Practice KNN A3.202-v1

Dr. Nishat Mohammad

01/24/2024

Task 1: Loading Prostate Cancer Dataset

Download the data set for the tutorial and save it in your project folder.

```
# Load data to variable
prsca_data1 <- (read.csv("Prostate_Cancer.csv", stringsAsFactors = FALSE))

# Look at the data
str(prsca_data1)
```

```
## '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 ...
```

```
# Check dimensions
dim(prsca_data1)
```

```
## [1] 100  10
```

```
# Check for missing values
any(is.na(prsca_data1))
```

```
## [1] FALSE
```

```
# FALSE means there are no missing values!
```

Task 2:

Follow this tutorial on applying kNN to prostate cancer detection and implement all of the steps in your R Notebook. Use appropriate headers to each step to structure your 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.).

Take off the id column

```
# Take the id  column (first column) out of the data frame
prsca_data1 <- prsca_data1[-1]

# Look at the editted data
str(prsca_data1)
```

```
## 'data.frame':    100 obs. of  9 variables:
##  $ 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 ...
```

The ID column has been removed leaving us with 9 variables.

Factor for Beingn and Malignant Tumor Catogories

```
# Factoring step
prsca_data1$diagnoses <- factor(prsca_data1$diagnosis_result, levels = c("B", "M"), labels = c("Benign"
#str(prsca_data1)

# Get the percentages
round(prop.table(table(prsca_data1$diagnosis)) * 100, digits = 1)
```

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

This shows that Benign tumors contribute to 38% of the observations while malignant contribute to 68% of the observations.

Normalizing numeric data

From the structure we can see the that 1st to 4th columns are integers and 5th to 8th columns are numeric.

```
# Check the summary of the data except the first and last columns
summary(prsca_data1[2:9])
```

```
##      radius         texture         perimeter           area
##  Min.   : 9.00   Min.   :11.00   Min.   : 52.00   Min.   : 202.0
##  1st Qu.:12.00   1st Qu.:14.00   1st Qu.: 82.50   1st Qu.: 476.8
##  Median :17.00   Median :17.50   Median : 94.00   Median : 644.0
```

```
##  Mean   :16.85   Mean   :18.23   Mean   : 96.78   Mean   : 702.9
##  3rd Qu.:21.00   3rd Qu.:22.25   3rd Qu.:114.25   3rd Qu.: 917.0
##  Max.   :25.00   Max.   :27.00   Max.   :172.00   Max.   :1878.0
##    smoothness       compactness        symmetry      fractal_dimension
##  Min.   :0.0700   Min.   :0.0380   Min.   :0.1350   Min.   :0.05300
##  1st Qu.:0.0935   1st Qu.:0.0805   1st Qu.:0.1720   1st Qu.:0.05900
##  Median :0.1020   Median :0.1185   Median :0.1900   Median :0.06300
##  Mean   :0.1027   Mean   :0.1267   Mean   :0.1932   Mean   :0.06469
##  3rd Qu.:0.1120   3rd Qu.:0.1570   3rd Qu.:0.2090   3rd Qu.:0.06900
##  Max.   :0.1430   Max.   :0.3450   Max.   :0.3040   Max.   :0.09700
```

This give me a quick look at all the statistics.

```
# Make a function
nrm <- function(x) {
return ((x - min(x)) / (max(x) - min(x)))
}

# Normalize
prsca_data_nrm <- as.data.frame(lapply(prsca_data1[2:9], nrm))
str(prsca_data_nrm)
```

```
## 'data.frame':    100 obs. of  8 variables:
##  $ radius           : num  0.875 0 0.75 0.312 0 ...
##  $ texture          : num  0.0625 0.125 1 0.3125 0.5 ...
##  $ perimeter        : num  0.825 0.675 0.65 0.217 0.692 ...
##  $ area             : num  0.449 0.671 0.597 0.11 0.653 ...
##  $ smoothness       : num  1 1 0.753 0 0.973 ...
##  $ compactness      : num  0.782 0.134 0.397 0.801 0.309 ...
##  $ symmetry         : num  0.633 0.272 0.426 0.74 0.272 ...
##  $ fractal_dimension: num  0.5909 0.0909 0.1591 1 0.1364 ...
```

```
summary(prsca_data_nrm$perimeter)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.2542  0.3500  0.3732  0.5188  1.0000
```

Now all the columns are numeric and values normalized. I checked the perimeter summary, values are normalized.

Creating Training and Test Data Set

This will be done with ratio of 65 for training and 35 for test data.

```
# divide the data set into 2 portions
# in the ratio of 65: 35 (assumed)
# for the training and test data set respectively
training_set <- prsca_data_nrm[1:65,]
test_set <- prsca_data_nrm[66:100,]
```

Now I will add labels for the training and test sets

```
# Make the labels in accordance with the diagnosis_result column (column 1)

training_labels <- prsca_data1[1:65, 1]
testing_labels <- prsca_data1[66:100, 1]
```

Training a model on data

```
library(class)
# Get value of k
#nrow(prsca_data1)
K <- sqrt(nrow(prsca_data1))
#any(is.na(prsca_data1))
# apply knn() function
test_knn_prediction <- knn(train = training_set, test = test_set, cl = training_labels, k= K)
```

Here, I found the k by taking the square root of the total number of observations. k is 10 form the 100 observations in this dataset.
I applied the knn function here using the both sets and the labels of the training set with k of 10. Now let us evaluate the model.

Evaluate the model performance

```
library(gmodels)

# use the cross table
prsc_cros_table <- CrossTable(
  x = testing_labels,
  y = test_knn_prediction,
  prop.chisq=FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  35
##
##
##                | test_knn_prediction
## testing_labels |         B |         M | Row Total |
## ---------------|-----------|-----------|-----------|
##              B |         7 |        12 |        19 |
##                |     0.368 |     0.632 |     0.543 |
```

```
##                |     0.875 |     0.444 |           |
##                |     0.200 |     0.343 |           |
## ---------------|-----------|-----------|-----------|
##              M |         1 |        15 |        16 |
##                |     0.062 |     0.938 |     0.457 |
##                |     0.125 |     0.556 |           |
##                |     0.029 |     0.429 |           |
## ---------------|-----------|-----------|-----------|
##   Column Total |         8 |        27 |        35 |
##                |     0.229 |     0.771 |           |
## ---------------|-----------|-----------|-----------|
##
##
```

The Cross table above shows the prediction of cancer samples in the data set provided.
There are a total of 19 benign observations in the test set, out of these our model predicted 7 correctly (true positive) and 12 incorrectly as malignant (false negative).
There are total of 16 malignant observations in the test set, out of there our model predicted 15 correctly (true negative) and 1 incorrectly as benign (false positive).

The overall accuracy of the model on the test set is the percentage of sum of true events fraction which is 7+ 15 divide by 35 which is approximately 62.9%. This is the precsion of the model.

Fluctuate the value of K around 10 to check for better accuracy:

```
K11 <- 11
test_knn_prediction_k11 <- knn(train = training_set, test = test_set, cl = training_labels, k= K11)
prsc_cros_table_k11 <- CrossTable(
  x = testing_labels,
  y = test_knn_prediction_k11,
  prop.chisq=FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  35
##
##
##                | test_knn_prediction_k11
## testing_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 |           |
## ---------------|-----------|-----------|-----------|
##
##
```

From this cross table we can see the true event fraction reduce to (5+16)/35 which is 60.0% this is the accuracy or precision of the model

```
K9 <- 9
test_knn_prediction_k9 <- knn(train = training_set, test = test_set, cl = training_labels, k= K9)
prsc_cros_table_k9 <- CrossTable(
  x = testing_labels,
  y = test_knn_prediction_k9,
  prop.chisq=FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  35
##
##
##                | test_knn_prediction_k9
## testing_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 |           |
## ---------------|-----------|-----------|-----------|
```

```
##
##
```

For this model we can see that the true event fraction is increased to $(8 + 16)/35$ which is 68.6% accuracy.

My conclusion is that the knn model will be more accurate if k is reduced. but the value of k should be carefully decided in order not to leave out some trivial patterns which may be equally important for later interpretation.

Task 3:

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

```r
# Load data
p_df <- read.csv("Prostate_Cancer.csv",stringsAsFactors=FALSE)
str(p_df)
```

```
## '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 ...
```

```r
# Take off id column
p_df <- p_df[-1]

# Load caret package
library(caret)
```

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

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

```r
# For reproducibility
set.seed(786)

# Train and test sets
caret_training_set <- createDataPartition(y = p_df$diagnosis_result, p= 0.64, list = FALSE)
train_df <- p_df[caret_training_set,]

testing_df <- p_df[-caret_training_set,]

# Factor the data according to tumor categories benign and malignant
train_df[["diagnosis_result"]] = factor(train_df[["diagnosis_result"]])
```

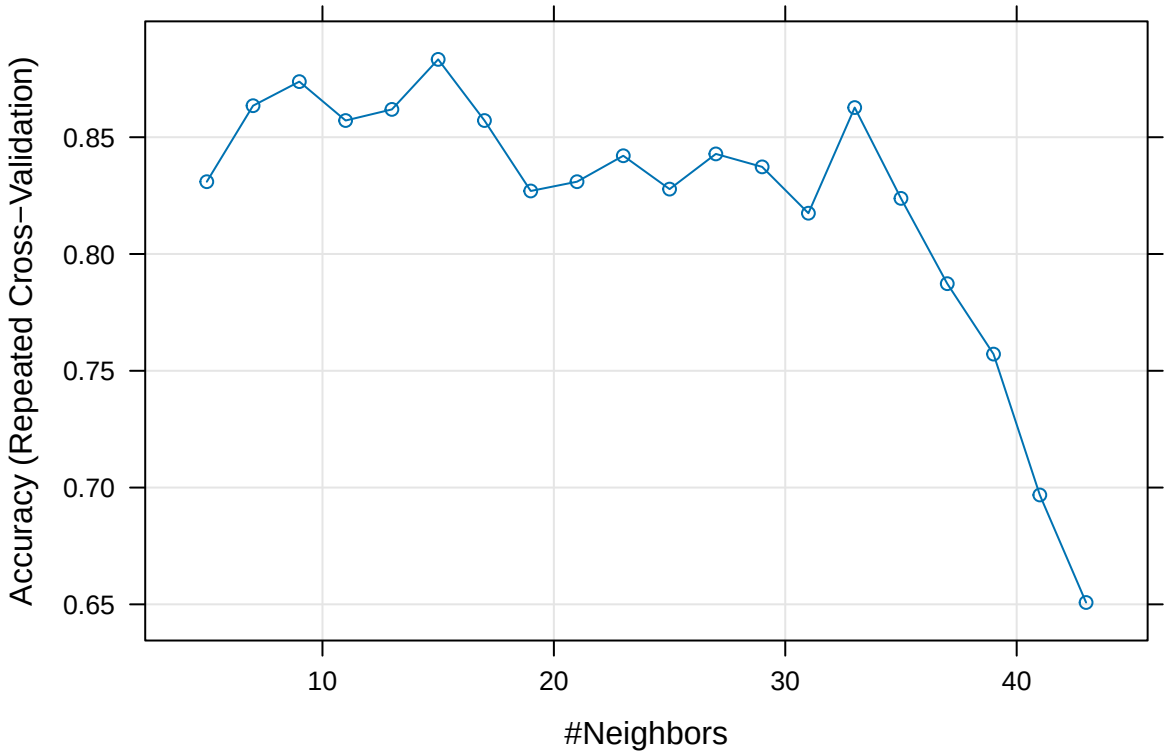
```r
testing_df[["diagnosis_result"]] = factor(testing_df[["diagnosis_result"]])


# Normalize and Train the model
#?trainControl
train_control <- trainControl(method = "repeatedcv", number=10, repeats = 3)

knn_pred_model <- train(diagnosis_result ~., data = train_df, method = "knn",
                 trControl=train_control,
                 preProcess = c("center", "scale"),
                 tuneLength = 20)
knn_pred_model
```

```
## 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, 58, 59, 59, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8309524  0.6371304
##    7  0.8634921  0.6952238
##    9  0.8738095  0.7227221
##   11  0.8571429  0.6822976
##   13  0.8619048  0.6867655
##   15  0.8833333  0.7454087
##   17  0.8571429  0.6929639
##   19  0.8269841  0.6110237
##   21  0.8309524  0.6124895
##   23  0.8420635  0.6392355
##   25  0.8277778  0.6033848
##   27  0.8428571  0.6398598
##   29  0.8373016  0.6227163
##   31  0.8174603  0.5716079
##   33  0.8626984  0.6756588
##   35  0.8238095  0.5856155
##   37  0.7873016  0.4821758
##   39  0.7571429  0.3939959
##   41  0.6968254  0.2372294
##   43  0.6507937  0.1004329
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 15.
```

```r
# Visualize the prediction
plot(knn_pred_model)
```




```
# Use the model on test set
knn_test_pred <- predict(knn_pred_model, newdata = testing_df)
```

Task 4:

Use the confusionMatrix() function from the caret package to determine the accuracy of both algorithms.

```
# test  the knn from caret package
length(testing_df$diagnosis_result)
```

```
## [1] 35
```

```
length(knn_test_pred)
```

```
## [1] 35
```

```
confusionMatrix(table(knn_test_pred, testing_df$diagnosis_result))
```

```
## Confusion Matrix and Statistics
##
##
## knn_test_pred  B  M
```

```
##             B  9  1
##             M  4 21
##
##                Accuracy : 0.8571
##                  95% CI : (0.6974, 0.9519)
##     No Information Rate : 0.6286
##     P-Value [Acc > NIR] : 0.002746
##
##                   Kappa : 0.6789
##
##  Mcnemar's Test P-Value : 0.371093
##
##             Sensitivity : 0.6923
##             Specificity : 0.9545
##          Pos Pred Value : 0.9000
##          Neg Pred Value : 0.8400
##              Prevalence : 0.3714
##          Detection Rate : 0.2571
##    Detection Prevalence : 0.2857
##       Balanced Accuracy : 0.8234
##
##        'Positive' Class : B
##
```

The accuracy of the KNN model using caret is 0.8571 when considering the benign Class is the positive class.

```
confusionMatrix(table(knn_test_pred, testing_labels))
```

```
## Confusion Matrix and Statistics
##
##              testing_labels
## knn_test_pred  B  M
##             B  2  8
##             M 17  8
##
##                Accuracy : 0.2857
##                  95% CI : (0.1464, 0.463)
##     No Information Rate : 0.5429
##     P-Value [Acc > NIR] : 0.9994
##
##                   Kappa : -0.378
##
##  Mcnemar's Test P-Value : 0.1096
##
##             Sensitivity : 0.10526
##             Specificity : 0.50000
##          Pos Pred Value : 0.20000
##          Neg Pred Value : 0.32000
##              Prevalence : 0.54286
##          Detection Rate : 0.05714
##    Detection Prevalence : 0.28571
##       Balanced Accuracy : 0.30263
##
```

```
##        'Positive' Class : B
##
```

The accuracy of this model is 0.28.
Comparing the accuracy of both models, the knn by the caret package has a higher accuracy.