Practice 3

Smit Patil

1. Download the data set for the tutorial.
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.

```
#Importing data
prc <- read.csv("prostate_cancer.csv")
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 ...
```

```
#Creating a new column with values Benign, Malignant
prc$diagnosis <- factor(prc$diagnosis_result, levels = c("B", "M"), labels = c("Benign", "Malignant"))

#Creating a function for normalization
normalize <- function(x)
{
  return ((x - min(x)) / (max(x) - min(x)))
}

#Converting the data into normalized form
prc_n <- as.data.frame(lapply(prc[3:10], 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
```

```
#Splitting the data for training and testing
prc_train <- prc_n[1:65,]
prc_test <- prc_n[66:100,]

#Getting labels for trainig and testing
prc_train_labels <- prc[1:65, 2]
```

```r
prc_test_labels <- prc[66:100, 2]

#Importing library "class" to perform kNN algorithm
library(class)
prc_test_pred <- knn(train = prc_train, test = prc_test,cl = prc_train_labels, k=10)

#Importing library "gmodels" to display te resulting output in a CrossTable
library(gmodels)
CrossTable(x = prc_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
## prc_test_labels |         B |         M | Row Total |
## ----------------|-----------|-----------|-----------|
##               B |         6 |        13 |        19 |
##                 |     0.316 |     0.684 |     0.543 |
##                 |     0.857 |     0.464 |           |
##                 |     0.171 |     0.371 |           |
## ----------------|-----------|-----------|-----------|
##               M |         1 |        15 |        16 |
##                 |     0.062 |     0.938 |     0.457 |
##                 |     0.143 |     0.536 |           |
##                 |     0.029 |     0.429 |           |
## ----------------|-----------|-----------|-----------|
##    Column Total |         7 |        28 |        35 |
##                 |     0.200 |     0.800 |           |
## ----------------|-----------|-----------|-----------|
##
##
```

```r
#Calculating accuracy
True_Negative <- 7
True_Positive <- 16
Accuracy <- ((True_Negative + True_Positive)/35)*100
Accuracy
```

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

3. Try another kNN implementation using caret package. Compare the accuracy of the two implementations.

```r
library(lattice)
library(ggplot2)
library(data.table)
library(dplyr)
```

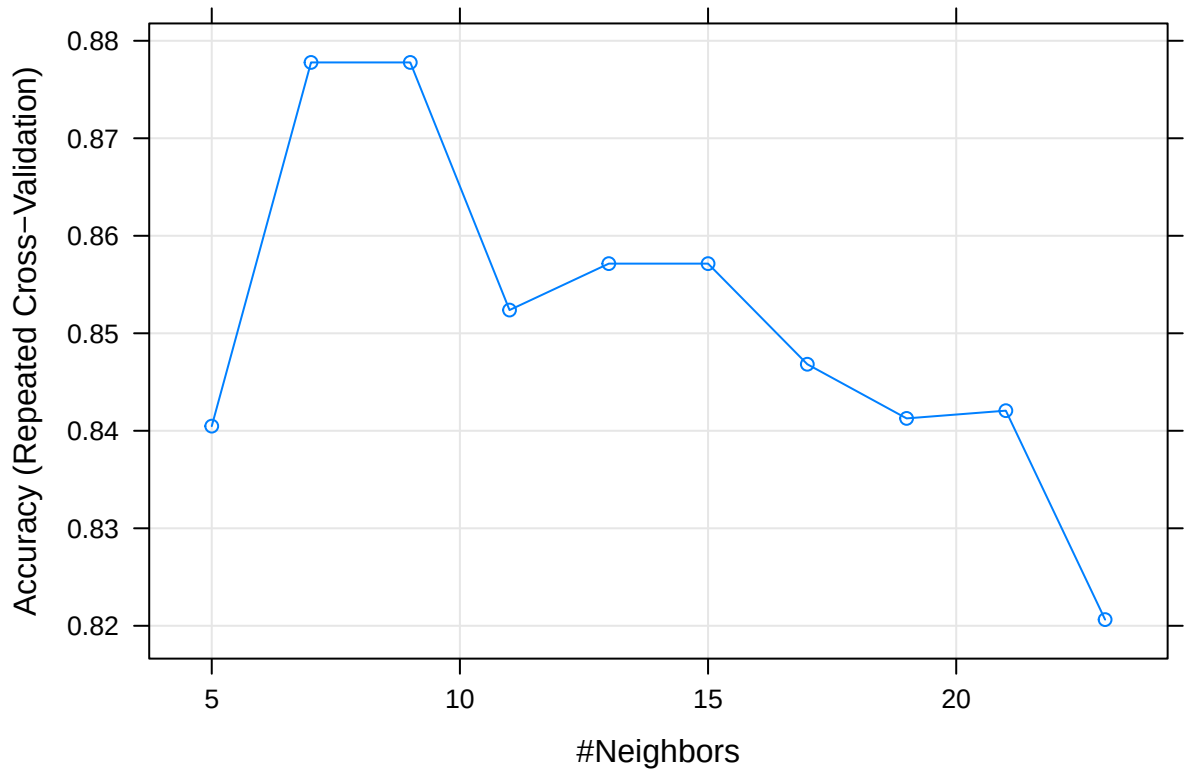
```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(caret)

#Creating training dataset
training <- prc_train
training$diagnosis_result <- as.factor(prc_train_labels)

#Creating testing dataset
testing <- prc_test
testing$diagnosis_result <- as.factor(prc_test_labels)

#Assigning train control values to variables
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
set.seed(882)

#Using train function from caret package to train the data
knn_fit <- train(diagnosis_result ~., data = training, method = "knn",
 trControl=trctrl,
 preProcess = c("center", "scale"),
 tuneLength = 10)

#Displaying accuracy of trained data and plotting the same
knn_fit
```

```
## 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: 58, 59, 58, 58, 59, 58, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8404762  0.5673572
##    7  0.8777778  0.6610080
##    9  0.8777778  0.6578334
##   11  0.8523810  0.5738698
##   13  0.8571429  0.5840147
##   15  0.8571429  0.5875953
##   17  0.8468254  0.5448222
##   19  0.8412698  0.5222031
##   21  0.8420635  0.5233005
##   23  0.8206349  0.4519559
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```
plot(knn_fit)
```




```
#testing the the model with testing data
test_pred <- predict(knn_fit, newdata = testing)
test_pred
```

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

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

```
#Using ConfusionMatrix function from caret package for both models
confusionMatrix(test_pred, testing$diagnosis_result )
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B 10  0
##          M  9 16
##
##                Accuracy : 0.7429
##                  95% CI : (0.5674, 0.8751)
##     No Information Rate : 0.5429
##     P-Value [Acc > NIR] : 0.012296
##
##                   Kappa : 0.5039
##
##  Mcnemar's Test P-Value : 0.007661
##
##             Sensitivity : 0.5263
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.6400
##              Prevalence : 0.5429
##          Detection Rate : 0.2857
##    Detection Prevalence : 0.2857
##       Balanced Accuracy : 0.7632
##
##        'Positive' Class : B
##
```

```
prc_test_labels_1 <- as.factor(prc_test_labels)
confusionMatrix(prc_test_pred, prc_test_labels_1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  B  M
##          B  6  1
##          M 13 15
##
##                Accuracy : 0.6
##                  95% CI : (0.4211, 0.7613)
##     No Information Rate : 0.5429
##     P-Value [Acc > NIR] : 0.307007
##
##                   Kappa : 0.2391
##
##  Mcnemar's Test P-Value : 0.003283
##
##             Sensitivity : 0.3158
```

```
##             Specificity : 0.9375
##          Pos Pred Value : 0.8571
##          Neg Pred Value : 0.5357
##              Prevalence : 0.5429
##          Detection Rate : 0.1714
##    Detection Prevalence : 0.2000
##       Balanced Accuracy : 0.6266
##
##        'Positive' Class : B
##
```