NAIVE BAYES to predict SPAM versus HAM

Bogdan Tanasa

THE SECTIONS IN THE RMARKDOWN DOCUMENT :

8. PERFORMING THE CONVERSIONS

9. TRAINING AND MAKING THE PREDICTIONS

10. TRAINING AND MAKING THE PREDICTIONS AFTER ADDING LAGRANGE ESTIMATOR

1. INTRODUCTION

We are using the data from **UCI** : !(https://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection)

We are reading a file about **SHAKIRA**, and we aim to predict whether these messages are **SPAM/HAM** by using **a NAIVE BAYES algorithm**;

2. READING THE DATA

```r
library(klaR)
library(MASS)
library(caret)

library(tm)
library(wordcloud)
library(e1071)
library(gmodels)
library(pander)

library(dplyr)
library(doMC)
registerDoMC(cores=4)
```

```r
######################################################################

sms_raw <- read.delim("Youtube05-Shakira_03oct2020.csv", header=TRUE, sep=",", stringsAsFactors=FALSE)
head(sms_raw)
```

```
##                                COMMENT_ID                        AUTHOR
## 1   z13lgffb5w3ddx1ul22qy1wxspy5cpkz504                     dharma pal
## 2     z123dbgb0mqjfxbtz22ucjc5jvzcv3ykj                  Tiza Arellano
## 3 z12quxxp2vutflkxv04cihggzt2azl34pms0k Prìñçeśś Âliś Łøvê Dømíñø Mâđiś
## 4     z133stly3kete3tly22petvwdpmghrlli                  Analena López
## 5     z12myn4rltf4ejddv23mwr3piuapcbl0r           jehoiada wellington
## 6     z135vzqy1yrjhluew23kibopnrmqsplux               Kara Cuthbertson
##                         DATE                          CONTENT CLASS
## 1 2015-05-29T02:30:18.971000                        Nice song     0
## 2 2015-05-29T00:14:48.748000                      I love song     0
## 3 2015-05-28T21:00:08.607000                      I love song     0
## 4 2015-05-28T17:08:29.827000     shakira is best for worldcup     0
## 5 2015-05-28T17:06:37.288000 The best world cup song ever!!!!     0
## 6 2015-05-28T15:46:42.482000                           I love     0
```

```
# to make the columns TYPE and TEXT, as it is easier to work with the RELEVANT DATA

sms_raw2 = subset(sms_raw, select=c("CONTENT", "CLASS"))
sms_raw2$type = ifelse(sms_raw2$CLASS > 0, "spam", "ham")
sms_raw2$text = sms_raw2$CONTENT

sms_raw3 = subset(sms_raw2, select=c("type", "text"))
sms_raw3$type <- factor(sms_raw3$type)
head(sms_raw3)
```

```
##   type                             text
## 1  ham                        Nice song
## 2  ham                      I love song
## 3  ham                      I love song
## 4  ham     shakira is best for worldcup
## 5  ham The best world cup song ever!!!!
## 6  ham                           I love
```

```
# write.table(sms_raw3, file="file.sms_raw3.for.verifications.txt", sep="\t", quote=F)

# for simplicity, to use again as a variable the name SMS_RAW
rm(sms_raw)

sms_raw = sms_raw3
head(sms_raw)
```

```
##   type                             text
## 1  ham                        Nice song
## 2  ham                      I love song
## 3  ham                      I love song
## 4  ham     shakira is best for worldcup
## 5  ham The best world cup song ever!!!!
## 6  ham                           I love
```

```
dim(sms_raw)
```

```
## [1] 367   2
```

3. DATA RANDOMIZATION

```
#########################################################################
# here we  randomize the lines of input file:

set.seed(12358)
sms_raw <- sms_raw[sample(nrow(sms_raw)),]
str(sms_raw)
```

```
## 'data.frame':    367 obs. of  2 variables:
##  $ type: Factor w/ 2 levels "ham","spam": 1 2 1 2 2 2 1 1 1 2 ...
##  $ text: chr  "waka waka" "1 753 682 421 GANGNAM STYLE ^^" "this song always gives me chills! :)" "C
```

```
dim(sms_raw)
```

```
## [1] 367   2
```

4. DATA TRANSFORMATION

```
#########################################################################
# we transform the text into a corpus that can later be used in the analysis,
# then we will convert all text to lowercase,
# remove numbers, remove some common stop words in english,
# remove punctuation and extra whitespace, and finally,
# we generate the document term that will be the basis for the classification task.

sms_corpus <- Corpus(VectorSource(sms_raw$text))

sms_corpus_clean <- sms_corpus %>%
    tm_map(content_transformer(tolower)) %>%
    tm_map(removeNumbers) %>%
    tm_map(removeWords, stopwords(kind="en")) %>%
    tm_map(removePunctuation) %>%
    tm_map(stripWhitespace) %>% tm_map(stemDocument)
```

```
## Warning in tm_map.SimpleCorpus(., content_transformer(tolower)): transformation
## drops documents

## Warning in tm_map.SimpleCorpus(., removeNumbers): transformation drops documents

## Warning in tm_map.SimpleCorpus(., removeWords, stopwords(kind = "en")):
## transformation drops documents

## Warning in tm_map.SimpleCorpus(., removePunctuation): transformation drops
## documents

## Warning in tm_map.SimpleCorpus(., stripWhitespace): transformation drops
## documents

## Warning in tm_map.SimpleCorpus(., stemDocument): transformation drops documents
```

```
sms_dtm <- DocumentTermMatrix(sms_corpus_clean)
```

5. TRAINING AND TEST SETS

```
##########################################################################################

dim(sms_dtm)[1]
```

```
## [1] 367
```

```
length(sms_corpus_clean)
```

```
## [1] 367
```

```
LENGTH_TRAIN = dim(sms_dtm)[1]  * 0.7
LENGTH_DATA = dim(sms_dtm)[1]

sms_dtm_train <- sms_dtm[1:LENGTH_TRAIN, ]
sms_dtm_test <- sms_dtm[(LENGTH_TRAIN+1):LENGTH_DATA, ]

sms_train_labels <- sms_raw[1:LENGTH_TRAIN, ]$type
sms_test_labels <- sms_raw[(LENGTH_TRAIN+1):LENGTH_DATA, ]$type

sms_train_labels <- sms_raw[1:LENGTH_TRAIN, ]$type
sms_test_labels <- sms_raw[(LENGTH_TRAIN+1):LENGTH_DATA, ]$type

head(sms_train_labels)
```

```
## [1] ham  spam ham  spam spam spam
## Levels: ham spam
```

```
head(sms_test_labels)
```

```
## [1] ham  spam ham  ham  spam spam
## Levels: ham spam
```

```
length(sms_train_labels )
```

```
## [1] 256
```

```
length(sms_test_labels )
```

```
## [1] 110
```

```
# in order to BALANCE the DATA
# to compare the proportion of SPAM and HAM in the training and test data frames:

prop.table(table(sms_train_labels))
```

```
## sms_train_labels
##      ham     spam
## 0.546875 0.453125
```

```
prop.table(table(sms_test_labels))
```

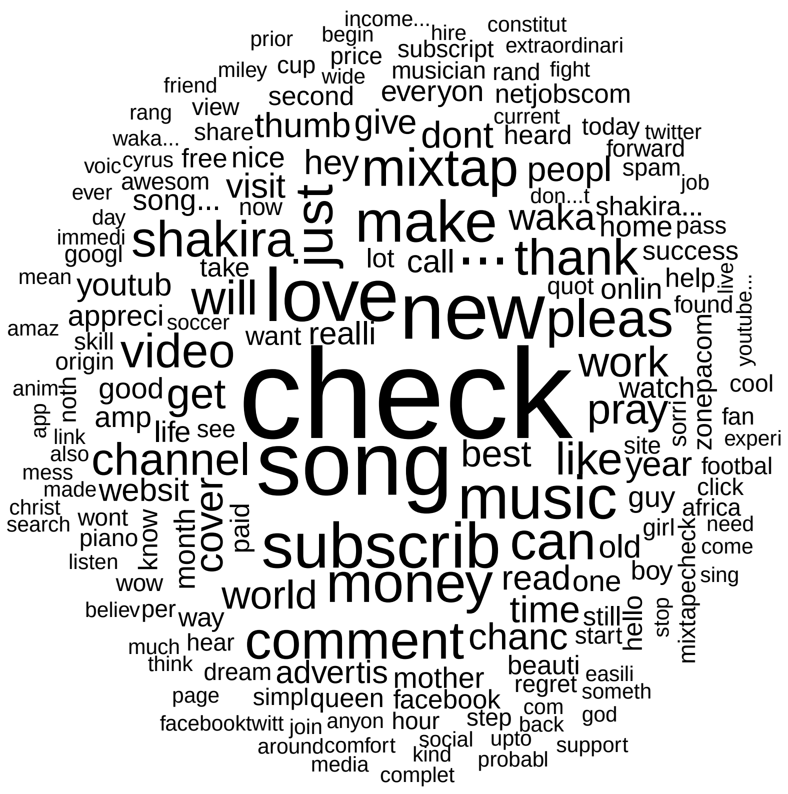
```
## sms_test_labels
##       ham      spam
## 0.4727273 0.5272727
```

6. TO VISUALIZE the WORD CLOUDS

```
##############################################################
# to represent the data as WORDCLOUDS :
# library(wordcloud)
```

```r
# png("word.cloud.all.png")
wordcloud(sms_corpus_clean, min.freq = 5, random.order = FALSE) ### changing MIN FREQ
```

```
## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '' in
## 'mbcsToSbcs': dot substituted for <ef>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '' in
## 'mbcsToSbcs': dot substituted for <bb>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '' in
## 'mbcsToSbcs': dot substituted for <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '' in 'mbcsToSbcs': dot substituted for <ef>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '' in 'mbcsToSbcs': dot substituted for <bb>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '' in 'mbcsToSbcs': dot substituted for <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+feff

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'song'
## in 'mbcsToSbcs': dot substituted for <ef>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'song'
## in 'mbcsToSbcs': dot substituted for <bb>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'song'
## in 'mbcsToSbcs': dot substituted for <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'song' in 'mbcsToSbcs': dot substituted for
## <ef>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'song' in 'mbcsToSbcs': dot substituted for
## <bb>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'song' in 'mbcsToSbcs': dot substituted for
## <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+feff

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'shakira' in 'mbcsToSbcs': dot substituted for <ef>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'shakira' in 'mbcsToSbcs': dot substituted for <bb>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'shakira' in 'mbcsToSbcs': dot substituted for <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'shakira' in 'mbcsToSbcs': dot substituted for
## <ef>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'shakira' in 'mbcsToSbcs': dot substituted for
## <bb>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'shakira' in 'mbcsToSbcs': dot substituted for
## <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+feff

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'youtube' in 'mbcsToSbcs': dot substituted for <ef>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'youtube' in 'mbcsToSbcs': dot substituted for <bb>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'youtube' in 'mbcsToSbcs': dot substituted for <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'youtube' in 'mbcsToSbcs': dot substituted for
## <ef>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'youtube' in 'mbcsToSbcs': dot substituted for
## <bb>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'youtube' in 'mbcsToSbcs': dot substituted for
## <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+feff

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'waka'
## in 'mbcsToSbcs': dot substituted for <ef>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'waka'
## in 'mbcsToSbcs': dot substituted for <bb>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'waka'
## in 'mbcsToSbcs': dot substituted for <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'waka' in 'mbcsToSbcs': dot substituted for
## <ef>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'waka' in 'mbcsToSbcs': dot substituted for
## <bb>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'waka' in 'mbcsToSbcs': dot substituted for
## <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+feff

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'income…' in 'mbcsToSbcs': dot substituted for <e2>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'income…' in 'mbcsToSbcs': dot substituted for <80>
## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'income…' in 'mbcsToSbcs': dot substituted for <a6>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'income…' in 'mbcsToSbcs': dot substituted for
## <e2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'income…' in 'mbcsToSbcs': dot substituted for
## <80>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'income…' in 'mbcsToSbcs': dot substituted for
## <a6>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+2026

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'don’t'
## in 'mbcsToSbcs': dot substituted for <e2>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'don’t'
## in 'mbcsToSbcs': dot substituted for <80>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'don’t'
## in 'mbcsToSbcs': dot substituted for <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'don’t' in 'mbcsToSbcs': dot substituted for
## <e2>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'don’t' in 'mbcsToSbcs': dot substituted for
## <80>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'don’t' in 'mbcsToSbcs': dot substituted for
## <99>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+2019
```



```
# dev.off()

spam <- subset(sms_raw, type == "spam")
ham <- subset(sms_raw, type == "ham")

dim(spam)
```

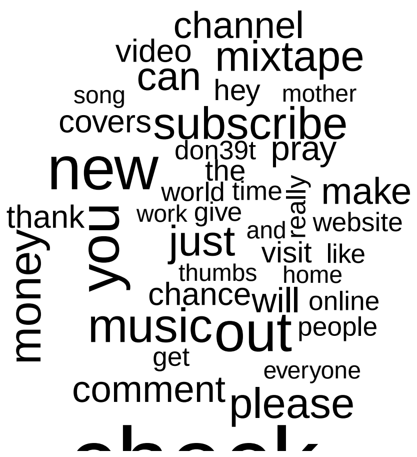
```
## [1] 174   2
```

```
dim(ham)
```

```
## [1] 193   2
```

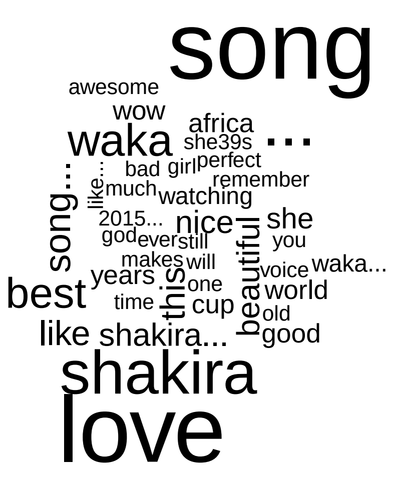
```
###########################################################
# to represent the SPAM data as WORDCLOUDS :
# png("word.cloud.spam.png")
wordcloud(spam$text, max.words = 40, scale = c(3, 0.5))
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents

## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents
```



```r
# dev.off()

################################################################
# to represent the HAM data as WORDCLOUDS :
# png("word.cloud.ham.png")
wordcloud(ham$text, max.words = 40, scale = c(3, 0.5))
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents

## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '2015'
## in 'mbcsToSbcs': dot substituted for <ef>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '2015'
## in 'mbcsToSbcs': dot substituted for <bb>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '2015'
## in 'mbcsToSbcs': dot substituted for <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '2015' in 'mbcsToSbcs': dot substituted for
## <ef>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '2015' in 'mbcsToSbcs': dot substituted for
## <bb>
## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '2015' in 'mbcsToSbcs': dot substituted for
## <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+feff

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'shakira' in 'mbcsToSbcs': dot substituted for <ef>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'shakira' in 'mbcsToSbcs': dot substituted for <bb>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on
## 'shakira' in 'mbcsToSbcs': dot substituted for <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'shakira' in 'mbcsToSbcs': dot substituted for
## <ef>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'shakira' in 'mbcsToSbcs': dot substituted for
## <bb>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'shakira' in 'mbcsToSbcs': dot substituted for
## <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+feff

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'song'
## in 'mbcsToSbcs': dot substituted for <ef>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'song'
## in 'mbcsToSbcs': dot substituted for <bb>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'song'
## in 'mbcsToSbcs': dot substituted for <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'song' in 'mbcsToSbcs': dot substituted for
## <ef>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'song' in 'mbcsToSbcs': dot substituted for
## <bb>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'song' in 'mbcsToSbcs': dot substituted for
## <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+feff

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'like'
## in 'mbcsToSbcs': dot substituted for <ef>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'like'
## in 'mbcsToSbcs': dot substituted for <bb>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'like'
## in 'mbcsToSbcs': dot substituted for <bf>
## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'like' in 'mbcsToSbcs': dot substituted for
## <ef>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'like' in 'mbcsToSbcs': dot substituted for
## <bb>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'like' in 'mbcsToSbcs': dot substituted for
## <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+feff

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '' in
## 'mbcsToSbcs': dot substituted for <ef>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '' in
## 'mbcsToSbcs': dot substituted for <bb>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on '' in
## 'mbcsToSbcs': dot substituted for <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '' in 'mbcsToSbcs': dot substituted for <ef>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '' in 'mbcsToSbcs': dot substituted for <bb>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on '' in 'mbcsToSbcs': dot substituted for <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+feff

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'waka'
## in 'mbcsToSbcs': dot substituted for <ef>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'waka'
## in 'mbcsToSbcs': dot substituted for <bb>

## Warning in strwidth(words[i], cex = size[i], ...): conversion failure on 'waka'
## in 'mbcsToSbcs': dot substituted for <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'waka' in 'mbcsToSbcs': dot substituted for
## <ef>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'waka' in 'mbcsToSbcs': dot substituted for
## <bb>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : conversion failure on 'waka' in 'mbcsToSbcs': dot substituted for
## <bf>

## Warning in text.default(x1, y1, words[i], cex = size[i], offset = 0, srt =
## rotWord * : font metrics unknown for Unicode character U+feff
```



```
# dev.off()
```

7. DATA FILTERING

```
# to find the FREQUENT WORDS with frequency > 2 :
# findFreqTerms(sms_dtm_train, 2)

sms_freq_words <- findFreqTerms(sms_dtm_train, 2)  ### we can change to 5

# as we desire all the rows, but only the columns representing the words
# in the sms_freq_words vector, we use the commands :

sms_dtm_freq_train <- sms_dtm_train[ , sms_freq_words]
sms_dtm_freq_test  <- sms_dtm_test[ , sms_freq_words]
```

8. PERFORMING THE CONVERSIONS

```
# to define a new FUNCTION : to convert the counts into Yes, No :

convert_counts <- function(x) { x <- ifelse(x > 0, "Yes", "No") }
```

```r
sms_train <- apply(sms_dtm_freq_train, MARGIN = 2, convert_counts)
sms_test <- apply(sms_dtm_freq_test, MARGIN = 2, convert_counts)
```

9. TRAINING AND MAKING THE PREDICTIONS

```r
# library(e1071)

sms_classifier <- naiveBayes(sms_train, sms_train_labels)

sms_test_pred <- predict(sms_classifier, sms_test)

# showing the CROSS TABLE :
# library(gmodels)

CrossTable(sms_test_pred, sms_test_labels,
           prop.chisq = FALSE, prop.t = FALSE,
           dnn = c("predicted", "actual"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  110
##
##
##              | actual
##    predicted |       ham |      spam | Row Total |
## -------------|-----------|-----------|-----------|
##          ham |        51 |        16 |        67 |
##              |     0.761 |     0.239 |     0.609 |
##              |     0.981 |     0.276 |           |
## -------------|-----------|-----------|-----------|
##         spam |         1 |        42 |        43 |
##              |     0.023 |     0.977 |     0.391 |
##              |     0.019 |     0.724 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        52 |        58 |       110 |
##              |     0.473 |     0.527 |           |
## -------------|-----------|-----------|-----------|
##
##
```

Here the **ACCURACY** is (51 +42)/(51+42+16+1) = **0.8454545**

10. TRAINING AND MAKING THE PREDICTIONS AFTER ADDING LAGRANGE ESTIMATOR

```
##############################################################

sms_classifier2 <- naiveBayes(sms_train, sms_train_labels, laplace = 1)

sms_test_pred2 <- predict(sms_classifier2, sms_test)

CrossTable(sms_test_pred2, sms_test_labels,
           prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
           dnn = c("predicted", "actual"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  110
##
##
##              | actual
##    predicted |       ham |      spam | Row Total |
## -------------|-----------|-----------|-----------|
##          ham |        51 |        22 |        73 |
##              |     0.981 |     0.379 |           |
## -------------|-----------|-----------|-----------|
##         spam |         1 |        36 |        37 |
##              |     0.019 |     0.621 |           |
## -------------|-----------|-----------|-----------|
## Column Total |        52 |        58 |       110 |
##              |     0.473 |     0.527 |           |
## -------------|-----------|-----------|-----------|
##
##
```

Here the **ACCURACY** is (51 +36)/(51+42+22+1) = **0.75**

ADDITIONAL and OTHER COMMENTS

```
##############################################################
### i have been trying also to use CARET on the dataset,
### although repetitively, I am getting the error below :

### when performing the training :
# Something is wrong; all the Accuracy metric values are missing:
#    Accuracy       Kappa
# Min.   : NA   Min.   : NA
# 1st Qu.: NA   1st Qu.: NA
# Median : NA   Median : NA
# Mean   :NaN   Mean   :NaN
# 3rd Qu.: NA   3rd Qu.: NA
# Max.   : NA   Max.   : NA
# NA's   :2     NA's   :2
# Error: Stopping
#################################################################
```

As a conclusion, by using a Naive Bayes approach to predict HAM versus SPAM in Shakira's messages, we have obtained a good ACCURACY of 0.84 (although adding the Lagrange estimator decreases the ACCURACY to 0.75).