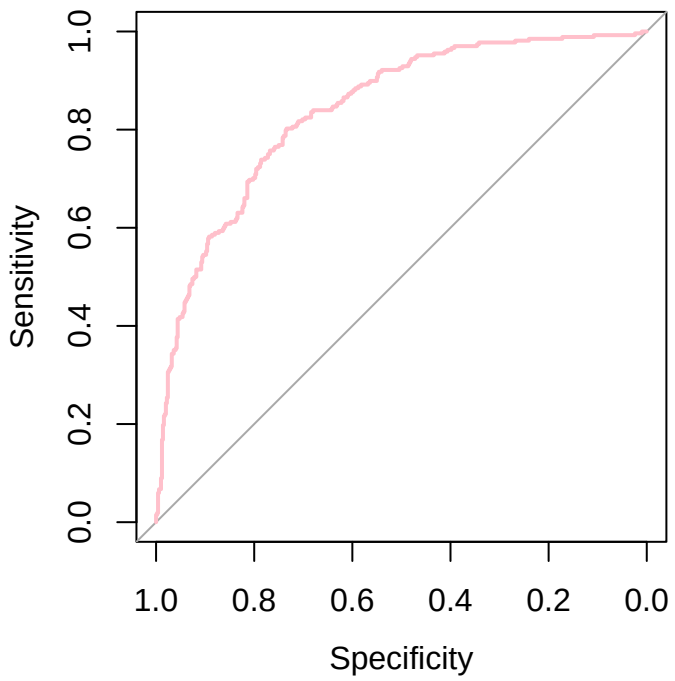
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
roc_lda_diab
```

```
##
## Call:
## roc.default(response = standardize_Diab$Outcome, predictor = lda_lr_prob)
##
## Data: lda_lr_prob in 500 controls (standardize_Diab$Outcome 0) < 268 cases (standardize_Diab$Outcome 1).
## Area under the curve: 0.8393
```

```r
#AUC : 0.8393

plot(roc_lda_diab, col = "pink")
```




```r
# Define the train control with LOOCV. We define a training control because we
#specify parameters for the training and includes the LOOCV method.
train_ctrl <- trainControl(method = "LOOCV")

# Train the LDA model using the train function defined above
diab_lda_train <- train(Outcome ~ Pregnancies + Glucose + BloodPressure +
```

```
            SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
            data = standardize_Diab, method = "lda", trControl = train_ctrl)

# Calculate the test error rate
test_error_rate <- 1 - diab_lda_train$results$Accuracy
print(paste("Test Error Rate using LOOCV:", test_error_rate))
```

```
## [1] "Test Error Rate using LOOCV: 0.225260416666667"
```

(e):

```
#QDA on the data for all predictors
diab_qda <- qda(Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
          Insulin + BMI + DiabetesPedigreeFunction + Age, data = standardize_Diab)

# Get predictions
qda_pred <- predict(diab_qda, standardize_Diab)$class

# Confusion matrix
cm_qda <- confusionMatrix(qda_pred, standardize_Diab$Outcome)
cm_qda
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 432 113
##          1  68 155
##
##                Accuracy : 0.7643
##                  95% CI : (0.7327, 0.7939)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 7.13e-12
##
##                   Kappa : 0.4603
##
##  Mcnemar's Test P-Value : 0.001074
##
##             Sensitivity : 0.8640
##             Specificity : 0.5784
##          Pos Pred Value : 0.7927
##          Neg Pred Value : 0.6951
##              Prevalence : 0.6510
##          Detection Rate : 0.5625
##    Detection Prevalence : 0.7096
##       Balanced Accuracy : 0.7212
##
##        'Positive' Class : 0
```

```
##
```

```
# Misclassification/error rate
mcr_qda <- 1 - cm_qda$overall['Accuracy']
mcr_qda
```

```
##  Accuracy
## 0.2356771
```

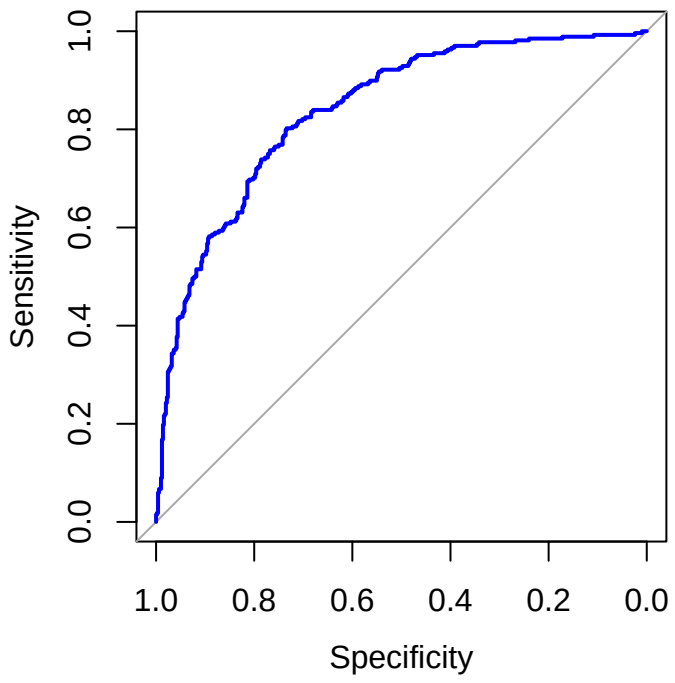
```
qda_lr_prob <- predict(diab_lda, standardize_Diab)$posterior[,2]

#ROC
roc_qda_diab <- roc(standardize_Diab$Outcome, qda_lr_prob)
```

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

## Setting direction: controls < cases
```

```
roc_qda_diab
```

```
##
## Call:
## roc.default(response = standardize_Diab$Outcome, predictor = qda_lr_prob)
##
## Data: qda_lr_prob in 500 controls (standardize_Diab$Outcome 0) < 268 cases (standardize_Diab$Outcome 1).
## Area under the curve: 0.8393
```

```
#AUC : 0.8393

plot(roc_qda_diab, col = "blue")
```



```r
# Train the QDA model using the train function defined above
diab_qda_train <- train(Outcome ~ Pregnancies + Glucose + BloodPressure +
          SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
          data = standardize_Diab, method = "qda", trControl = train_ctrl)

# Calculate the test error rate
test_error_rate <- 1 - diab_qda_train$results$Accuracy
print(paste("Test Error Rate using LOOCV:", test_error_rate))
```

```
## [1] "Test Error Rate using LOOCV: 0.260416666666667"
```

(f):

```r
#Naive bayes on training and test data for all predictors
diab_nbc <- naiveBayes(Outcome ~ Pregnancies + Glucose + BloodPressure +
          SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
          data = standardize_Diab)

# Get predictions for test data and training data
nbc_pred <- predict(diab_nbc, standardize_Diab)

# Confusion matrix for training data and test data
cm_nbc <- confusionMatrix(nbc_pred, standardize_Diab$Outcome)
cm_nbc
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 421 104
##          1  79 164
##
##                Accuracy : 0.7617
##                  95% CI : (0.73, 0.7914)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 2.156e-11
##
##                   Kappa : 0.464
##
##  Mcnemar's Test P-Value : 0.07604
##
##             Sensitivity : 0.8420
##             Specificity : 0.6119
##          Pos Pred Value : 0.8019
##          Neg Pred Value : 0.6749
##              Prevalence : 0.6510
##          Detection Rate : 0.5482
##    Detection Prevalence : 0.6836
```

```
##       Balanced Accuracy : 0.7270
##
##        'Positive' Class : 0
##
```

```
# Misclassification rate for training data and test data
mcr_nbc <- 1 - cm_nbc$overall['Accuracy']
mcr_nbc
```

```
##  Accuracy
## 0.2382812
```

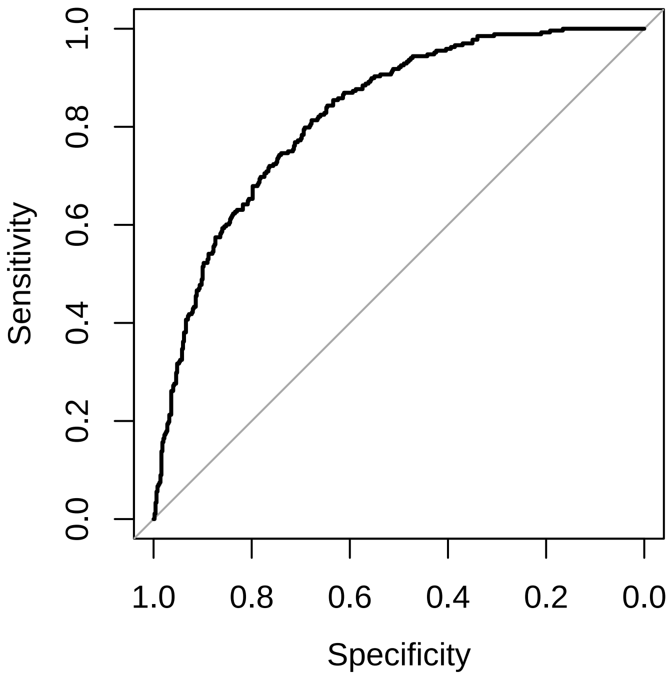
```
nbc_lr_prob <- predict(diab_nbc, standardize_Diab, type = "raw")[,2]

#ROC
roc_nbc_diab <- roc(standardize_Diab$Outcome, nbc_lr_prob)
```

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

## Setting direction: controls < cases
```

```
roc_nbc_diab
```

```
##
## Call:
## roc.default(response = standardize_Diab$Outcome, predictor = nbc_lr_prob)
##
## Data: nbc_lr_prob in 500 controls (standardize_Diab$Outcome 0) < 268 cases (standardize_Diab$Outcome 1).
## Area under the curve: 0.8243
```

```
#AUC : 0.8243

plot(roc_nbc_diab)
```




```r
# Train the Naive bayes model using the train function defined above
diab_nbc_train <- train(Outcome ~ Pregnancies + Glucose + BloodPressure +
            SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
            data = standardize_Diab, method = "nb", trControl = train_ctrl)
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
```

```
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
## Warning in FUN(X[[i]], ...): Numerical 0 probability for all classes with
## observation 1
```

```r
# Calculate the test error rate
test_error_rate <- 1 - diab_nbc_train$results$Accuracy
print(paste("Test Error Rate using LOOCV:", test_error_rate))
```

```
## [1] "Test Error Rate using LOOCV: 0.247395833333333"
## [2] "Test Error Rate using LOOCV: 0.25"
```

(g):

```
#LDA on data for all predictors
diab_lda_all <- lda(Outcome ~ Pregnancies + Glucose + BloodPressure + Insulin +
                BMI + DiabetesPedigreeFunction + Age, data = standardize_Diab)

# Get predictions
lda_pred_all <- predict(diab_lda_all, standardize_Diab)$class

# Confusion matrix
cm_lda_all <- confusionMatrix(lda_pred_all, standardize_Diab$Outcome)
cm_lda_all
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 446 112
##          1  54 156
##
##                Accuracy : 0.7839
##                  95% CI : (0.753, 0.8125)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : 7.051e-16
##
##                   Kappa : 0.4992
##
##  Mcnemar's Test P-Value : 9.686e-06
##
##             Sensitivity : 0.8920
##             Specificity : 0.5821
##          Pos Pred Value : 0.7993
##          Neg Pred Value : 0.7429
##              Prevalence : 0.6510
##          Detection Rate : 0.5807
##    Detection Prevalence : 0.7266
##       Balanced Accuracy : 0.7370
##
##        'Positive' Class : 0
##
```

```
# Misclassification rate
mcr_lda_all <- 1 - cm_lda_all$overall['Accuracy']
mcr_lda_all
```

```
##  Accuracy
## 0.2161458
```

```
lda_lr_prob_all <- predict(diab_lda_all, standardize_Diab)$posterior[,2]

#ROC
roc_lda_diab_all <- roc(standardize_Diab$Outcome, lda_lr_prob_all)
```

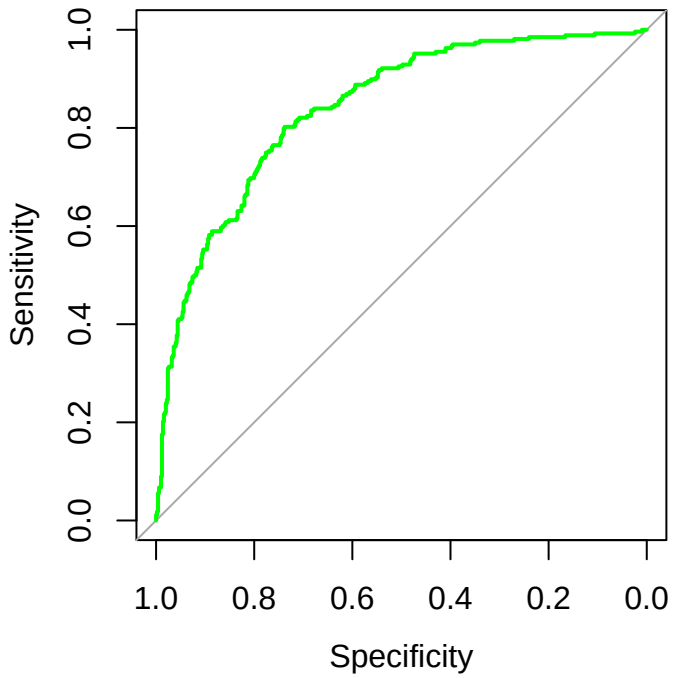
```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
roc_lda_diab_all
```

```
##
## Call:
## roc.default(response = standardize_Diab$Outcome, predictor = lda_lr_prob_all)
##
## Data: lda_lr_prob_all in 500 controls (standardize_Diab$Outcome 0) < 268 cases (standardize_Diab$Outcome 1)
## Area under the curve: 0.8394
```

```
#AUC : 0.8394

plot(roc_lda_diab_all, col = "green")
```




```
# Train the LDA model using the train function defined above
diab_lda_good_train <- train(Outcome ~ Pregnancies + Glucose + BloodPressure +
          SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
```

```
          data = standardize_Diab, method = "lda", trControl = train_ctrl)

# Calculate the test error rate
test_error_rate <- 1 - diab_lda_good_train$results$Accuracy
print(paste("Test Error Rate using LOOCV:", test_error_rate))
```

```
## [1] "Test Error Rate using LOOCV: 0.225260416666667"
```

(h):

```
set.seed(2)

#Train a KNN model using method = knn for k = 1 to 100 for finding optimal K and loocv error rate all at once
knn_loocv_h <- train(form = Outcome ~ Pregnancies + Glucose + BloodPressure +
      SkinThickness + Insulin + BMI + DiabetesPedigreeFunction + Age,
      data = standardize_Diab, method = "knn", trControl = train_ctrl,
      tuneGrid = expand.grid(k = seq(from = 1, to = 100, by = 1)))

#Optimal knn = 23
knn_loocv_h
```

```
## k-Nearest Neighbors
##
## 768 samples
##   8 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 767, 767, 767, 767, 767, 767, ...
## Resampling results across tuning parameters:
##
##   k    Accuracy   Kappa
##     1  0.7070312  0.3432854
##     2  0.6861979  0.3015125
##     3  0.7356771  0.4074975
##     4  0.7356771  0.4000924
##     5  0.7421875  0.4111889
##     6  0.7278646  0.3767976
##     7  0.7395833  0.4084299
##     8  0.7486979  0.4265730
##     9  0.7395833  0.4030964
##    10  0.7382812  0.4006523
##    11  0.7486979  0.4245069
##    12  0.7408854  0.4033915
##    13  0.7369792  0.3938485
##    14  0.7395833  0.3965680
```

```
##     15  0.7369792  0.3883036
##     16  0.7408854  0.3990374
##     17  0.7447917  0.4086366
##     18  0.7460938  0.4078760
##     19  0.7421875  0.3982144
##     20  0.7486979  0.4117909
##     21  0.7591146  0.4330407
##     22  0.7565104  0.4269114
##     23  0.7604167  0.4355828
##     24  0.7591146  0.4288103
##     25  0.7578125  0.4294479
##     26  0.7552083  0.4200887
##     27  0.7460938  0.3979352
##     28  0.7526042  0.4128251
##     29  0.7460938  0.3956807
##     30  0.7526042  0.4095176
##     31  0.7591146  0.4266714
##     32  0.7552083  0.4179191
##     33  0.7500000  0.4010528
##     34  0.7539062  0.4120735
##     35  0.7526042  0.4061727
##     36  0.7513021  0.4047333
##     37  0.7539062  0.4087399
##     38  0.7500000  0.3953549
##     39  0.7500000  0.3965032
##     40  0.7513021  0.3979314
##     41  0.7526042  0.4027896
##     42  0.7526042  0.4027896
##     43  0.7500000  0.3942023
##     44  0.7513021  0.3979314
##     45  0.7552083  0.4045530
##     46  0.7539062  0.4019579
##     47  0.7578125  0.4108875
##     48  0.7539062  0.4008124
##     49  0.7526042  0.3970647
##     50  0.7513021  0.3944718
##     51  0.7552083  0.4034113
##     52  0.7539062  0.4008124
##     53  0.7460938  0.3806042
##     54  0.7578125  0.4086242
##     55  0.7552083  0.4034113
##     56  0.7460938  0.3794132
##     57  0.7552083  0.4011149
##     58  0.7526042  0.3935767
##     59  0.7578125  0.4074861
##     60  0.7526042  0.3912289
##     61  0.7591146  0.4078223
##     62  0.7578125  0.4040451
##     63  0.7565104  0.4014204
##     64  0.7552083  0.3952959
```

```
##    65  0.7539062  0.3914881
##    66  0.7526042  0.3900482
##    67  0.7591146  0.4055230
##    68  0.7591146  0.4043666
##    69  0.7565104  0.3955759
##    70  0.7578125  0.3993946
##    71  0.7591146  0.4008704
##    72  0.7552083  0.3917498
##    73  0.7565104  0.3943933
##    74  0.7565104  0.3967539
##    75  0.7578125  0.3982205
##    76  0.7591146  0.4020403
##    77  0.7552083  0.3893626
##    78  0.7526042  0.3840751
##    79  0.7513021  0.3802265
##    80  0.7526042  0.3852791
##    81  0.7486979  0.3725065
##    82  0.7513021  0.3814391
##    83  0.7460938  0.3660040
##    84  0.7460938  0.3647562
##    85  0.7500000  0.3763703
##    86  0.7486979  0.3761826
##    87  0.7500000  0.3775917
##    88  0.7513021  0.3814391
##    89  0.7513021  0.3814391
##    90  0.7513021  0.3814391
##    91  0.7500000  0.3763703
##    92  0.7473958  0.3735621
##    93  0.7486979  0.3725065
##    94  0.7486979  0.3737367
##    95  0.7500000  0.3775917
##    96  0.7460938  0.3647562
##    97  0.7513021  0.3777868
##    98  0.7486979  0.3725065
##    99  0.7539062  0.3855116
##   100  0.7552083  0.3905585
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 23.
```

```r
# Extract predictions for our knn model above
diab_knn_pred <- knn_loocv_h$pred

# Calculate the test error rate using loocv
error_rate_loocv <- mean(diab_knn_pred$pred != diab_knn_pred$obs)
error_rate_loocv
```

```
## [1] 0.2501823
```

```r
# Confusion matrix
cm_knn <- confusionMatrix(diab_knn_pred$pred, diab_knn_pred$obs)
cm_knn
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 45497 14711
##          1  4503 12089
##
##                Accuracy : 0.7498
##                  95% CI : (0.7467, 0.7529)
##     No Information Rate : 0.651
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.396
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9099
##             Specificity : 0.4511
##          Pos Pred Value : 0.7557
##          Neg Pred Value : 0.7286
##              Prevalence : 0.6510
##          Detection Rate : 0.5924
##    Detection Prevalence : 0.7840
##       Balanced Accuracy : 0.6805
##
##        'Positive' Class : 0
##
```

```r
# Misclassification/error rate matches the loocv error rate
mcr_knn <- 1 - cm_knn$overall['Accuracy']
mcr_knn
```

```
##  Accuracy
## 0.2501823
```

```r
#ROC
roc_knn_diab <- roc(diab_knn_pred$obs, as.numeric(diab_knn_pred$pred))
```

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

```
## Setting direction: controls < cases
```

```r
roc_knn_diab
```

```
##
```

```
## Call:
## roc.default(response = diab_knn_pred$obs, predictor = as.numeric(diab_knn_pred$pred))
##
## Data: as.numeric(diab_knn_pred$pred) in 50000 controls (diab_knn_pred$obs 0) < 26800 cases (diab_knn_pred$o
## Area under the curve: 0.6805
```

```
#AUC : 0.6804

plot(roc_knn_diab, col = "red")
```