##Assignment: Random Forests (with Parameter Tuning)

#install.packages(c("tidyverse","caret","ranger","randomForest","e1071","rpart","rattle","RColorBrewer"))  
library(tidyverse)

## ── Attaching packages ──────────────────────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ ggplot2 3.3.0 ✓ purrr 0.3.3  
## ✓ tibble 3.0.0 ✓ dplyr 0.8.5  
## ✓ tidyr 1.0.2 ✓ stringr 1.4.0  
## ✓ readr 1.3.1 ✓ forcats 0.5.0

## ── Conflicts ─────────────────────────────────────────────────────────── tidyverse\_conflicts() ──  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(ranger)  
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ranger':  
##   
## importance

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

library(e1071)  
library(rpart)  
library(rattle)

## Rattle: A free graphical interface for data science with R.  
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

##   
## Attaching package: 'rattle'

## The following object is masked from 'package:randomForest':  
##   
## importance

## The following object is masked from 'package:ranger':  
##   
## importance

library(RColorBrewer)

For this assignment, we’ll use the same modified version of the Framingham heart disease dataset that we used in a previous assignment. The file is already uploaded to RStudio Cloud as “framingham\_edit.csv”. Read it in to R as a data frame named “heart”.

heart = read\_csv("framingham\_edit.csv")

## Parsed with column specification:  
## cols(  
## gender = col\_character(),  
## age = col\_double(),  
## currentSmoker = col\_character(),  
## cigsPerDay = col\_double(),  
## BPMeds = col\_character(),  
## prevalentStroke = col\_character(),  
## prevalentHyp = col\_character(),  
## diabetes = col\_character(),  
## totChol = col\_double(),  
## sysBP = col\_double(),  
## diaBP = col\_double(),  
## BMI = col\_double(),  
## heartRate = col\_double(),  
## glucose = col\_double(),  
## TenYearCHD = col\_character()  
## )

heart = heart %>% drop\_na()  
  
heart = heart %>% mutate\_at(c("gender","currentSmoker","BPMeds","prevalentStroke","prevalentHyp","diabetes","TenYearCHD"),as\_factor)  
  
str(heart)

## tibble [3,751 × 15] (S3: tbl\_df/tbl/data.frame)  
## $ gender : Factor w/ 2 levels "Male","Female": 1 2 1 2 2 2 2 2 1 1 ...  
## $ age : num [1:3751] 39 46 48 61 46 43 63 45 52 43 ...  
## $ currentSmoker : Factor w/ 2 levels "No","Yes": 1 1 2 2 2 1 1 2 1 2 ...  
## $ cigsPerDay : num [1:3751] 0 0 20 30 23 0 0 20 0 30 ...  
## $ BPMeds : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ prevalentStroke: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ prevalentHyp : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 2 1 1 2 2 ...  
## $ diabetes : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ totChol : num [1:3751] 195 250 245 225 285 228 205 313 260 225 ...  
## $ sysBP : num [1:3751] 106 121 128 150 130 ...  
## $ diaBP : num [1:3751] 70 81 80 95 84 110 71 71 89 107 ...  
## $ BMI : num [1:3751] 27 28.7 25.3 28.6 23.1 ...  
## $ heartRate : num [1:3751] 80 95 75 65 85 77 60 79 76 93 ...  
## $ glucose : num [1:3751] 77 76 70 103 85 99 85 78 79 88 ...  
## $ TenYearCHD : Factor w/ 2 levels "No","Yes": 1 1 1 2 1 1 2 1 1 1 ...

summary(heart)

## gender age currentSmoker cigsPerDay BPMeds   
## Male :1670 Min. :32.00 No :1919 Min. : 0.000 No :3637   
## Female:2081 1st Qu.:42.00 Yes:1832 1st Qu.: 0.000 Yes: 114   
## Median :49.00 Median : 0.000   
## Mean :49.57 Mean : 9.009   
## 3rd Qu.:56.00 3rd Qu.:20.000   
## Max. :70.00 Max. :70.000   
## prevalentStroke prevalentHyp diabetes totChol sysBP   
## No :3730 No :2581 No :3649 Min. :113.0 Min. : 83.5   
## Yes: 21 Yes:1170 Yes: 102 1st Qu.:206.0 1st Qu.:117.0   
## Median :234.0 Median :128.0   
## Mean :236.9 Mean :132.4   
## 3rd Qu.:264.0 3rd Qu.:144.0   
## Max. :696.0 Max. :295.0   
## diaBP BMI heartRate glucose TenYearCHD  
## Min. : 48.00 Min. :15.54 Min. : 44.0 Min. : 40.00 No :3179   
## 1st Qu.: 75.00 1st Qu.:23.09 1st Qu.: 68.0 1st Qu.: 71.00 Yes: 572   
## Median : 82.00 Median :25.41 Median : 75.0 Median : 78.00   
## Mean : 82.94 Mean :25.81 Mean : 75.7 Mean : 81.88   
## 3rd Qu.: 90.00 3rd Qu.:28.06 3rd Qu.: 82.0 3rd Qu.: 87.00   
## Max. :142.50 Max. :56.80 Max. :143.0 Max. :394.00

Split the data into training and testing sets using a random number seed of 999 and place **80%** of the data into the training set.

set.seed(999)   
train.rows = createDataPartition(y = heart$TenYearCHD, p = 0.8, list = FALSE)   
train = slice(heart,train.rows)   
test = slice(heart,-train.rows)

*Q1* Let’s start by building a single classification tree (using rpart in caret). Use a random number seed of 999. Use **5** fold cross-validation. Do basic parameter tuning on the cp parameter by setting a tuneLength of 10. Plot the tree. What value of cp did caret select as optimal?

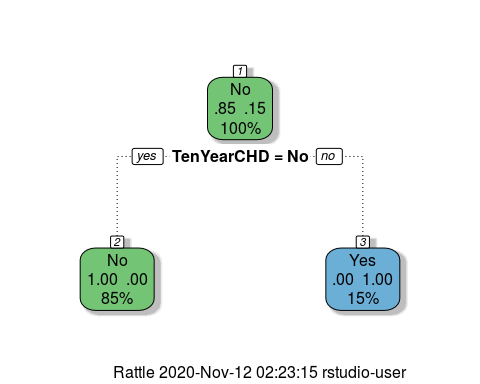
The final value selected by caret was cp = 0.8888889.

set.seed(999)  
ctrl = trainControl(method = "cv",  
 number = 5) #5 fold, k-fold cross-validation  
  
tree1 = train(x=as.data.frame(train[,-1]),y=train$TenYearCHD,  
 method = "rpart",  
 tuneLength = 10, #easiest way to set up cp tuning for trees is using tuneLength  
 #tuneLength of 5 means that R will try 5 different cp values  
 #larger values of tuneLength will take longer to run for large datasets  
 trControl = ctrl)

tree1

## CART   
##   
## 3002 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 2403, 2401, 2402, 2401, 2401   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa  
## 0.0000000 1.0000000 1   
## 0.1111111 1.0000000 1   
## 0.2222222 1.0000000 1   
## 0.3333333 1.0000000 1   
## 0.4444444 1.0000000 1   
## 0.5555556 1.0000000 1   
## 0.6666667 1.0000000 1   
## 0.7777778 1.0000000 1   
## 0.8888889 1.0000000 1   
## 1.0000000 0.8474358 0   
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.8888889.

fancyRpartPlot(tree1$finalModel)



*Q2* What is the accuracy of your tree from Q1 on the training and testing sets? Be sure to also note the naive accuracy on the training and testing sets (Hint: Look at the “No Information Rate” in the confusionMatrix output).

This tuned tree model gave an accuracy of 100%, rounded up from 99.88%, when used on the training data. When used on testing data, accuracy hit 99.52%. Before tuning, the naive model was 84.74% accurate. This model is optimized for determining if patients will have heart disease in the next 10 years.

trainpredtree = predict(tree1, type = "raw")  
confusionMatrix(trainpredtree, train$TenYearCHD,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2544 0  
## Yes 0 458  
##   
## Accuracy : 1   
## 95% CI : (0.9988, 1)  
## No Information Rate : 0.8474   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.1526   
## Detection Rate : 0.1526   
## Detection Prevalence : 0.1526   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##

testpredtree = predict(tree1, newdata = test, type = "raw")  
confusionMatrix(testpredtree, test$TenYearCHD,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 635 0  
## Yes 0 114  
##   
## Accuracy : 1   
## 95% CI : (0.9951, 1)  
## No Information Rate : 0.8478   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.1522   
## Detection Rate : 0.1522   
## Detection Prevalence : 0.1522   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##

*Q3* Build a random forest model using the randomForest package in caret. Use a random number seed of 999 and **5** fold cross-validation. Tune with mtry values from 1 to 14. What mtry value did caret select as optimal? NOTE: This model make take several minutes to run, be patient! It will also take this long to run when you knit (unless you choose to save to an RDS file as discussed in the lecture).

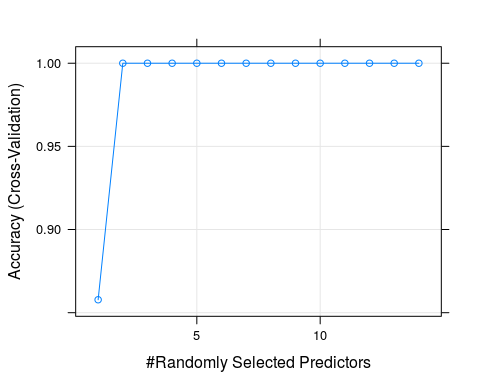
Caret selected mtry = 2 as the optimal value.

#only parameter we can tune in the randomForest package is mtry  
tgrid = expand.grid(  
 mtry = 1:14 #mtry CANNOT be larger than the possible number of predictor variables (6 in this case)  
 )  
  
#NOTE: Bigger tuning grids will take longer and make it more likely that you will run out of memory in RStudio Cloud on larger datasets.  
  
set.seed(999)  
rf1 = train(x=as.data.frame(train[,-1]),y=train$TenYearCHD,   
 method = "rf",   
 tuneGrid = tgrid,  
 trControl = ctrl) #same cross-validation set-up from before

print(rf1)

## Random Forest   
##   
## 3002 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 2403, 2401, 2402, 2401, 2401   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 1 0.8577625 0.1087928  
## 2 1.0000000 1.0000000  
## 3 1.0000000 1.0000000  
## 4 1.0000000 1.0000000  
## 5 1.0000000 1.0000000  
## 6 1.0000000 1.0000000  
## 7 1.0000000 1.0000000  
## 8 1.0000000 1.0000000  
## 9 1.0000000 1.0000000  
## 10 1.0000000 1.0000000  
## 11 1.0000000 1.0000000  
## 12 1.0000000 1.0000000  
## 13 1.0000000 1.0000000  
## 14 1.0000000 1.0000000  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 2.

plot(rf1)



*Q4* What is the most important variable from the model created in Q3?

varImp(rf1)

## rf variable importance  
##   
## Overall  
## TenYearCHD 100.0000  
## age 4.1486  
## sysBP 3.9347  
## diaBP 3.3007  
## BMI 2.9836  
## glucose 2.9740  
## totChol 2.7363  
## heartRate 1.9742  
## cigsPerDay 1.3938  
## prevalentHyp 0.9030  
## currentSmoker 0.2176  
## BPMeds 0.1956  
## diabetes 0.1954  
## prevalentStroke 0.0000

Age is the most significant variable in the model at 4.1486499 out of age, sysBP, diaBP, BMI, glucose, totChol, heartRate, cigsPerDay, and prevalentHyp.

*Q5* What is the accuracy of your random forest from Q3 on the training and testing sets? IMPORTANT NOTE: Random forest models using caret will sometimes display a perfect accuracy (accuracy = 1) on the training set when you use the confusionMatrix command. It is possible that this could happen here under certain circumstances. This accuracy is NOT correct. To get a better idea of accuracy on the training set, look back to where the accuracy values for the various mtry values are reported.

This model produces an accuracy of 99.88%

trainpredrf = predict(rf1, type = "raw")   
confusionMatrix(trainpredrf,train$TenYearCHD,  
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2544 0  
## Yes 0 458  
##   
## Accuracy : 1   
## 95% CI : (0.9988, 1)  
## No Information Rate : 0.8474   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.1526   
## Detection Rate : 0.1526   
## Detection Prevalence : 0.1526   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##

testpredrf = predict(rf1, newdata = test, type = "raw")   
confusionMatrix(testpredrf,test$TenYearCHD,  
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 635 0  
## Yes 0 114  
##   
## Accuracy : 1   
## 95% CI : (0.9951, 1)  
## No Information Rate : 0.8478   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.1522   
## Detection Rate : 0.1522   
## Detection Prevalence : 0.1522   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##

*Q6* Build a random forest model using the ranger package in caret. Use a random number seed of 999 and **5** fold cross-validation. Tune with the following: mtry values of 2,4,6, and 8 (use c(2,4,6,8) in the code) to 14, splitrule values of “gini” and “hellinger” (exclude extratrees), and min.node.size values from 1 to 5.  
What parameter values did caret select as optimal?

#more parameters available to tune using the ranger package for random forests  
tgrid = expand.grid(  
 mtry = c(2,4,6,8,10,12,14), #mtry CANNOT be larger than the possible number of predictor variables (6 in this case)  
 splitrule = c("gini", "hellinger"),  
 min.node.size = 1:5)  
  
#NOTE: Bigger tuning grids will take longer and make it more likely that you will run out of memory in RStudio Cloud on larger datasets.  
  
rf2 = train(x=as.data.frame(train[,-1]),y=train$TenYearCHD,   
 method = "ranger",   
 tuneGrid = tgrid,  
 importance = "permutation", #necessary for ranger  
 trControl = ctrl) #same cross-validation set-up from before

saveRDS(rf2, "rf2.rds")

NOTE: This model make take several minutes to run, be patient!

HINT: You might want to save your model to an RDS file. If you do this, you can comment out your model so that it does not run again when you knit. Be sure comment out the saving code too. You would only leave the code to read the model from the RDS file.

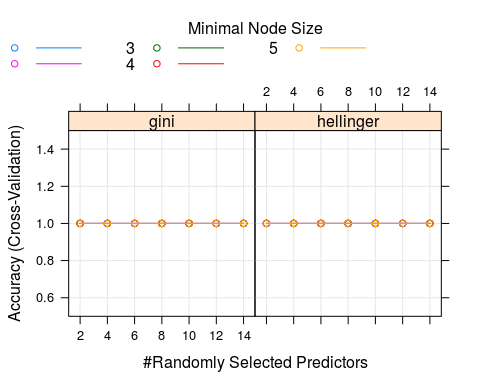
*Q7* What is the accuracy of your random forest from Q6 on the training and testing sets?

rf2 = readRDS("rf2.rds")

print(rf2)

## Random Forest   
##   
## 3002 samples  
## 14 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (5 fold)   
## Summary of sample sizes: 2402, 2402, 2402, 2401, 2401   
## Resampling results across tuning parameters:  
##   
## mtry splitrule min.node.size Accuracy Kappa  
## 2 gini 1 1 1   
## 2 gini 2 1 1   
## 2 gini 3 1 1   
## 2 gini 4 1 1   
## 2 gini 5 1 1   
## 2 hellinger 1 1 1   
## 2 hellinger 2 1 1   
## 2 hellinger 3 1 1   
## 2 hellinger 4 1 1   
## 2 hellinger 5 1 1   
## 4 gini 1 1 1   
## 4 gini 2 1 1   
## 4 gini 3 1 1   
## 4 gini 4 1 1   
## 4 gini 5 1 1   
## 4 hellinger 1 1 1   
## 4 hellinger 2 1 1   
## 4 hellinger 3 1 1   
## 4 hellinger 4 1 1   
## 4 hellinger 5 1 1   
## 6 gini 1 1 1   
## 6 gini 2 1 1   
## 6 gini 3 1 1   
## 6 gini 4 1 1   
## 6 gini 5 1 1   
## 6 hellinger 1 1 1   
## 6 hellinger 2 1 1   
## 6 hellinger 3 1 1   
## 6 hellinger 4 1 1   
## 6 hellinger 5 1 1   
## 8 gini 1 1 1   
## 8 gini 2 1 1   
## 8 gini 3 1 1   
## 8 gini 4 1 1   
## 8 gini 5 1 1   
## 8 hellinger 1 1 1   
## 8 hellinger 2 1 1   
## 8 hellinger 3 1 1   
## 8 hellinger 4 1 1   
## 8 hellinger 5 1 1   
## 10 gini 1 1 1   
## 10 gini 2 1 1   
## 10 gini 3 1 1   
## 10 gini 4 1 1   
## 10 gini 5 1 1   
## 10 hellinger 1 1 1   
## 10 hellinger 2 1 1   
## 10 hellinger 3 1 1   
## 10 hellinger 4 1 1   
## 10 hellinger 5 1 1   
## 12 gini 1 1 1   
## 12 gini 2 1 1   
## 12 gini 3 1 1   
## 12 gini 4 1 1   
## 12 gini 5 1 1   
## 12 hellinger 1 1 1   
## 12 hellinger 2 1 1   
## 12 hellinger 3 1 1   
## 12 hellinger 4 1 1   
## 12 hellinger 5 1 1   
## 14 gini 1 1 1   
## 14 gini 2 1 1   
## 14 gini 3 1 1   
## 14 gini 4 1 1   
## 14 gini 5 1 1   
## 14 hellinger 1 1 1   
## 14 hellinger 2 1 1   
## 14 hellinger 3 1 1   
## 14 hellinger 4 1 1   
## 14 hellinger 5 1 1   
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were mtry = 2, splitrule = gini  
## and min.node.size = 1.

plot(rf2)



varImp(rf2)

## ranger variable importance  
##   
## Overall  
## TenYearCHD 100.00000  
## sysBP 2.89693  
## diaBP 2.07034  
## prevalentHyp 1.95698  
## age 1.12837  
## BMI 0.56576  
## cigsPerDay 0.47359  
## totChol 0.31571  
## currentSmoker 0.29566  
## glucose 0.19242  
## BPMeds 0.11385  
## diabetes 0.08484  
## heartRate 0.03932  
## prevalentStroke 0.00000

trainpredrf2 = predict(rf2, type = "raw")   
confusionMatrix(trainpredrf2,train$TenYearCHD,  
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2544 0  
## Yes 0 458  
##   
## Accuracy : 1   
## 95% CI : (0.9988, 1)  
## No Information Rate : 0.8474   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.1526   
## Detection Rate : 0.1526   
## Detection Prevalence : 0.1526   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##

testpredrf2 = predict(rf2, newdata = test, type="raw")  
confusionMatrix(testpredrf2,test$TenYearCHD,  
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 635 0  
## Yes 0 114  
##   
## Accuracy : 1   
## 95% CI : (0.9951, 1)  
## No Information Rate : 0.8478   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.1522   
## Detection Rate : 0.1522   
## Detection Prevalence : 0.1522   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##

*Q8* Use the optimal parameters selected in *Q6* and add a max.depth = 5 parameter to the model. Recall that this is NOT added to the tuning grid. See the example from the lecture for where to put this parameter.

tgrid = expand.grid(  
 mtry = c(2,4,6,8,10,12,14), #mtry CANNOT be larger than the possible number of predictor variables (6 in this case)  
 splitrule = c("gini", "hellinger"),  
 min.node.size = 1:5)  
  
#NOTE: Bigger tuning grids will take longer and make it more likely that you will run out of memory in RStudio Cloud on larger datasets.  
  
rf3 = train(x=as.data.frame(train[,-1]),y=train$TenYearCHD,   
 method = "ranger",   
 max.depth = 5, #try 5 first and see what happens  
 tuneGrid = tgrid,  
 importance = "permutation", #necessary for ranger  
 trControl = ctrl) #same cross-validation set-up from before

saveRDS(rf3, "rf3.rds")

*Q9* What is the accuracy of your random forest from *Q8* on the training and testing sets?

rf3 = readRDS("rf3.rds")

trainpredrf3 = predict(rf3, type = "raw")   
confusionMatrix(trainpredrf3,train$TenYearCHD,  
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 2544 0  
## Yes 0 458  
##   
## Accuracy : 1   
## 95% CI : (0.9988, 1)  
## No Information Rate : 0.8474   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.1526   
## Detection Rate : 0.1526   
## Detection Prevalence : 0.1526   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##

testpredrf3 = predict(rf3, newdata = test, type="raw")  
confusionMatrix(testpredrf3,test$TenYearCHD,  
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 635 0  
## Yes 0 114  
##   
## Accuracy : 1   
## 95% CI : (0.9951, 1)  
## No Information Rate : 0.8478   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 1   
##   
## Mcnemar's Test P-Value : NA   
##   
## Sensitivity : 1.0000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 1.0000   
## Prevalence : 0.1522   
## Detection Rate : 0.1522   
## Detection Prevalence : 0.1522   
## Balanced Accuracy : 1.0000   
##   
## 'Positive' Class : Yes   
##