Summary of Decision Tree and Random Forest in Practice

Truc Huynh

11/27/2020

Table of Contents

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
library (DescTools)

## Warning: package 'DescTools' was built under R version 4.0.3
```

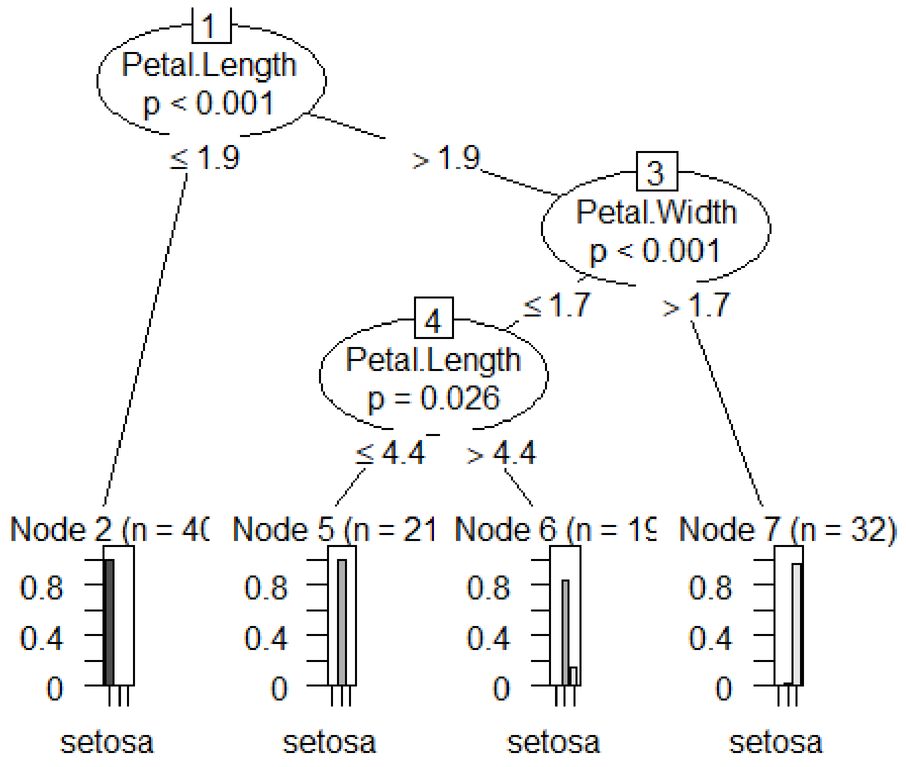
Part 0: Code Practice

Decision Trees with Package party

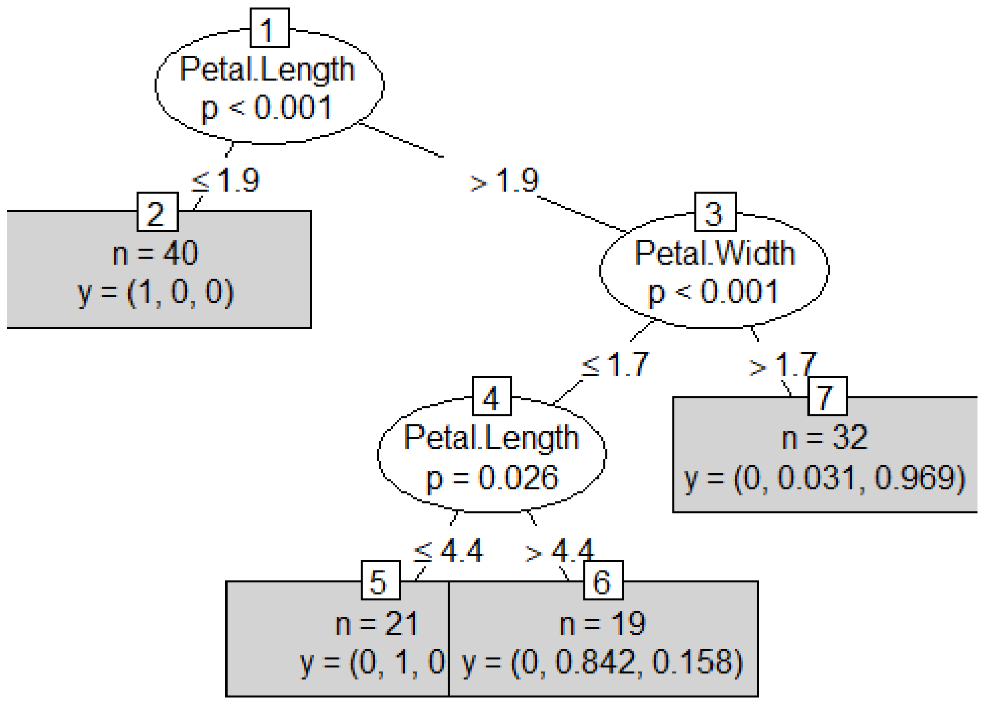
build a decision tree for the iris data with function ctree() in package party [Hothorn et al., 2010].

```
str(iris)

## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1
1 1 1 1 ...

set.seed(1234)
ind<- sample(2,nrow(iris), replace = TRUE, prob=c(0.7,0.3))
trainData <- iris[ind==1,]
testData<- iris[ind==2,]

# Use diffrent setting to build a decision Tree.
library(party)

## Loading required package: grid

## Loading required package: mvtnorm

## Loading required package: modeltools

## Loading required package: stats4

##
## Attaching package: 'modeltools'

## The following object is masked from 'package:DescTools':
##
##     ParseFormula

## Loading required package: strucchange

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Loading required package: sandwich
```

```
myFormula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length +
Petal.Width
iris_ctree <- ctree(myFormula, data=trainData)

#Check the prediction:
table(predict(iris_ctree), trainData$Species)

##
##                 setosa versicolor virginica
##   setosa           40          0         0
##   versicolor        0         37         3
##   virginica         0          1        31

print(iris_ctree)

##
##   Conditional inference tree with 4 terminal nodes
##
## Response:  Species
## Inputs:  Sepal.Length, Sepal.Width, Petal.Length, Petal.Width
## Number of observations:  112
##
## 1) Petal.Length <= 1.9; criterion = 1, statistic = 104.643
##   2)*  weights = 40
## 1) Petal.Length > 1.9
##   3) Petal.Width <= 1.7; criterion = 1, statistic = 48.939
##     4) Petal.Length <= 4.4; criterion = 0.974, statistic = 7.397
##       5)*  weights = 21
##     4) Petal.Length > 4.4
##       6)*  weights = 19
##   3) Petal.Width > 1.7
##     7)*  weights = 32

plot(iris_ctree)
```



```r
plot(iris_ctree, type="simple")
```



```
# predict on test data
testPred <- predict(iris_ctree, newdata = testData)
table(testPred, testData$Species)

##
## testPred     setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         12         2
##   virginica       0          0        14
```

Decision Trees with Package rpart

```
data("bodyfat", package = "TH.data")
dim(bodyfat)

## [1] 71 10

attributes(bodyfat)

## $names
##  [1] "age"          "DEXfat"       "waistcirc"    "hipcirc"
"elbowbreadth"
##  [6] "kneebreadth"  "anthro3a"     "anthro3b"     "anthro3c"     "anthro4"
##
## $row.names
##  [1] "47"  "48"  "49"  "50"  "51"  "52"  "53"  "54"  "55"  "56"  "57"
"58"
## [13] "59"  "60"  "61"  "62"  "63"  "64"  "65"  "66"  "67"  "68"  "69"
"70"
## [25] "71"  "72"  "73"  "74"  "75"  "76"  "77"  "78"  "79"  "80"  "81"
"82"
## [37] "83"  "84"  "85"  "86"  "87"  "88"  "89"  "90"  "91"  "92"  "93"
"94"
## [49] "95"  "96"  "97"  "98"  "99"  "100" "101" "102" "103" "104" "105"
"106"
## [61] "107" "108" "109" "110" "111" "112" "113" "114" "115" "116" "117"
##
## $class
## [1] "data.frame"

bodyfat[1:5,]

##    age DEXfat waistcirc hipcirc elbowbreadth kneebreadth anthro3a anthro3b
## 47  57  41.68     100.0   112.0          7.1         9.4     4.42     4.95
## 48  65  43.29      99.5   116.5          6.5         8.9     4.63     5.01
## 49  59  35.41      96.0   108.5          6.2         8.9     4.12     4.74
## 50  58  22.79      72.0    96.5          6.1         9.2     4.03     4.48
## 51  60  36.42      89.5   100.5          7.1        10.0     4.24     4.68
##    anthro3c anthro4
## 47     4.50    6.13
## 48     4.48    6.37
## 49     4.60    5.82
## 50     3.91    5.66
## 51     4.15    5.91
```

```
set.seed(1234)
ind <- sample(2, nrow(bodyfat), replace=TRUE, prob=c(0.7, 0.3))
bodyfat.train <- bodyfat[ind==1,]
bodyfat.test <- bodyfat[ind==2,]
# train a decision tree
library(rpart)
myFormula <- DEXfat ~ age + waistcirc + hipcirc + elbowbreadth + kneebreadth
bodyfat_rpart <- rpart(myFormula, data = bodyfat.train,control =
rpart.control(minsplit = 10))
attributes(bodyfat_rpart)
```

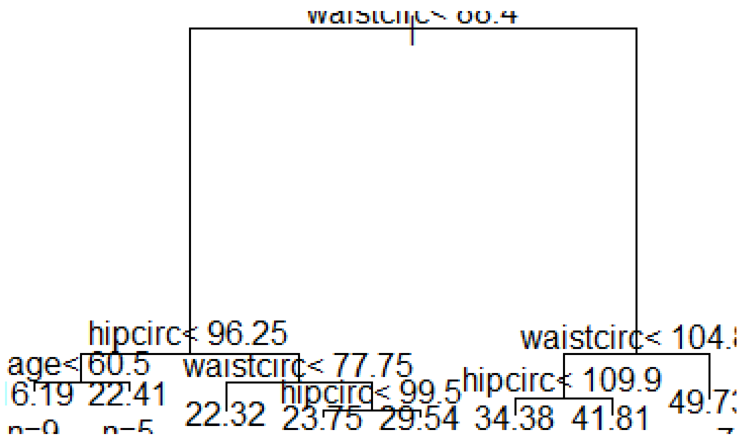
```
## $names
##  [1] "frame"               "where"               "call"
##  [4] "terms"               "cptable"             "method"
##  [7] "parms"               "control"             "functions"
## [10] "numresp"             "splits"              "variable.importance"
## [13] "y"                   "ordered"
##
## $xlevels
## named list()
##
## $class
## [1] "rpart"
```

```
print(bodyfat_rpart$cptable)
```

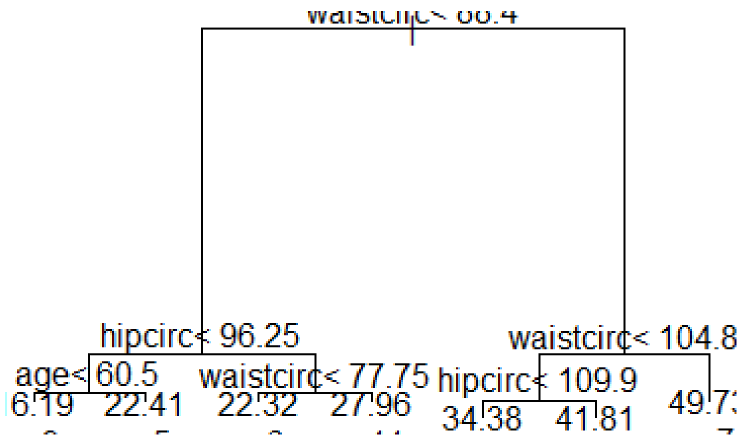
```
##           CP nsplit  rel error    xerror       xstd
## 1 0.67272638      0 1.00000000 1.0427457 0.19016187
## 2 0.09390665      1 0.32727362 0.5081173 0.11702581
## 3 0.06037503      2 0.23336696 0.4522296 0.09801847
## 4 0.03420446      3 0.17299193 0.3967005 0.09676249
## 5 0.01708278      4 0.13878747 0.3015476 0.07385485
## 6 0.01695763      5 0.12170469 0.2929969 0.06850104
## 7 0.01007079      6 0.10474706 0.2713231 0.06690466
## 8 0.01000000      7 0.09467627 0.2713231 0.06690466
```

```
print(bodyfat_rpart)
```

```
## n= 56
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 56 7265.0290000 30.94589
##    2) waistcirc< 88.4 31  960.5381000 22.55645
##      4) hipcirc< 96.25 14  222.2648000 18.41143
##        8) age< 60.5 9   66.8809600 16.19222 *
##        9) age>=60.5 5   31.2769200 22.40600 *
##       5) hipcirc>=96.25 17  299.6470000 25.97000
##        10) waistcirc< 77.75 6   30.7345500 22.32500 *
##        11) waistcirc>=77.75 11  145.7148000 27.95818
##          22) hipcirc< 99.5 3    0.2568667 23.74667 *
##          23) hipcirc>=99.5 8   72.2933500 29.53750 *
##    3) waistcirc>=88.4 25 1417.1140000 41.34880
##      6) waistcirc< 104.75 18  330.5792000 38.09111
##       12) hipcirc< 109.9 9   68.9996200 34.37556 *
##       13) hipcirc>=109.9 9   13.0832000 41.80667 *
##      7) waistcirc>=104.75 7  404.3004000 49.72571 *

plot(bodyfat_rpart)
text(bodyfat_rpart, use.n=T)
```

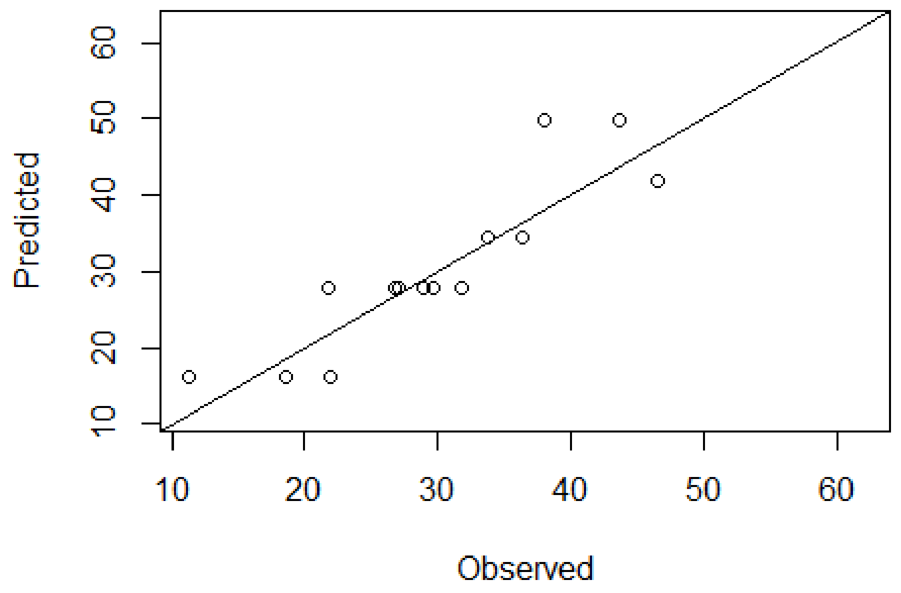


```
opt <- which.min(bodyfat_rpart$cptable[,"xerror"])
cp <- bodyfat_rpart$cptable[opt, "CP"]
bodyfat_prune <- prune(bodyfat_rpart, cp = cp)
print(bodyfat_prune)

## n= 56
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 56 7265.02900 30.94589
##    2) waistcirc< 88.4 31  960.53810 22.55645
##      4) hipcirc< 96.25 14  222.26480 18.41143
##         8) age< 60.5 9   66.88096 16.19222 *
##         9) age>=60.5 5   31.27692 22.40600 *
##       5) hipcirc>=96.25 17  299.64700 25.97000
##        10) waistcirc< 77.75 6   30.73455 22.32500 *
##        11) waistcirc>=77.75 11  145.71480 27.95818 *
##     3) waistcirc>=88.4 25 1417.11400 41.34880
##       6) waistcirc< 104.75 18  330.57920 38.09111
##        12) hipcirc< 109.9 9   68.99962 34.37556 *
##        13) hipcirc>=109.9 9   13.08320 41.80667 *
##       7) waistcirc>=104.75 7  404.30040 49.72571 *

plot(bodyfat_prune)
text(bodyfat_prune, use.n=T)
```



The predictions of a good model are expected to be equal or very close to their actual values, that is, most points should be on or close to the diagonal line.

```
DEXfat_pred <- predict(bodyfat_prune, newdata=bodyfat.test)
xlim <- range(bodyfat$DEXfat)
plot(DEXfat_pred ~ DEXfat, data=bodyfat.test, xlab="Observed",
ylab="Predicted", ylim=xlim, xlim=xlim)
abline(a=0, b=1)
```



Random Forest

Again, the iris data is first split into two subsets: training (70%) and test (30%).

```
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
trainData <- iris[ind==1,]
testData <- iris[ind==2,]
```

the formula is set to “Species ~ .”, which means to predict Species with all other variables in the data.

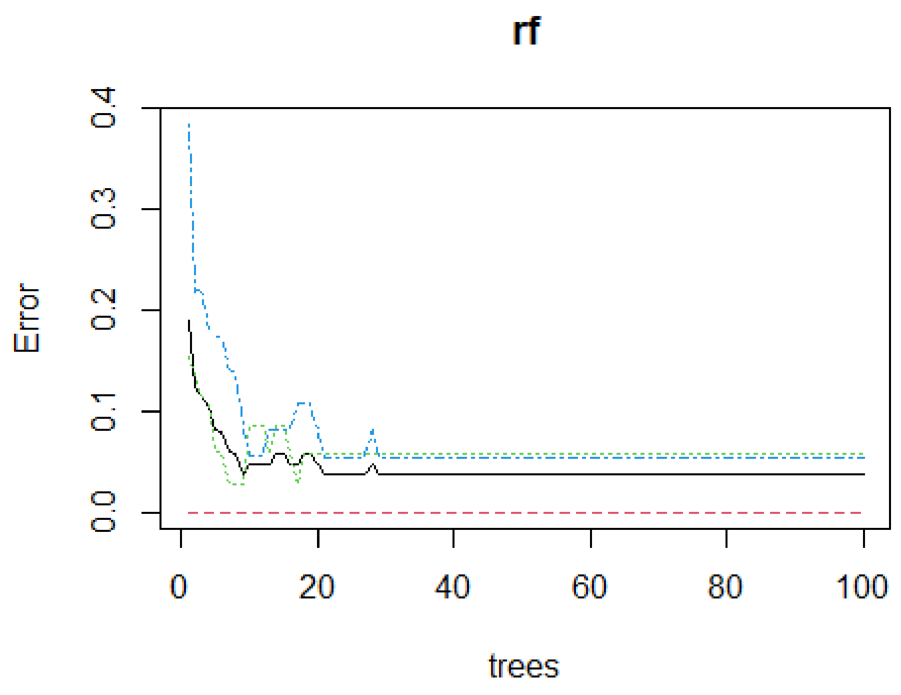
```
library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

rf <- randomForest(Species ~ ., data=trainData, ntree=100, proximity=TRUE)
table(predict(rf), trainData$Species)

##
##              setosa versicolor virginica
##   setosa         33          0         0
##   versicolor      0         33         2
##   virginica       0          2        35

print(rf)
```

```
##
## Call:
##  randomForest(formula = Species ~ ., data = trainData, ntree = 100,
proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3.81%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         33          0         0  0.00000000
## versicolor      0         33         2  0.05714286
## virginica       0          2        35  0.05405405
```

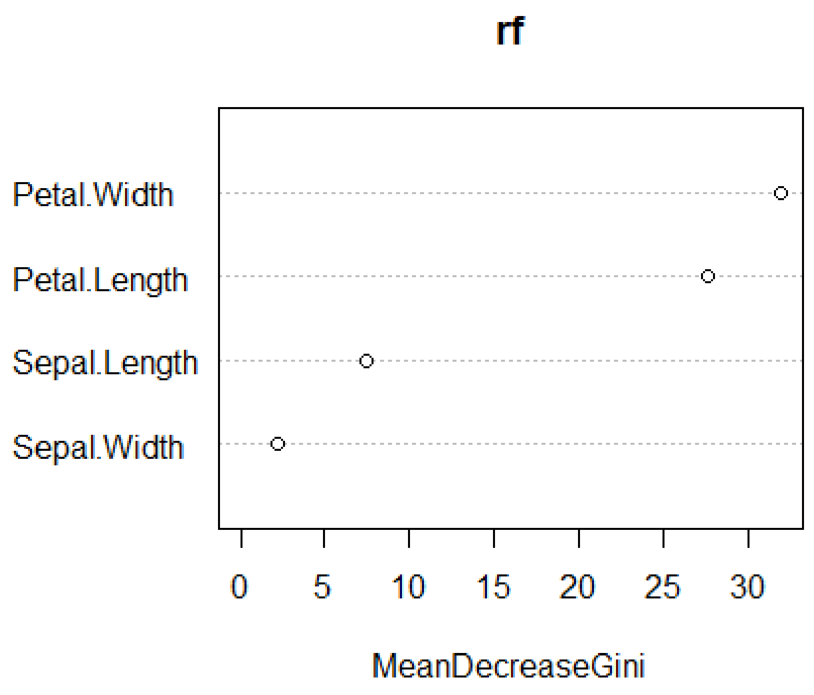
```
attributes(rf)
```

```
## $names
##  [1] "call"            "type"            "predicted"       "err.rate"
##  [5] "confusion"       "votes"           "oob.times"       "classes"
##  [9] "importance"      "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"          "y"
## [17] "test"            "inbag"           "terms"
##
## $class
## [1] "randomForest.formula" "randomForest"
```

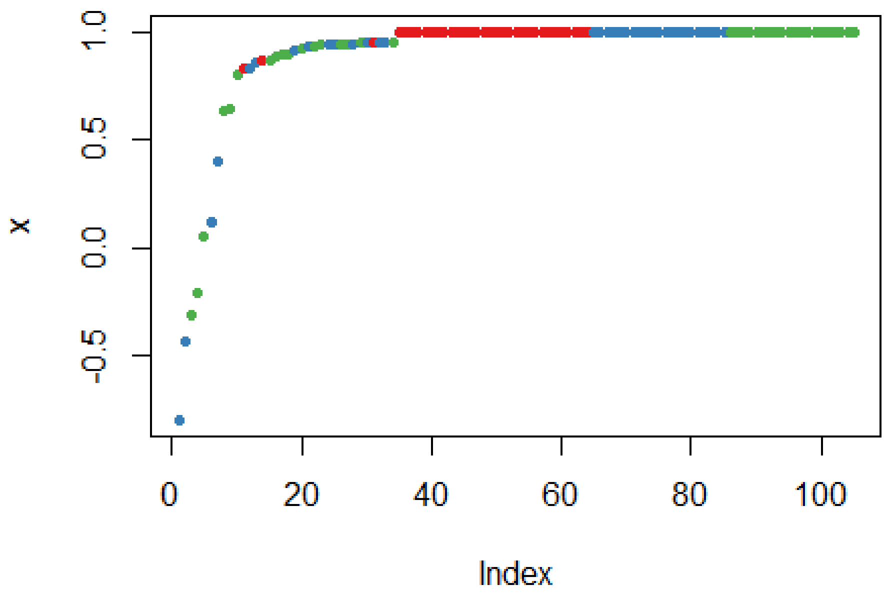
```
plot(rf)
```



```
importance(rf)

##              MeanDecreaseGini
## Sepal.Length         7.435541
## Sepal.Width          2.225215
## Petal.Length        27.629030
## Petal.Width         31.926785

 varImpPlot(rf)
```



```
 irisPred <- predict(rf, newdata=testData)
 table(irisPred, testData$Species)

##
## irisPred     setosa versicolor virginica
##   setosa         17          0         0
##   versicolor      0         15         2
##   virginica       0          0        11

plot(margin(rf, testData$Species))
```



Part I: Reading

Read V. Chandola, A. Banerjee and V. Kumar, "Outlier Detection: A Survey", ACM Computing Surveys Vol.41, no. 3. 2009. Answer the following questions:

Summarize three main outlier detection techniques

- (2.2.1) Supervised outlier detection techniques: Assume the training dataset contains normal and outlier class. The normal approach is to build predictive models for both normal and abnormal classes; data will be compared with both models to see where it belongs. Pros: techniques can build accurate models. Cons: labeled training data might be prohibitively expensive to obtain since it requires a lot of human effort to obtain the labeled training data set.
- (2.2.2) Semi-supervised outlier detection techniques: Assume the availability of only one class. The normal approach is to model only the available class and decare any test instance which does not fit this model to belong to the other class. The reason is very hard to obtain all the possible outlying behaviors that can occur in the training dataset. The techniques which model only the normal instances are very popular since they are easy to obtain. Also, normal behavior is easier to construct than abnormal.
- (2.2.3) Unsupervised outlier detection techniques: These techniques in this category does not make any assumption about label training, and it focus on making assumption about the data. For examples, normal instances are far more frequent than outliers. Thus a frequently occurring pattern is typically considered normal while a rare occurrence is an outlier. Unsupervised outlier detection techniques and semi-

supervised outlier detection techniques are widely used than supervised outlier detection techniques.

Summarize different types of outliers

- (2.3.1) Type I outliers: A data instance is an outlier due to its attribute values which are inconsistent with values taken by normal instances. Techniques that detect Type I outliers analyze the relation of an individual instance with respect to rest of the data instances.
- (2.3.2) Type II outliers : These outliers are caused due to the occurrence of an individual data instance in a specific context in the given data. Type II outliers are defined with respect to a context. In fact, a data instance may be a Type II outlier in a specific context (fall under some conditions); but its identical data instance (in terms of behavioral attributes) could be considered normal in a different context. Mostly explored in time-series data, and spatial data. For example, a credit card user usually spends $ 20 at a gas station and $ 200 at a jewelry store. If he spends $ 200 at someday, it considers a type II outliers.
- (2.3.3) Type III outliers: When a subset of data instances is outlying with respect to the entire data set, it is considered type III outliers. Its instance data is not outlying by itself, but their occurrences is the whole structure is anomalous. For example, normal shopping pattern of a credit card user is that the gas station, the grocery store, and the restaurant. Three of more purchase at a gas station on the same day (at different time or under more conditions) is potential card theft. This sequence of transactions is a Type III outlier. Type I outliers may be detected in any type of data. Type II and Type III outliers require the presence of sequential or spatial structure in the data. But the nature of outliers actually required to be detected by a particular outlier detection algorithm is determined by the specific problem formulation.

Summarize two strategies for evaluation of outlier detection techniques

- (15.1) Detecting outliers in a given data set: A labeling type of outlier detection technique is typically evaluated using any of the evaluation techniques from 2-class classification literature. In the following steps:

  First, choose the dataset.

  Second, split the data set into training and test (if invoke training).

  Third, use the training dataset to train the predicted model.

  Fourth, applying the outlier detection techniques to the test dataset to detect the outliers and normal.

  Finally, Use the predicted label compare with the actual labels to construct a confusion matrix.
- Conclusion: here are many evaluation metrics such as precision, recall, accuracy, false positive rate, false negative rate, detection rates, ROC-curve… have been applied in outlier detection literature. In fact, the choice of an evaluation data set depends on what type of data is the outlier detection technique targeted for. The UCI KDD archive contain many benchmark data sets that use to evaluate outlier algorithm. However,

most of them are only available for Type I outlier detection techniques; some can be used for Type II; none for type III.

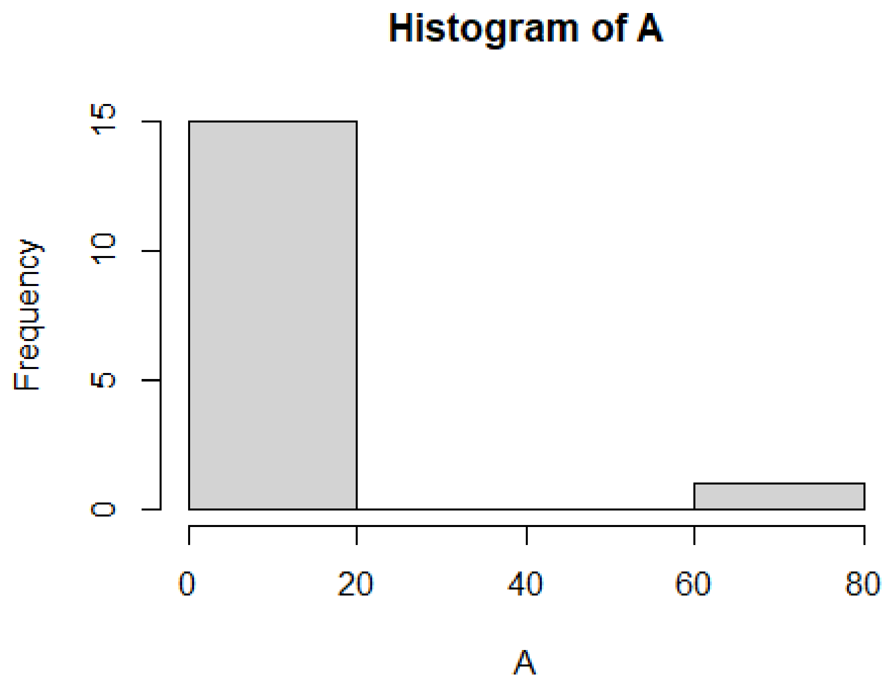
- (15.2) Evaluation in the application domain : Outlier detection is a highly application domain-oriented concept, and there is no techniques can be applied for all possible setting. One approach is to choose a dataset (sample) belong to the target application domain. Evaluation measures the performance of the entire outlier detection setting (including the technique, the features chosen and other related parameters/assumptions). It is also possible that evaluation technique that perform well in subsection, might not perform well in the application domain. But such evaluation is necessary to establish the usefulness of the technique in the target application domain. Challenges are benchmark data sets are not available publicly, and evaluation from application domain perspective is that labeled validation data is often not available at all.

Part II: Outlier Analysis

Question 1:

- Create a histogram of the data distribution. Using R Studio, and dplyr package to draw the histogram.

```
library(dplyr)

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':
##
##     combine

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

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

# Import data
A <- c(1, 3, 2, 1, 3, 2, 75, 1, 3, 2, 2, 1, 2, 3, 2, 1)
View (A)

library(ggplot2)

##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:randomForest':
##
##     margin

hist(A)
```



```
# Outlier has the value of 75
```

- Answer: Outlier has the value of 75.

Question 2:

- Compute the Z-value of each data point:

```
# 2.    Compute the Z-value of each data point.
# Which of these values can be considered the most extreme value?
Data <- data.frame(A)

Data %>%
  mutate(zscore = (A- mean(A))/sd(A))

##      A     zscore
## 1    1 -0.3008265
## 2    3 -0.1914351
## 3    2 -0.2461308
## 4    1 -0.3008265
## 5    3 -0.1914351
## 6    2 -0.2461308
## 7   75  3.7466579
## 8   1 -0.3008265
## 9   3 -0.1914351
## 10  2 -0.2461308
## 11  2 -0.2461308
## 12  1 -0.3008265
## 13  2 -0.2461308
## 14  3 -0.1914351
## 15  2 -0.2461308
## 16  1 -0.3008265
```

- Answer: Index 7(75) and z-score is 3.7466579 can be considered the most extreme value.

Question 3:

- Determine the nearest neighbor (k=1):

```
# 3.    Determine the nearest neighbor (k=1) of each data point.
# Which data points have the largest value of the nearest neighbor distance?
# Are they the correct outliers?
library(FNN)
get.knn(Data, k=1,algorithm=c("brute"))

## $nn.index
##       [,1]
##  [1,]    4
##  [2,]    5
##  [3,]    6
##  [4,]    4
##  [5,]    5
##  [6,]    6
##  [7,]    2
##  [8,]    4
##  [9,]    5
## [10,]    6
## [11,]    6
## [12,]    4
## [13,]    6
## [14,]    5
## [15,]    6
## [16,]    4
##
## $nn.dist
##       [,1]
##  [1,]    0
##  [2,]    0
##  [3,]    0
##  [4,]    0
##  [5,]    0
##  [6,]    0
##  [7,]   72
##  [8,]    0
##  [9,]    0
## [10,]    0
## [11,]    0
## [12,]    0
## [13,]    0
## [14,]    0
## [15,]    0
## [16,]    0

# The data show that value with index [7] have the largest value of the
nearest
# neighbor distance. Thus it is a outlier.
```

- Answer: The data show that value with index [7] have the largest value of the nearest neighbor distance. Thus it is a outlier.

Question 4:

```
# 4.    Apply a k-means clustering algorithm (k=2) to the data set.
# Which data points lie furthest from the two means (the centroids of the two
# clusters) found? Are they the correct outliers?
clusters <- kmeans(Data[,1], 2)
str(clusters)

## List of 9
##  $ cluster     : int [1:16] 2 2 2 2 2 2 1 2 2 2 ...
##  $ centers     : num [1:2, 1] 75 1.93
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : NULL
##  $ totss       : num 5014
##  $ withinss    : num [1:2] 0 8.93
##  $ tot.withinss: num 8.93
##  $ betweenss   : num 5005
##  $ size        : int [1:2] 1 15
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"

Data$Borough <- as.factor(clusters$cluster)
clusters[["centers"]]

##        [,1]
## 1 75.000000
## 2  1.933333
```

- Answer: Data points lie furthest from the two means (the centroids of the two clusters) found: There is two cluster found. However, the 75 is only contain 1 value (itself). Therefore, the value 75 itself is outlier.

Question 5:

- Answer: There is no node that consider outlier because all the node is connect to point 1.

Part III: Classification

Question 1-6:

- On paper.

Question 7-11:

- Import the data for Classification

```
train.data <- read.csv("~/R/DataMining/Decision_Tree/DATA.csv")
train.data $Instance <-NULL
train.data$a3 <- factor(train.data$a3)
train.data $class <- factor(train.data$class)
train.data

##      a1 a2 a3 class
## 1  TRUE  1  A     +
## 2  TRUE  6  A     +
## 3  TRUE  5  A     -
## 4 FALSE  4  B     +
## 5 FALSE  7  B     -
## 6 FALSE  3  B     -
## 7 FALSE  8  C     -
## 8  TRUE  7  C     +
## 9 FALSE  5  C     -
```

- Calculate the entropy using R

```
Entropy(train.data$a1,train.data$class)

## [1] 1.752715

Entropy(train.data$a2,train.data$class)

## [1] 2.947703

Entropy(train.data$a3,train.data$class)

## [1] 2.503258
```

Question 8

- Import the test data

```
# Create test data frame to test the data
a1<- c(TRUE,TRUE,TRUE,FALSE,FALSE,FALSE)
a2<- c(1,6,5,4,7,3)
a3<- c('A','A','B','B','C','C')
```

```
class<-c('+','+','-','+','-','-')
test.data<-data.frame(a1,a2,a3,class)
test.data$a2 <- as.integer(test.data$a2)
test.data$a3 <- factor(test.data$a3)
test.data$class <- factor(test.data$class)
test.data
```

```
##      a1 a2 a3 class
## 1  TRUE  1  A     +
## 2  TRUE  6  A     +
## 3  TRUE  5  B     -
## 4 FALSE  4  B     +
## 5 FALSE  7  C     -
## 6 FALSE  3  C     -
```

- Training Model

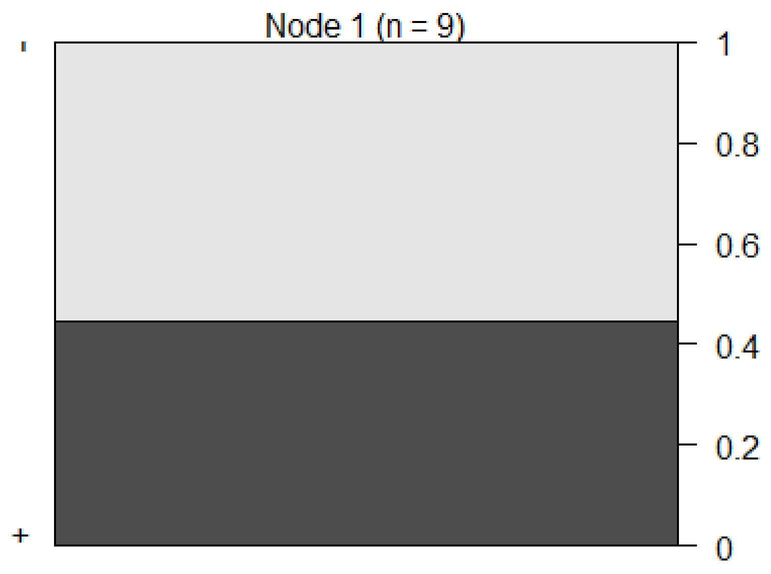
```
# Training Model
myFormula <- class ~ a1+a2+a3
small_ctree <- ctree(myFormula, data=train.data)

# Check the prediction:
table(predict(small_ctree), train.data$class)
```

```
##
##     - +
##   - 5 4
##   + 0 0
```

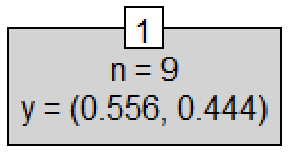
```
print(small_ctree)
```

```
##
##   Conditional inference tree with 1 terminal nodes
##
## Response:  class
## Inputs:  a1, a2, a3
## Number of observations:  9
##
## 1)*  weights = 9
```

```
# Draw the plot for the final tree generated by the decision tree classifier
plot(small_ctree)
```



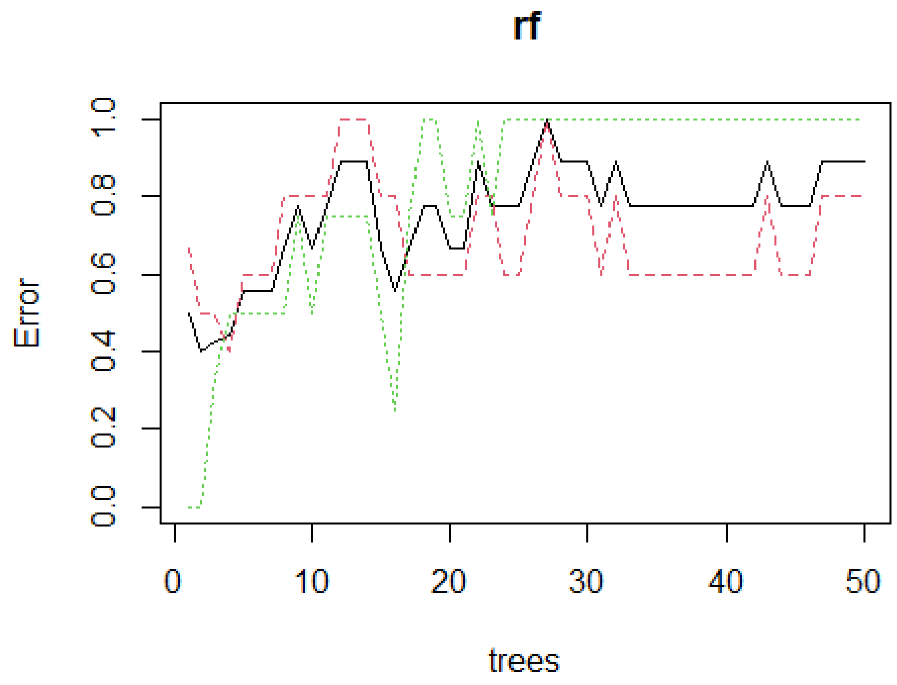
```
plot(small_ctree, type="simple")
```



```
# predict on test data
testPred <- predict(small_ctree, newdata = test.data)

# test the model prediction
table(testPred, test.data$class)

##
## testPred - +
##        - 3 3
##        + 0 0
```

Question 9,10,11:

```
rf <- randomForest(class ~ ., data=train.data, ntree=50, proximity=TRUE)
table(predict(rf), train.data$class)

##
##     - +
##   - 1 4
##   + 4 0

print(rf)

##
## Call:
##  randomForest(formula = class ~ ., data = train.data, ntree = 50,
proximity = TRUE)
##                Type of random forest: classification
##                      Number of trees: 50
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 88.89%
## Confusion matrix:
##   - + class.error
## - 1 4         0.8
## + 4 0         1.0

attributes(rf)

## $names
##  [1] "call"            "type"            "predicted"       "err.rate"
##  [5] "confusion"       "votes"           "oob.times"       "classes"
##  [9] "importance"      "importanceSD"    "localImportance" "proximity"
## [13] "ntree"           "mtry"            "forest"          "y"
## [17] "test"            "inbag"           "terms"
##
## $class
## [1] "randomForest.formula" "randomForest"

plot(rf)
```


rf
Error
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50
trees