

Practice 3

Jordan Lian

2/14/2021

1. Download the [data set for the tutorial](#).

```
library(tidyverse)
```

```
## -- Attaching packages -----
```

```
## v ggplot2 3.3.3    v purrr  0.3.4
## v tibble  3.0.3    v dplyr  1.0.2
## v tidyr   1.1.2    v stringr 1.4.0
## v readr   1.3.1    v forcats 0.5.0
```

```
## -- Conflicts -----
```

```
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
origin_prostate <- read.csv("prostate_cancer.csv", stringsAsFactors = FALSE)
prostate <- origin_prostate
head(prostate)
```

```
##   id diagnosis_result radius texture perimeter area smoothness compactness
## 1  1                M    23     12        151  954      0.143      0.278
## 2  2                B     9     13        133 1326      0.143      0.079
## 3  3                M    21     27        130 1203      0.125      0.160
## 4  4                M    14     16         78  386      0.070      0.284
## 5  5                M     9     19        135 1297      0.141      0.133
## 6  6                B    25     25         83  477      0.128      0.170
##   symmetry fractal_dimension
## 1    0.242           0.079
## 2    0.181           0.057
## 3    0.207           0.060
## 4    0.260           0.097
## 5    0.181           0.059
## 6    0.209           0.076
```

2. Follow this [tutorial on applying kNN to prostate cancer detection](#) and implement all of the steps in an R Notebook. Make sure to explain each step and what it does. (*Note: The data set provided as part of this assignment has been slightly modified from the one used in the tutorial, so small deviations in the result can be expected.*)

Step 1: Data Collection

See Question 1

Step 2: Preparing and Exploring the Data

```
# remove the first variable (id) from the dataset
prostate <- prostate[-1]

# get table of number of patients based on diagnosis
table(prostate$diagnosis_result)
```

```
##
##  B  M
## 38 62
```

```
# Rename B and M to Benign and Malignant
prostate$diagnosis <- factor(prostate$diagnosis_result, levels = c("B", "M"),
                             labels = c("Benign", "Malignant"))
# Get percentage of B/M diagnoses
round(prop.table(table(prostate$diagnosis)) * 100, digits = 1)
```

```
##
##      Benign Malignant
##      38      62
```

Normalize Numeric Data

```
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }
norm_prostate <- as.data.frame(lapply(prostate[2:9], normalize))
summary(norm_prostate$radius)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.0000 0.1875 0.5000 0.4906 0.7500 1.0000
```

Create test and training data set

```
# Split training and test dataset by 65/35
prostate_train <- norm_prostate[1:65,]
prostate_test <- norm_prostate[66:100,]

# Target variable is 'diagnosis_result', which is not in the training/test datasets
train_labels <- prostate[1:65, 1]
test_labels <- prostate[66:100, 1]
```

Step 3 - Train the model on data

```
library(class)
prc_test_pred <- knn(train = prostate_train,
                     test = prostate_test,
                     cl = train_labels,
                     k=10)
```

Step 4 - Evaluate the model performance

```
library(gmodels)
CrossTable(x = test_labels, y = prc_test_pred, prop.chisq = FALSE)
```

```
##
##
##   Cell Contents
## |-----|
## |               N |
## |   N / Row Total |
## |   N / Col Total |
## |   N / Table Total |
## |-----|
##
##
## Total Observations in Table:  35
##
##
##      | prc_test_pred
## test_labels |      B |      M | Row Total |
## -----|-----|-----|-----|
##      B |      6 |     13 |      19 |
##      |    0.316 |    0.684 |    0.543 |
##      |    1.000 |    0.448 |          |
##      |    0.171 |    0.371 |          |
## -----|-----|-----|-----|
##      M |      0 |     16 |      16 |
##      |    0.000 |    1.000 |    0.457 |
##      |    0.000 |    0.552 |          |
##      |    0.000 |    0.457 |          |
## -----|-----|-----|-----|
## Column Total |      6 |     29 |      35 |
##      |    0.171 |    0.829 |          |
## -----|-----|-----|-----|
##
##
```

Step 5 - Improve the performance of the model, change the k-value

```
# k=8
prc_test_pred <- knn(train = prostate_train,
                     test = prostate_test,
                     cl = train_labels,
                     k = 8)
CrossTable(x = test_labels, y = prc_test_pred, prop.chisq = FALSE)
```

```
##
##
##      Cell Contents
## |-----|
## |                      N |
## |          N / Row Total |
## |          N / Col Total |
## |          N / Table Total |
## |-----|
##
##
## Total Observations in Table:  35
##
##
##      | prc_test_pred
## test_labels |      B |      M | Row Total |
## -----|-----|-----|-----|
##          B |      9 |     10 |      19 |
##          |    0.474 |    0.526 |    0.543 |
##          |    1.000 |    0.385 |          |
##          |    0.257 |    0.286 |          |
## -----|-----|-----|-----|
##          M |      0 |     16 |      16 |
##          |    0.000 |    1.000 |    0.457 |
##          |    0.000 |    0.615 |          |
##          |    0.000 |    0.457 |          |
## -----|-----|-----|-----|
## Column Total |      9 |     26 |      35 |
##          |    0.257 |    0.743 |          |
## -----|-----|-----|-----|
##
##
```

```
# k=9
prc_test_pred <- knn(train = prostate_train,
                     test = prostate_test,
                     cl = train_labels,
                     k = 9)
CrossTable(x = test_labels, y = prc_test_pred, prop.chisq = FALSE)
```

```
##
##
##      Cell Contents
## |-----|
## |                      N |
```

```
## |          N / Row Total |
## |          N / Col Total |
## |          N / Table Total |
## |-----|
##
##
## Total Observations in Table:  35
##
##
##          | prc_test_pred
## test_labels |          B |          M | Row Total |
## -----|-----|-----|-----|
##          B |          8 |         11 |         19 |
##          |         0.421 |         0.579 |         0.543 |
##          |         1.000 |         0.407 |         |
##          |         0.229 |         0.314 |         |
## -----|-----|-----|-----|
##          M |          0 |         16 |         16 |
##          |         0.000 |         1.000 |         0.457 |
##          |         0.000 |         0.593 |         |
##          |         0.000 |         0.457 |         |
## -----|-----|-----|-----|
## Column Total |          8 |         27 |         35 |
##          |         0.229 |         0.771 |         |
## -----|-----|-----|-----|
##
##
```

```
# k=11
prc_test_pred <- knn(train = prostate_train,
                     test = prostate_test,
                     cl = train_labels,
                     k = 11)
CrossTable(x = test_labels, y = prc_test_pred, prop.chisq = FALSE)
```

```
##
##
##      Cell Contents
## |-----|
## |          N |
## |          N / Row Total |
## |          N / Col Total |
## |          N / Table Total |
## |-----|
##
##
## Total Observations in Table:  35
##
##
##          | prc_test_pred
## test_labels |          B |          M | Row Total |
## -----|-----|-----|-----|
##          B |          5 |         14 |         19 |
##          |         0.263 |         0.737 |         0.543 |
```

```
##          |      1.000 |      0.467 |          |
##          |      0.143 |      0.400 |          |
## -----|-----|-----|-----|
##          M |          0 |          16 |          16 |
##          |      0.000 |          1.000 |          0.457 |
##          |      0.000 |          0.533 |          |
##          |      0.000 |          0.457 |          |
## -----|-----|-----|-----|
## Column Total |          5 |          30 |          35 |
##          |      0.143 |          0.857 |          |
## -----|-----|-----|-----|
##
##
```

```
# k=12
prc_test_pred <- knn(train = prostate_train,
                     test = prostate_test,
                     cl = train_labels,
                     k = 12)
CrossTable(x = test_labels, y = prc_test_pred, prop.chisq = FALSE)
```

```
##
##
##      Cell Contents
## |-----|
## |                      N |
## |      N / Row Total |
## |      N / Col Total |
## |      N / Table Total |
## |-----|
##
##
## Total Observations in Table:  35
##
##
##      | prc_test_pred
## test_labels |      B |      M | Row Total |
## -----|-----|-----|-----|
##      B |          5 |          14 |          19 |
##      |      0.263 |          0.737 |          0.543 |
##      |      1.000 |          0.467 |          |
##      |      0.143 |          0.400 |          |
## -----|-----|-----|-----|
##      M |          0 |          16 |          16 |
##      |      0.000 |          1.000 |          0.457 |
##      |      0.000 |          0.533 |          |
##      |      0.000 |          0.457 |          |
## -----|-----|-----|-----|
## Column Total |          5 |          30 |          35 |
##      |      0.143 |          0.857 |          |
## -----|-----|-----|-----|
##
##
```

k=9 produced the least amount of false positives and most accurate results overall, so I would use k=9 for kNN for this dataset.

3. Once you have complete the tutorial, try another kNN implementation from another package, such as the **caret** package. Compare the accuracy of the two implementations.

```
library(caret)
```

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

```
##
```

```
## Attaching package: 'caret'
```

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

```
##
```

```
## lift
```

Sampling

```
# Split data as training and test set. Using createDataPartition() function from caret
indxTrain <- createDataPartition(y = prostate$diagnosis_result, p = 0.75, list = FALSE)
training <- prostate[indxTrain,]
testing <- prostate[-indxTrain,]
```

```
# Check distribution in original data and partitioned data
prop.table(table(prostate$diagnosis_result)) * 100
```

```
##
```

```
## B M
```

```
## 38 62
```

```
prop.table(table(testing$diagnosis_result)) * 100
```

```
##
```

```
## B M
```

```
## 37.5 62.5
```

```
prop.table(table(prostate$diagnosis_result)) * 100
```

```
##
```

```
## B M
```

```
## 38 62
```

Preprocessing

```
trainX <- training[,names(training) != "diagnosis_result"]
preProcValues <- preProcess(x = trainX, method = c("center", "scale"))
preProcValues
```

```
## Created from 76 samples and 9 variables
##
## Pre-processing:
##   - centered (8)
##   - ignored (1)
##   - scaled (8)
```

Training and train control

```
set.seed(400)
ctrl <- trainControl(method="repeatedcv", repeats = 3)
knnFit <- train(diagnosis_result ~ .,
               data = training,
               method = "knn",
               trControl = ctrl,
               preProcess = c("center","scale"),
               tuneLength = 20)
```

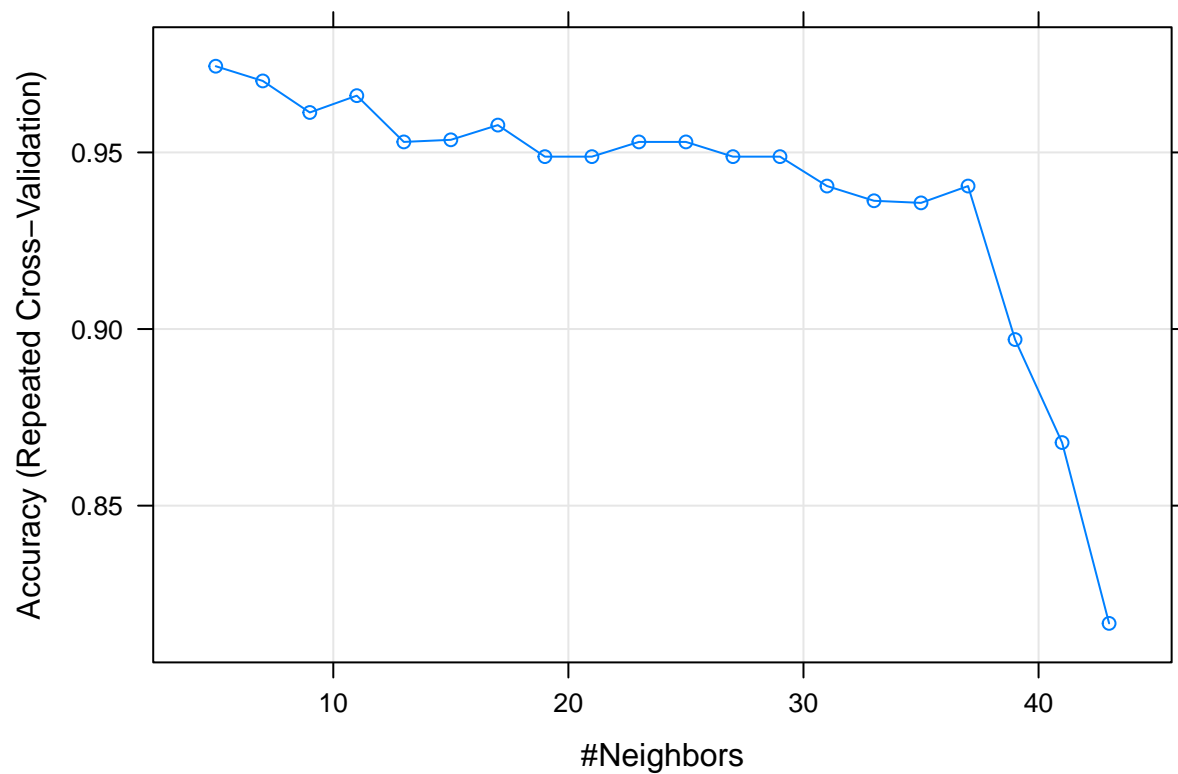
```
# kNNFit output
knnFit
```

```
## k-Nearest Neighbors
##
## 76 samples
## 9 predictor
## 2 classes: 'B', 'M'
##
## Pre-processing: centered (9), scaled (9)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 69, 69, 68, 68, 68, 69, ...
## Resampling results across tuning parameters:
##
##  k  Accuracy  Kappa
##  5  0.9744048  0.9371903
##  7  0.9702381  0.9276665
##  9  0.9613095  0.9079977
## 11  0.9660714  0.9181426
## 13  0.9529762  0.8889501
## 15  0.9535714  0.8895712
## 17  0.9577381  0.8990950
## 19  0.9488095  0.8794263
## 21  0.9488095  0.8794263
## 23  0.9529762  0.8889501
## 25  0.9529762  0.8889501
## 27  0.9488095  0.8794263
## 29  0.9488095  0.8794263
```



```
## 31 0.9404762 0.8589135
## 33 0.9363095 0.8493897
## 35 0.9357143 0.8487685
## 37 0.9404762 0.8589135
## 39 0.8970238 0.7574164
## 41 0.8678571 0.6839667
## 43 0.8166667 0.5470036
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

```
#Plot yields Number of Neighbors Vs accuracy (based on repeated cross validation)
plot(knnFit)
```



4. Try the `confusionMatrix` function from the **caret** package to determine the accuracy of both algorithms.

```
knnPredict <- predict(knnFit, newdata = testing)

# Get the confusion matrix to see accuracy value and other parameter values
confusionMatrix(knnPredict, as.factor(testing$diagnosis_result))
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction  B  M
##           B  8  1
##           M  1 14
##
##           Accuracy : 0.9167
##           95% CI : (0.73, 0.9897)
##           No Information Rate : 0.625
##           P-Value [Acc > NIR] : 0.001448
##
##           Kappa : 0.8222
##
## Mcnemar's Test P-Value : 1.000000
##
##           Sensitivity : 0.8889
##           Specificity : 0.9333
##           Pos Pred Value : 0.8889
##           Neg Pred Value : 0.9333
##           Prevalence : 0.3750
##           Detection Rate : 0.3333
##           Detection Prevalence : 0.3750
##           Balanced Accuracy : 0.9111
##
##           'Positive' Class : B
##
```

```
mean(knnPredict == testing$diagnosis_result)
```

```
## [1] 0.9166667
```

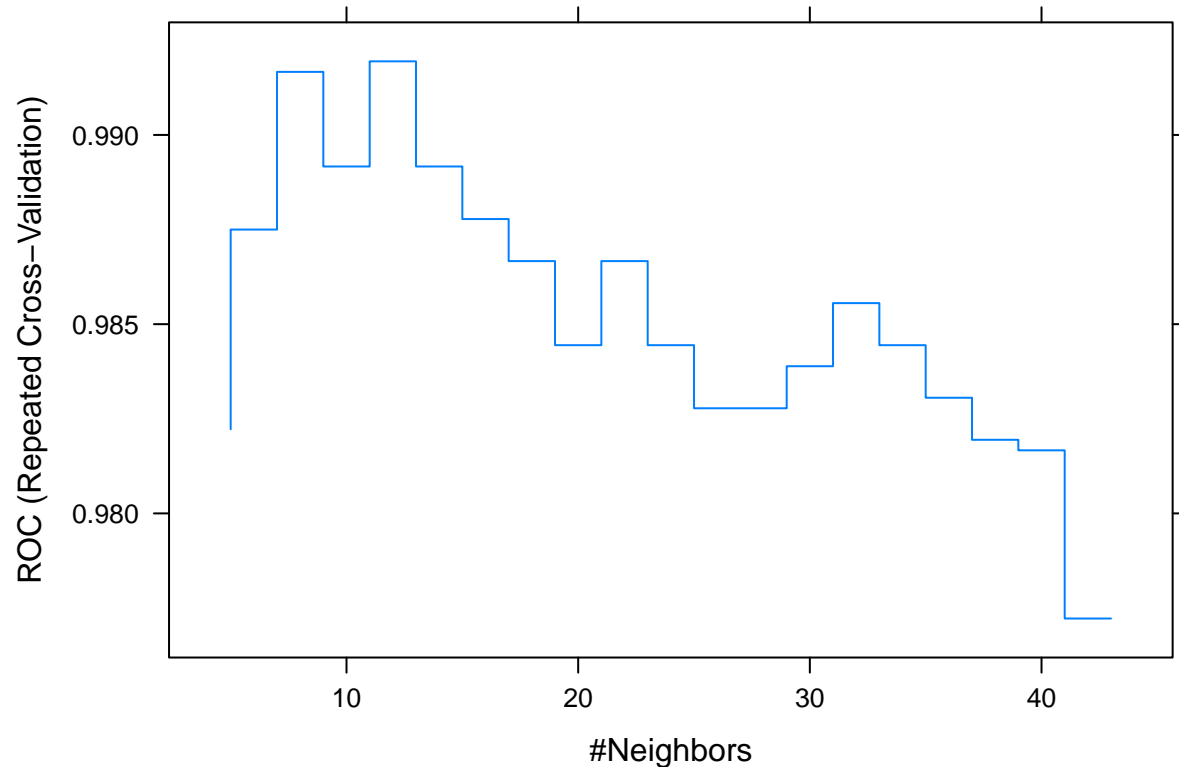
```
# Verify 2 class summary function
ctrl <- trainControl(method="repeatedcv",
                     repeats = 3,
                     classProbs = TRUE,
                     summaryFunction = twoClassSummary)
knnFit <- train(diagnosis_result ~ .,
               data = training,
               method = "knn",
               trControl = ctrl,
               preProcess = c("center", "scale"),
               tuneLength = 20)

knnFit
```

```
## k-Nearest Neighbors
##
## 76 samples
## 9 predictor
## 2 classes: 'B', 'M'
##
## Pre-processing: centered (9), scaled (9)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 69, 68, 68, 68, 68, 69, ...
## Resampling results across tuning parameters:
```

```
##
## k ROC Sens Spec
## 5 0.9822222 0.9277778 1
## 7 0.9875000 0.9000000 1
## 9 0.9916667 0.8777778 1
## 11 0.9891667 0.8777778 1
## 13 0.9919444 0.8333333 1
## 15 0.9891667 0.8555556 1
## 17 0.9877778 0.8555556 1
## 19 0.9866667 0.8666667 1
## 21 0.9844444 0.8555556 1
## 23 0.9866667 0.8555556 1
## 25 0.9844444 0.8722222 1
## 27 0.9827778 0.8722222 1
## 29 0.9827778 0.8555556 1
## 31 0.9838889 0.8555556 1
## 33 0.9855556 0.8333333 1
## 35 0.9844444 0.8333333 1
## 37 0.9830556 0.8222222 1
## 39 0.9819444 0.7111111 1
## 41 0.9816667 0.6888889 1
## 43 0.9772222 0.5888889 1
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 13.
```

```
plot(knnFit, print.thres = 0.5, type="S")
```



```
knnPredict <- predict(knnFit,newdata = testing)
```

```
# Get the confusion matrix to see accuracy value and other parameter values
confusionMatrix(knnPredict, as.factor(testing$diagnosis_result))
```

```
## Confusion Matrix and Statistics
```

```
##
```

```
##           Reference
```

```
## Prediction  B  M
```

```
##           B  8  0
```

```
##           M  1 15
```

```
##
```

```
##           Accuracy : 0.9583
```

```
##           95% CI : (0.7888, 0.9989)
```

```
## No Information Rate : 0.625
```

```
## P-Value [Acc > NIR] : 0.0001944
```

```
##
```

```
##           Kappa : 0.9091
```

```
##
```

```
## McNemar's Test P-Value : 1.0000000
```

```
##
```

```
##           Sensitivity : 0.8889
```

```
##           Specificity : 1.0000
```

```
## Pos Pred Value : 1.0000
```

```
## Neg Pred Value : 0.9375
```

```
##           Prevalence : 0.3750
##       Detection Rate : 0.3333
## Detection Prevalence : 0.3333
##       Balanced Accuracy : 0.9444
##
##       'Positive' Class : B
##
```

```
mean(knnPredict == testing$diagnosis_result)
```

```
## [1] 0.9583333
```