Naive Bayes

Alejandro Campoy Nieves

30 de enero de 2019

```
library(tm)
library(RTextTools)
library(e1071)
library(dplyr)
library(caret)
library(quanteda)
```

Naive Bayes

En esta sección vamos a explicar un clasificador basado en el teorema de Bayes, denominado Naive Bayes. Este es un modelo predictivo, por lo que tendremos la posibilidad de deducir las clases de los items del conjunto del test por medio de un clasificador desarrollado con esta técnica.

Hemos decido realizar esta técnica para barajar la posible componente aleatoria que puedan tener las clases de los medicamentos. Además, nos viene perfecto ya que esta técnica funciona con variables discretas para poder crear el clasificador.

Vamos a realizar esta técnica directamente con el texto, no debería ser muy complicado adaptarlo a variables numéricas (incluso más sencillo del que hemos realizado), ya que nuestro dataset solamente tiene variables discretas, por tanto, preferimos centrarnos directamente en el texto.

```
datos_train <- read.table("datos/datos_train_preprocesado.csv", sep=",",
                          comment.char="",quote = "\"", header=TRUE)
datos_test <- read.table("datos/datos_test_preprocesado.csv", sep=",",
                         comment.char="",quote = "\"", header=TRUE)
# Establecemos la semilla
set.seed(3)
```

El algoritmo realizado es el siguiente:

```
algoritmo.naiveBayes <- function(texto_train,texto_test,label_train,label_test){

  # Calculamos los corpus de los comentarios pasados
  corpus_train = tm::VCorpus(tm::VectorSource(texto_train))
  corpus_test = tm::VCorpus(tm::VectorSource(texto_test))

  # Obtenemos la matriz de términos de esos comentarios
  dtm_train <- DocumentTermMatrix(corpus_train)
  dtm_test <- DocumentTermMatrix(corpus_test)

  # Obtenemos los términos más frecuentes
  # (Aquellos en los que se repiten en más de 5 textos)
  freq.words <- findFreqTerms(dtm_train, 5)

  # Recalculamos la matriz de términos en función de este concepto (Sparsity)
  dtm_freq_train <- DocumentTermMatrix(corpus_train, control=list(dictionary = freq.words))
  dtm_freq_test <- DocumentTermMatrix(corpus_test, control=list(dictionary = freq.words))
```

```
  #Función para convertir el peso de los términos en valores binarios:
  # yes=presente, no=ausente
  convert_counts <- function(x) {
    x <- ifelse(x > 0, "Yes", "No")
  }

  # Obtenemos matriz de términos con valores binarios en lugar de pesos (continuos)
  trainNB <- apply(dtm_freq_train, MARGIN = 2, convert_counts)
  testNB <- apply(dtm_freq_test, MARGIN = 2, convert_counts)

  # Entrenamos el clasificador con el conjunto de entrenamiento
  classifier <- naiveBayes(trainNB, as.factor(label_train), laplace = 1)

  # Hacemos las predicciones sobre el conjunto de test y calculamos los errores que tienen
  pred_test <- predict(classifier, newdata=testNB)
  pred_train <- predict(classifier, newdata=trainNB)
  Etest <- mean(pred_test!=label_test)
  Etrain <- mean(pred_train!=label_train)

  cat("--------------------------------\n")
  cat("**MATRIZ DE CONFUSIÓN TEST**\n")

  print(table(pred=pred_test,real=label_test))

  cat(paste("Error de Test: ",Etest*100," %\n\n"))

  cat("-------------------------------------\n")
  cat("**MATRIZ DE CONFUSIÓN TRAIN**\n")

  print(table(pred=pred_train,real=label_train))

  cat(paste("Error de Train: ",Etrain*100," %\n"))
  cat("-------------------------------------")
}
```

Lo primero que hacemos es calcular los corpus correspondientes, tanto para el conjunto de entrenamiento como para el de prueba. Acto seguido nos generamos nuestras matrices de términos del mismo modo que en otras técnicas.

En un intento de hacer esta matriz más pequeña para que fuese algo más eficiente, tratamos de encontrar aquellos términos que se encuentran en menos de 5 documentos (comentarios) en la totalidad de nuestro dataset de entrenamiento. Entonces, recalculamos la matriz de términos despreciando estos términos. En otras palabras, aplicamos la técnica de Sparsity. Es cierto que esto es una parte que podría ir perfectamente en el preprocesamiento, tal y como se ha hablado en el apartado correspondiente. No obstante, no vimos conveniente realizarla en el resto de técnicas salvo en esta. Por lo que lo hacemos de forma específica y exclusiva para este problema.

Como ya hemos mencionado antes, Naive Bayes es un clasificador que funciona bien con variables discretas. Por ello, es importante discretizar nuestras matrices de términos para este problema. Tenemos por cada documento un vector de pesos que varía dependiendo de las frecuencias de cada término en cada documento. Entonces, tal y como se menciona en la teoría de esta asignatura, decidimos discretizar estos valores bajo el criterio de si los términos se encuentran presentes (valor “yes”) o ausentes (valor “no”) en cada documento.

En otras palabras, transformamos nuestras matrices de términos en matrices binarias basándonos en este concepto de presente o ausente. Con esto ya tendríamos los comentarios de nuestro dataset preparados para

poder crear el clasificador basado en el teorema de Bayes. Mencionar que las etiquetas pasadas deben de ser de tipo "factor", de lo contrario no funcionará (se realiza la conversión dentro de la función por si no estuviera en este tipo). Este fue un error que nos retrasó mucho en el desarrollo.

Después, simplemente tendríamos que realizar las predicciones correspondientes, obtener las matrices de confusión y el error derivado de las mismas.

Ha sido realizado en una función de tal forma que podamos pasarle cualquier texto y cualquier etiqueta de nuestro dataset que sea apta para clasificar. Ahora vamos a ver los resultados que nos proporciona esta técnica para los comentarios de beneficios y efectos secundarios.

Vamos a deducir el **el nivel de efectividad en función de los comentarios sobre los beneficios**.

```
algoritmo.naiveBayes(datos_train$benefits_preprocesado,datos_test$benefits_preprocesado,
                     datos_train$effectivenessNumber,datos_test$effectivenessNumber)
```

```
--------------------------------
**MATRIZ DE CONFUSIÓN TEST**
    real
pred   1   2   3   4   5
   1  44  13   7  12   6
   2   3   3   4   7   1
   3   6   8  21  29  27
   4  14  32  80 150 148
   5  14  20  45 111 229
Error de Test:  56.7698259187621  %

-------------------------------------
**MATRIZ DE CONFUSIÓN TRAIN**
    real
pred   1   2   3   4   5
   1 196  20  13  26  34
   2   1  55   1   7   9
   3   3  23 198  41  55
   4  25  60 131 665 359
   5  22  28  72 187 873
Error de Train:  35.985824742268  %
-------------------------------------
```

Figure 1: Matriz de confusión y error para la predicción del nivel de efectividad de los medicamentos a partir de comentarios sobre beneficios.

Como podemos observar en los resultados obtenidos, tenemos un error de 35.9858% para el conjunto de entrenamiento y 56.7698% para el conjunto de test. Los resultados pueden parecer a priori nefastos, ya que podemos decir que se equivoca en un poco más de la mitad de las ocasiones. No obstante, debemos decir en defensa de esta técnica que debe poner una nota de efectividad (del 1 al 5) del medicamento en función del comentario que haya realizado esa persona sobre el mismo. En otras palabras, de 5 posibles valores de nota que puede poner debe decidir una.

Consideramos que es más interesante fijarse en la matriz de confusión que en el error que obtiene. La diagonal principal corresponde con los aciertos obtenidos por el clasificador. Lo interesante es que hay también valores altos cerca de esta diagonal. ¿Qué significa esto?. Quiere decir que el clasificador estima bastante bien las notas incluso cuando se equivoca. Cuando la nota real del medicamento es un 5, el clasificador suele equivocarse (cuando se equivoca) con una mayor frecuencia poniéndole un 4, no con un 1, por ejemplo, el cual es un error más pronunciado.

Otro ejemplo, solo se ha equivocado una vez poniendo un 2, cuando el nivel de efectividad del medicamento era un 5 en realidad. También hay que tener en cuenta que deducir el nivel de efectividad exacto del medicamento en función del comentario de una persona puede ser una tarea difícil incluso para una persona, ya que podemos deducir que tiene una efectividad alta o baja en función del comentario, pero saber el valor excato de efectividad es otra historia.

Ahora, vamos a deducir **el nivel de efectividad en función de los comentarios sobre los efectos secundarios**.

```
algoritmo.naiveBayes(datos_train$effects_preprocesado,datos_test$effects_preprocesado,
                     datos_train$effectivenessNumber,datos_test$effectivenessNumber)
```

```
--------------------------------
**MATRIZ DE CONFUSIÓN TEST**
    real
pred   1   2   3   4   5
   1  25  14  16  26  21
   2   3   5   7  10  13
   3   4   9  17  24  20
   4  18  17  31  58  62
   5  31  31  86 191 295
Error de Test:  61.3152804642166  %

-------------------------------------
**MATRIZ DE CONFUSIÓN TRAIN**
    real
pred    1    2    3    4    5
   1  103    2   20   54   61
   2    1   58    6   21   37
   3   13   11  145   27   55
   4   25   33   48  360   91
   5  105   82  196  464 1086
Error de Train:  43.5567010309278  %
-------------------------------------
```

Figure 2: Matriz de confusión y error para la predicción del nivel de efectividad de los medicamentos a partir de comentarios sobre efectos secundarios.

Vemos que las predicciones son parecidas, aunque con un poco más de error tanto en el test como el train. Parece ser que los efectos secundarios que refleja la gente en los comentarios son menos útiles para deducir el nivel de efectividad del medicamento. En realidad le vemos mucho sentido a este hecho, porque que un medicamento tenga unos efectos secundarios leves o graves no quiere decir en principio que sea efectivo o no. Por lo que podemos llegar a la conclusión de que este tipo de comentarios aporta un conocimiento menos útil al clasificador para poder deducir este tipo de etiqueta en concreto.

Aun así, vemos que no suele tener muchos errores "graves" ya que hay valores más altos cerca de la diagonal principal de la matriz de confusión para el conjunto de test, de la misma forma que en el caso anterior.

Ahora, vamos a deducir **el ratingLabel en función de los comentarios sobre los beneficios**.

```
algoritmo.naiveBayes(datos_train$benefits_preprocesado,datos_test$benefits_preprocesado,
                     datos_train$ratingLabel,datos_test$ratingLabel)
```

Vemos que los errores son más bajos que en los dos casos anteriores, esto es algo que consideramos normal. ya que ahora el clasificador solo tiene que decir si el rating que pone esa persona está por encima (1) o por debajo (0) del 5. Es como decir si un medicamento es *bueno* o *malo* para la persona que lo prueba en función del comentario que escribe.

Otro dato interesante es que el clasificador comete muchos más falsos negativos que positivos. En otras palabras, tiende a pensar que las personas establecen ratings más bajos de lo que realmente son, en general.

Vamos a analizar **el ratingLabel en función de los comentarios sobre efectos secundarios**.

```
algoritmo.naiveBayes(datos_train$effects_preprocesado,datos_test$effects_preprocesado,
                     datos_train$ratingLabel,datos_test$ratingLabel)
```

Vemos que los errores son más bajos aún, con un 22.4371% en el error de test. Para el ratingLabel ocurre al contrario, generamos un mejor modelo basándonos en los comentarios sobre efectos secundarios que en los beneficios. Suponemos que el rating lo establece las personas, no solo en función de su efectividad, sino de las

```
--------------------------------
**MATRIZ DE CONFUSIÓN TEST**
    real
pred   0   1
   0 163 215
   1  77 579
Error de Test:  28.2398452611219  %

-------------------------------------
**MATRIZ DE CONFUSIÓN TRAIN**
    real
pred    0    1
   0  531  587
   1  129 1857
Error de Train:  23.0670103092784  %
-------------------------------------
```

Figure 3: Matriz de confusión y error para la predicción del ratingLabel de los medicamentos a partir de comentarios sobre beneficios.

```
--------------------------------
**MATRIZ DE CONFUSIÓN TEST**
    real
pred   0   1
   0 116 108
   1 124 686
Error de Test:  22.4371373307544  %

-------------------------------------
**MATRIZ DE CONFUSIÓN TRAIN**
    real
pred    0    1
   0  346  283
   1  314 2161
Error de Train:  19.2332474226804  %
-------------------------------------
```

Figure 4: Matriz de confusión y error para la predicción del ratingLabel de los medicamentos a partir de comentarios sobre efectos secundarios.

sensaciones que el medicamento transmite en su bienestar general. Por lo que no nos parece un resultado descabellado.