Mod4\_Assign2-Quiz2-ClassificationTrees

Grace Williams

2023-06-07

## : For this assignment you will need the following libraries: tidyverse, tidymodels, caret, rpart, rpart.plot, rattle, and RColorBrewer.

## Before beginning the assignment tasks, you should read-in the data for the assignment into a data frame called “heart”.

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.2 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(tidymodels)

## ── Attaching packages ────────────────────────────────────── tidymodels 1.1.0 ──  
## ✔ broom 1.0.5 ✔ rsample 1.1.1  
## ✔ dials 1.2.0 ✔ tune 1.1.1  
## ✔ infer 1.0.4 ✔ workflows 1.1.3  
## ✔ modeldata 1.1.0 ✔ workflowsets 1.0.1  
## ✔ parsnip 1.1.0 ✔ yardstick 1.2.0  
## ✔ recipes 1.0.6

## Warning: package 'broom' was built under R version 4.3.1

## ── Conflicts ───────────────────────────────────────── tidymodels\_conflicts() ──  
## ✖ scales::discard() masks purrr::discard()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ recipes::fixed() masks stringr::fixed()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ yardstick::spec() masks readr::spec()  
## ✖ recipes::step() masks stats::step()  
## • Dig deeper into tidy modeling with R at https://www.tmwr.org

library(caret)

## Loading required package: lattice  
##   
## Attaching package: 'caret'  
##   
## The following objects are masked from 'package:yardstick':  
##   
## precision, recall, sensitivity, specificity  
##   
## The following object is masked from 'package:purrr':  
##   
## lift

library(rpart)

##   
## Attaching package: 'rpart'  
##   
## The following object is masked from 'package:dials':  
##   
## prune

library(rpart.plot)  
library(rattle)

## Loading required package: bitops  
## Rattle: A free graphical interface for data science with R.  
## Version 5.5.1 Copyright (c) 2006-2021 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(RColorBrewer)  
library(readr)  
library(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

library(mice)

##   
## Attaching package: 'mice'  
##   
## The following object is masked from 'package:stats':  
##   
## filter  
##   
## The following objects are masked from 'package:base':  
##   
## cbind, rbind

library(VIM)

## Loading required package: colorspace  
## Loading required package: grid  
## The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,  
## which was just loaded, will retire in October 2023.  
## Please refer to R-spatial evolution reports for details, especially  
## https://r-spatial.org/r/2023/05/15/evolution4.html.  
## It may be desirable to make the sf package available;  
## package maintainers should consider adding sf to Suggests:.  
## The sp package is now running under evolution status 2  
## (status 2 uses the sf package in place of rgdal)  
## VIM is ready to use.  
##   
## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues  
##   
## Attaching package: 'VIM'  
##   
## The following object is masked from 'package:rattle':  
##   
## wine  
##   
## The following object is masked from 'package:recipes':  
##   
## prepare  
##   
## The following object is masked from 'package:datasets':  
##   
## sleep

heart <- read\_csv("heart\_disease-1.csv")

## Rows: 918 Columns: 12  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (5): Sex, ChestPainType, RestingECG, ExerciseAngina, ST\_Slope  
## dbl (7): Age, RestingBP, Cholesterol, FastingBS, MaxHR, Oldpeak, HeartDisease  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

## Then carefully convert the “sex”, “ChestPainType”, “RestingECG”, “ExerciseAngina”, “ST\_Slope”, and “HeartDisease” variables to factors. Recode the levels of the “HeartDisease” variable from “0” to “No” and “1” to “Yes”

heart = heart %>% mutate(Sex = as\_factor(Sex)) %>%  
 mutate(ChestPainType = as\_factor(ChestPainType)) %>%  
 mutate(RestingECG = as\_factor(RestingECG)) %>%  
 mutate(ExerciseAngina = as\_factor(ExerciseAngina)) %>%  
 mutate(ST\_Slope = as\_factor(ST\_Slope)) %>%  
 mutate(HeartDisease = as\_factor(HeartDisease)) %>%  
 mutate(HeartDisease = fct\_recode(HeartDisease, "No" = "0", "Yes" = "1" ))

## Question 1: Split the data into training and testing sets. Your training set should have 70% of the data. Use a random number (set.seed) of 12345. Stratify your split by the response variable “HeartDisease”.

## How many rows are in the training set?

### 642

set.seed(12345)   
heart\_split = initial\_split(heart, prop = 0.7, strata = HeartDisease) #70% in training  
train = training(heart\_split)   
test = testing(heart\_split)

## Question 2: Create a classification tree to predict “HeartDisease” in the training set (using all of the other variables as predictors). Plot the tree. You do not need to manually tune the complexity parameter (i.e., it’s OK to allow R to try different cp values on its own). Do not use k-folds at this point.

## The first split in the tree is a split on which variable?

## A. Sex

## B. ST\_Slope

## C. ChestPainType

## D. ExerciseAngina

### B

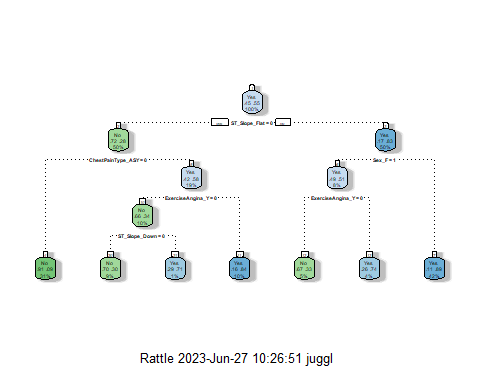
train\_recipe = recipe(HeartDisease ~ Sex +  
 ChestPainType +  
 RestingECG +  
 ExerciseAngina +  
 ST\_Slope, train) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification")  
  
train\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(train\_recipe)  
  
train\_fit = fit(train\_wflow, train)  
summary(train\_fit$fit$fit$fit)

## Call:  
## rpart::rpart(formula = ..y ~ ., data = data, model = ~TRUE)  
## n= 642   
##   
## CP nsplit rel error xerror xstd  
## 1 0.49477352 0 1.0000000 1.0000000 0.04389406  
## 2 0.07317073 1 0.5052265 0.5052265 0.03691584  
## 3 0.06968641 2 0.4320557 0.4738676 0.03607408  
## 4 0.01742160 3 0.3623693 0.3797909 0.03314572  
## 5 0.01045296 5 0.3275261 0.3449477 0.03188366  
## 6 0.01000000 6 0.3170732 0.3449477 0.03188366  
##   
## Variable importance  
## ST\_Slope\_Flat ExerciseAngina\_Y ChestPainType\_ASY ST\_Slope\_Down   
## 33 24 23 10   
## Sex\_F RestingECG\_ST ChestPainType\_NAP   
## 8 2 1   
##   
## Node number 1: 642 observations, complexity param=0.4947735  
## predicted class=Yes expected loss=0.4470405 P(node) =1  
## class counts: 287 355  
## probabilities: 0.447 0.553   
## left son=2 (322 obs) right son=3 (320 obs)  
## Primary splits:  
## ST\_Slope\_Flat < 0.5 to the left, improve=96.61556, (0 missing)  
## ExerciseAngina\_Y < 0.5 to the left, improve=88.29278, (0 missing)  
## ChestPainType\_ASY < 0.5 to the left, improve=82.97138, (0 missing)  
## Sex\_F < 0.5 to the right, improve=28.83194, (0 missing)  
## ChestPainType\_NAP < 0.5 to the right, improve=11.31223, (0 missing)  
## Surrogate splits:  
## ExerciseAngina\_Y < 0.5 to the left, agree=0.693, adj=0.384, (0 split)  
## ChestPainType\_ASY < 0.5 to the left, agree=0.643, adj=0.284, (0 split)  
## ST\_Slope\_Down < 0.5 to the right, agree=0.572, adj=0.141, (0 split)  
## Sex\_F < 0.5 to the right, agree=0.542, adj=0.081, (0 split)  
## RestingECG\_ST < 0.5 to the left, agree=0.522, adj=0.041, (0 split)  
##   
## Node number 2: 322 observations, complexity param=0.07317073  
## predicted class=No expected loss=0.2795031 P(node) =0.5015576  
## class counts: 232 90  
## probabilities: 0.720 0.280   
## left son=4 (197 obs) right son=5 (125 obs)  
## Primary splits:  
## ChestPainType\_ASY < 0.5 to the left, improve=37.887450, (0 missing)  
## ExerciseAngina\_Y < 0.5 to the left, improve=36.648680, (0 missing)  
## ST\_Slope\_Down < 0.5 to the left, improve=23.817100, (0 missing)  
## Sex\_F < 0.5 to the right, improve= 7.070518, (0 missing)  
## ChestPainType\_NAP < 0.5 to the right, improve= 4.531742, (0 missing)  
## Surrogate splits:  
## ExerciseAngina\_Y < 0.5 to the left, agree=0.761, adj=0.384, (0 split)  
## ST\_Slope\_Down < 0.5 to the left, agree=0.683, adj=0.184, (0 split)  
## ChestPainType\_NAP < 0.5 to the right, agree=0.627, adj=0.040, (0 split)  
## RestingECG\_ST < 0.5 to the left, agree=0.624, adj=0.032, (0 split)  
##   
## Node number 3: 320 observations, complexity param=0.0174216  
## predicted class=Yes expected loss=0.171875 P(node) =0.4984424  
## class counts: 55 265  
## probabilities: 0.172 0.828   
## left son=6 (53 obs) right son=7 (267 obs)  
## Primary splits:  
## Sex\_F < 0.5 to the right, improve=12.902810, (0 missing)  
## ChestPainType\_ASY < 0.5 to the left, improve= 9.359420, (0 missing)  
## ExerciseAngina\_Y < 0.5 to the left, improve= 7.507435, (0 missing)  
## ChestPainType\_NAP < 0.5 to the right, improve= 4.335220, (0 missing)  
## RestingECG\_LVH < 0.5 to the right, improve= 1.505707, (0 missing)  
##   
## Node number 4: 197 observations  
## predicted class=No expected loss=0.08629442 P(node) =0.3068536  
## class counts: 180 17  
## probabilities: 0.914 0.086   
##   
## Node number 5: 125 observations, complexity param=0.06968641  
## predicted class=Yes expected loss=0.416 P(node) =0.194704  
## class counts: 52 73  
## probabilities: 0.416 0.584   
## left son=10 (64 obs) right son=11 (61 obs)  
## Primary splits:  
## ExerciseAngina\_Y < 0.5 to the left, improve=15.1396900, (0 missing)  
## ST\_Slope\_Down < 0.5 to the left, improve= 8.8502860, (0 missing)  
## Sex\_F < 0.5 to the right, improve= 2.1009240, (0 missing)  
## RestingECG\_LVH < 0.5 to the left, improve= 0.5760000, (0 missing)  
## RestingECG\_ST < 0.5 to the left, improve= 0.1921404, (0 missing)  
## Surrogate splits:  
## ST\_Slope\_Down < 0.5 to the left, agree=0.680, adj=0.344, (0 split)  
## Sex\_F < 0.5 to the right, agree=0.560, adj=0.098, (0 split)  
## RestingECG\_ST < 0.5 to the left, agree=0.544, adj=0.066, (0 split)  
## RestingECG\_LVH < 0.5 to the left, agree=0.520, adj=0.016, (0 split)  
##   
## Node number 6: 53 observations, complexity param=0.0174216  
## predicted class=Yes expected loss=0.490566 P(node) =0.08255452  
## class counts: 26 27  
## probabilities: 0.491 0.509   
## left son=12 (30 obs) right son=13 (23 obs)  
## Primary splits:  
## ExerciseAngina\_Y < 0.5 to the left, improve=4.28766700, (0 missing)  
## ChestPainType\_ASY < 0.5 to the left, improve=2.12245000, (0 missing)  
## ChestPainType\_NAP < 0.5 to the right, improve=1.90448500, (0 missing)  
## RestingECG\_LVH < 0.5 to the right, improve=0.24880780, (0 missing)  
## RestingECG\_ST < 0.5 to the left, improve=0.06199461, (0 missing)  
## Surrogate splits:  
## ChestPainType\_ASY < 0.5 to the left, agree=0.717, adj=0.348, (0 split)  
## ChestPainType\_NAP < 0.5 to the right, agree=0.623, adj=0.130, (0 split)  
## RestingECG\_ST < 0.5 to the left, agree=0.623, adj=0.130, (0 split)  
## RestingECG\_LVH < 0.5 to the right, agree=0.585, adj=0.043, (0 split)  
##   
## Node number 7: 267 observations  
## predicted class=Yes expected loss=0.1086142 P(node) =0.4158879  
## class counts: 29 238  
## probabilities: 0.109 0.891   
##   
## Node number 10: 64 observations, complexity param=0.01045296  
## predicted class=No expected loss=0.34375 P(node) =0.09968847  
## class counts: 42 22  
## probabilities: 0.656 0.344   
## left son=20 (57 obs) right son=21 (7 obs)  
## Primary splits:  
## ST\_Slope\_Down < 0.5 to the left, improve=2.15820800, (0 missing)  
## Sex\_F < 0.5 to the right, improve=0.41647810, (0 missing)  
## RestingECG\_LVH < 0.5 to the left, improve=0.15705130, (0 missing)  
## RestingECG\_ST < 0.5 to the right, improve=0.04242081, (0 missing)  
##   
## Node number 11: 61 observations  
## predicted class=Yes expected loss=0.1639344 P(node) =0.09501558  
## class counts: 10 51  
## probabilities: 0.164 0.836   
##   
## Node number 12: 30 observations  
## predicted class=No expected loss=0.3333333 P(node) =0.04672897  
## class counts: 20 10  
## probabilities: 0.667 0.333   
##   
## Node number 13: 23 observations  
## predicted class=Yes expected loss=0.2608696 P(node) =0.03582555  
## class counts: 6 17  
## probabilities: 0.261 0.739   
##   
## Node number 20: 57 observations  
## predicted class=No expected loss=0.2982456 P(node) =0.08878505  
## class counts: 40 17  
## probabilities: 0.702 0.298   
##   
## Node number 21: 7 observations  
## predicted class=Yes expected loss=0.2857143 P(node) =0.01090343  
## class counts: 2 5  
## probabilities: 0.286 0.714

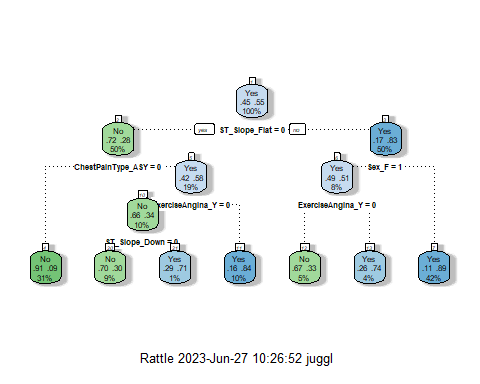
tree = train\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

## Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
## ℹ Please use `extract\_fit\_parsnip()` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

fancyRpartPlot(tree)



fancyRpartPlot(tree, tweak=1.5) #tweak makes the tree a little easier to read



## Question 3: Examine the complexity parameter (cp) values tried by R.

## Which cp value is “optimal” (recall that the optimal cp corresponds to the minimized “xerror” value)? Report your answer to two decimal places.

### 0.01

train\_fit$fit$fit$fit$cptable

## CP nsplit rel error xerror xstd  
## 1 0.49477352 0 1.0000000 1.0000000 0.04389406  
## 2 0.07317073 1 0.5052265 0.5052265 0.03691584  
## 3 0.06968641 2 0.4320557 0.4738676 0.03607408  
## 4 0.01742160 3 0.3623693 0.3797909 0.03314572  
## 5 0.01045296 5 0.3275261 0.3449477 0.03188366  
## 6 0.01000000 6 0.3170732 0.3449477 0.03188366

## Question 4: Use a tuning grid (as we did in the Titanic problem) to allow R to try 25 different values for the complexity parameter (cp). R will select reasonable values. Use 5-fold k-fold cross-validation (don’t forget to set up your folds). Use a seed of 123 when setting up your folds.

## Hint: You can reuse the vast majority of the code that I provided for you. Be careful to change names and you should be “good to go”. Note: This model took about two minutes to run on my computer. Your run time will vary by your computational power :) Plot the relationship between the complexity parameter (cp) and model performance (given by accuracy and by ROC AUC). I have provided code in the lectures that uses the “collect\_metrics” functions to help you do this.

## From this plot, what is the accuracy of the model (to two decimal places) if a cp value of 0.1 is selected? You will need to “eyeball” this answer. I have included a bit of a tolerance in the answer on Canvas. As long as you are “close” to the correct accuracy, you will see your answer marked as correct.

### 0.79

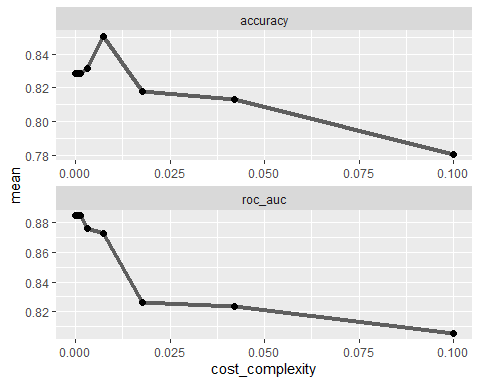
set.seed(123)  
folds = vfold\_cv(train, v = 5)

train\_recipe = recipe(HeartDisease ~., train) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification")  
  
tree\_grid = grid\_regular(cost\_complexity(),  
 levels = 25)  
  
  
train\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(train\_recipe)  
  
tree\_res =   
 train\_wflow %>%   
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid  
 )  
  
tree\_res

## # Tuning results  
## # 5-fold cross-validation   
## # A tibble: 5 × 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [513/129]> Fold1 <tibble [50 × 5]> <tibble [0 × 3]>  
## 2 <split [513/129]> Fold2 <tibble [50 × 5]> <tibble [0 × 3]>  
## 3 <split [514/128]> Fold3 <tibble [50 × 5]> <tibble [0 × 3]>  
## 4 <split [514/128]> Fold4 <tibble [50 × 5]> <tibble [0 × 3]>  
## 5 <split [514/128]> Fold5 <tibble [50 × 5]> <tibble [0 × 3]>

tree\_res %>%  
 collect\_metrics() %>%  
 ggplot(aes(cost\_complexity, mean)) +  
 geom\_line(size = 1.5, alpha = 0.6) +  
 geom\_point(size = 2) +  
 facet\_wrap(~ .metric, scales = "free", nrow = 2)

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



best\_tree = tree\_res %>%  
 select\_best("accuracy")  
  
best\_tree

## # A tibble: 1 × 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.00750 Preprocessor1\_Model22

## Question 5: Which cp value (to four decimal places) yields the “optimal” accuracy value?

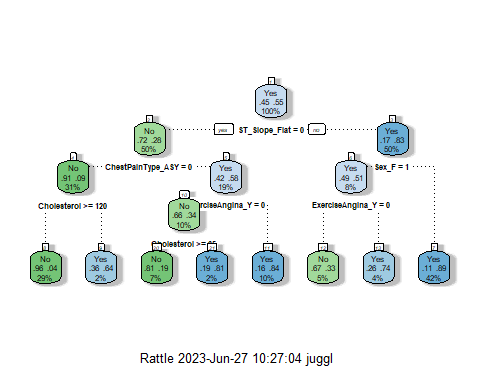
### 0.0075

## Question 6: Plot the tree that corresponds to the cp value from Task 5. Don’t forget to finalize your workflow and generate your final fit before trying to plot.

## How would you classify a patient that is “Male” with an “ST\_Slope” this not “Flat”?

### Yes

final\_wf =  
 train\_wflow %>%  
 finalize\_workflow(best\_tree)  
  
final\_fit = fit(final\_wf, train)  
  
tree = final\_fit %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")  
  
fancyRpartPlot(tree, tweak = 1.5)



## Question 7: What is the accuracy (on the training set) of the “tree” that you generated in Question 6?

## Take your time and think about how to determine this value. Report your answer to four decimal places.

### 0.8754

treepred = predict(final\_fit, train, type = "class")  
head(treepred)

## # A tibble: 6 × 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 No   
## 4 No   
## 5 No   
## 6 No

confusionMatrix(treepred$.pred\_class, train$HeartDisease, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 234 27  
## Yes 53 328  
##   
## Accuracy : 0.8754   
## 95% CI : (0.8473, 0.8999)  
## No Information Rate : 0.553   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.7457   
##   
## Mcnemar's Test P-Value : 0.005189   
##   
## Sensitivity : 0.9239   
## Specificity : 0.8153   
## Pos Pred Value : 0.8609   
## Neg Pred Value : 0.8966   
## Prevalence : 0.5530   
## Detection Rate : 0.5109   
## Detection Prevalence : 0.5935   
## Balanced Accuracy : 0.8696   
##   
## 'Positive' Class : Yes   
##

## Question 8 What is the sensitivity of your model from Question 6 (on the training set)? Report your answer to four decimal places.

### 0.9239

## Question 9 What is the naive accuracy of your model from Question 6 (on the training set)? Report your answer to four decimal places.

### 0.

treepred\_test = predict(final\_fit, test, type = "class")  
head(treepred\_test)

## # A tibble: 6 × 1  
## .pred\_class  
## <fct>   
## 1 No   
## 2 No   
## 3 No   
## 4 Yes   
## 5 No   
## 6 Yes

confusionMatrix(treepred\_test$.pred\_class, test$HeartDisease, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 99 18  
## Yes 24 135  
##   
## Accuracy : 0.8478   
## 95% CI : (0.7999, 0.8881)  
## No Information Rate : 0.5543   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.6905   
##   
## Mcnemar's Test P-Value : 0.4404   
##   
## Sensitivity : 0.8824   
## Specificity : 0.8049   
## Pos Pred Value : 0.8491   
## Neg Pred Value : 0.8462   
## Prevalence : 0.5543   
## Detection Rate : 0.4891   
## Detection Prevalence : 0.5761   
## Balanced Accuracy : 0.8436   
##   
## 'Positive' Class : Yes   
##

## Question 10 What is the accuracy of your model from Question 6 on the testing set (to four decimal places)?

### 0.8478