k-NN and Decision Tree on Kyphosis Dataset

Abhay Padda

3 March 2018

##install.packages("gam") which has the Kyphosis dataset  
library(gam)

## Loading required package: splines

## Loading required package: foreach

## Loaded gam 1.14

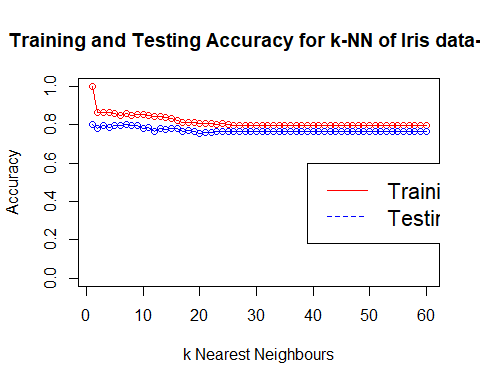
## Load data-set Kyphosis  
data("kyphosis")  
summary(kyphosis)

## Kyphosis Age Number Start   
## absent :64 Min. : 1.00 Min. : 2.000 Min. : 1.00   
## present:17 1st Qu.: 26.00 1st Qu.: 3.000 1st Qu.: 9.00   
## Median : 87.00 Median : 4.000 Median :13.00   
## Mean : 83.65 Mean : 4.049 Mean :11.49   
## 3rd Qu.:130.00 3rd Qu.: 5.000 3rd Qu.:16.00   
## Max. :206.00 Max. :10.000 Max. :18.00

## k-NN for Kyphosis dataset  
source('knn\_functions1.R')  
  
m <- avgTrnTst(kyphosis, 0.8, 1)  
dim(m)

## [1] 60 3

plotFn(m, 'Training and Testing Accuracy for k-NN of Iris data-set')



## Decision Tree for Kyphosis dataset  
library(rpart)

##   
## Attaching package: 'rpart'

## The following object is masked \_by\_ '.GlobalEnv':  
##   
## kyphosis

library(rpart.plot)  
  
v <- kyphosis$Kyphosis  
  
table(v)

## v  
## absent present   
## 64 17

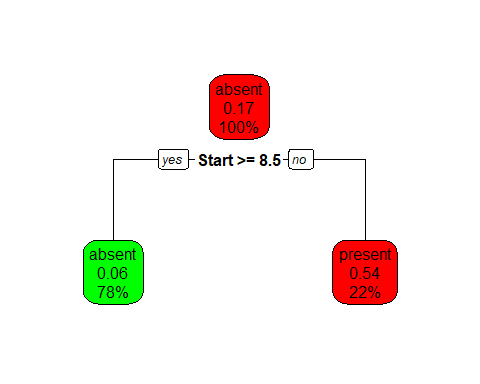
set.seed(522)  
  
kyphosis[, 'train'] <- ifelse(runif(nrow(kyphosis)) < 0.75, 1, 0)  
  
trainSet <- kyphosis[kyphosis$train == 1,]  
testSet <- kyphosis[kyphosis$train == 0, ]  
  
trainColNum <- grep('train', names(trainSet))  
  
trainSet <- trainSet[, -trainColNum]  
testSet <- testSet[, -trainColNum]  
  
treeFit <- rpart(Kyphosis~.,data=trainSet,method = 'class')  
print(treeFit)

## n= 60   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 60 10 absent (0.83333333 0.16666667)   
## 2) Start>=8.5 47 3 absent (0.93617021 0.06382979) \*  
## 3) Start< 8.5 13 6 present (0.46153846 0.53846154) \*

rpart.plot(treeFit, box.col=c("red", "green"))  
  
Prediction1 <- predict(treeFit,newdata=testSet[-15],type = 'class')  
  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2



confusionMatrix(Prediction1,testSet$Kyphosis)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction absent present  
## absent 12 3  
## present 2 4  
##   
## Accuracy : 0.7619   
## 95% CI : (0.5283, 0.9178)  
## No Information Rate : 0.6667   
## P-Value [Acc > NIR] : 0.2486   
##   
## Kappa : 0.4444   
## Mcnemar's Test P-Value : 1.0000   
##   
## Sensitivity : 0.8571   
## Specificity : 0.5714   
## Pos Pred Value : 0.8000   
## Neg Pred Value : 0.6667   
## Prevalence : 0.6667   
## Detection Rate : 0.5714   
## Detection Prevalence : 0.7143   
## Balanced Accuracy : 0.7143   
##   
## 'Positive' Class : absent   
##

## As tree length is just one, so no pruning required for Kyphosis data-set

#### knn\_function.R file contents

# 'caTools' package provides us with functions to split dataset uniformly to test and training  
library(caTools)  
  
# Load library 'class' that has the knn() function  
library(class)  
  
# Function to split the dataset randomly  
splitFile <- function(dataset, trProp, classColPos) {  
 # split the dataset  
 sample = sample.split(iris[, classColPos], SplitRatio = trProp)  
   
 # create training and testing dataset  
 train = subset(iris, sample == TRUE)  
 test = subset(iris, sample == FALSE)  
   
 # save the target labels and remove from the train and test dataset  
 trainLabels <- train[, classColPos]  
 testLabels <- test[, classColPos]  
 train <- train[, -classColPos]  
 test <- test[, -classColPos]  
   
 # Nomalize function  
 normalize <- function(x) {  
 return( (x-min(x))/(max(x)-min(x)))  
 }  
 train  
 test  
 # Normalize test and training dataset  
 gtrn <- as.data.frame(lapply(train, normalize))  
 gtsn <- as.data.frame(lapply(test, normalize))  
   
 return(list(trn=gtrn, trL=trainLabels, val=gtsn, tsL=testLabels))  
}  
  
# Function to plot graph  
plotFn <- function(dataSet, graphTitle = '', ylimLo=0) {  
 plot(dataSet[, 1], dataSet[, 2], main = graphTitle, xlab = 'k Nearest Neighbours',  
 ylab = 'Accuracy', ylim = c(ylimLo, 1), type = 'o', col = 'red')  
 lines(dataSet[, 1], dataSet[, 3], type = 'o', col = 'blue')  
 legend(26, 0.6, legend=c("Training Accuracy", "Testing Accuracy"),  
 col=c("red", "blue"), lty=1:2, cex=1.4)  
}  
  
  
# Function to use k-NN and return training and testing results  
train\_test <- function(trainData,trainLabels,testData,testLabels) {  
 train <- c()  
 test <- c()  
 for (k in 1:40) {  
 knntr <- knn(trainData, trainData, trainLabels, k=k)  
 knnts <- knn(trainData, testData, trainLabels, k=k)  
 trTable <- table(knntr, trainLabels)  
 tsTable <- table(knnts, testLabels)  
 trTable <- prop.table(trTable)  
 tsTable <- prop.table(tsTable)  
 trainAccuracy <- sum(trTable[1,1], trTable[2,2], trTable[3,3])/sum(trTable)  
 testAccuracy <- sum(tsTable[1,1], tsTable[2,2], trTable[3,3])/sum(tsTable)  
 train <- c(train, trainAccuracy)  
 test <- c(test, testAccuracy)  
 }  
 acc <- data.frame('k' = 1:40, 'trAc' = train, 'tsAc' = test)  
 return(acc = acc)  
}  
  
# Single function to split data and then call train\_test function  
avgTrnTst <- function(dataset, trProp, classColPos) {  
 for (i in 1:30) {  
 a <- splitFile(dataset, trProp, classColPos)  
 b <- train\_test(a$trn, a$trL, a$val, a$tsL)  
 if (i==1) acd <- b  
 else acd <- rbind(acd, b)  
 }  
 library(plyr)  
   
 a1 <- ddply(acd,.(k), summarize, meanV = mean(trAc))  
 a2 <- ddply(acd,.(k), summarize, meanV = mean(tsAc))  
 m <- merge(a1,a2,by='k')  
   
 return(m)  
}