Data Mining Part 2

Veerasak Kritsanapraphan

4/6/2017

Agenda

- Data Mining Techniques using R
- Predictive Modeling Performance
- Clustering
  - K-Means Clustering
  - Hierarchical Clustering
- Association Rules (Market Basket Analysis)
- Multi-model Learning
- Text-Mining

Slide and Sample Data

https://github.com/vkrit/chula_datamining.

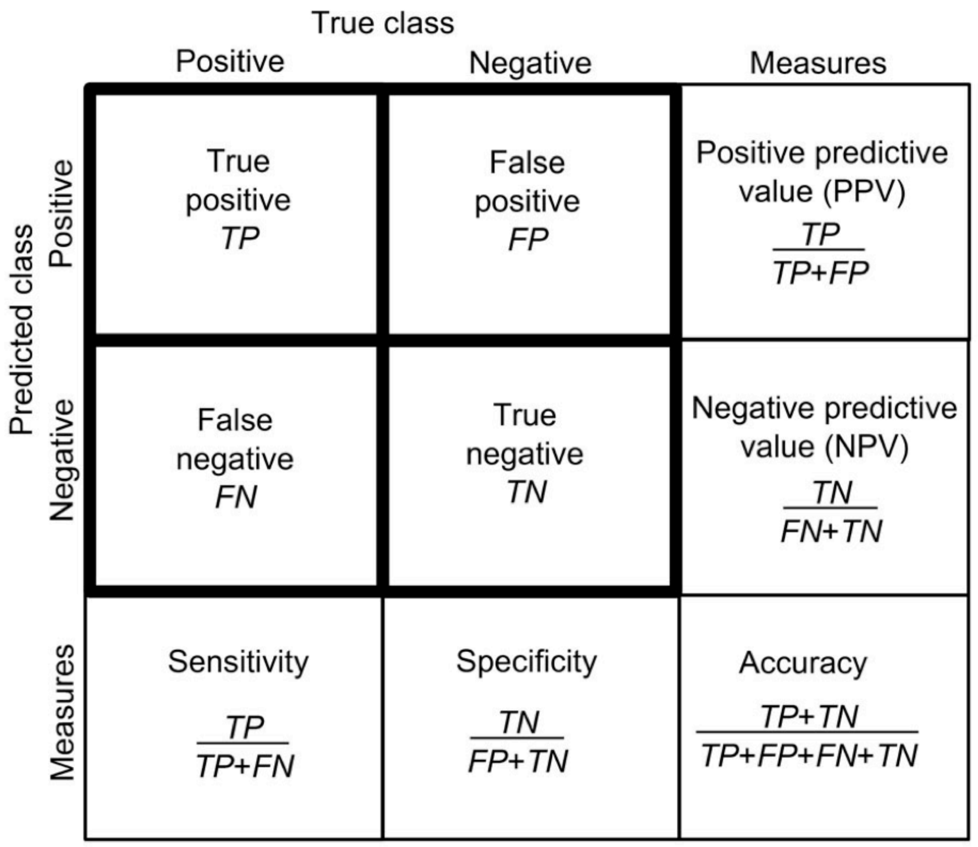
3/51

Prepare Data

```r
# Prepare iris
set.seed(567)
ind <- sample(2, nrow(iris), replace = TRUE, prob = c(0.7,
    0.3))
traindata <- iris[ind == 1, ]
testdata <- iris[ind == 2, ]
table(traindata$Species)
```

```
##
##     setosa versicolor  virginica
##         35         37         35
```

Predictive Modeling Performance



Predicted class	True class: Positive	True class: Negative	Measures
Positive	True positive *TP*	False positive *FP*	Positive predictive value (PPV) $\frac{TP}{TP+FP}$
Negative	False negative *FN*	True negative *TN*	Negative predictive value (NPV) $\frac{TN}{FN+TN}$
Measures	Sensitivity $\frac{TP}{TP+FN}$	Specificity $\frac{TN}{FP+TN}$	Accuracy $\frac{TP+TN}{TP+FP+FN+TN}$

Create Decision Tree Model

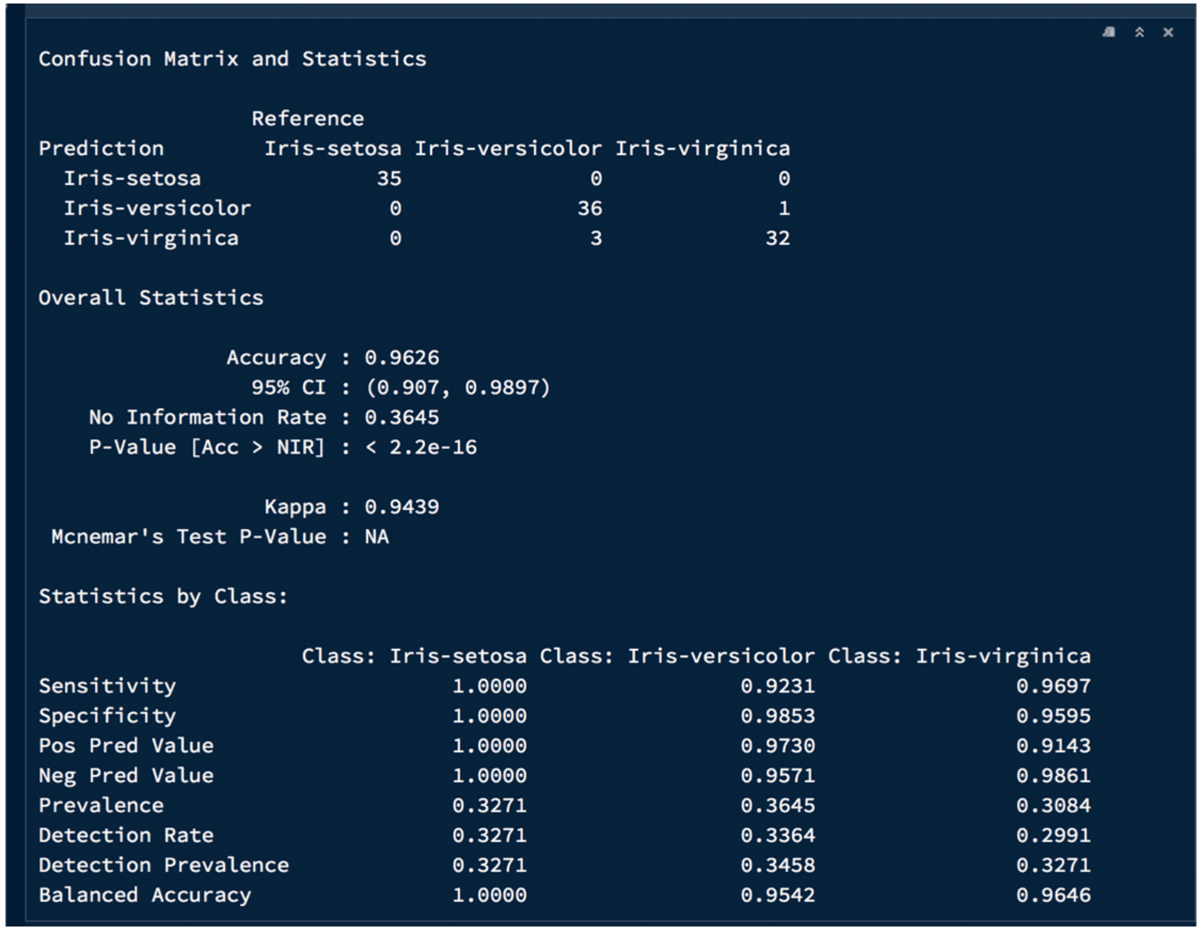
```
library(party)

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

Create Confusion Matrix from Training Data

```
library(caret)
trainPred = predict(iris_ctree, traindata)
confusionMatrix(traindata$Species, trainPred)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         35          0         0
##   versicolor      0         36         1
##   virginica       0          3        32
##
## Overall Statistics
##
##                Accuracy : 0.9626
```

Confusion Matrix



```
Confusion Matrix and Statistics

                 Reference
Prediction        Iris-setosa Iris-versicolor Iris-virginica
  Iris-setosa              35               0              0
  Iris-versicolor           0              36              1
  Iris-virginica            0               3             32

Overall Statistics

               Accuracy : 0.9626
                 95% CI : (0.907, 0.9897)
    No Information Rate : 0.3645
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9439
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: Iris-setosa Class: Iris-versicolor Class: Iris-virginica
Sensitivity                      1.0000                 0.9231                0.9697
Specificity                      1.0000                 0.9853                0.9595
Pos Pred Value                   1.0000                 0.9730                0.9143
Neg Pred Value                   1.0000                 0.9571                0.9861
Prevalence                       0.3271                 0.3645                0.3084
Detection Rate                   0.3271                 0.3364                0.2991
Detection Prevalence             0.3271                 0.3458                0.3271
Balanced Accuracy                1.0000                 0.9542                0.9646
```

K-Means Clustering

1. Pick an initial set of K centroids (this can be random or any other means)
2. For each data point, assign it to the member of the closest centroid according to the given distance function
3. Adjust the centroid position as the mean of all its assigned member data points. Go back to (2) until the membership isn't change and centroid position is stable.
4. Output the centroids

K-Means (cont.)

```r
library(stats)
set.seed(101)
km <- kmeans(iris[, 1:4], 3)
plot(iris[, 1], iris[, 2], col = km$cluster)
points(km$centers[, c(1, 2)], col = 1:2, pch = 19, cex = 2)
```

K-Means (cont.)

```
table(km$cluster, iris$Species)
```

```
##
##      setosa versicolor virginica
##   1       0         48        14
##   2      50          0         0
##   3       0          2        36
```

K-Means (second round)

```
set.seed(900)
km <- kmeans(iris[, 1:4], 3)
plot(iris[, 1], iris[, 2], col = km$cluster)
points(km$centers[, c(1, 2)], col = 1:3, pch = 19, cex = 2)
```

K-Means (second round - cont.)

```
##
##     setosa versicolor virginica
##   1      0         46        50
##   2     17          4         0
##   3     33          0         0
```

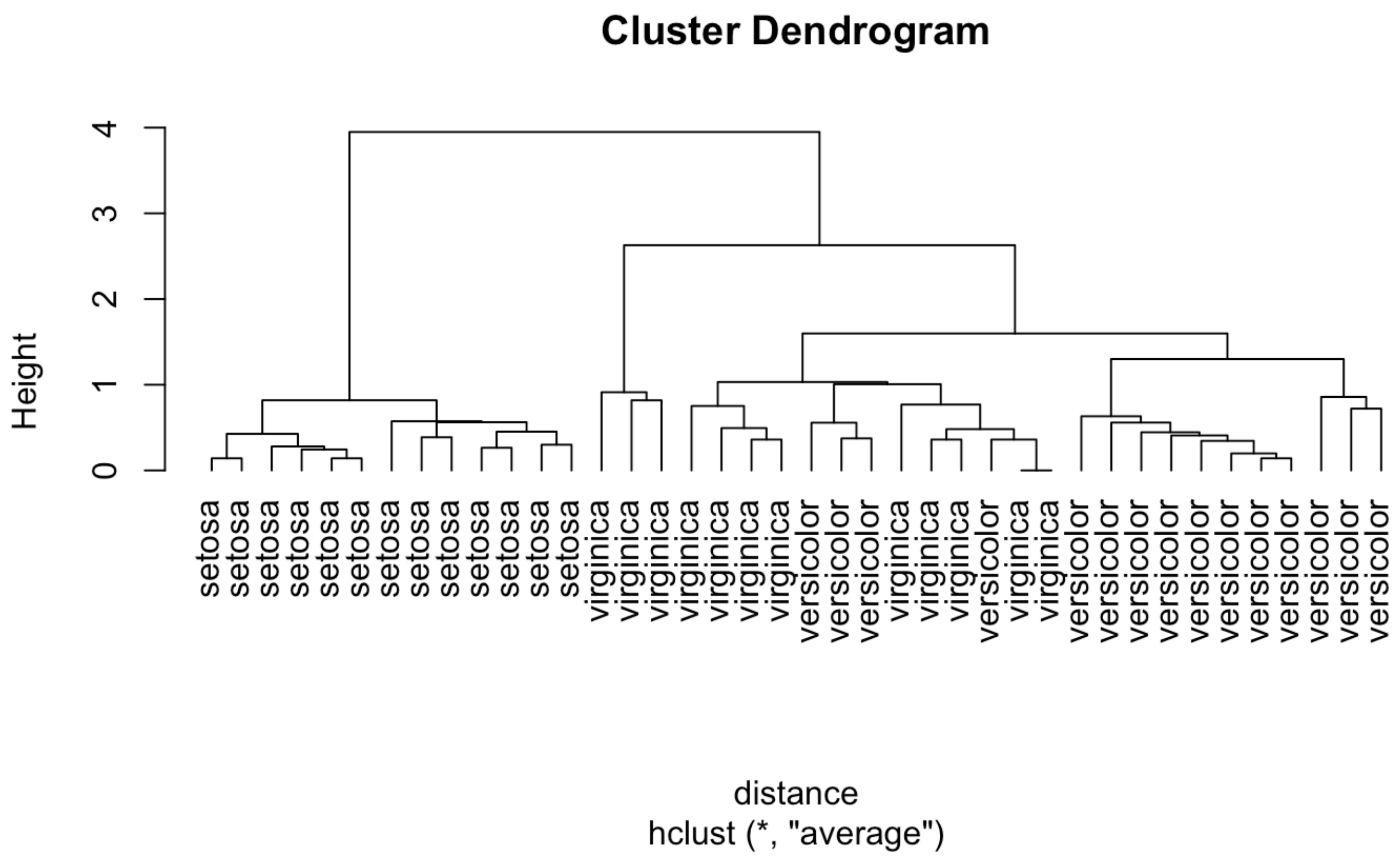
Hierarchical Clustering

1. Compute distance between every pairs of point/cluster
   - Distance between point is just using the distance function.
   - Compute distance between point A to cluster B may involve many choices (such as the min/max/avg distance between the point A and points in the cluster B)
   - Compute distance between cluster A to cluster B may first compute distance of all points pairs (one from cluster A and the the other from cluster B) and then pick either min/max/avg of these pairs.
2. Combine the two closest point/cluster into a cluster. Go back to (1) until only one big cluster remains

Hierarchical Clustering

```r
set.seed(101)
sampleiris <- iris[sample(1:150, 40), ]  # get samples from iris dataset
# each observation has 4 variables, ie, they are
# interpreted as 4-D points
distance <- dist(sampleiris[, -5], method = "euclidean")
cluster <- hclust(distance, method = "average")
```

```
plot(cluster, hang = -1, label = sampleiris$Species)
```



Prune the result tree to 3 groups

```
group.3 <- cutree(cluster, k = 3)  # prune the tree
table(group.3, sampleiris$Species)
```

```
##
## group.3 setosa versicolor virginica
##       1      0         15         9
##       2     13          0         0
##       3      0          0         3
```

Plot cluster by column 1 and 2

```
par(mfrow = c(1, 2))
plot(sampleiris[, c(1, 2)], col = group.3, pch = 19, cex = 1,
    main = "3 clusters")
plot(sampleiris[, c(1, 2)], col = sampleiris$Species, pch = 19,
    cex = 1, main = "real clusters")
```

Association Rules (Market Basket Analysis)

Support: The rule holds with support sup in T (the transaction data set) if sup % of transactions contain X Y.

Confidence: The rule holds in T with confidence conf if conf% of tranactions that contain X also contain Y.

Lift : The Lift of the rule is X=>Y is the confidence of the rule divided by the expected confidence, assuming that the item sets are independent.

Apriori Algorithm

```r
# Loead the libraries
library(registry)
library(Matrix)
library(arules)
library(arulesViz)
library(datasets)

# Load the data set
data(Groceries)
```

Data Format

Explore Groceries Data

```r
# Create an item frequency plot for the top 20 items
itemFrequencyPlot(Groceries, topN = 20, type = "absolute")
```

Create Association Rules

```
rules <- apriori(Groceries, parameter = list(supp = 0.001,
    conf = 0.8))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.8    0.1    1 none FALSE            TRUE   0.001      1     10
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
```

Plot Rules

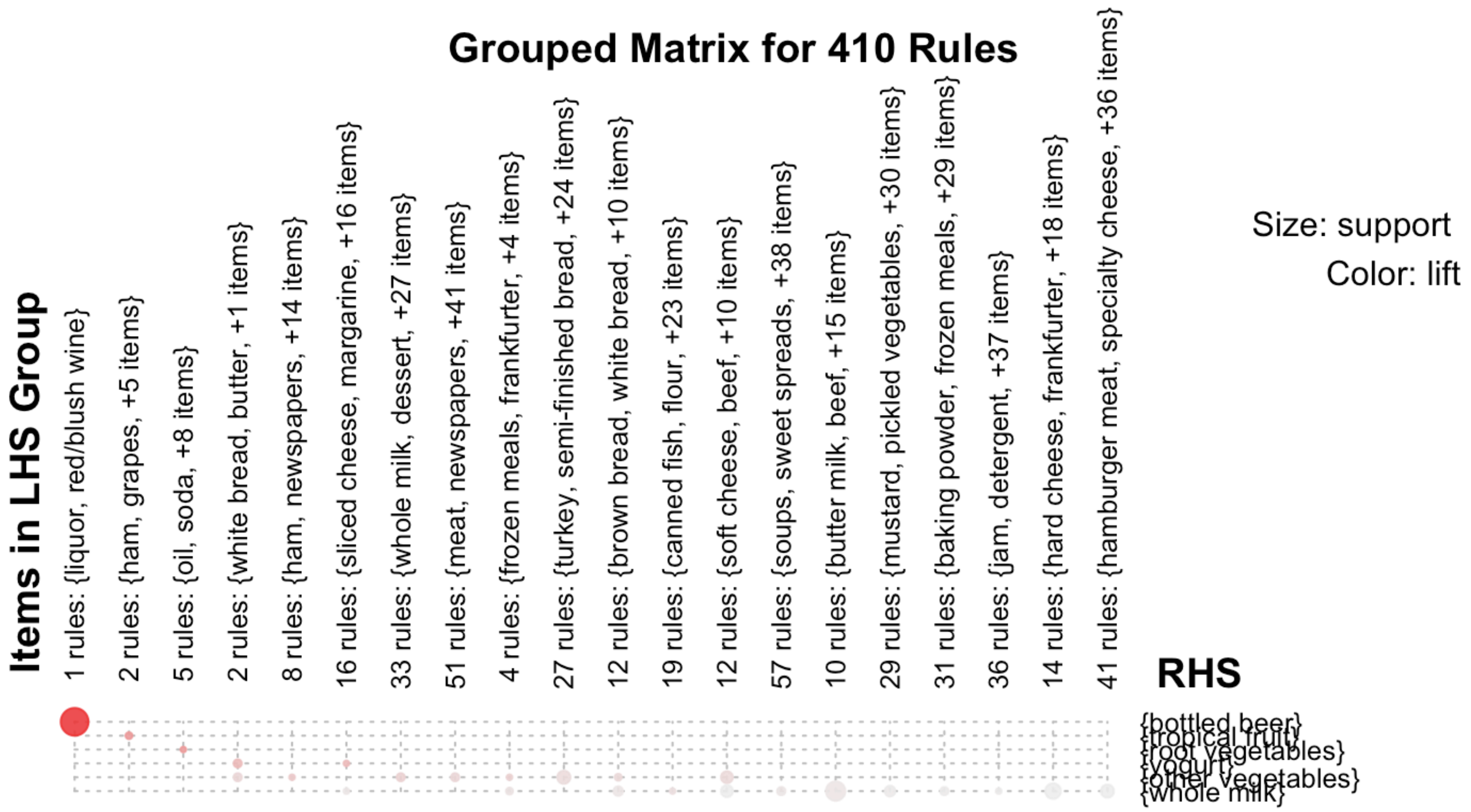
```
plot(rules)
```

Sort Rules

```
rules <- sort(rules, by = "confidence", decreasing = TRUE)
inspect(rules[1:5])
```

```
##   lhs                        rhs           support confidence lift
## 1 {rice,
##    sugar}                 => {whole milk}  0.0012          1  3.9
## 2 {canned fish,
##    hygiene articles}      => {whole milk}  0.0011          1  3.9
## 3 {root vegetables,
##    butter,
##    rice}                  => {whole milk}  0.0010          1  3.9
## 4 {root vegetables,
##    whipped/sour cream,
##    flour}                 => {whole milk}  0.0017          1  3.9
## 5 {butter,
```

```r
plot(rules, method = "grouped")
```

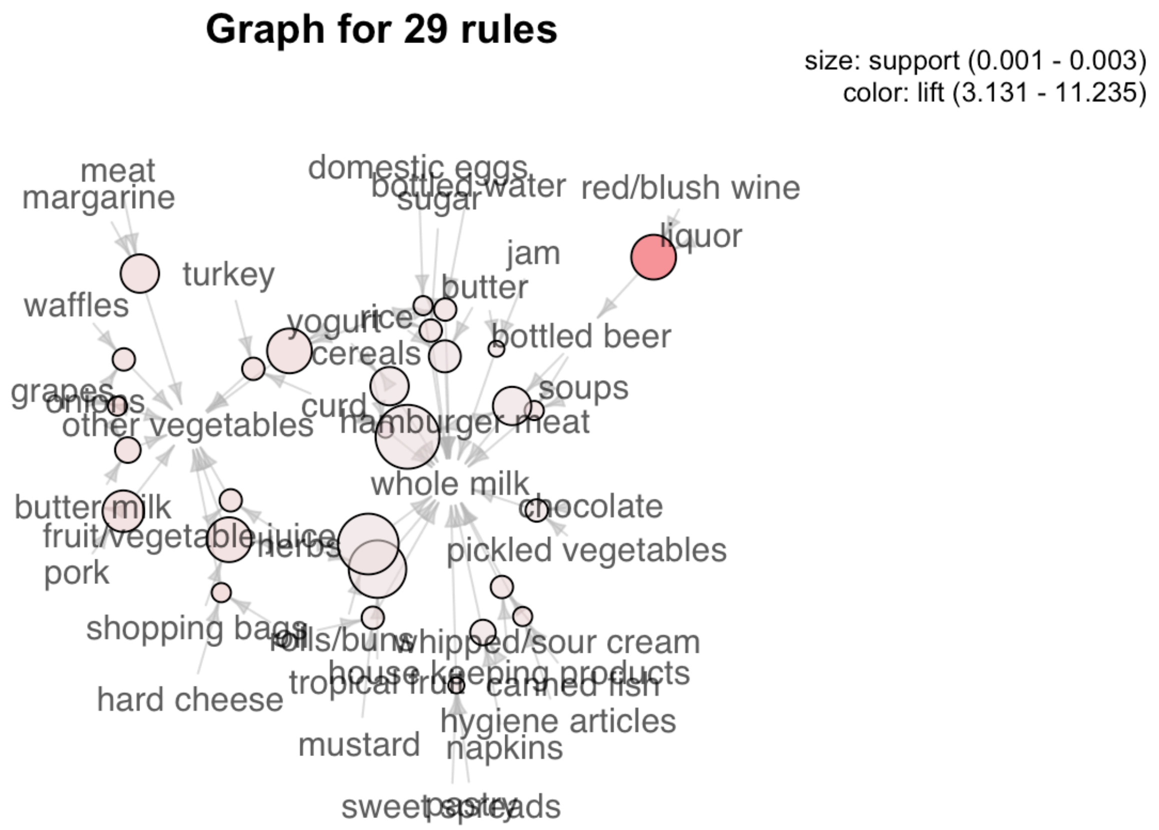


Change to have limit association in one rule

```
rules <- apriori(Groceries, parameter = list(supp = 0.001,
    conf = 0.8, maxlen = 3))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.8    0.1    1 none FALSE            TRUE   0.001      1      3
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
```

```
plot(rules, method = "graph")
```



Rules pruned

```
subset.matrix <- is.subset(rules, rules)
subset.matrix[lower.tri(subset.matrix, diag = T)] <- NA
redundant <- colSums(subset.matrix, na.rm = T) >= 1
rules.pruned <- rules[!redundant]
rules <- rules.pruned
```

```
summary(rules)
```

```
## set of 29 rules
##
## rule length distribution (lhs + rhs):sizes
##  3
## 29
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       3       3       3       3       3       3
##
## summary of quality measures:
##     support           confidence        lift
##  Min.   :0.00102   Min.   :0.80   Min.   : 3.1
##  1st Qu.:0.00112   1st Qu.:0.81   1st Qu.: 3.3
##  Median :0.00122   Median :0.85   Median : 3.6
##  Mean   :0.00147   Mean   :0.86   Mean   : 4.0
```

Targeting Items

- What are customers likely to buy before buying whole milk?
- What are customers likely to buy if they purchase whole milk?
- This essentially means we want to set either the Left Hand Side adn Right Hand Side. This is not difficult to do with R!

Find whole milk's antecedents

```
rules <- apriori(data = Groceries, parameter = list(supp = 0.001,
    conf = 0.08), appearance = list(default = "lhs", rhs = "whole milk"),
    control = list(verbose = F))
rules <- sort(rules, decreasing = TRUE, by = "confidence")
inspect(rules[1:5])
```

```
##   lhs                        rhs           support confidence lift
## 1 {rice,
##    sugar}                 => {whole milk}   0.0012          1   3.9
## 2 {canned fish,
##    hygiene articles}      => {whole milk}   0.0011          1   3.9
## 3 {root vegetables,
##    butter,
##    rice}                  => {whole milk}   0.0010          1   3.9
## 4 {root vegetables,
```

Likely to buy after buy whole milk

```
rules <- apriori(data = Groceries, parameter = list(supp = 0.001,
    conf = 0.15, minlen = 2), appearance = list(default = "rhs",
    lhs = "whole milk"), control = list(verbose = F))
rules <- sort(rules, decreasing = TRUE, by = "confidence")
inspect(rules[1:5])
```

```
##   lhs             rhs                support confidence lift
## 6 {whole milk} => {other vegetables} 0.075   0.29       1.5
## 5 {whole milk} => {rolls/buns}       0.057   0.22       1.2
## 4 {whole milk} => {yogurt}           0.056   0.22       1.6
## 2 {whole milk} => {root vegetables}  0.049   0.19       1.8
## 1 {whole milk} => {tropical fruit}   0.042   0.17       1.6
```

Bagging and Boosting using R

Multi-model Learning

Ensemble : Bagging

Random Forest

- Here is how such a system is trained; for some number of trees T:
- Sample N cases at random with replacement to create a subset of the data. The subset should be about 66% of the total set.
- At each node:
  - For some number m (see below), m predictor variables are selected at random from all the predictor variables
  - The predictor variable that provides the best split, according to some objective function, is used to do a binary split on that node.
  - At the next node, choose another m variables at random from all predictor variables and do the same.

36/51

Bagging

```r
library(ggplot2)
library(randomForest)
# Train 500 trees, random selected attributes
model <- randomForest(Species ~ ., data = traindata, nTree = 500)
prediction <- predict(model, newdata = testdata, type = "class")
table(prediction, testdata$Species)
```

```
##
## prediction   setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         13         2
##   virginica       0          0        13
```

Boosting

```
library(adabag)
iris.adaboost <- boosting(Species ~ ., data = traindata, boost = TRUE,
    mfinal = 5)
iris.adaboost
```

```
## $formula
## Species ~ .
##
## $trees
## $trees[[1]]
## n= 107
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
```

Plot variables important

```
barplot(iris.adaboost$imp[order(iris.adaboost$imp, decreasing = TRUE)],
    ylim = c(0, 100), main = "Variables Relative Importance",
    col = "lightblue")
```

Boosting (compare result)

```
table(iris.adaboost$class, traindata$Species, dnn = c("Predicted Class",
    "Observed Class"))
```

```
##                  Observed Class
## Predicted Class setosa versicolor virginica
##      setosa         35          0         0
##      versicolor      0         35         0
##      virginica       0          2        35
```

Text Mining

- Get Text Mining Library

```
# Needed <- c('tm', 'SnowballCC', 'RColorBrewer',
# 'wordcloud', 'biclust', 'igraph', 'fpc')
# install.packages(Needed, dependencies = TRUE)
```

Load file (Shakespear's Plays)

```
TEXTFILE = "t8.shakespeare.txt"
if (!file.exists(TEXTFILE)) {
    download.file("https://ocw.mit.edu/ans7870/6/6.006/s08/lecturenotes/files/t8.shake
        destfile = TEXTFILE)
}
shakespeare = readLines(TEXTFILE)
length(shakespeare)
```

```
## [1] 124456
```

```
shakespeare = shakespeare[-(1:173)]
shakespeare = shakespeare[-(124195:length(shakespeare))]
shakespeare = paste(shakespeare, collapse = " ")
shakespeare = strsplit(shakespeare, "<<[^>]*>>")[[1]]
```

Text Mining

```
library(tm)
docs.vec <- VectorSource(shakespeare)
docs.corpus <- Corpus(docs.vec)
summary(docs.corpus)
```

```
##      Length Class             Mode
## 1    2      PlainTextDocument list
## 2    2      PlainTextDocument list
## 3    2      PlainTextDocument list
## 4    2      PlainTextDocument list
## 5    2      PlainTextDocument list
## 6    2      PlainTextDocument list
## 7    2      PlainTextDocument list
## 8    2      PlainTextDocument list
## 9    2      PlainTextDocument list
```

Text Mining Basic

```
# Remove Punctuation
docs.corpus <- tm_map(docs.corpus, removePunctuation)
head(docs.corpus)
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 6
```

```
# Remove Number
docs.corpus <- tm_map(docs.corpus, removeNumbers)
docs.corpus <- tm_map(docs.corpus, tolower)
# Remove Stopwords
docs.corpus <- tm_map(docs.corpus, removeWords, stopwords("english"))
```

Clean Data

```
# remove ing s, es
library(SnowballC)
docs.corpus <- tm_map(docs.corpus, stemDocument)
docs.corpus <- tm_map(docs.corpus, stripWhitespace)
```

Step of Text Mining

Create Document Term Matrix

```
# Create Document Term Matrix
dtm <- DocumentTermMatrix(docs.corpus)
inspect(dtm[1:10, 1:10])
```

```
## <<DocumentTermMatrix (documents: 10, terms: 10)>>
## Non-/sparse entries: 19/81
## Sparsity           : 81%
## Maximal term length: 6
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs accept addit agent allow alter altern appli aris asi associ
##   1       1     1     1     1     1      2     1    1   1      1
```

Create Term Document Matrix

```
# Create Term Document Matrix
tdm <- TermDocumentMatrix(docs.corpus)
inspect(tdm[1:10, 1:10])
```

```
## <<TermDocumentMatrix (terms: 10, documents: 10)>>
## Non-/sparse entries: 19/81
## Sparsity           : 81%
## Maximal term length: 6
## Weighting          : term frequency (tf)
## Sample             :
##         Docs
## Terms    1 10 2 3 4 5 6 7 8 9
##   accept 1  0 2 0 0 0 0 0 0 0
##   addit  1  0 2 0 0 1 0 1 0 0
##   agent  1  0 0 0 0 0 0 0 0 0
```

Explore Data

```
# Explore Data
freq <- colSums(as.matrix(dtm))
length(freq)
```

```
## [1] 18786
```

```
ord <- order(freq)
head(ord)
```

```
## [1]  9 11 13 14 15 20
```

Removing sparse terms

```
# Start by removing sparse terms:
TDM.common = removeSparseTerms(tdm, 0.1)
dim(tdm)
```

```
## [1] 18786   219
```

```
dim(TDM.common)
```

```
## [1]   0 219
```

```
m = as.matrix(tdm)
v = sort(rowSums(m), decreasing = TRUE)
d <- data.frame(word = names(v), freq = v)
head(d, 10)
```

Create Word Cloud

```
library(wordcloud)
set.seed(1234)
wordcloud(words = d$word, freq = d$freq, min.freq = 1, max.words = 200,
    random.order = FALSE, rot.per = 0.35, colors = brewer.pal(8,
        "Dark2"))
```

50/51

email : veerasak.kr568@cbs.chula.ac.th

Thank you