kNN to predict prostate cancer

Irin Ann Paul

02-04-24

Data Collection

The prostate cancer data is loaded in csv format.

Data Preparation and Exploration

```
# set directory as the path to the data file
setwd("C:/Users/irina/Documents/DA5030")

#read file, convert every character vector to a factor
url_pros <- "https://s3.us-east-2.amazonaws.com/artificium.us/datasets/Prostate_Cancer.csv"
prc <- read.csv(url_pros, stringsAsFactors = FALSE)

# view the structure of 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 ...
```

There are 100 observations and 10 variables in this data set. The first column is for 'id', which can be removed in the next step since it isn't significant in making a kNN model. The patient diagnosis_result column shows the nature of the tumor in their prostate: wither Benign or Malignant. All other variables are numerical measures of the tumor.

```
# removing the id column from the data set
prc <- prc[-1]
```

```
# view the number of patients in categories 'Benign' (B), and 'Malignant' (M)
# this is the target variable
table(prc$diagnosis_result)
```

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

```
# renaming
prc$diagnosis_result <- factor(prc$diagnosis_result, levels = c("B", "M"), labels = c("Benign", "Malign

# convert it to percentage and round up to 1 decimal point
round(prop.table(table(prc$diagnosis)) * 100, digits = 1)
```

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

Out of the 100 observations, 38 are Benign tumors and 62 are Malignant.

```
# function to normalize all variables using min-max normalization:
normalize <- function(x){
  return((x - min(x)) / (max(x) - min(x)))
}

# normalization
# starting from 2nd variable, since 1st is not numeric
prc_norm <- as.data.frame(lapply(prc[2:9], normalize))

# check status
summary(prc_norm)
```

```
##      radius          texture         perimeter           area
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.1875   1st Qu.:0.1875   1st Qu.:0.2542   1st Qu.:0.1639
##  Median :0.5000   Median :0.4062   Median :0.3500   Median :0.2637
##  Mean   :0.4906   Mean   :0.4519   Mean   :0.3732   Mean   :0.2989
##  3rd Qu.:0.7500   3rd Qu.:0.7031   3rd Qu.:0.5188   3rd Qu.:0.4266
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##    smoothness      compactness        symmetry      fractal_dimension
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.3219   1st Qu.:0.1384   1st Qu.:0.2189   1st Qu.:0.1364
##  Median :0.4384   Median :0.2622   Median :0.3254   Median :0.2273
##  Mean   :0.4484   Mean   :0.2889   Mean   :0.3442   Mean   :0.2657
##  3rd Qu.:0.5753   3rd Qu.:0.3876   3rd Qu.:0.4379   3rd Qu.:0.3636
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

The data set is normalized to convert all numerical variables to a 0 - 1 scale. This ensures uniformity in scale.

```
# divide data into training and testing in 65:35 ratio
prc_train <- prc_norm[1:65,]
prc_test <- prc_norm[66:100,]


# label the train and test datasets:
```

```r
# takes the target variable value of each observation
prc_train_label <- prc[1:65, 1]
prc_test_label <- prc[66:100, 1]
```

The prc data set is manually divided into train and test data in the ratio 65:35.

Training model on data

```r
# load package
library(class)

# use knn function to train the model:
# k value is usually sqr root of no. of obs
prc_pred <- knn(train = prc_train, test = prc_test, cl = prc_train_label, k = 10)
```

The data is used in training a kNN model using the knn() function, with a k value of 10. This can be tweaked to improve the performance (accuracy) of the prediction.

Model Evaluation

```r
# load packages
library(gmodels)

ct_prc <- CrossTable(prc_test_label, prc_pred, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  35
##
##
##                | prc_pred
## prc_test_label |    Benign | Malignant | Row Total |
## ---------------|-----------|-----------|-----------|
##         Benign |         7 |        12 |        19 |
##                |     0.368 |     0.632 |     0.543 |
##                |     1.000 |     0.429 |           |
##                |     0.200 |     0.343 |           |
## ---------------|-----------|-----------|-----------|
##      Malignant |         0 |        16 |        16 |
##                |     0.000 |     1.000 |     0.457 |
```

```
##                |     0.000 |     0.571 |           |
##                |     0.000 |     0.457 |           |
## ---------------|-----------|-----------|-----------|
##   Column Total |         7 |        28 |        35 |
##                |     0.200 |     0.800 |           |
## ---------------|-----------|-----------|-----------|
##
##
```

```
ct_prc
```

```
## $t
##            y
## x           Benign Malignant
##   Benign         7        12
##   Malignant      0        16
##
## $prop.row
##            y
## x              Benign Malignant
##   Benign    0.3684211 0.6315789
##   Malignant 0.0000000 1.0000000
##
## $prop.col
##            y
## x              Benign Malignant
##   Benign    1.0000000 0.4285714
##   Malignant 0.0000000 0.5714286
##
## $prop.tbl
##            y
## x              Benign Malignant
##   Benign    0.2000000 0.3428571
##   Malignant 0.0000000 0.4571429
```

```
# accuracy
accuracy <- ((ct_prc$t[1, 1] + ct_prc$t[2, 2]) / 35) * 100
accuracy
```

```
## [1] 65.71429
```

There are 35 observations in the test data, out of which 7 were True negatives(0.2%) and 16 were True positives(0.5%). The model becomes dangerous if there are False negatives.

The total accuracy of the model was 65.71% ((TP + TN)/35) Making changes to the k value and re-assigning train and test data might remove the False Negatives and increase the percentage of True positives.

kNN using Caret package

```
# load package
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
# reproducibility
set.seed(123)

# partitioning data in 65:35 ratio
prc_pred_caret <- createDataPartition(prc$diagnosis_result, times = 1, p = 0.65, list = FALSE)
prc_train_caret <- prc[prc_pred_caret, ]
prc_test_caret <- prc[ - prc_pred_caret, ]

# pre-processing data (scaling the data, such that the mean is 0 and sd is 1)
preProcPrc <- preProcess(prc_train_caret, method = c("center", "scale"))
train_transf <- predict(preProcPrc, prc_train_caret)
test_transf <- predict(preProcPrc, prc_test_caret)

k_values <- c(1:12)


# model training
knnModel <- train(diagnosis_result ~ .,
                  data = train_transf,
                  method = "knn",
                  trControl = trainControl(method = "cv"), # cross-validation
                  tuneGrid = data.frame(k = k_values)) # try different values of k

best_model<- knn3(
                  diagnosis_result ~ .,
                  data = train_transf,
                  k = knnModel$bestTune$k
                 )

# Prediction
prediction <- predict(best_model, test_transf, type = "class") # predict class labels of the test data

# Calculate confusion matrix
cm <- confusionMatrix(prediction, test_transf$diagnosis_result, positive = "Malignant")
cm
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign        11         0
##   Malignant      2        21
##
##                Accuracy : 0.9412
##                  95% CI : (0.8032, 0.9928)
##     No Information Rate : 0.6176
##     P-Value [Acc > NIR] : 1.82e-05
##
##                   Kappa : 0.8717
```

```
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.0000
##             Specificity : 0.8462
##          Pos Pred Value : 0.9130
##          Neg Pred Value : 1.0000
##              Prevalence : 0.6176
##          Detection Rate : 0.6176
##    Detection Prevalence : 0.6765
##       Balanced Accuracy : 0.9231
##
##        'Positive' Class : Malignant
##
```

Here, I select the best model out of 3 based on different k values. 10 is taken since there are 100 observations. Numbers nearest to 10, 9 and 11 are considered here. 12 gives the best model.

The confusion matrix for the kNN model using caret package is significantly high. Here, 11 were True Negatives, and 21 True Positives. The number of False negative is very low here (0) and therefore makes the model more reliable.

The model's accuracy is: 94.1

Comparison of both models

```
cm_both <- confusionMatrix(prc_pred, prc_test_label, positive = "Malignant")
cm_both
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Benign Malignant
##   Benign         7         0
##   Malignant     12        16
##
##                Accuracy : 0.6571
##                  95% CI : (0.4779, 0.8087)
##     No Information Rate : 0.5429
##     P-Value [Acc > NIR] : 0.116877
##
##                   Kappa : 0.3478
##
##  Mcnemar's Test P-Value : 0.001496
##
##             Sensitivity : 1.0000
##             Specificity : 0.3684
##          Pos Pred Value : 0.5714
##          Neg Pred Value : 1.0000
##              Prevalence : 0.4571
##          Detection Rate : 0.4571
##    Detection Prevalence : 0.8000
```

```
##       Balanced Accuracy : 0.6842
##
##        'Positive' Class : Malignant
##
```

The confusion matrix for kNN model using the kNN() function shows a lower accuracy: 65.7