DA5030.A3.Parpattedar

Shruti Parpattedar

February 3, 2019

Question 1

Downloading and loading the dataset into R

```
#setwd("D:/NEU/DA5030/Assignment3")
prc <- read.csv("prostate_cancer.csv", stringsAsFactors = FALSE)
```

Question 2

Preparing and exploring the data

```
str(prc)
```

```
## 'data.frame':    100 obs. of  10 variables:
##  $ id               : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ diagnosis_result : chr  "M" "B" "M" "M" ...
##  $ radius           : int  23 9 21 14 9 25 16 15 19 25 ...
##  $ texture          : int  12 13 27 16 19 25 26 18 24 11 ...
##  $ perimeter        : int  151 133 130 78 135 83 120 90 88 84 ...
##  $ area             : int  954 1326 1203 386 1297 477 1040 578 520 476 ...
##  $ smoothness       : num  0.143 0.143 0.125 0.07 0.141 0.128 0.095 0.119 0.127 0.119 ...
##  $ compactness      : num  0.278 0.079 0.16 0.284 0.133 0.17 0.109 0.165 0.193 0.24 ...
##  $ symmetry         : num  0.242 0.181 0.207 0.26 0.181 0.209 0.179 0.22 0.235 0.203 ...
##  $ fractal_dimension: num  0.079 0.057 0.06 0.097 0.059 0.076 0.057 0.075 0.074 0.082 ...
```

```
prc <- prc[-1]
table(prc$diagnosis_result)
```

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

```
prc$diagnosis <- factor(prc$diagnosis_result, levels = c("B", "M"),
                        labels = c("Benign", "Malignant"))
round(prop.table(table(prc$diagnosis)) * 100, digits = 1)
```

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

Normalizing numeric data

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

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

Creating training and test data set

```
prc_train <- prc_n[1:65,]
prc_test <- prc_n[66:100,]
prc_train_labels <- prc[1:65, 1]
prc_test_labels <- prc[66:100, 1]
```

Training a model on data

```
#install.packages("class")
library(class)
prc_test_pred <- knn(train = prc_train, test = prc_test,cl = prc_train_labels, k=10)
```

Evaluate the model performance

Accuracy - ((TN+TP)/35) = 0.63%

```
#install.packages("gmodels")
library(gmodels)
CrossTable(prc_test_labels, 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
## prc_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 |           |
## ----------------|-----------|-----------|-----------|
##
##
```

Improve the performance of the model

Using k=9, I am getting 0 false negatives which is an improvement over 1 false negative which was being observed k=10.

Accuracy - ((TN+TP)/35) = 0.69%

Using k=11, I am getting 0 false negatives but 14 false positives.

Accuracy - ((TN+TP)/35) = 0.6%

```r
prc_test_pred2 <- knn(train = prc_train, test = prc_test,cl = prc_train_labels, k=9)
CrossTable(prc_test_labels, prc_test_pred2, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  35
##
##
##                 | prc_test_pred2
## prc_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 |           |
## ----------------|-----------|-----------|-----------|
##
##
```

```r
prc_test_pred3 <- knn(train = prc_train, test = prc_test,cl = prc_train_labels, k=11)
CrossTable(prc_test_labels, prc_test_pred3, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
```

```
## Total Observations in Table:  35
##
##
##                  | prc_test_pred3
## prc_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 |           |
## -----------------|-----------|-----------|-----------|
##
##
```

Question 3

Using the kNN algorithm from the caret package.

```r
library(caret)
```

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

```r
library(doSNOW)
```

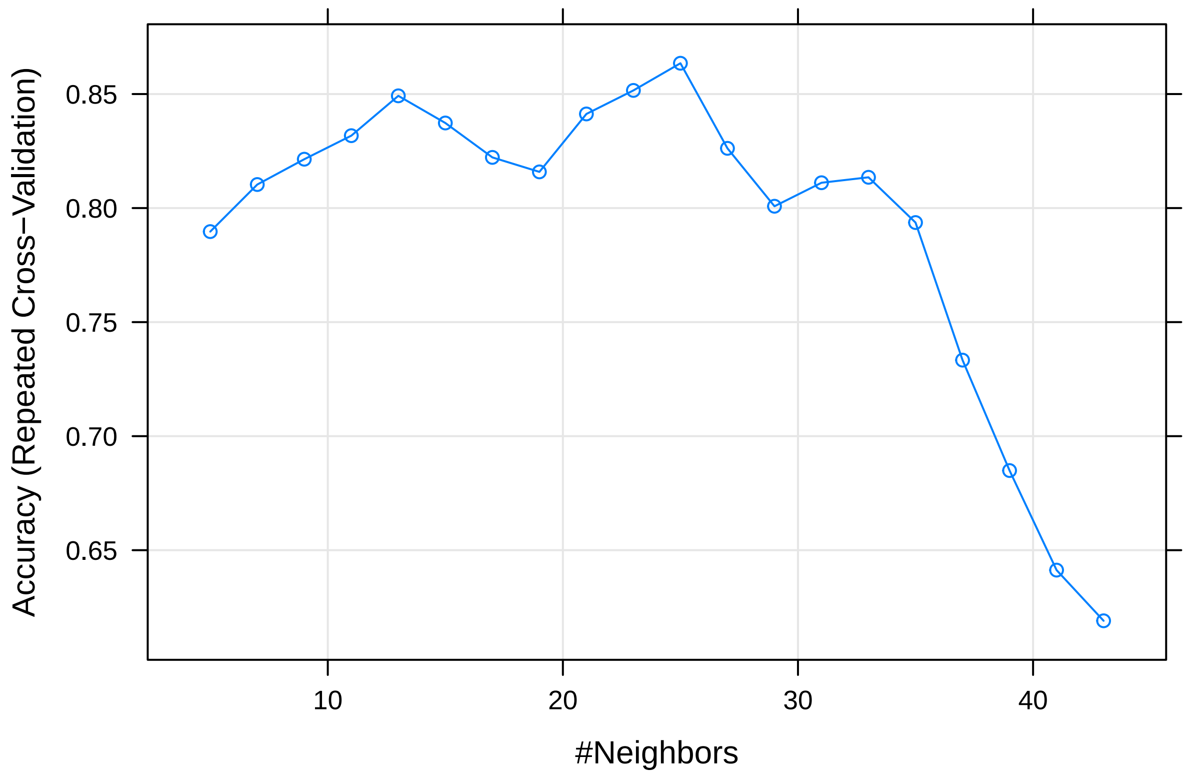
```
## Loading required package: foreach
## Loading required package: iterators
## Loading required package: snow
```

```r
library(xgboost)

# Loading the data onto a different variable in R and removing the id column from it.
data <- read.csv("prostate_cancer.csv", stringsAsFactors = FALSE)
#str(data)
data <- data[,-1]

# Using the diagnosis_result column as a factor instead of a plain character.
data$diagnosis_result <- as.factor(data$diagnosis_result)

# Partioning the data into a 65-35 training and testing dataset.
set.seed(300)
indexes <- createDataPartition(data$diagnosis_result, p = 0.64, list = FALSE)
data.train <- data[indexes,]
data.test <- data[-indexes,]
#prop.table(table(data$diagnosis_result))
#prop.table(table(data.train$diagnosis_result))
```

```r
#prop.table(table(data.test$diagnosis_result))

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

# Training and training control from the dataset.
set.seed(400)
train.control <- trainControl(method = "repeatedcv", repeats = 3)

# Finding the knn fit for the training set and then plotting it.
knnFit <- train(diagnosis_result ~ .,
                data = data.train,
                method = "knn",
                trControl = train.control,
                preProcess = c("center","scale"),
                tuneLength = 20)
knnFit
```

```
## k-Nearest Neighbors
##
## 65 samples
##  8 predictor
##  2 classes: 'B', 'M'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 59, 58, 58, 59, 59, 58, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.7896825  0.53348190
##    7  0.8103175  0.56985871
##    9  0.8214286  0.59207886
##   11  0.8317460  0.60709267
##   13  0.8492063  0.64939839
##   15  0.8373016  0.62536458
##   17  0.8222222  0.59572432
##   19  0.8158730  0.57391273
##   21  0.8412698  0.62402754
##   23  0.8515873  0.65318627
##   25  0.8634921  0.67637468
##   27  0.8261905  0.57681912
##   29  0.8007937  0.51057155
##   31  0.8111111  0.53963235
##   33  0.8134921  0.54240542
##   35  0.7936508  0.48910220
##   37  0.7333333  0.33073593
##   39  0.6849206  0.18744589
##   41  0.6412698  0.07142857
##   43  0.6190476  0.00000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 25.
```

```r
plot(knnFit)
```




```r
# Using the model to predict data on the testing set.
knnPredict <- predict(knnFit, newdata = data.test)
knnPredict
```

```
##  [1] M M M M M M M M M B M B M M M B M M M B B M M B M M M M M M M M B M B
## Levels: B M
```

Question 4

Generting confusion matrices for the kNN predictions made using the two algorithms above.

```r
prc_test_labels <- as.factor(prc_test_labels)
confusionMatrix(prc_test_pred, prc_test_labels)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B  6  0
##          M 13 16
##
##                Accuracy : 0.6286
##                  95% CI : (0.4492, 0.7853)
##     No Information Rate : 0.5429
##     P-Value [Acc > NIR] : 0.1987130
```

```
##
##                  Kappa : 0.2968
##  Mcnemar's Test P-Value : 0.0008741
##
##            Sensitivity : 0.3158
##            Specificity : 1.0000
##         Pos Pred Value : 1.0000
##         Neg Pred Value : 0.5517
##             Prevalence : 0.5429
##         Detection Rate : 0.1714
##   Detection Prevalence : 0.1714
##      Balanced Accuracy : 0.6579
##
##       'Positive' Class : B
##
```

```r
confusionMatrix(knnPredict, data.test$diagnosis_result)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B  8  0
##          M  5 22
##
##                Accuracy : 0.8571
##                  95% CI : (0.6974, 0.9519)
##     No Information Rate : 0.6286
##     P-Value [Acc > NIR] : 0.002746
##
##                   Kappa : 0.6679
##  Mcnemar's Test P-Value : 0.073638
##
##             Sensitivity : 0.6154
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.8148
##              Prevalence : 0.3714
##          Detection Rate : 0.2286
##    Detection Prevalence : 0.2286
##       Balanced Accuracy : 0.8077
##
##        'Positive' Class : B
##
```