Holsenbeck_S_4

Stephen Synchronicity

2018-02-20

```r
# Set Assignment html below
Q <- read_html("https://da5030.weebly.com/assignment-4.html") %>% html_nodes(xpath = "//div[contains(@cl
for (i in seq_along(Q)) {
    Q[i] <- Q[i] %>% gsub("<li>", paste("## ", i, "\n<div class='q'>", sep = ""),
        ., perl = T) %>% gsub("</li>", paste("\n</div>\n<p class='a'>\n```{r '",
        i, "'}\n```\n</p>", sep = ""), ., perl = T) %>% str_split("\n")
}
sapply(Q, FUN = "cat", sep = "\n", simplify = T)
```

1

Build an R Notebook of the SMS message filtering example in the textbook on pages 103 to 123. Show each step and add appropriate documentation. This is the same as Lesson 4.

```r
# SMS data from http://dcomp.sor.ufscar.br/talmeida/smspamcollection/ Tiago
# Agostinho de Almeida and José María Gómez Hidalgo hold the copyright (c) for
# the SMS Spam Collection v.1.  -------------------- Fri Feb 16 20:15:00 2018
# --------------------# Load Data
sms <- read.csv("sms_spam.csv", stringsAsFactors = FALSE)
# data structure
str(sms)
```

```
## 'data.frame':    5559 obs. of  2 variables:
##  $ type: chr  "ham" "ham" "ham" "spam" ...
##  $ text: chr  "Hope you are having a good week. Just checking in" "K..give back my thanks." "Am also
```

```r
# turn type into a factor
sms$type <- factor(sms$type, levels = c("spam", "ham"))
# verify factor
str(sms$type)
```

```
##  Factor w/ 2 levels "spam","ham": 2 2 2 1 1 2 2 2 1 2 ...
```

```r
# count frequencies
table(sms$type) %>% print %>% sapply(FUN = function(x) {
    x/length(sms$type)
})
```

```
##
## spam  ham
##  747 4812

##      spam       ham
## 0.1343767 0.8656233
```

```r
# load text mining package
library(tm)
# Create a volatile (stored in RAM for rapid manipulation) corpus from a vector
```

```
# document source. Parentheses around variable assignment prints result
(sms.crp <- VCorpus(VectorSource(sms$text)))
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 5559
```

```
# ------------------- Fri Feb 16 21:35:27 2018 --------------------# Look at
# individual corpus items
inspect(sms.crp[1:2])
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 2
##
## [[1]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 49
##
## [[2]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 23
```

```
# View actual text
lapply(sms.crp[c(1:5)], as.character)
```

```
## $`1`
## [1] "Hope you are having a good week. Just checking in"
##
## $`2`
## [1] "K..give back my thanks."
##
## $`3`
## [1] "Am also doing in cbe only. But have to pay."
##
## $`4`
## [1] "complimentary 4 STAR Ibiza Holiday or Â£10,000 cash needs your URGENT collection. 09066364349 N
##
## $`5`
## [1] "okmail: Dear Dave this is your final notice to collect your 4* Tenerife Holiday or #5000 CASH a
```

```
# Map the tolower function to the corpus to convert all text to lowercase
sms.crpClean <- tm_map(sms.crp, content_transformer(tolower))
# Show the transformation
as.character(sms.crp[[1]])
```

```
## [1] "Hope you are having a good week. Just checking in"
```

```
as.character(sms.crpClean[[1]])
```

```
## [1] "hope you are having a good week. just checking in"
```

```
# View all transformation types
getTransformations()
```

```
## [1] "removeNumbers"     "removePunctuation" "removeWords"
## [4] "stemDocument"      "stripWhitespace"
```

```r
# remove numbers
sms.crpClean <- tm_map(sms.crpClean, removeNumbers)
# remove stop words: removeWords removes any list of words, stopwords genereates
# vector of stopwords (defaults to english stopwords.)
sms.crpClean <- tm_map(sms.crpClean, removeWords, stopwords())
# remove punctuation, but we would rather replace it first: sms.crpClean <-
# tm_map(sms.crpClean, removePunctuation)
replacePunctuation <- function(x) {
    (gsub("[[:punct:]]+", " ", x))
}
# Replace punctuation with a space
sms.crpClean <- tm_map(sms.crpClean, content_transformer(replacePunctuation))
# If any additional punctuation characters were not replaced by this function,
# remove them
sms.crpClean <- tm_map(sms.crpClean, removePunctuation)
# Verify it worked
as.character(sms.crpClean[[1]])
```

```
## [1] "hope     good week  just checking "
```

```r
# ------------------- Fri Feb 16 21:57:14 2018 --------------------# Demo
# snowballc
library(SnowballC)
# strips endings such that only rootword remains
wordStem(c("learn", "learned", "learning", "learns"))
```

```
## [1] "learn" "learn" "learn" "learn"
```

```r
# map stemDocument to the corpus
sms.crpClean <- tm_map(sms.crpClean, stemDocument)
# Test stripWhitespace
tm_map(sms.crpClean, stripWhitespace)[[1]]
```

```
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 25
```

```r
# Map it to corpus
sms.crpClean <- tm_map(sms.crpClean, stripWhitespace)
# Ensure all transformations worked
lapply(sms.crp[1:3], as.character)
```

```
## $`1`
## [1] "Hope you are having a good week. Just checking in"
##
## $`2`
## [1] "K..give back my thanks."
##
## $`3`
## [1] "Am also doing in cbe only. But have to pay."
```

```r
lapply(sms.crpClean[1:3], as.character)
```

```
## $`1`
## [1] "hope good week just check"
##
```

```
## $`2`
## [1] "k give back thank"
##
## $`3`
## [1] "also cbe pay"
```

```
# -------------------- Fri Feb 16 22:07:07 2018 --------------------# Document
# Term Matrix create a document-term sparse matrix sms.dtm <-
# DocumentTermMatrix(sms.crpClean) throwing an error Error in .tolower(txt) :
# invalid input 'Ã«â€' in 'utf8towcs' it appears that some invalid UTF8
# characters are in the corpus We will try to remove them removeSpecial <-
# function(x){gsub('Ã|«|â|€',' ',x)} sms.crpClean <- tm_map(sms.crpClean,
# content_transformer(removeSpecial)) if this created any extra whitespace remove
# it sms.crpClean <- tm_map(sms.crpClean, stripWhitespace) Try again sms.dtm <-
# DocumentTermMatrix(sms.crpClean) Still error. This post may help:
# https://stackoverflow.com/questions/9637278/r-tm-package-invalid-input-in-utf8towcs
sms.crpClean <- tm_map(sms.crpClean, content_transformer(function(x) iconv(enc2utf8(x),
    sub = "byte")))
sms.dtm <- DocumentTermMatrix(sms.crpClean)
# Seems to have worked! stripWhitespace was run again but is not copied here.


# alternative solution: create a document-term sparse matrix directly from the
# SMS corpus
sms.dtm2 <- DocumentTermMatrix(sms.crp, control = list(tolower = TRUE, removeNumbers = TRUE,
    stopwords = TRUE, removePunctuation = TRUE, stemming = TRUE))

# No errors using this method -------------------- Mon Feb 19 19:11:44 2018
# --------------------# Compare the two matrices
sms.dtm
```

```
## <<DocumentTermMatrix (documents: 5559, terms: 6138)>>
## Non-/sparse entries: 42844/34078298
## Sparsity           : 100%
## Maximal term length: 34
## Weighting          : term frequency (tf)
```

```
sms.dtm2
```

```
## <<DocumentTermMatrix (documents: 5559, terms: 6971)>>
## Non-/sparse entries: 43240/38708549
## Sparsity           : 100%
## Maximal term length: 40
## Weighting          : term frequency (tf)
```

```
# Using the same stopword removal method as with sms.dtm
sms.dtm2 <- DocumentTermMatrix(sms.crp, control = list(tolower = TRUE, removeNumbers = TRUE,
    stopwords = function(x) {
        removeWords(x, stopwords())
    }, removePunctuation = TRUE, stemming = TRUE))
# Still a difference in the number of entries but I think this might have to do
# with us manuallyremoving some non-unicode characters with gsub due to the
# error. Also, the order of pre-processing steps makes a difference as well.
# -------------------- Mon Feb 19 19:18:35 2018 --------------------# Creating
# train and test partitions Data is random so it can be split sequentially
train <- 1:round(nrow(sms.dtm) * 0.75)
```

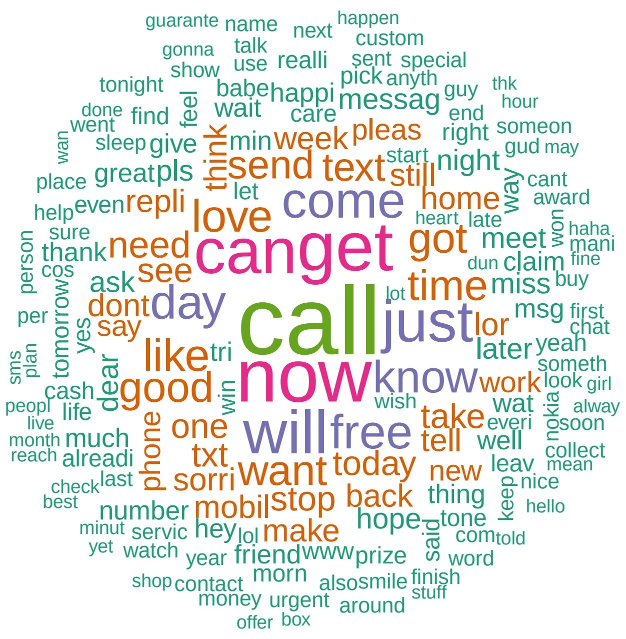
```r
test <- 1:round(nrow(sms.dtm) * 0.25)
sms.trn <- sms.dtm[train, ]
sms.tst <- sms.dtm[test, ]
# Create the class verification vectors
sms.vtrn <- sms[train, ]$type
sms.vtst <- sms[test, ]$type
# Check the distribution of classes
prop.table(table(sms.vtrn))
```

```
## sms.vtrn
##      spam       ham
## 0.1352842 0.8647158
```

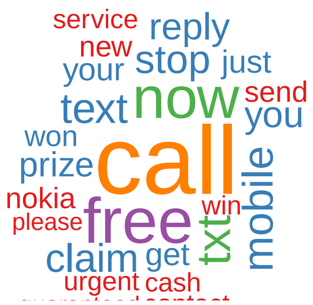
```r
prop.table(table(sms.vtst))
```

```
## sms.vtst
##      spam       ham
## 0.1345324 0.8654676
```

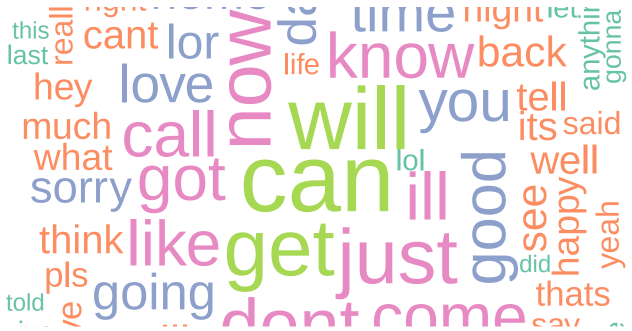
```r
# ------------------- Mon Feb 19 19:29:57 2018 --------------------# Word cloud
library(wordcloud)
wordcloud::wordcloud(words = sms.crpClean, scale = c(3, 0.4), min.freq = 50, random.order = F,
    random.color = F, colors = RColorBrewer::brewer.pal(n = 5, name = "Dark2"))
```



```r
# Subsetting types of SMs
spam <- subset(sms, type == "spam")
ham <- subset(sms, type == "ham")
# Wordclouds for each type
wordcloud(spam$text, scale = c(3, 0.4), min.freq = 50, random.order = F, random.color = F,
    colors = RColorBrewer::brewer.pal(n = 5, name = "Set1"))
```



```r
wordcloud(ham$text, scale = c(3, 0.4), min.freq = 50, random.order = F, random.color = F,
    colors = RColorBrewer::brewer.pal(n = 5, name = "Set2"))
```



```r
# -------------------- Mon Feb 19 20:05:08 2018 --------------------# Find
# frequent terms
sms.frq <- findFreqTerms(sms.trn, 5)
# View(sms.frq) It looks like the first 8 strings are strange characters.
sms.frq <- sms.frq[-c(1:8)]
str(sms.frq)
```

```
##  chr [1:1167] "abt" "acc" "accept" "access" "account" "across" "act" ...
```

```r
# That looks better -------------------- Mon Feb 19 20:20:41 2018
# --------------------# Subset the DTM's with the frequent terms
sms.frq.trn <- sms.trn[, sms.frq]
sms.frq.tst <- sms.tst[, sms.frq]
# convert counts to a factor
convert_counts <- function(x) {
    x <- ifelse(x > 0, "Yes", "No")
}
# apply() convert_counts() to columns of train/test data
sms.trn <- apply(sms.frq.trn, MARGIN = 2, convert_counts)
sms.tst <- apply(sms.frq.tst, MARGIN = 2, convert_counts)
# ------------------- Mon Feb 19 20:42:23 2018 --------------------# Model
# Training
library("e1071")
# Train the naiveBayes model
sms.cls <- naiveBayes(sms.trn, sms.vtrn)
# Make a prediction about the test set
sms.pred <- predict(sms.cls, newdata = sms.tst)
# Evaluate the model, confusionMatrix is preferred to Crosstable
library("caret")
confusionMatrix(sms.pred, sms.vtst)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction spam  ham
##       spam  171    5
##       ham    16 1198
##
##                Accuracy : 0.9849
##                  95% CI : (0.977, 0.9906)
##     No Information Rate : 0.8655
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9335
##  Mcnemar's Test P-Value : 0.0291
##
##             Sensitivity : 0.9144
##             Specificity : 0.9958
##          Pos Pred Value : 0.9716
##          Neg Pred Value : 0.9868
##              Prevalence : 0.1345
##          Detection Rate : 0.1230
##    Detection Prevalence : 0.1266
##       Balanced Accuracy : 0.9551
##
##        'Positive' Class : spam
##
```

```r
# The performance is slightly better than the run in the text, but there are 5
# messages that are legitimate that were filtered.  ------------------- Mon Feb
# 19 20:58:01 2018 --------------------# Attempt to improve model performance
# with laplace estimator and train
trn.ctrl <- trainControl(sms.vtrn, number = 3, repeats = 2, method = "repeatedcv",
    allowParallel = T)
```

Note: The chunk below is set to eval=F because knitting continues to fail after implementing multiple recommended fixes. The chunk runs fine in R.

```r
# -------------------- Tue Feb 20 21:38:15 2018 --------------------# After
# troubleshooting a cryptic error: Warning: predictions failed for Fold3.Rep1:
# fL=1, usekernel=TRUE, adjust=1 Error in log(sapply(1:nattribs, tempfoo)) :
# non-numeric argument to mathematical function Warning in In
# nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, There were
# missing values in resampled performance measures.  for nearly an hour. I found
# this post: https://github.com/topepo/caret/issues/793 and implemented the code
# below but the knitting still fails.

# cnames <- gsub('[^a-zA-Z0-9]','',colnames(sms.trn),perl=T)
# any(str_detect(cnames,'â|€')) colnames(sms.trn) <- cnames
options(warn = 1)

library(parallel)
library(doParallel)
cluster <- makeCluster(detectCores() - 1)  # Use 7/8 cores
registerDoParallel(cluster)
system.time(nb.model <- train(sms.trn, sms.vtrn, method = "nb", trControl = trn.ctrl,
    tuneGrid = expand.grid(fL = 1, usekernel = T, adjust = c(0.5, 1, 2, 3))))
nb.pred <- predict(nb.model, newdata = sms.tst)
confusionMatrix(nb.pred, sms.vtst)
stopCluster(cluster)
registerDoSEQ()
# A marginal improvement with the Laplace estimator and repeatedcv, 1 less
# misclassified ham message.
```

2

Install the requisite packages to execute the following code that classifies the built-in iris data using Naive Bayes. Build an R Notebook and explain in detail what each step does. Be sure to look up each function to understand how it is used.

```r
# -------------------- Mon Feb 19 21:22:02 2018 --------------------# Load the
# klaR library
library(klaR)
# Load the iris dataset
data(iris)
# Row count
nrow(iris)
```

```
## [1] 150
```

```r
# Summary of data
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width   
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100  
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300  
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300  
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199  
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

```
# Top 6 rows
head(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

```
# identify indexes to be in testing dataset every index of 5th, 10th, 15th ..
# will be the testing dataset the rest are training dataset 80% Training data,
# 20% Test Data, using a striated sample. %% returns the modulus of the division,
# if there's no modulus, it attributes a T, otherwise F. which provides the
# number index of the values marked as T.
testidx <- which(1:length(iris[, 1])%%5 == 0)

# separate into training and testing datasets Subsets all rows from iris that
# arent in the test index to make a training set
iristrain <- iris[-testidx, ]
# Subsets iris using the test index rows to make a test set
iristest <- iris[testidx, ]

# apply Naive Bayes, formula specifies species as the classification, and all
# other rows as the data to train on
nbmodel <- NaiveBayes(Species ~ ., data = iristrain)

# Create a prediction using all the data in the test set except the Species
# column
prediction <- predict(nbmodel, iristest[, -5])
# Check the accuracy with table, shows only 2 misclassifications
table(prediction$class, iristest[, 5])
```

```
##
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         2
##   virginica       0          0         8
```

2a

How would you make a prediction for a new case with the above package?

Build the model, make a data frame with values in each of the predictor variables columns and set the newdata argument for the predict function to the data frame.

2b

b. How does this package deal with numeric features?

From inspecting the nbmodel and the resources that I have read, I am fairly certain that the KlaR implementation uses a Gaussian (normal) distribution density curve per class for values of each variable and determines the probability based on the number of values falling within the area under the curve (the normal probability density function).

2c

c. How does it specify a Laplace estimator?

The fL (factor Laplace) variable can be specified with a factor indicating the Laplace correction value using the following syntax fL = n where n is a positive integer.

3

What are Laplace estimators and why are they used in Naive Bayes classification? Provide an example of how they might be used and when. (You do not need to write any code. Instead explain their use in the R Notebook.)

A Laplace estimator will attribute a value equal to the specified Laplace factor value to each variable/class combination such that probabilities amounting to 0 or 1 can be avoided. Thus the Bayes probability equation for a given class/variable combination when considering the probability of the unknown having the class would look like the following:

$$\frac{No.of Vars where Class = T + L}{Total No.of Obs for Var + L * V_i}$$
$$\text{Where } V_i = \text{ The No. of classes concerned with the variable in question}$$

This formula will be used to recalculate all probabilities. The Laplace correction factor is used to avoid 0 probabilities, divide by 0 situations, and probabilities of 1 and generally creates a more accurate model when there are class/variable combinations in the dataset with a probablity of 0.