Unidad 4

Amelia Martínez Sequera

9/11/2020

Índice de contenidos

1. El algoritmo Naive Bayes.

¿Qué es?

El algoritmo de Naive Bayes se fundamenta en los conocidos como métodos bayesianos, un conjunto de principios matemáticos fundamentales desarrollados por Thomas Bayes cuyo fin es la descripción de eventos probabilísticos. Sobre esta base los clasificadores basados en los métodos bayesianos utilizan datos de entrenamiento para calcular una probabilidad observada de cada clase basada en los valores de las variables. Cuando el clasificador se utiliza posteriormente en datos no etiquetados, utiliza las probabilidades observadas para predecir la clase más probable de los nuevos datos dados para estas variables.

Se han utilizado clasificadores bayesianos para:

- La clasificación de textos, como el filtrado de correo basura (spam), la identificación de autores o la categorización de temas.
- La detección de intrusos o la detección de anomalías en las redes informáticas.
- El diagnóstico de las condiciones médicas mediante conjuntos de síntomas observados.

Típicamente, los clasificadores bayesianos se aplican mejor a los problemas en los que la información de numerosos atributos debe considerarse simultáneamente para estimar la probabilidad de un resultado. Mientras que muchos algoritmos ignoran aquellas características que tienen efectos débiles, los métodos bayesianos utilizan todas las pruebas disponibles para cambiar sutilmente las predicciones. Si un gran número de variables tienen efectos relativamente menores, en conjunto, su impacto combinado podría ser considerable.

Naive Bayes asume que todas las características del conjunto de datos son igualmente importantes e independientes. Estos supuestos rara vez son verdaderos en el mundo real.

El algoritmo de Naive Bayes presenta un problema importante que surge si un evento nunca ocurre para uno o más niveles de la clase, y es que, debido a que las probabilidades en los algoritmos de Naive Bayes se multiplican, un valor del cero por ciento causa que la probabilidad posterior de que se de x suceso sea cero, lo que da a este evento nulo la capacidad para anular y dominar efectivamente sobre todas las demás evidencias.

Una solución a este problema consiste en el uso de un elemento conocido como el Estimador de Laplace, que lleva el nombre del matemático francés Pierre-Simon Laplace. El estimador de Laplace lo que hace es añadir un pequeño número, una cifra residual, a cada uno de los recuentos realizados en la frecuencia, lo que asegura que cada característica tiene una probabilidad no nula de ocurrir con cada clase. Típicamente, el estimador de Laplace se fija en 1, lo que asegura que cada combinación de clases y características se encuentra en los datos al menos una vez.

El estimador de Laplace se puede ajustar a cualquier valor, y no necesariamente tiene que ser el mismo para cada una de las características. Se podría usar el estimador de Laplace para reflejar una presunta probabilidad a priori de cómo la característica se relaciona con la clase. En la práctica, dado un conjunto de datos de entrenamiento lo suficientemente grande, este paso es innecesario, y casi siempre se utiliza el valor de uno.

Puesto que el Naive Bayes utiliza tablas de frecuencia para el aprendizaje, cada característica debe ser categórica a fin de crear las combinaciones de valores de clase y característica que componen la matriz. Como los rasgos numéricos no tienen categorías de valores, este algoritmo no funciona directamente con los datos numéricos.

Una solución fácil y eficaz es la de discretizar las variables numéricas, lo que significa simplemente que los números se colocan en categorías conocidas como contenedores (bins). Hay que tener en consideración es que la discretización de una variable numérica siempre da lugar a una reducción de la información, ya que la granularidad original de la variable se reduce a un conjunto de categorías. Es importante lograr un equilibrio, un número demasiado reducido de bins puede dar lugar a que se oculten tendencias importantes, mientras que un número demasiado elevado de bins puede dar lugar a recuentos pequeños en la tabla de frecuencias de Naive Bayes.

Tabla de fortaleza y debilidades

Fortalezas	Debilidades
Simple, rápido y muy efectivo	Se basa en una suposición a menudo errónea de igualdad de importancia y variables independientes
Funciona bien con ruido y datos faltantes	No es ideal para conjuntos de datos con muchas variables numéricas
Requiere relativamente pocos ejemplos para entrenamiento, pero también funciona bien con una gran cantidad de ejemplos	Las probabilidades estimadas son menos fiables que las clases predichas
Fácil de obtener la probabilidad estimada de una predicción	

2. Ejemplo.

Step 1: Obtención de los datos

```
#Descargamos los archivos
library(readr)
flowering_time <- read_csv("C:/Users/Meli/Downloads/flowering_time.csv",
    col_names = FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_double()
## )
```

```
str(flowering_time)
```

```
## tibble [697 x 1] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ X1: num [1:697] 45 32 29 24 54 43 40 26 42 28 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   X1 = col_double()
##   .. )
```

Step 2: Análisis y procesado de los datos

```
# crearemos un vector vacío donde vamos a recodificar los valores
# correspondientes a los días de floración en rápida (=< 40)
# o lenta (>40 días), en 0 y 1 respectivamente

floracion <- vector() # Se crea un vector vacío
floracion[flowering_time$X1 == 40] <- "0"
floracion[flowering_time$X1 < 40] <- "0"
floracion[flowering_time$X1 > 40] <- "1"

floracion <- as.factor(floracion)
table(floracion)
```

```
## floracion
##   0   1
## 437 260
```

```
str(floracion)
```

```
##  Factor w/ 2 levels "0","1": 2 1 1 1 2 2 1 1 2 1 ...
```

```
Genotypes <- read.csv(file.path("C:/Users/Meli/Downloads/genotype.csv"),
                      stringsAsFactors = TRUE, header = F)
for (i in 1:ncol(Genotypes)) {
    Genotypes[, i] <- factor(Genotypes[, i])
}
```

```
genotipo <- cbind(floracion, Genotypes) #unimos los dos archivos
```

```
library(e1071)

#Separamos los datos en 2/3 training y 1/3 test.
set.seed(12345)

train<-sample(1:nrow(genotipo),round(2*nrow(genotipo)/3,0))

training<- Genotypes[train,]
test<- Genotypes[-train,]

train_labels<-genotipo[train,1]
test_labels<-genotipo[-train,1]
```

Step 3: Entrenando el modelo

```
classifier <- naiveBayes(training, train_labels)
```

Step 4: Evaluando el modelo

Para la evaluación del modelo Naive Bayes creado se requiere de hacer una predicción con el marco de datos de prueba de los genotipos mediante la función predict(), que tomará como parámetros el modelo creado y el mencionado marco de datos.

```
test_pred <- predict(classifier, test)

library(gmodels)

CrossTable(test_pred, test_labels,nprop.chisq = FALSE,
           prop.t = FALSE, dnn = c('predicted', 'actual'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## | Chi-square contribution |
## |           N / Row Total |
## |           N / Col Total |
## |-------------------------|
##
```

```
##
## Total Observations in Table:  232
##
##
##              | actual
##    predicted |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       115 |        43 |       158 |
##              |     3.460 |     5.460 |           |
##              |     0.728 |     0.272 |     0.681 |
##              |     0.810 |     0.478 |           |
## -------------|-----------|-----------|-----------|
##            1 |        27 |        47 |        74 |
##              |     7.388 |    11.657 |           |
##              |     0.365 |     0.635 |     0.319 |
##              |     0.190 |     0.522 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       142 |        90 |       232 |
##              |     0.612 |     0.388 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```
library(caret)
confusionMatrix(test_pred,test_labels, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 115  43
##          1  27  47
##
##                Accuracy : 0.6983
##                  95% CI : (0.6348, 0.7566)
##     No Information Rate : 0.6121
##     P-Value [Acc > NIR] : 0.003857
##
##                   Kappa : 0.3433
##
##  Mcnemar's Test P-Value : 0.072998
##
##             Sensitivity : 0.5222
##             Specificity : 0.8099
##          Pos Pred Value : 0.6351
##          Neg Pred Value : 0.7278
##              Prevalence : 0.3879
##          Detection Rate : 0.2026
##    Detection Prevalence : 0.3190
##       Balanced Accuracy : 0.6660
##
##        'Positive' Class : 1
##
```

```
#La categoría positiva es floración lenta (1)
```

Se observa que de 232 sujetos analizados, se obtienen 27 falsos positivos y 43 falsos negativos. Es una tasa de error alta, luego conviene implementar estrategias para la mejora del modelo. Para ello, se recurre al estimador de Laplace, estableciendo para este un valor de 1. Se realiza la predicción y se crea la tabla de confusión.

Step 5: Mejorando el modelo

```
#laplace = 1:
classifier1 <- naiveBayes(training, train_labels,laplace = 1)

test_pred1 <- predict(classifier1, test)

#we'll compare the predicted classes to the actual classifications using a

CrossTable(test_pred1, 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:  232
##
##
##              | actual
##    predicted |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       116 |        42 |       158 |
##              |     0.817 |     0.467 |           |
## -------------|-----------|-----------|-----------|
##            1 |        26 |        48 |        74 |
##              |     0.183 |     0.533 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       142 |        90 |       232 |
##              |     0.612 |     0.388 |           |
## -------------|-----------|-----------|-----------|
##
##
```

```
confusionMatrix(test_pred1,test_labels, positive = "1")
```

```
## Confusion Matrix and Statistics
```

```
##
##            Reference
## Prediction   0   1
##          0 116  42
##          1  26  48
##
##                Accuracy : 0.7069
##                  95% CI : (0.6438, 0.7646)
##     No Information Rate : 0.6121
##     P-Value [Acc > NIR] : 0.001619
##
##                   Kappa : 0.362
##
##  Mcnemar's Test P-Value : 0.068909
##
##             Sensitivity : 0.5333
##             Specificity : 0.8169
##          Pos Pred Value : 0.6486
##          Neg Pred Value : 0.7342
##              Prevalence : 0.3879
##          Detection Rate : 0.2069
##    Detection Prevalence : 0.3190
##       Balanced Accuracy : 0.6751
##
##        'Positive' Class : 1
##
```

Se ha disminuido el número de falsos negativos de 43 a 42, y el número de falsos positivos de 27 a 26. Se prueba con otro valor para el estimador de laplace, como 0, pero se observa que la diferencia respecto a aplicar un laplace de 1 es mínima: igual número de errores tipo I y II, aunque obtenemos mejores valores estadísticos para laplace= 1.

```
#laplace = 0:
classifier0 <- naiveBayes(training, train_labels,laplace = 0)

test_pred0 <- predict(classifier0, test)

#we'll compare the predicted classes to the actual classifications using a

CrossTable(test_pred0, 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:  232
```

```
##
##
##              | actual
##    predicted |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       115 |        43 |       158 |
##              |     0.810 |     0.478 |           |
## -------------|-----------|-----------|-----------|
##            1 |        27 |        47 |        74 |
##              |     0.190 |     0.522 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       142 |        90 |       232 |
##              |     0.612 |     0.388 |           |
## -------------|-----------|-----------|-----------|
##
##
```

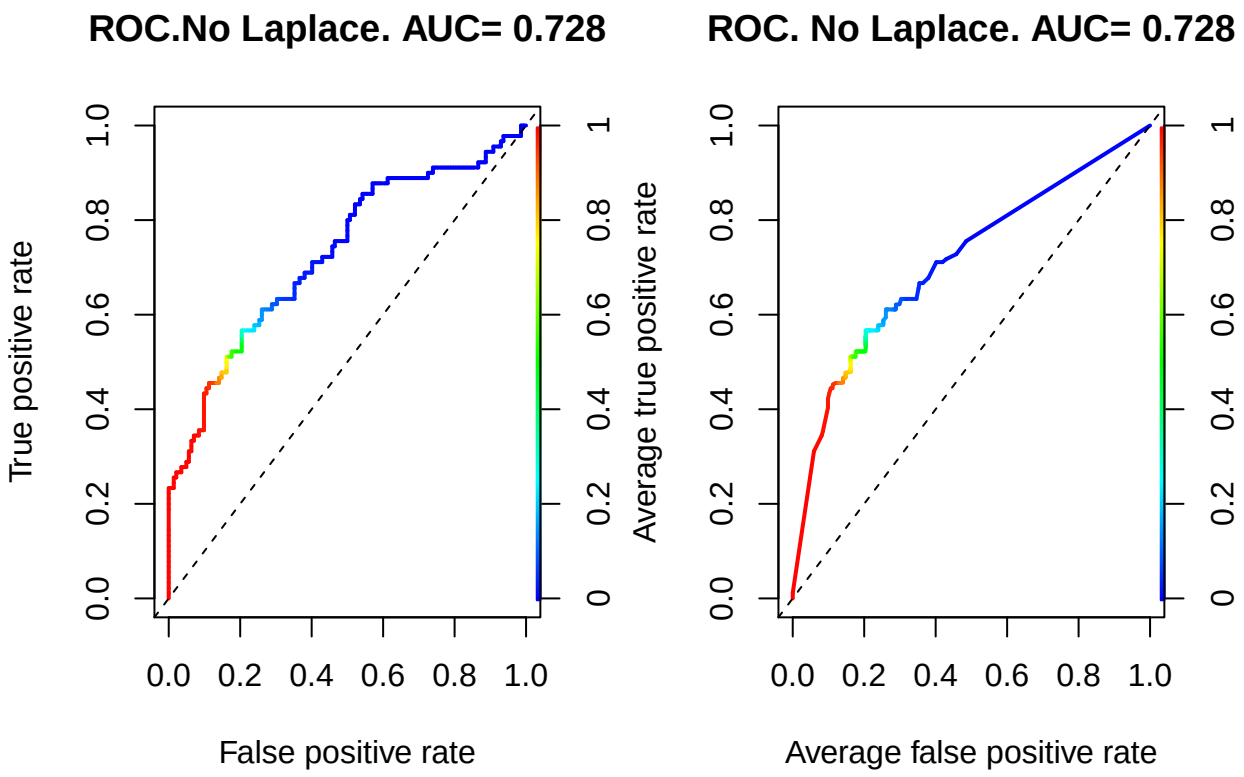
```
confusionMatrix(test_pred0,test_labels, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 115  43
##          1  27  47
##
##                Accuracy : 0.6983
##                  95% CI : (0.6348, 0.7566)
##     No Information Rate : 0.6121
##     P-Value [Acc > NIR] : 0.003857
##
##                   Kappa : 0.3433
##
##  Mcnemar's Test P-Value : 0.072998
##
##             Sensitivity : 0.5222
##             Specificity : 0.8099
##          Pos Pred Value : 0.6351
##          Neg Pred Value : 0.7278
##              Prevalence : 0.3879
##          Detection Rate : 0.2026
##    Detection Prevalence : 0.3190
##       Balanced Accuracy : 0.6660
##
##        'Positive' Class : 1
##
```

Step 6: Curvas ROC(Receiver Operating Characteristic)

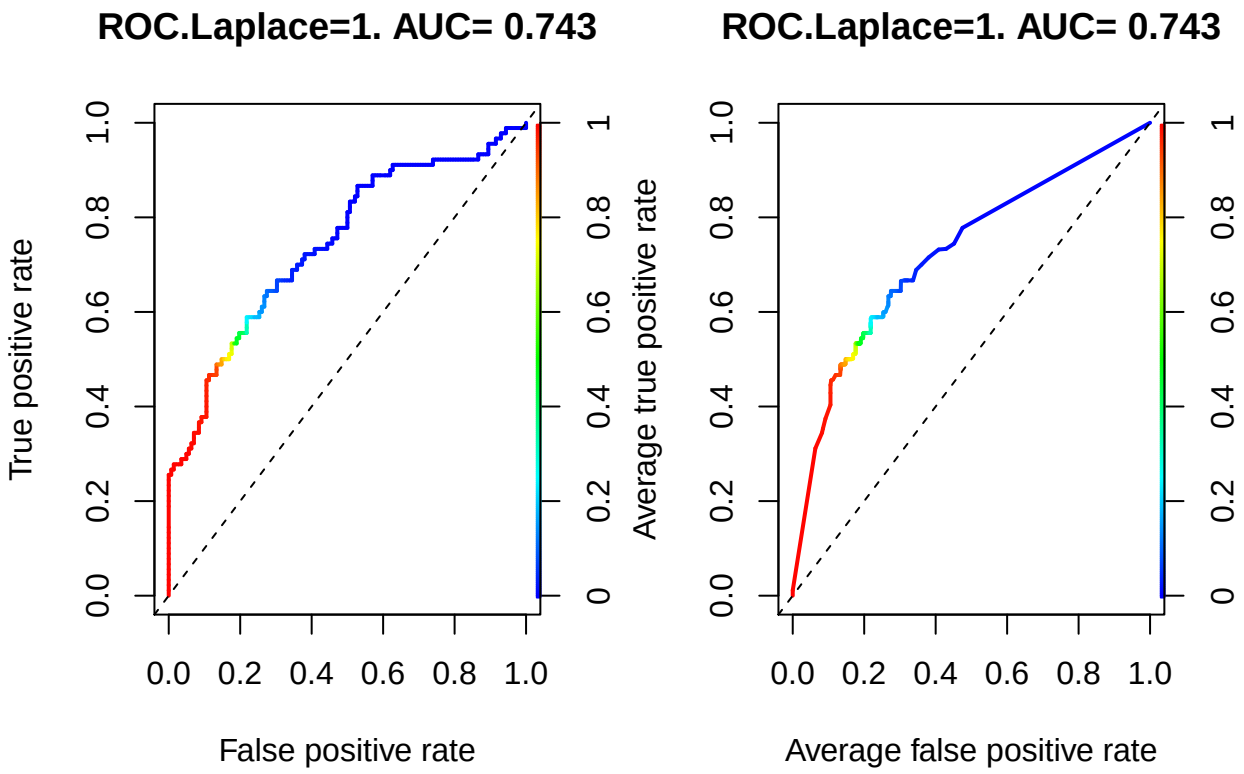
Usar el argumento type =”raw” de la función predict() para obtener las probabilidades.

```r
#Para no laplace:
library(ROCR)
par(mfrow = c(1, 2))
Pred.Prob <- predict(classifier, test, type = "raw")
Pred.Prob <- as.data.frame(Pred.Prob)
pred<- prediction(predictions = Pred.Prob[,2], labels= test_labels)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
perf.auc <- performance(pred, measure = "auc")
perf.auc <- unlist(perf.auc@y.values)
plot(perf, colorize = TRUE, lwd = 2, main = paste("ROC.No Laplace. AUC=", round(perf.auc,
    3)))
abline(a = 0, b = 1, lwd = 1, lty = 2)
plot(perf, avg = "threshold", colorize = TRUE, lwd = 2, main = paste("ROC. No Laplace. AUC=",
    round(perf.auc, 3)))
abline(a = 0, b = 1, lwd = 1, lty = 2)
```




```r
#Para laplace=1
par(mfrow = c(1, 2))
Pred.Prob <- predict(classifier1, test, type = "raw")
Pred.Prob <- as.data.frame(Pred.Prob)
pred<- prediction(predictions = Pred.Prob[,2], labels= test_labels)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
perf.auc <- performance(pred, measure = "auc")
perf.auc <- unlist(perf.auc@y.values)
plot(perf, colorize = TRUE, lwd = 2, main = paste("ROC.Laplace=1. AUC=", round(perf.auc,
```

```
    3)))
abline(a = 0, b = 1, lwd = 1, lty = 2)
plot(perf, avg = "threshold", colorize = TRUE, lwd = 2, main = paste("ROC.Laplace=1. AUC=",
    round(perf.auc, 3)))
abline(a = 0, b = 1, lwd = 1, lty = 2)
```



Aunque cualitativamente no se observa una gran mejora en la gráfica, vemos un aumento cuantitativo en el valor del AUC.