Clasificación usando Naive Bayes

Code

1. Introducción

- Vimos que el algoritmo Naive Bayes clasifica de manera probabilística
- La probabilidad de que un objeto pertenezca a cierta clase depende de las características que tenga
- Por medio del cálculo de probabilidades posteriores en función de los datos de entrenamiento que se tengan
- Para filtrar correos spam, por ejemplo, las palabras contenidas en el correo determinan el nivel al que pertenece
- Esto en función de la frecuencia de dichas palabras en correos spam previamente clasificados
- Aplicaremos esta técnica filtrando mensajes de texto (SMS) spam

2. Datos

- Usaremos una base de datos adaptada del SMS Spam Collection
- La base contiene los mensajes, además de un indicador de si es spam o no
- Para poder implementar el algoritmo debemos procesar los datos
- Debemos transformar los textos en una bolsa de palabras, que ignora el orden de las palabras
- Simplemente indica si la palabra aparece o no
- Como siempre, arranquemos estableciendo el directorio de trabajo
- Leamos la base llamada `sms_spam_.csv`

Hide

Hide

```
sms_raw <- read.csv("sms_spam.csv", stringsAsFactors=FALSE)
```

- Echemos un vistazo a la base:

Hide

Hide

```
str(sms_raw)
```

```
'data.frame':   5574 obs. of  2 variables:
 $ type: chr  "ham" "ham" "spam" "ham" ...
 $ text: chr  "Go until jurong point, crazy.. Available only in bugis n great world l
a e buffet... Cine there got amore wat..." "Ok lar... Joking wif u oni..." "Free entr
y in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receiv
e entry question(std txt rate)T&C"| __truncated__ "U dun say so early hor... U c alre
ady then say..." ...
```

Hide

Hide

```
head(sms_raw)
```

	type <chr>
1	ham
2	ham
3	spam
4	ham
5	ham
6	spam

6 rows | 1-2 of 2 columns

- 5574 SMS. La primera variable indica si es spam o no; la segunda contiene el texto
- La variable `type` aparece como string, pero de hecho es un factor. La podemos transformar

Hide

Hide

```
sms_raw$type <- factor(sms_raw$type)
```

- Inspeccionemos la variable:

Hide

Hide

```
table(sms_raw$type)
```

```
 ham spam
4827  747
```

- 4827 mensajes deseados frente a 747 indeseados

3. Limpieza y estandarización de los datos

- Los mensajes se componen de palabras, espacios, números y puntuación
- Debemos remover lo que no nos interesa, como números y puntuación
- También aquellas palabras que no nos dicen mucho, como *and*, *but*, *or*
- Usaremos el paquete `tm` para este propósito

Hide

Hide

```
install.packages("tm")
```

```
Error in install.packages : Updating loaded packages
```

Hide

Hide

```
library(tm)
```

- El primer paso es crear un **corpus**: colección de documentos
- En este caso, el corpus es una colección de mensajes de texto

- Para definir el corpus usamos la función `VCorpus()`

Hide

Hide

```
sms_corpus <- VCorpus(VectorSource(sms_raw$text))
```

- Usamos la función `VectorSource()` porque los textos ya están cargados en R en la base
- Pero podríamos usar otras fuentes de textos, como PDFs, Words, web, etc.
- Podemos echar un vistazo a nuestro corpus:

Hide

Hide

```
print(sms_corpus)
```

```
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 5574
```

- Tenemos 5574 documentos (SMS)
- En esencia nuestro corpus es una lista compleja
- Con la función `inspect()` podemos resumir algunos elementos del corpus

Hide

Hide

```
inspect(sms_corpus[1:2])
```

```
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 2

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 111

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 29
```

- Podemos ver el mensaje con la función `as.character()`

Hide

Hide

```
as.character(sms_corpus[[1]])
```

```
[1] "Go until jurong point, crazy.. Available only in bugis n great world la e buffet
... Cine there got amore wat..."
```

- Para ver múltiples mensajes debemos usar la función `lapply()`

Hide

Hide

```
lapply(sms_corpus[1:3], as.character)
```

```
$`1`
[1] "Go until jurong point, crazy.. Available only in bugis n great world la e buffet
... Cine there got amore wat..."

$`2`
[1] "Ok lar... Joking wif u oni..."

$`3`
[1] "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 8
7121 to receive entry question(std txt rate)T&C's apply 08452810075over18's"
```

- Limpiemos los datos para poder analizarlos
- Primero, expresemos todas las palabras en minúsculas, porque *HELLO*, *Hello* o *hello* son la misma palabra
- Con la función `tm_map()` podemos aplicar una serie de transformaciones al corpus
- Empecemos conviertiendo todas las letras en minúsculas
- Esto con la función `tolower()`

Hide

Hide

```
sms_corpus_clean <- tm_map(sms_corpus, content_transformer(tolower))
```

- Podemos comparar los textos

Hide

Hide

```
as.character(sms_corpus[[3]])
```

```
[1] "Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 8
7121 to receive entry question(std txt rate)T&C's apply 08452810075over18's"
```

Hide

Hide

```
as.character(sms_corpus_clean[[3]])
```

```
[1] "free entry in 2 a wkly comp to win fa cup final tkts 21st may 2005. text fa to 8
7121 to receive entry question(std txt rate)t&c's apply 08452810075over18's"
```

- Ahora removamos los números de los mensajes

Hide

Hide

```
sms_corpus_clean <- tm_map(sms_corpus_clean, removeNumbers)
```

- Revisemos:

Hide

Hide

```
as.character(sms_corpus_clean[[3]])
```

```
[1] "free entry in  a wkly comp to win fa cup final tkts st may . text fa to  to rece
ive entry question(std txt rate)t&c's apply over's"
```

- Removamos ahora las *stopwords*. Son aquellas palabras que aparecen con mucha frecuencia pero proveen poca información para clasificar

Hide

Hide

```
sms_corpus_clean <- tm_map(sms_corpus_clean, removeWords, stopwords())
install.packages("tm")
```

```
probando la URL 'https://cran.rstudio.com/bin/macosx/mavericks/contrib/3.3/tm_0.7-1.t
gz'
Content type 'application/x-gzip' length 876346 bytes (855 KB)
==================================================
downloaded 855 KB

tar: Failed to set default locale
```

```


The downloaded binary packages are in
    /var/folders/cw/_9zpljws58jbbv49mcb86l6c0000gn/T//RtmprCm2Nz/downloaded_packages
```

- Hay unos caracteres raros que debemos remover:

Hide

Hide

Hide

```
sms_corpus_clean <- tm_map(sms_corpus_clean, removeWords, c("<c2><a3>", "<c2><a3>wk",
"<c3><9c>", "<c3><bc>", "<e2><80><93>", "<e2><80><a6>"))
sms_corpus_clean <- tm_map(sms_corpus_clean, removeWords, c("<c2><a3>", "<c2><a3>wk",
"<c3><9c>", "<c3><bc>", "<e2><80><93>", "<e2><80><a6>"))
```

Hide

Hide

```
as.character(sms_corpus_clean[[3]])
```

```
[1] "free entry    wkly comp  win fa cup final tkts st may . text fa    receive entry
question(std txt rate)t&c's apply 's"
```

- Ahora eliminemos la puntuación:

Hide

Hide

```
sms_corpus_clean <- tm_map(sms_corpus_clean, removePunctuation)
```

- Verifiquemos:

Hide

Hide

```
as.character(sms_corpus_clean[[3]])
```

```
[1] "free entry    wkly comp  win fa cup final tkts st may  text fa    receive entry
questionstd txt ratetcs apply s"
```

- En el análisis de texto es común reducir las palabras a sus raíces
- En inglés esto se llama *stemming*. Esto se hace para que el algoritmo de machine learning trate a palabras relacionadas como un mismo concepto
- La idea es eliminar los sufijos de las palabras y dejar solo la raíz

- Para hacer este proceso necesitamos el paquete `SnowballC`

Hide

Hide

```
install.packages("SnowballC")
```

```
Error in install.packages : Updating loaded packages
```

Hide

Hide

```
library(SnowballC)
```

- Dentro del paquete `SnowballC` usamos la función `wordStem()`
- Para un vector de caracteres, esta función nos devuelve el mismo vector en su forma de raíz
- Por ejemplo:

Hide

Hide

```
wordStem(c("learn", "learned", "learning", "learns"))
```

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

- Para aplicar esto a todo el corpus, debemos usar la función `stemDocument()`

Hide

Hide

```
sms_corpus_clean <- tm_map(sms_corpus_clean, stemDocument)
```

- Y verifiquemos:

Hide

Hide

```
as.character(sms_corpus_clean[[3]])
```

```
[1] "free entri wkli comp win fa cup final tkts st may text fa receiv entri questions
td txt ratetc appli s"
```

- Por último, removamos los espacios en blanco que quedaron luego de esta limpieza
- Esto lo hacemos con la función `stripWhitespace()`

Hide

Hide

```
sms_corpus_clean <- tm_map(sms_corpus_clean, stripWhitespace)
```

- Verifiquemos:

Hide

Hide

```
as.character(sms_corpus_clean[[3]])
```

```
[1] "free entri wkli comp win fa cup final tkts st may text fa receiv entri questions
td txt ratetc appli s"
```

- Nótese que solo remueve los nuevos espacios en blanco, pero deja los originales entre palabras
- Comparemos el mensaje 1 original con el que quedó tras la limpieza

Hide

Hide

```
as.character(sms_corpus[[1]])
```

```
[1] "Go until jurong point, crazy.. Available only in bugis n great world la e buffet
... Cine there got amore wat..."
```

Hide

Hide

```
as.character(sms_corpus_clean[[1]])
```

```
[1] "go jurong point crazi avail bugi n great world la e buffet cine got amor wat"
```

- Tras limpiar nos acercamos más a la bolsa de palabras de la que hablamos antes y con la que analizamos este tipo de datos

4. Tokenization

- Hemos avanzado bastante en el tipo de limpieza que debemos hacer a los datos
- Pero falta un último paso. El proceso conocido como **tokenization**
- Un **token** es un elemento único de **string** de texto. En el caso de estos mensajes, una palabra
- Nos interesa saber qué palabras aparecen o no en cada documento (mensaje)
- Para esto, usamos una función de `tm` que nos permite hacer **tokenization** de nuestro corpus
- Dicha función es `DocumentTermMatrix()`
- Crea una matriz en la que las filas son los documentos (cada SMS) y las columnas son las palabras
- Así, cada celda indica si la palabra asociada a la columna aparece en el mensaje asociado a la fila
- Como lo muestra la siguiente tabla:

Ejemplo Document Term Matrix

message #	balloon	balls	bam	bambling	band
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

Fuente: Lantz (2015)

- Naturalmente, esta será una matriz con muchos 0s.

- Creemos esta matriz:

Hide

Hide

```
sms_dtm <- DocumentTermMatrix(sms_corpus_clean)
install.packages("SnowballC")
```

```
probando la URL 'https://cran.rstudio.com/bin/macosx/mavericks/contrib/3.3/SnowballC_
0.5.1.tgz'
Content type 'application/x-gzip' length 3156091 bytes (3.0 MB)
==================================================
downloaded 3.0 MB

tar: Failed to set default locale
```

```

The downloaded binary packages are in
    /var/folders/cw/_9zpljws58jbbv49mcb86l6c0000gn/T//RtmprCm2Nz/downloaded_packages
```

- Nota: nosotros limpiamos manualmente los mensajes. Lo hubiésemos podidos hacer directamente con la función `DocumentTermMatrix()` fijando los parámetros del caso

5. Conjuntos de datos de entrenamiento y prueba

- Para poder implementar el algoritmo, necesitamos datos de entrenamiento y de prueba
- Usaremos el 75% de los datos para entrenar y el 25% para probar
- Los mensajes ya venían en orden aleatorio

Hide

Hide

```
sms_dtm_train <- sms_dtm[1:4181, ]
sms_dtm_test <- sms_dtm[4182:5574, ]
```

- También debemos crear un par de vectores con los outcomes de interés para estos mensaje (si son spam o no)

Hide

Hide

```
sms_train_labels <- sms_raw[1:4181, ]$type
sms_test_labels <- sms_raw[4182:5574, ]$type
```

- Podemos verificar que los conjuntos de prueba y entrenamiento se representan a sí mismos, al ser elegidos aleatoriamente

Hide

Hide

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

```
sms_train_labels
      ham      spam
0.8648649 0.1351351
```

Hide

Hide

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

```
sms_test_labels
      ham      spam
0.8693467 0.1306533
```

- Prácticamente las mismas proporciones. 13% de spam en ambas bases

6. Visualización de los datos: nubes de palabras

- Las nubes de palabras son una forma gráfica de presentar la frecuencia de palabras

- Pueden servir para encontrar patrones en los datos que representan a los textos
- R tiene el paquete `wordcloud` para hacer este tipo de nubes

Hide

Hide

```
install.packages("wordcloud")
```

```
Error in install.packages : Updating loaded packages
```

Hide

Hide

```
library(wordcloud)
```

- Creemos la nube de palabras para nuestro corpus limpio

Hide

Hide

```
wordcloud(sms_corpus_clean, min.freq=50, random.order=FALSE)
install.packages("wordcloud")
```

```
probando la URL 'https://cran.rstudio.com/bin/macosx/mavericks/contrib/3.3/wordcloud_
2.5.tgz'
Content type 'application/x-gzip' length 127386 bytes (124 KB)
==================================================
downloaded 124 KB

tar: Failed to set default locale
```

```


The downloaded binary packages are in
    /var/folders/cw/_9zpljws58jbbv49mcb86l6c0000gn/T//RtmprCm2Nz/downloaded_packages
```

- Puede ser interesante comparar las nubes de palabras para cada tipo de mensaje
- Antes no separamos los dos tipos de datos
- Lo podemos hacer con la función `subset()`

Hide

Hide

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

- Algo interesante de la función `wordcloud()` es que así no hayamos limpiados los datos, automáticamente aplica ciertos procesos

Hide

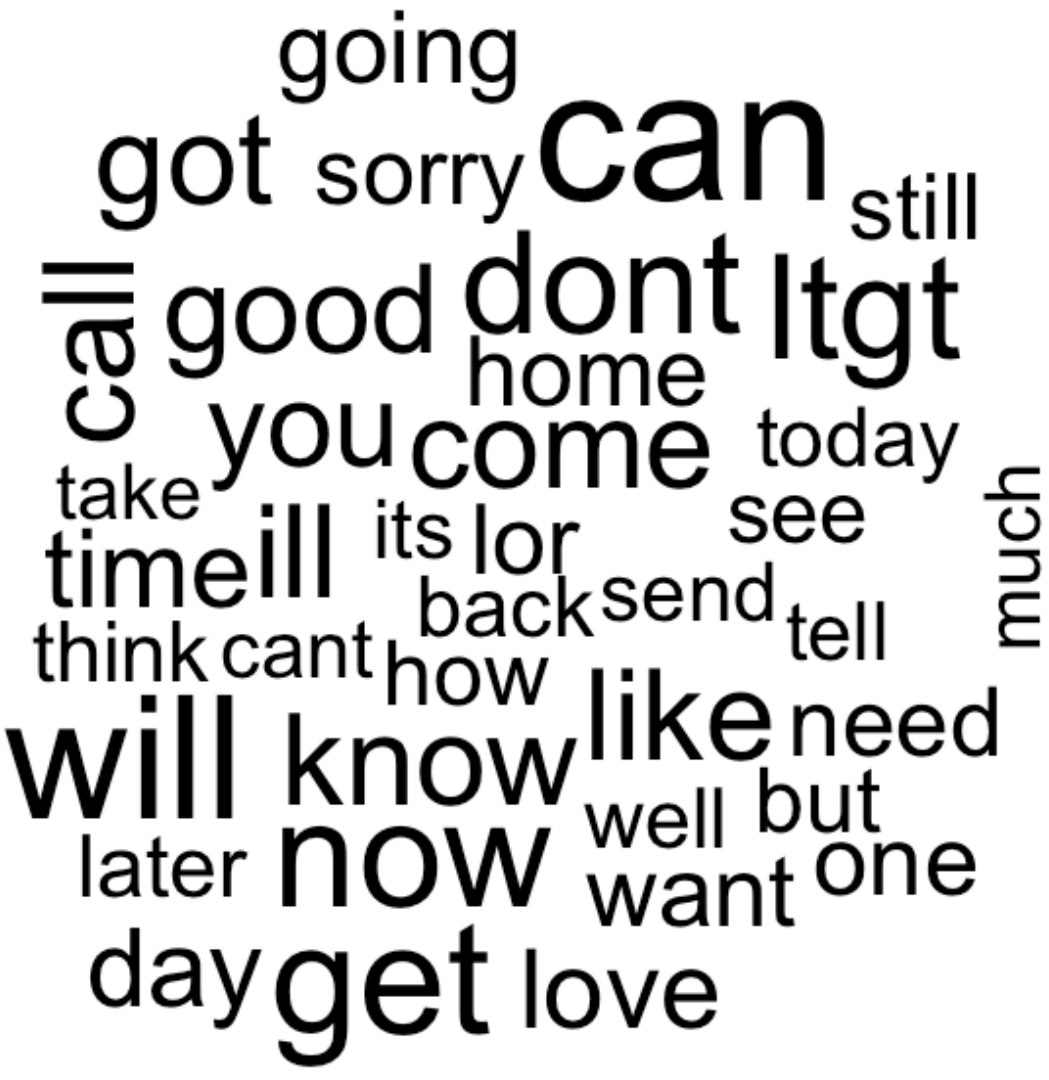
Hide

```
wordcloud(spam$text, max.words=40, scale=c(3, 0.5))
```

Hide

Hide

```
wordcloud(ham$text, max.words=40, scale=c(3, 0.5))
```



- Es claro que las palabras frecuentes en los spam son distintas a las de los ham
- Un modelo **Naive Bayes** encontrará un buen número de palabras para filtrar

7. Palabras frecuentes como características de los mensajes

- La matriz DTM contiene más de 6500 características
- Cada característica indica si una palabra está o no contenida en un mensaje
- Pero esto no es muy informativo pues contiene muchos 0s
- Debemos reducir el número de palabras
- Para quedarnos con aquellas que aparecen en al menos 5 mensajes
- Eliminamos las que aparecen en menos de 5 mensajes, es decir, en menos del 0.1% de los casos de los datos de entrenamiento
- Para esto usamos la función `findFreqTerms()` del paquete `tm`

- Toma una DTM y devuelve un vector con las palabras que aparecen al menos el número especificado de veces
- Por ejemplo:

Hide

Hide

```
findFreqTerms(sms_dtm_train, 5)
```

```
  [1] "<c2><a3>"         "<c2><a3>wk"
  [3] "<c3><9c>"         "<c3><bc>"
  [5] "<e2><80><93>"     "<e2><80><a6>"
  [7] "abiola"           "abl"
  [9] "abt"              "accept"
 [11] "access"           "account"
 [13] "across"           "activ"
 [15] "actual"           "add"
 [17] "address"          "admir"
 [19] "adult"            "advanc"
 [21] "aft"              "afternoon"
 [23] "aftr"             "age"
 [25] "ago"              "ahead"
 [27] "ahmad"            "aight"
 [29] "aint"             "air"
 [31] "aiyah"            "alex"
 [33] "almost"           "alon"
 [35] "alreadi"          "alright"
 [37] "alrit"            "also"
 [39] "alway"            "amp"
 [41] "angri"            "announc"
 [43] "anoth"            "answer"
 [45] "anybodi"          "anymor"
 [47] "anyon"            "anyth"
 [49] "anytim"           "anyway"
 [51] "apart"            "app"
 [53] "appli"            "appoint"
 [55] "appreci"          "april"
 [57] "ard"              "area"
 [59] "argument"         "arm"
 [61] "around"           "arrang"
 [63] "arrest"           "arriv"
 [65] "asap"             "ask"
 [67] "askd"             "asleep"
 [69] "ass"              "attempt"
 [71] "auction"          "avail"
 [73] "ave"              "avoid"
 [75] "await"            "award"
```

```
 [77] "away"            "awesom"
 [79] "babe"            "babi"
 [81] "back"            "bad"
 [83] "bag"             "bak"
 [85] "balanc"          "bank"
 [87] "bare"            "bath"
 [89] "batteri"         "bcoz"
 [91] "bcum"            "bday"
 [93] "beauti"          "becom"
 [95] "bed"             "bedroom"
 [97] "begin"           "believ"
 [99] "belli"           "best"
[101] "better"          "bid"
[103] "big"             "bill"
[105] "bird"            "birthday"
[107] "bit"             "black"
[109] "blank"           "bless"
[111] "blue"            "bluetooth"
[113] "bodi"            "bold"
[115] "bonus"           "boo"
[117] "book"            "bore"
[119] "boss"            "bother"
[121] "bout"            "bowl"
[123] "box"             "boy"
[125] "boytoy"          "brand"
[127] "break"           "breath"
[129] "brilliant"       "bring"
[131] "brother"         "bslvyl"
[133] "btnationalr"     "budget"
[135] "bugi"            "bus"
[137] "busi"            "buy"
[139] "buzz"            "cabin"
[141] "cafe"            "cal"
[143] "call"            "caller"
[145] "callertun"       "camcord"
[147] "came"            "camera"
[149] "can"             "cancel"
[151] "cant"            "car"
[153] "card"            "care"
[155] "carlo"           "case"
[157] "cash"            "cashbal"
[159] "catch"           "caus"
[161] "chanc"           "chang"
[163] "charact"         "charg"
[165] "chariti"         "chat"
[167] "cheap"           "check"
[169] "cheer"           "chennai"
[171] "chikku"          "childish"
[173] "children"        "chines"
[175] "choic"           "choos"
```

```
[177] "christma"        "cine"
[179] "cinema"          "claim"
[181] "class"           "clean"
[183] "clear"           "click"
[185] "clock"           "close"
[187] "club"            "code"
[189] "coffe"           "coin"
[191] "cold"            "colleagu"
[193] "collect"         "colleg"
[195] "colour"          "come"
[197] "comin"           "comp"
[199] "compani"         "competit"
[201] "complet"         "complimentari"
[203] "comput"          "concentr"
[205] "condit"          "confid"
[207] "confirm"         "congrat"
[209] "congratul"       "connect"
[211] "contact"         "content"
[213] "convey"          "cook"
[215] "cool"            "copi"
[217] "correct"         "cos"
[219] "cost"            "countri"
[221] "coupl"           "cours"
[223] "cover"           "coz"
[225] "crave"           "crazi"
[227] "credit"          "cri"
[229] "croydon"         "cuddl"
[231] "cum"             "cup"
[233] "current"         "custcar"
[235] "custom"          "cut"
[237] "cute"            "cuz"
[239] "dad"             "daddi"
[241] "damn"            "darl"
[243] "darlin"          "darren"
[245] "dat"             "date"
[247] "day"             "dead"
[249] "deal"            "dear"
[251] "decid"           "deep"
[253] "definit"         "del"
[255] "delet"           "deliv"
[257] "deliveri"        "den"
[259] "depend"          "detail"
[261] "dey"             "didnt"
[263] "die"             "differ"
[265] "difficult"       "digit"
[267] "din"             "dinner"
[269] "direct"          "dis"
[271] "discount"        "discuss"
[273] "disturb"         "dnt"
[275] "doctor"          "doesnt"
```

```
[277] "dog"               "doin"
[279] "dollar"            "don"
[281] "don<e2><80><98>t" "done"
[283] "dont"              "door"
[285] "doubl"             "download"
[287] "draw"              "dream"
[289] "drink"             "drive"
[291] "drop"              "drug"
[293] "dude"              "dun"
[295] "dunno"             "dvd"
[297] "earli"             "earlier"
[299] "easi"              "eat"
[301] "eatin"             "either"
[303] "els"               "email"
[305] "embarass"          "empti"
[307] "end"               "enemi"
[309] "energi"            "england"
[311] "enjoy"             "enough"
[313] "enter"             "entri"
[315] "envelop"           "especi"
[317] "etc"               "euro"
[319] "eve"               "even"
[321] "ever"              "everi"
[323] "everyon"           "everyth"
[325] "exact"             "exam"
[327] "excel"             "excit"
[329] "excus"             "expect"
[331] "experi"            "expir"
[333] "extra"             "eye"
[335] "face"              "facebook"
[337] "fact"              "fall"
[339] "famili"            "fanci"
[341] "fantasi"           "fantast"
[343] "far"               "fast"
[345] "fat"               "father"
[347] "fault"             "feel"
[349] "felt"              "fetch"
[351] "fight"             "figur"
[353] "file"              "fill"
[355] "film"              "final"
[357] "find"              "fine"
[359] "finger"            "finish"
[361] "first"             "five"
[363] "fix"               "flight"
[365] "flirt"             "flower"
[367] "follow"            "fone"
[369] "food"              "forev"
[371] "forget"            "forgot"
[373] "forward"           "found"
[375] "free"              "freemsg"
```

```
[377] "freephon"        "fren"
[379] "fri"             "friday"
[381] "friend"          "friendship"
[383] "frm"             "frnd"
[385] "frnds"           "fuck"
[387] "full"            "fullonsmscom"
[389] "fun"             "funni"
[391] "futur"           "gal"
[393] "game"            "gap"
[395] "gas"             "gave"
[397] "gay"             "gentl"
[399] "get"             "gettin"
[401] "gift"            "girl"
[403] "give"            "glad"
[405] "god"             "goe"
[407] "goin"            "gone"
[409] "gonna"           "good"
[411] "goodmorn"        "goodnight"
[413] "got"             "goto"
[415] "gotta"           "great"
[417] "green"           "greet"
[419] "grin"            "group"
[421] "guarante"        "gud"
[423] "guess"           "guy"
[425] "gym"             "haf"
[427] "haha"            "hai"
[429] "hair"            "half"
[431] "hand"            "hang"
[433] "happen"          "happi"
[435] "hard"            "hav"
[437] "havent"          "head"
[439] "hear"            "heard"
[441] "heart"           "heavi"
[443] "hee"             "hell"
[445] "hello"           "help"
[447] "hey"             "hgsuiteland"
[449] "high"            "hit"
[451] "hiya"            "hmm"
[453] "hmmm"            "hmv"
[455] "hol"             "hold"
[457] "holder"          "holiday"
[459] "home"            "honey"
[461] "hook"            "hop"
[463] "hope"            "horni"
[465] "hospit"          "hot"
[467] "hotel"           "hour"
[469] "hous"            "housemaid"
[471] "how"             "howev"
[473] "howz"            "hrs"
[475] "hug"             "huh"
```

```
[477] "hungri"          "hurri"
[479] "hurt"            "iam"
[481] "ice"             "idea"
[483] "identifi"        "ignor"
[485] "ill"             "imagin"
[487] "imma"            "immedi"
[489] "import"          "inc"
[491] "inch"            "includ"
[493] "india"           "indian"
[495] "info"            "inform"
[497] "insid"           "instead"
[499] "interest"        "interview"
[501] "invit"           "ipod"
[503] "irrit"           "ish"
[505] "issu"            "ive"
[507] "izzit"           "januari"
[509] "jay"             "job"
[511] "john"            "join"
[513] "joke"            "joy"
[515] "jus"             "just"
[517] "juz"             "kalli"
[519] "kate"            "keep"
[521] "kept"            "key"
[523] "kick"            "kid"
[525] "kill"            "kind"
[527] "kinda"           "king"
[529] "kiss"            "knew"
[531] "know"            "knw"
[533] "ladi"            "land"
[535] "landlin"         "laptop"
[537] "lar"             "last"
[539] "late"            "later"
[541] "latest"          "laugh"
[543] "lazi"            "ldn"
[545] "lead"            "learn"
[547] "least"           "leav"
[549] "lect"            "left"
[551] "leh"             "lei"
[553] "lemm"            "less"
[555] "lesson"          "let"
[557] "letter"          "liao"
[559] "librari"         "lick"
[561] "lie"             "life"
[563] "lift"            "light"
[565] "like"            "line"
[567] "link"            "list"
[569] "listen"          "littl"
[571] "live"            "load"
[573] "loan"            "local"
[575] "locat"           "log"
```

```
[577] "login"           "lol"
[579] "long"            "longer"
[581] "look"            "lor"
[583] "lose"            "lost"
[585] "lot"             "lovabl"
[587] "love"            "lover"
[589] "loverboy"        "loyalti"
[591] "ltd"             "ltdecimalgt"
[593] "ltgt"            "lttimegt"
[595] "luck"            "lucki"
[597] "lunch"           "luv"
[599] "made"            "mah"
[601] "mail"            "make"
[603] "man"             "mani"
[605] "march"           "mark"
[607] "marri"           "marriag"
[609] "match"           "mate"
[611] "matter"          "maxim"
[613] "may"             "mayb"
[615] "mean"            "meant"
[617] "med"             "medic"
[619] "meet"            "meh"
[621] "mell"            "member"
[623] "men"             "menu"
[625] "merri"           "messag"
[627] "met"             "mid"
[629] "midnight"        "might"
[631] "min"             "mind"
[633] "mine"            "minut"
[635] "miracl"          "miss"
[637] "mistak"          "moan"
[639] "mob"             "mobil"
[641] "mobileupd"       "mode"
[643] "mom"             "moment"
[645] "mon"             "monday"
[647] "money"           "month"
[649] "mood"            "moon"
[651] "morn"            "motorola"
[653] "move"            "movi"
[655] "mrng"            "mrt"
[657] "msg"             "msgs"
[659] "mths"            "much"
[661] "mum"             "murder"
[663] "music"           "must"
[665] "muz"             "nah"
[667] "nake"            "name"
[669] "nation"          "natur"
[671] "naughti"         "near"
[673] "need"            "net"
[675] "network"         "neva"
```

```
[677] "never"           "new"
[679] "news"            "next"
[681] "nice"            "nigeria"
[683] "night"           "nite"
[685] "nobodi"          "noe"
[687] "nokia"           "none"
[689] "noon"            "nope"
[691] "normal"          "noth"
[693] "notic"           "now"
[695] "ntt"             "num"
[697] "number"          "nxt"
[699] "nyt"             "offer"
[701] "offic"           "offici"
[703] "okay"            "oki"
[705] "old"             "omw"
[707] "one"             "onlin"
[709] "oop"             "open"
[711] "oper"            "opinion"
[713] "opt"             "optout"
[715] "orang"           "orchard"
[717] "order"           "oredi"
[719] "oso"             "other"
[721] "otherwis"        "outsid"
[723] "pack"            "page"
[725] "paid"            "pain"
[727] "paper"           "parent"
[729] "park"            "part"
[731] "parti"           "partner"
[733] "pass"            "passion"
[735] "password"        "past"
[737] "pay"             "peac"
[739] "peopl"           "per"
[741] "person"          "pete"
[743] "phone"           "photo"
[745] "pic"             "pick"
[747] "pictur"          "piec"
[749] "pix"             "pizza"
[751] "place"           "plan"
[753] "plane"           "play"
[755] "player"          "pleas"
[757] "pleasur"         "plenti"
[759] "pls"             "plus"
[761] "plz"             "pmin"
[763] "pmsg"            "pobox"
[765] "poboxwwq"        "point"
[767] "poli"            "polic"
[769] "poor"            "pop"
[771] "possibl"         "post"
[773] "pound"           "power"
[775] "pple"            "ppm"
```

```
[777] "practic"          "pray"
[779] "prefer"           "prepar"
[781] "press"            "pretti"
[783] "price"            "princess"
[785] "privat"           "prize"
[787] "prob"             "probabl"
[789] "problem"          "process"
[791] "project"          "promis"
[793] "pub"              "put"
[795] "qualiti"          "question"
[797] "quick"            "quit"
[799] "quiz"             "quot"
[801] "rain"             "rate"
[803] "rather"           "rcvd"
[805] "reach"            "read"
[807] "readi"            "real"
[809] "realiz"           "realli"
[811] "reason"           "receipt"
[813] "receiv"           "recent"
[815] "record"           "refer"
[817] "regard"           "regist"
[819] "remain"           "rememb"
[821] "remind"           "remov"
[823] "rent"             "rental"
[825] "repli"            "repres"
[827] "request"          "respond"
[829] "respons"          "rest"
[831] "result"           "return"
[833] "reveal"           "review"
[835] "right"            "ring"
[837] "rington"          "rite"
[839] "road"             "rock"
[841] "room"             "roommat"
[843] "rose"             "round"
[845] "rowwjhl"          "rpli"
[847] "rreveal"          "run"
[849] "sad"              "sae"
[851] "safe"             "said"
[853] "sale"             "sam"
[855] "sat"              "saturday"
[857] "savamob"          "save"
[859] "saw"              "say"
[861] "sch"              "school"
[863] "score"            "scream"
[865] "sea"              "search"
[867] "season"           "sec"
[869] "second"           "secret"
[871] "see"              "seem"
[873] "seen"             "select"
[875] "self"             "sell"
```

```
[877] "semest"          "send"
[879] "sens"            "sent"
[881] "serious"         "servic"
[883] "set"             "settl"
[885] "sex"             "sexi"
[887] "shall"           "share"
[889] "shd"             "ship"
[891] "shirt"           "shit"
[893] "shop"            "short"
[895] "show"            "shower"
[897] "shuhui"          "sick"
[899] "side"            "sigh"
[901] "sight"           "sign"
[903] "silent"          "simpl"
[905] "sinc"            "sing"
[907] "singl"           "sir"
[909] "sis"             "sister"
[911] "sit"             "situat"
[913] "sky"             "skype"
[915] "slave"           "sleep"
[917] "slept"           "slow"
[919] "slowli"          "small"
[921] "smile"           "smoke"
[923] "sms"             "smth"
[925] "snow"            "sofa"
[927] "solv"            "somebodi"
[929] "someon"          "someth"
[931] "sometim"         "somewher"
[933] "song"            "soni"
[935] "sonyericsson"    "soon"
[937] "sorri"           "sort"
[939] "sound"           "space"
[941] "speak"           "special"
[943] "specialcal"      "spend"
[945] "spent"           "spoke"
[947] "sport"           "spree"
[949] "stand"           "star"
[951] "start"           "statement"
[953] "station"         "stay"
[955] "std"             "still"
[957] "stock"           "stop"
[959] "store"           "stori"
[961] "str"             "straight"
[963] "street"          "strong"
[965] "student"         "studi"
[967] "stuff"           "stupid"
[969] "style"           "sub"
[971] "subscrib"        "success"
[973] "summer"          "sun"
[975] "sunday"          "sunshin"
```

```
 [977] "support"           "suppos"
 [979] "sure"              "surpris"
 [981] "sweet"             "swing"
 [983] "system"            "take"
 [985] "talk"              "tampa"
 [987] "tcs"               "teach"
 [989] "team"              "tear"
 [991] "teas"              "tel"
 [993] "tell"              "ten"
 [995] "tenerif"           "term"
 [997] "test"              "text"
 [999] "thank"             "thanx"
 [ reached getOption("max.print") -- omitted 166 entries ]
```

- Guardemos este objeto pero eliminemos los caracteres extraños:

Hide

Hide

```
sms_freq_words <- findFreqTerms(sms_dtm_train, 5)
sms_freq_words <- sms_freq_words[7:1166]
str(sms_freq_words)
```

```
 chr [1:1160] "abiola" "abl" "abt" "accept" ...
```

- Ahora filtremos nuestra DTM para dejar solamente las palabras frecuentes:

Hide

Hide

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

- Hemos eliminado la columnas que no tienen palabras frecuentes
- El algoritmo Naive Bayes está creado para características categorícas
- Pero la matriz está contando el número de veces que aparece una palabra
- Esto lo podemos resolver con la siguiente función:

Hide

Hide

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

- Para aplicar esta función a nuestra matriz usamos `apply()`
- Funciona como `lapply()` pero para matrices
- Debemos especificar si queremos aplicar la instrucci'on a las filas o las columnas
- Esto lo hacemos con el par'ametro `MARGIN`
- Como queremos hacerlo a las columnas, fijamos `MARGIN=2`

Hide

Hide

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

- Tenemos dos matrices de caracteres, con las celdas indicando si la palabra está o no contenida (columnas) en cada mensaje (filas)

8. Entrenamiento del modelo y predicción

- Ahora sí estamos listos para utilizar el algoritmo
- Con base en la presencia o no de las palabras, se estimará la probabilidad de un que SMS sea spam
- Para esto usaremos el paquete `e1071` , que incluye una variedad de funciones de machine learning

Hide

Hide

```
install.packages("e1071")
```

```
Error in install.packages : Updating loaded packages
```

Hide

Hide

Hide

```
library(e1071)
```

- En este paquete los procesos de entrenamiento y predicción se hacen en etapas separadas
- Para entrenar, usamos la función `naiveBayes()` con la siguiente sintaxis:

```
m <- naiveBayes(train, class, laplace)
```

- Donde `m` es el nombre que le damos al objeto
- `train` es el data frame o matriz con los datos de entrenamiento
- `class` es un vector tipo factor indicando la clase de cada fila en los datos de entrenamiento
- `laplace` es el parámetro que controla por el estimador de Laplace
- El default es `laplace=0`
- El objeto que nos devueve esta función nos sirve para hacer las predicciones
- Esto lo hacemos con la funcion `predict()`, que tiene la siguiente sintaxis:

```
p <- predict(m, test, type)
```

- donde `m` es el objeto estimado con la función `naivebayes()`
- `test` es el data frame o matriz con los datos de prueba
- `type` es un parámetro que puede ser igual a `class` o `raw`: `class` indica que la predicciones deben ser la categoría, mientras que `raw` es la probabilidad
- El default es `type=class`
- Con esto, podemos estimar nuestro modelo. Primero el entrenamiento:

Hide

Hide

```
sms_classifier <- naiveBayes(sms_train, sms_train_labels)
```

- Ya tenemos el objeto que nos sirve para hacer las predicciones
- Ahora usemos la función `predict()`

Hide

Hide

```
sms_test_pred <- predict(sms_classifier, sms_test)
install.packages("e1071")
```

```
probando la URL 'https://cran.rstudio.com/bin/macosx/mavericks/contrib/3.3/e1071_1.6-
8.tgz'
Content type 'application/x-gzip' length 742779 bytes (725 KB)
==================================================
downloaded 725 KB

tar: Failed to set default locale
```

```

The downloaded binary packages are in
    /var/folders/cw/_9zpljws58jbbv49mcb86l6c0000gn/T//RtmprCm2Nz/downloaded_packages
```

- Veamos cómo nos fue, con la función `CrossTable()` del paquete `gmodels`
- Usamos `dnn` para redifinir los nombres de filas y columnas

Hide

Hide

```
install.packages("gmodels")
```

```
also installing the dependencies 'gtools', 'gdata'

probando la URL 'https://cran.rstudio.com/bin/macosx/mavericks/contrib/3.3/gtools_3.5
.0.tgz'
Content type 'application/x-gzip' length 134356 bytes (131 KB)
==================================================
downloaded 131 KB

probando la URL 'https://cran.rstudio.com/bin/macosx/mavericks/contrib/3.3/gdata_2.17
.0.tgz'
Content type 'application/x-gzip' length 1136842 bytes (1.1 MB)
==================================================
downloaded 1.1 MB

probando la URL 'https://cran.rstudio.com/bin/macosx/mavericks/contrib/3.3/gmodels_2.
16.2.tgz'
Content type 'application/x-gzip' length 72626 bytes (70 KB)
==================================================
downloaded 70 KB

tar: Failed to set default locale
tar: Failed to set default locale
tar: Failed to set default locale

The downloaded binary packages are in
    /var/folders/cw/_9zpljws58jbbv49mcb86l6c0000gn/T//RtmprCm2Nz/downloaded_packages
```

Hide

Hide

```
library(gmodels)
CrossTable(sms_test_pred, sms_test_labels, prop.chisq=FALSE, prop.t=FALSE, dnn=c('pre
dicted', 'actual'))
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|-------------------------|


Total Observations in Table:  1393


             | actual
   predicted |       ham |      spam | Row Total |
-------------|-----------|-----------|-----------|
         ham |      1202 |        20 |      1222 |
             |     0.984 |     0.016 |     0.877 |
             |     0.993 |     0.110 |           |
-------------|-----------|-----------|-----------|
        spam |         9 |       162 |       171 |
             |     0.053 |     0.947 |     0.123 |
             |     0.007 |     0.890 |           |
-------------|-----------|-----------|-----------|
Column Total |      1211 |       182 |      1393 |
             |     0.869 |     0.131 |           |
-------------|-----------|-----------|-----------|
```

- 1202/1211 (99.3%) de los verdaderos ham fueron clasificados correctamente
- 162/182 (89%) de los verdaderos spam fueron clasificados correctamente
- 20/182 (11%) spams fueron incorrectamente clasificados como hams (falsos negativos)
- 9/1211 (0.7%) hams fueron incorrectamente clasificados como spams (falso positivos)
- En total, el modelo clasifica incorrectamente 29/1393 casos. Una precisión del 98%. Nada mal
- Sin mucho esfuerzo, Naive Bayes lo hace muy bien. Por algo este algortimo es el estándar en clasifición de textos
- Además en este caso son más graves los falsos positivos (un ham que se va al spam). La tasa es bajísima en este ejemplo. Menos de uno de cada 100 hams se van al spam
- Igual les recomiendo revisar esa bandeja de cuando en cuando!

8. Mejorando el desemepeño del modelo

- Podemos intentar mejorar el desempeño de nuestro modelo
- En este caso, usando el estimador de Laplace
- Recordemos que este estimador añade un caso a cada característica
- Para evitar que haya ceros, los cuales hacen que automáticamente un mensaje sea clasificado como Spam o Ham sin importar que otra palabras tiene
- Simplemente debemos fijar el parámetro `laplace=1` en la función `naiveBayes()`:

Hide

Hide

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

- Hacemos las predicciones:

Hide

Hide

```
sms_test_pred2 <- predict(sms_classifier2, sms_test)
```

- Y contruimos nuestra tabla de resultados:

Hide

Hide

```
CrossTable(sms_test_pred2, sms_test_labels, prop.chisq=FALSE, prop.t=FALSE, dnn=c('pr
edicted', 'actual'))
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|-------------------------|


Total Observations in Table:  1393


             | actual
   predicted |       ham |      spam | Row Total |
-------------|-----------|-----------|-----------|
         ham |      1204 |        28 |      1232 |
             |     0.977 |     0.023 |     0.884 |
             |     0.994 |     0.154 |           |
-------------|-----------|-----------|-----------|
        spam |         7 |       154 |       161 |
             |     0.043 |     0.957 |     0.116 |
             |     0.006 |     0.846 |           |
-------------|-----------|-----------|-----------|
Column Total |      1211 |       182 |      1393 |
             |     0.869 |     0.131 |           |
-------------|-----------|-----------|-----------|
```

- Comparando, el número de falsos negativos sube de 20 a 28
- Pero el de falsos positivos pasa de 9 a 7
- Es decisión nuestra si sacrificamos un poco de precisión total a cambio de menos falsos positivos