Algoritmo de Naive Bayes

Alejandro Navas González

Fecha: 11 de septiembre, 2020

Contents

Introducción al Proyecto

Marco de Datos

Se conoce el genotipo de 697 plantas y su momento de floración en días. El fichero **floweringTime.zip** contiene:

- **genotype.csv**: Es el genotipo para cada planta (697 x 149). Hay tres posibles estados como máximo: 0, 1 y 2 que se corresponden con *Homocigoto dominante*, *Heterocigoto* y *Homocigoto recesivo*, respectivamente.
- **flowering_time.csv**: Es el tiempo transcurrido hasta la floración en días para cada planta (697 x 1).

Objetivo

El objetivo marcado en este trabajo es la predicción del tipo de floración (rápida o lenta) en función del genotipo de la planta.

Procedimiento

Para resolver este problema se realizará un informe dinámico donde se aplicarán los mismos pasos que se siguieron para el algoritmo k-NN. El informe comenzará con un índice y una sección que incluya la tabla de fortalezas y debilidades del algoritmo Naive Bayes y una explicación del mismo. A continuación, ya se plantea la resolución del problema con el algoritmo de Naive Bayes aplicando los pasos que sugiere realizar en su obra Brett Lantz con el marco de datos del spam.

Información Importante

- Floración rápida es cuando el número de días trascurrido hasta la floración es menor o igual a 40 días, en caso contrario es floración lenta. Se codifica la floración rápida como 0 y la floración lenta como 1.
- El dataset se dividirá en dos tercios para el marco de datos training y el tercio restante para el marco de datos de prueba. Para utilizar la misma serie de registros de training y de test usar como semilla inicial el valor de *set.seed(12345)*.
- Para evaluar el rendimiento del modelo se usará la función *confusionMatrix()* del paquete **caret**, que da más información. La **categoría positiva** es **floración lenta**.
- Se debe probar el modelo de Naive Bayes con laplace=0 y laplace=1.
- Crear un nuevo apartado titulado "Curvas ROC" donde se obtengan las curvas ROC para el modelo de Naive Bayes con laplace=0 y laplace= 1 y el área bajo la curva. Recordar que el package ROCR sirve para hacer las curvas ROC (ver Unidad 3). Importante: Usar el argumento **type ="raw"** de la función *predict()* para obtener las probabilidades.

Introducción al Algoritmo de Naive Bayes

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; y es esta simple idea la que da lugar a un método que a menudo tiene resultados similares a los ofrecidos por algoritmos más sofisticados. De hecho, 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.

Fortalezas & Debilidades

El **algoritmo de Naive Bayes** (**NB**) describe así una aplicación de los métodos bayesianos simple para la clasificación. Si bien no es el único método de aprendizaje automático que se fundamenta en el **Teorema de Bayes**, sí que es el más común, en particular para la clasificación de textos, donde se ha convertido en la norma de facto. Sus fortalezas y deblidades son las siguientes:

Fortalezas	**Debilidades**
Es simple, rápido y muy efectivo	Se basa en una suposición a menudo errónea de características igualmente importantes e independientes
Funciona bien con los datos ruidosos y faltantes	No es ideal para conjuntos de datos con gran número de características numéricas
Requiere de relativamente pocos sujetos para el entrenamiento, y también funciona bien para un gran número de éstos	Las probabilidades estimadas son menos fiables que las clases previstas
Es fácil obtener la probabilidad estimada de una predicción	

Así, el algoritmo de Naive Bayes recibe tal nombre porque hace un par de suposiciones "ingenuas" (*naive*) sobre los datos. En particular, el 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. Sin embargo, en la mayoría de los casos, a pesar de que se violen estas suposiciones, el Naive Bayes se desempeña bastante bien. Esto es cierto incluso en circunstancias extremas donde se encuentran fuertes dependencias entre las variables predictoras. De este modo, dadas la versatilidad y precisión de este algoritmo en muchos tipos de condiciones, Naive Bayes es a menudo un fuerte candidato inicial para las tareas de aprendizaje de clasificación. La razón por la que el Naive Bayes funciona bien a pesar de sus suposiciones erróneas ha sido objeto de gran especulación. Una explicación es que no es importante obtener una estimación cuidadosa de la probabilidad mientras los valores de la clase predichos sean verdaderos.

El Estimador de Laplace

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.

Naive Bayes para Variables Cuantitativas

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. Sin embargo, hay formas de abordar esta cuestión.

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**). Por esta razón, la discretización también se denomina a veces "*binning*". Este método es ideal cuando hay grandes cantidades de datos de entrenamiento, una condición común cuando se trabaja con Naive Bayes.

Hay varias maneras de discretizar una variable numérica. La más común es explorar los datos para las categorías naturales o los puntos de corte en la distribución de los datos. La elección de los puntos de corte y el establecimiento del tipo y número de bins es arbitraria. De hecho, si no hay puntos de corte obvios, una opción es discretizar la variable mediante el uso de cuantiles, dividiendo tal que así los datos en tres bins con tertiles, cuatro bins con cuartiles, cinco bins con quintiles, etcétera.

Por último, una cuestión que 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, es decir, 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, luego es preciso buscar un punto medio entre ambas situaciones.

Naive Bayes & la Floración

Recolección de los Datos

Se importan dos ficheros. Uno es genotype.csv, que contiene 697 genotipos de plantas en el que se recogen 149 genes, que pueden tomar los valores 0, 1 y 2 si son *homocigoto dominante*, *heterocigoto* y *homocigoto recesivo*, respectivamente. El otro fichero a importar es flowering_time.csv, que recoge el tiempo transcurrido hasta la floración en días para cada uno de los sujetos del primer fichero. Se recurre a la función *read.csv()* para importar ambos ficheros.

```
Flower<-read.csv(file.path(params$folder.data, params$file1), stringsAsFactors = TRUE)
Genotypes<-read.csv(file.path(params$folder.data, params$file2), stringsAsFactors = TRUE)
```

Exploración & Preparación

La primera etapa del análisis para construir un clasificador de Bayes consiste en el procesamiento de los raw data o datos crudos para realizar el análisis. Esto es de especial relevancia en el análisis de textos.

Análisis del Marco de Datos sobre la Floración

En primer lugar, se verifica que el marco de datos importado se corresponda con lo que se ha descrito previamente, para lo cual se utilizan las funciones *dim()* y *str()*. A continuación, se crea una variable para clasificar la floración de tal modo que si el sujeto tarda 40 días o menos en florecer se dice que la floración es rápida (*Fast*), y si la floración tarda más de 40 días se tendrá una floración lenta (*Slow*). Se vuelve a explorar el marco de datos con las mismas funciones para ver que la variable creada, a saber, Flowering, tiene coherencia con lo que se pretende realizar y se mira mediante las funciones *table()* y *prop.table()* cuántos de los sujetos tienen floración temprana y cuántos tienen floración tardía.

```
dim(Flower)
```

```
[1] 696   1
```

```
str(Flower)
```

```
'data.frame':   696 obs. of  1 variable:
 $ X45.0: num  32 29 24 54 43 40 26 42 28 33 ...
```

```
names(Flower)<-c("Days")
head(Flower)
```

```
Flower$Flowering<-NA
for (i in 1:length(Flower$Days)){
  if (Flower$Days[i]<=40){
    Flower$Flowering[i]<-"Fast"
  } else{
    Flower$Flowering[i]<-"Slow"
  }
}
Flower$Flowering<-factor(Flower$Flowering)
dim(Flower)
```

```
[1] 696   2
```

```
str(Flower)
```

```
'data.frame':   696 obs. of  2 variables:
 $ Days     : num  32 29 24 54 43 40 26 42 28 33 ...
 $ Flowering: Factor w/ 2 levels "Fast","Slow": 1 1 1 2 2 1 1 2 1 1 ...
```

```
head(Flower)
```

```
table(Flower$Flowering)
```

```

Fast Slow
 437  259
```

```
prop.table(table(Flower$Flowering))
```

```

     Fast      Slow
0.6278736 0.3721264
```

Análisis del Marco de Datos sobre los Genotipos

```
dim(Genotypes)
```

```
[1] 696 149
```

```
str(Genotypes)
```

```
'data.frame':   696 obs. of  149 variables:
 $ X0.0   : num  0 2 2 2 0 2 0 0 0 2 ...
 $ X2.0   : num  1 2 0 2 0 0 0 2 2 2 ...
 $ X2.0.1 : num  2 2 0 0 0 0 2 2 0 0 ...
 $ X2.0.2 : num  2 0 0 0 0 2 0 2 2 0 ...
 $ X0.0.1 : num  0 0 0 0 0 0 0 0 2 2 ...
 $ X2.0.3 : num  0 2 0 2 0 0 2 0 0 0 ...
 $ X0.0.2 : num  1 0 0 0 2 2 0 0 0 0 ...
 $ X0.0.3 : num  0 0 0 0 0 0 0 0 2 2 ...
 $ X0.0.4 : num  0 2 0 0 2 0 2 0 2 2 ...
 $ X0.0.5 : num  0 2 0 2 0 0 0 2 0 2 ...
 $ X2.0.4 : num  2 2 0 2 0 2 0 0 0 2 ...
 $ X0.0.6 : num  0 0 0 0 2 0 0 2 2 0 ...
 $ X0.0.7 : num  0 2 2 2 2 0 2 0 0 2 ...
```

```
$ X2.0.5 : num  0 0 0 0 0 0 0 0 2 0 ...
$ X0.0.8 : num  1 0 2 0 0 0 0 0 0 0 ...
$ X0.0.9 : num  2 2 0 0 0 2 0 0 2 0 ...
$ X0.0.10: num  2 0 0 2 2 2 0 2 2 0 ...
$ X2.0.6 : num  0 0 0 0 2 0 0 0 0 2 ...
$ X2.0.7 : num  1 2 0 0 0 2 2 2 0 2 ...
$ X0.0.11: num  0 0 0 2 0 0 2 0 2 0 ...
$ X0.0.12: num  0 0 0 0 0 0 0 2 0 0 ...
$ X2.0.8 : num  0 0 0 0 0 0 2 0 0 0 ...
$ X0.0.13: num  0 2 2 0 0 0 2 2 2 0 ...
$ X0.0.14: num  0 2 2 0 2 0 0 0 0 0 ...
$ X2.0.9 : num  0 0 0 0 0 2 2 2 0 0 ...
$ X2.0.10: num  2 0 0 2 0 2 2 0 0 0 ...
$ X0.0.15: num  0 0 0 2 2 0 0 0 0 0 ...
$ X0.0.16: num  0 0 0 0 0 0 2 0 2 0 ...
$ X2.0.11: num  0 0 2 0 0 2 0 0 0 2 ...
$ X0.0.17: num  0 2 0 0 0 2 2 0 0 2 ...
$ X0.0.18: num  0 2 0 0 0 0 0 0 0 0 ...
$ X2.0.12: num  0 0 0 0 2 2 0 0 0 0 ...
$ X0.0.19: num  0 2 2 0 2 0 0 2 0 0 ...
$ X0.0.20: num  0 2 0 0 2 2 0 2 0 2 ...
$ X0.0.21: num  0 0 2 2 2 0 0 2 0 2 ...
$ X0.0.22: num  1 0 0 0 0 2 0 2 0 0 ...
$ X0.0.23: num  0 0 2 0 0 0 0 2 0 2 ...
$ X0.0.24: num  2 2 2 0 0 0 2 2 2 0 ...
$ X0.0.25: num  2 0 2 2 0 2 2 0 0 0 ...
$ X2.0.13: num  2 0 0 0 2 2 2 0 2 0 ...
$ X2.0.14: num  0 0 0 2 2 0 0 0 0 2 ...
$ X0.0.26: num  2 2 2 2 2 2 0 0 2 0 ...
$ X0.0.27: num  0 0 0 0 0 0 0 2 2 0 ...
$ X0.0.28: num  2 2 0 0 0 2 0 0 2 0 ...
$ X2.0.15: num  2 0 0 0 0 0 0 0 0 2 ...
$ X0.0.29: num  2 2 2 0 2 2 2 0 0 0 ...
$ X2.0.16: num  1 0 2 0 0 0 0 0 2 0 ...
$ X2.0.17: num  0 0 0 2 0 0 2 2 0 2 ...
$ X2.0.18: num  2 0 2 2 2 2 2 2 2 2 ...
$ X2.0.19: num  0 2 0 2 2 0 0 0 0 0 ...
$ X0.0.30: num  2 2 2 0 0 2 2 2 0 0 ...
$ X2.0.20: num  0 0 2 2 0 2 2 0 0 0 ...
$ X2.0.21: num  0 0 0 0 0 0 0 0 0 0 ...
$ X0.0.31: num  2 0 2 2 2 2 0 0 0 2 ...
$ X0.0.32: num  0 0 2 2 0 0 0 2 0 0 ...
$ X0.0.33: num  0 2 2 2 2 0 2 0 0 0 ...
$ X0.0.34: num  0 0 0 2 0 0 2 2 0 0 ...
$ X2.0.22: num  0 2 0 0 2 0 0 0 0 0 ...
$ X0.0.35: num  0 0 2 2 0 0 2 2 0 2 ...
$ X0.0.36: num  0 0 0 0 0 0 0 2 0 0 0 ...
$ X2.0.23: num  0 2 2 2 2 2 2 2 0 0 0 ...
$ X0.0.37: num  2 0 0 2 0 2 0 2 2 2 ...
$ X0.0.38: num  2 0 0 2 0 2 0 2 2 2 ...
$ X2.0.24: num  2 2 2 2 0 2 0 0 2 0 ...
$ X0.0.39: num  0 2 0 2 0 2 2 0 0 2 ...
$ X0.0.40: num  0 0 0 0 0 0 0 0 2 0 0 ...
$ X0.0.41: num  2 0 0 0 2 2 0 0 2 0 ...
```

```
 $ X2.0.25: num  0 0 2 0 0 2 2 2 0 0 ...
 $ X2.0.26: num  2 2 0 2 0 2 0 2 2 0 ...
 $ X0.0.42: num  0 0 2 2 0 0 0 0 2 0 ...
 $ X0.0.43: num  0 0 0 0 2 0 0 0 0 0 ...
 $ X0.0.44: num  0 0 2 2 2 0 0 0 0 2 ...
 $ X0.0.45: num  2 2 0 0 0 2 0 2 0 0 ...
 $ X0.0.46: num  0 0 2 0 0 0 0 2 0 2 ...
 $ X0.0.47: num  2 2 0 0 0 2 0 0 0 0 ...
 $ X2.0.27: num  1 0 0 2 0 0 0 2 2 2 ...
 $ X0.0.48: num  0 2 0 0 0 2 0 0 2 0 ...
 $ X0.0.49: num  0 2 0 0 0 0 0 0 0 0 ...
 $ X0.0.50: num  0 0 0 0 0 0 0 0 0 2 ...
 $ X0.0.51: num  2 0 0 0 0 2 0 2 2 0 ...
 $ X0.0.52: num  1 2 0 0 0 0 2 0 0 0 ...
 $ X0.0.53: num  1 0 0 0 2 0 0 0 0 2 ...
 $ X0.0.54: num  0 0 0 2 0 0 0 0 2 0 ...
 $ X2.0.28: num  0 0 0 0 2 0 0 0 0 2 ...
 $ X2.0.29: num  2 0 0 0 0 2 2 0 0 2 ...
 $ X0.0.55: num  2 2 0 2 0 2 0 0 0 2 ...
 $ X2.0.30: num  0 2 1 2 2 0 2 2 0 0 ...
 $ X0.0.56: num  1 0 0 2 2 0 0 0 0 2 ...
 $ X0.0.57: num  0 2 0 2 0 0 0 2 0 0 ...
 $ X0.0.58: num  0 2 2 0 0 0 2 0 0 0 ...
 $ X0.0.59: num  0 0 2 0 0 0 0 2 0 2 ...
 $ X0.0.60: num  2 2 0 2 0 2 2 2 2 2 ...
 $ X0.0.61: num  2 2 2 0 0 0 2 0 2 0 ...
 $ X0.0.62: num  2 0 0 0 0 2 0 0 2 0 ...
 $ X0.0.63: num  2 2 2 0 0 0 2 0 0 0 ...
 $ X0.0.64: num  0 2 2 0 0 0 2 0 0 2 ...
 $ X0.0.65: num  0 0 2 2 0 0 2 0 0 0 ...
 $ X0.0.66: num  2 0 0 0 0 2 0 2 0 0 ...
 $ X2.0.31: num  0 2 2 0 0 0 2 0 2 0 ...
  [list output truncated]
```

```
for (i in 1:ncol(Genotypes)){
  Genotypes[,i]<-factor(Genotypes[,i])
}
str(Genotypes)
```

```
'data.frame':   696 obs. of  149 variables:
 $ X0.0   : Factor w/ 3 levels "0","1","2": 1 3 3 3 1 3 1 1 1 3 ...
 $ X2.0   : Factor w/ 3 levels "0","1","2": 2 3 1 3 1 1 1 3 3 3 ...
 $ X2.0.1 : Factor w/ 3 levels "0","1","2": 3 3 1 1 1 1 3 3 1 1 ...
 $ X2.0.2 : Factor w/ 3 levels "0","1","2": 3 1 1 1 1 3 1 3 3 1 ...
 $ X0.0.1 : Factor w/ 2 levels "0","2": 1 1 1 1 1 1 1 1 2 2 ...
 $ X2.0.3 : Factor w/ 3 levels "0","1","2": 1 3 1 3 1 1 3 1 1 1 ...
 $ X0.0.2 : Factor w/ 3 levels "0","1","2": 2 1 1 1 3 3 1 1 1 1 ...
 $ X0.0.3 : Factor w/ 2 levels "0","2": 1 1 1 1 1 1 1 1 2 2 ...
 $ X0.0.4 : Factor w/ 3 levels "0","1","2": 1 3 1 1 3 1 3 1 3 3 ...
 $ X0.0.5 : Factor w/ 3 levels "0","1","2": 1 3 1 3 1 1 1 3 1 3 ...
 $ X2.0.4 : Factor w/ 3 levels "0","1","2": 3 3 1 3 1 3 1 1 1 3 ...
 $ X0.0.6 : Factor w/ 3 levels "0","1","2": 1 1 1 1 3 1 1 3 3 1 ...
 $ X0.0.7 : Factor w/ 3 levels "0","1","2": 1 3 3 3 3 1 3 1 1 3 ...
 $ X2.0.5 : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 3 1 ...
 $ X0.0.8 : Factor w/ 3 levels "0","1","2": 2 1 3 1 1 1 1 1 1 1 ...
```

```
$ X0.0.9 : Factor w/ 3 levels "0","1","2": 3 3 1 1 1 3 1 1 3 1 ...
$ X0.0.10: Factor w/ 3 levels "0","1","2": 3 1 1 3 3 3 1 3 3 1 ...
$ X2.0.6 : Factor w/ 3 levels "0","1","2": 1 1 1 1 3 1 1 1 1 3 ...
$ X2.0.7 : Factor w/ 3 levels "0","1","2": 2 3 1 1 1 3 3 3 1 3 ...
$ X0.0.11: Factor w/ 3 levels "0","1","2": 1 1 1 3 1 1 3 1 3 1 ...
$ X0.0.12: Factor w/ 2 levels "0","2": 1 1 1 1 1 1 1 2 1 1 ...
$ X2.0.8 : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 3 1 1 1 ...
$ X0.0.13: Factor w/ 3 levels "0","1","2": 1 3 3 1 1 1 3 3 3 1 ...
$ X0.0.14: Factor w/ 3 levels "0","1","2": 1 3 3 1 3 1 1 1 1 1 ...
$ X2.0.9 : Factor w/ 3 levels "0","1","2": 1 1 1 1 1 3 3 3 1 1 ...
$ X2.0.10: Factor w/ 3 levels "0","1","2": 3 1 1 3 1 3 3 1 1 1 ...
$ X0.0.15: Factor w/ 3 levels "0","1","2": 1 1 1 3 3 1 1 1 1 1 ...
$ X0.0.16: Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 3 1 3 1 ...
$ X2.0.11: Factor w/ 3 levels "0","1","2": 1 1 3 1 1 3 1 1 1 3 ...
$ X0.0.17: Factor w/ 3 levels "0","1","2": 1 3 1 1 1 3 3 1 1 3 ...
$ X0.0.18: Factor w/ 3 levels "0","1","2": 1 3 1 1 1 1 1 1 1 1 ...
$ X2.0.12: Factor w/ 3 levels "0","1","2": 1 1 1 1 3 3 1 1 1 1 ...
$ X0.0.19: Factor w/ 2 levels "0","2": 1 2 2 1 2 1 1 2 1 1 ...
$ X0.0.20: Factor w/ 3 levels "0","1","2": 1 3 1 1 3 3 1 3 1 3 ...
$ X0.0.21: Factor w/ 3 levels "0","1","2": 1 1 3 3 3 1 1 3 1 3 ...
$ X0.0.22: Factor w/ 3 levels "0","1","2": 2 1 1 1 1 3 1 3 1 1 ...
$ X0.0.23: Factor w/ 3 levels "0","1","2": 1 1 3 1 1 1 1 3 1 3 ...
$ X0.0.24: Factor w/ 3 levels "0","1","2": 3 3 3 1 1 1 3 3 3 1 ...
$ X0.0.25: Factor w/ 3 levels "0","1","2": 3 1 3 3 1 3 3 1 1 1 ...
$ X2.0.13: Factor w/ 2 levels "0","2": 2 1 1 1 2 2 2 1 2 1 ...
$ X2.0.14: Factor w/ 3 levels "0","1","2": 1 1 1 3 3 1 1 1 1 3 ...
$ X0.0.26: Factor w/ 3 levels "0","1","2": 3 3 3 3 3 3 1 1 3 1 ...
$ X0.0.27: Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 3 3 1 ...
$ X0.0.28: Factor w/ 3 levels "0","1","2": 3 3 1 1 1 3 1 1 3 1 ...
$ X2.0.15: Factor w/ 3 levels "0","1","2": 3 1 1 1 1 1 1 1 1 3 ...
$ X0.0.29: Factor w/ 3 levels "0","1","2": 3 3 3 1 3 3 3 1 1 1 ...
$ X2.0.16: Factor w/ 3 levels "0","1","2": 2 1 3 1 1 1 1 1 3 1 ...
$ X2.0.17: Factor w/ 3 levels "0","1","2": 1 1 1 3 1 1 3 3 1 3 ...
$ X2.0.18: Factor w/ 3 levels "0","1","2": 3 1 3 3 3 3 3 3 3 3 ...
$ X2.0.19: Factor w/ 3 levels "0","1","2": 1 3 1 3 3 1 1 1 1 1 ...
$ X0.0.30: Factor w/ 3 levels "0","1","2": 3 3 3 1 1 3 3 3 1 1 ...
$ X2.0.20: Factor w/ 3 levels "0","1","2": 1 1 3 3 1 3 3 1 1 1 ...
$ X2.0.21: Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 1 1 1 ...
$ X0.0.31: Factor w/ 3 levels "0","1","2": 3 1 3 3 3 3 1 1 1 3 ...
$ X0.0.32: Factor w/ 3 levels "0","1","2": 1 1 3 3 1 1 1 3 1 1 ...
$ X0.0.33: Factor w/ 3 levels "0","1","2": 1 3 3 3 3 1 3 1 1 1 ...
$ X0.0.34: Factor w/ 3 levels "0","1","2": 1 1 1 3 1 1 3 3 1 1 ...
$ X2.0.22: Factor w/ 3 levels "0","1","2": 1 3 1 1 3 1 1 1 1 1 ...
$ X0.0.35: Factor w/ 3 levels "0","1","2": 1 1 3 3 1 1 3 3 1 3 ...
$ X0.0.36: Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 3 1 1 1 ...
$ X2.0.23: Factor w/ 3 levels "0","1","2": 1 3 3 3 3 3 3 1 1 1 ...
$ X0.0.37: Factor w/ 3 levels "0","1","2": 3 1 1 3 1 3 1 3 3 3 ...
$ X0.0.38: Factor w/ 3 levels "0","1","2": 3 1 1 3 1 3 1 3 3 3 ...
$ X2.0.24: Factor w/ 3 levels "0","1","2": 3 3 3 3 1 3 1 1 3 1 ...
$ X0.0.39: Factor w/ 3 levels "0","1","2": 1 3 1 3 1 3 3 1 1 3 ...
$ X0.0.40: Factor w/ 3 levels "0","1","2": 1 1 1 1 1 1 1 3 1 1 ...
$ X0.0.41: Factor w/ 3 levels "0","1","2": 3 1 1 1 3 3 1 1 3 1 ...
$ X2.0.25: Factor w/ 2 levels "0","2": 1 1 2 1 1 2 2 2 1 1 ...
$ X2.0.26: Factor w/ 3 levels "0","1","2": 3 3 1 3 1 3 1 3 3 1 ...
```

```
 $ X0.0.42: Factor w/ 3 levels "0","1","2": 1 1 3 3 1 1 1 1 3 1 ...
 $ X0.0.43: Factor w/ 3 levels "0","1","2": 1 1 1 1 3 1 1 1 1 1 ...
 $ X0.0.44: Factor w/ 3 levels "0","1","2": 1 1 3 3 3 1 1 1 1 3 ...
 $ X0.0.45: Factor w/ 2 levels "0","2": 2 2 1 1 1 2 1 2 1 1 ...
 $ X0.0.46: Factor w/ 2 levels "0","2": 1 1 2 1 1 1 1 2 1 2 ...
 $ X0.0.47: Factor w/ 3 levels "0","1","2": 3 3 1 1 1 3 1 1 1 1 ...
 $ X2.0.27: Factor w/ 3 levels "0","1","2": 2 1 1 3 1 1 1 3 3 3 ...
 $ X0.0.48: Factor w/ 3 levels "0","1","2": 1 3 1 1 1 3 1 1 3 1 ...
 $ X0.0.49: Factor w/ 2 levels "0","2": 1 2 1 1 1 1 1 1 1 1 ...
 $ X0.0.50: Factor w/ 2 levels "0","2": 1 1 1 1 1 1 1 1 1 2 ...
 $ X0.0.51: Factor w/ 3 levels "0","1","2": 3 1 1 1 1 3 1 3 3 1 ...
 $ X0.0.52: Factor w/ 3 levels "0","1","2": 2 3 1 1 1 1 3 1 1 1 ...
 $ X0.0.53: Factor w/ 3 levels "0","1","2": 2 1 1 1 3 1 1 1 1 3 ...
 $ X0.0.54: Factor w/ 3 levels "0","1","2": 1 1 1 3 1 1 1 1 3 1 ...
 $ X2.0.28: Factor w/ 2 levels "0","2": 1 1 1 1 2 1 1 1 1 2 ...
 $ X2.0.29: Factor w/ 3 levels "0","1","2": 3 1 1 1 1 3 3 1 1 3 ...
 $ X0.0.55: Factor w/ 3 levels "0","1","2": 3 3 1 3 1 3 1 1 1 3 ...
 $ X2.0.30: Factor w/ 3 levels "0","1","2": 1 3 2 3 3 1 3 3 1 1 ...
 $ X0.0.56: Factor w/ 3 levels "0","1","2": 2 1 1 3 3 1 1 1 1 3 ...
 $ X0.0.57: Factor w/ 3 levels "0","1","2": 1 3 1 3 1 1 1 3 1 1 ...
 $ X0.0.58: Factor w/ 3 levels "0","1","2": 1 3 3 1 1 1 3 1 1 1 ...
 $ X0.0.59: Factor w/ 3 levels "0","1","2": 1 1 3 1 1 1 1 3 1 3 ...
 $ X0.0.60: Factor w/ 3 levels "0","1","2": 3 3 1 3 1 3 3 3 3 3 ...
 $ X0.0.61: Factor w/ 3 levels "0","1","2": 3 3 3 1 1 1 3 1 3 1 ...
 $ X0.0.62: Factor w/ 3 levels "0","1","2": 3 1 1 1 1 3 1 1 3 1 ...
 $ X0.0.63: Factor w/ 3 levels "0","1","2": 3 3 3 1 1 1 3 1 1 1 ...
 $ X0.0.64: Factor w/ 3 levels "0","1","2": 1 3 3 1 1 1 3 1 1 3 ...
 $ X0.0.65: Factor w/ 3 levels "0","1","2": 1 1 3 3 1 1 3 1 1 1 ...
 $ X0.0.66: Factor w/ 3 levels "0","1","2": 3 1 1 1 1 3 1 3 1 1 ...
 $ X2.0.31: Factor w/ 3 levels "0","1","2": 1 3 3 1 1 1 3 1 3 1 ...
  [list output truncated]
```

Creación de los Marcos de Entrenamiento & Prueba

El siguiente paso es crear los marcos de entrenamiento (para construir el modelo) y el de prueba (para su evaluación). Para estos marcos de datos, dos tercios de se destinarán al primer conjunto y el tercio restante al de testeo. Se verifica en cada caso mediante la función *table()* que los porcentajes de floración son similares en ambos conjuntos creados a partir del marco que trata la floración. En este caso los sujetos se encuentran ordenados en los marcos de floración y genotipos, es decir, se sabe que el sujeto 1 del marco de la floración es también el sujeto 1 del marco de genotipos. Si la situación no fuera tal habría de hacerse *merge()* por el ID de los sujetos, y crear un marco común a partir del cual crear los marcos de entrenamiento y prueba.

```
# Se establece una semilla de pseudoaleatorización.
set.seed(12345)

# Se crean los marcos de entrenamiento y prueba para el marco de floración.
Flower.Train<-Flower[1:465, ]
Flower.Test<-Flower[465:696, ]

dim(Flower.Train)
```

```
[1] 465   2
```

```
dim(Flower.Test)
```

```
[1] 232   2
```

```
table(Flower.Train$Flowering)
```

```
Fast Slow
 289  176
```

```
table(Flower.Test$Flowering)
```

```
Fast Slow
 149   83
```

```
prop.table(table(Flower.Train$Flowering))
```

```
     Fast      Slow
0.6215054 0.3784946
```

```
prop.table(table(Flower.Test$Flowering))
```

```
     Fast      Slow
0.6422414 0.3577586
```

```
# Se crean los marcos de entrenamiento y prueba para el marco de genotipos.
Genotypes.Train<-Genotypes[1:465, ]
Genotypes.Test<-Genotypes[465:696, ]
```

Entrenamiento del Modelo

La implementación de Naive Bayes que se empleará para este estudio se encuentra en el paquete **e1701**. Este paquete fue desarrollado en el departamento de estadística de la **Universidad Tecnológica de Viena** (TU Wien), e incluye una variedad de funciones para machine learning. Existen otras alternativas como la función *NaiveBayes()* del paquete **klaR**, que es prácticamente idéntica a la aproximación que se va a aplicar con el paquete **e1701**.

A diferencia del algoritmo k-NN, la fase de entrenamiento y la de clasificación en Naive Bayes transcurren en diferentes etapas. Entonces, primero, se lleva a cabo la construcción del modelo mediante la función *naiveBayes()* del paquete **e1701**, que toma por parámetros el marco de datos de los genotipos reservados para el entrenamiento, y los datos de floración de estos sujetos, que se encuentran en el marco de entrenamiento de la floración.

```
library(e1071)
Flower.Bayes <- naiveBayes(Genotypes.Train, Flower.Train$Flowering)
```

El objeto creado se almacena en una variable, llamada en este caso Flower.Bayes, que contendrá un objeto *naiveBayes* que servirá para realizar una predicción.

Evaluación del 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.

```
Bayes.Pred <- predict(Flower.Bayes, Genotypes.Test)
```

Para comparar los valores que se han predicho con los valores que se hallan realmente en el marco de prueba se puede recurrir entre otros, a la función *CrossTable()*, del paquete **gmodels**. Se añaden parámetros adicionales como FALSE para eliminar valores innecesarios en las celdas, y se utiliza el parámetro *dnn* para renombrar las filas y columnas de la matriz de confusión.

```
library(gmodels)
Kross1<-CrossTable(Bayes.Pred, Flower.Test$Flowering, prop.chisq = FALSE, prop.t = FALSE, dnn = c('Predi
```

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


Total Observations in Table:  232


             | Actual
   Predicted |      Fast |      Slow | Row Total |
-------------|-----------|-----------|-----------|
        Fast |       129 |        37 |       166 |
             |     0.777 |     0.223 |     0.716 |
             |     0.866 |     0.446 |           |
-------------|-----------|-----------|-----------|
        Slow |        20 |        46 |        66 |
             |     0.303 |     0.697 |     0.284 |
             |     0.134 |     0.554 |           |
-------------|-----------|-----------|-----------|
Column Total |       149 |        83 |       232 |
             |     0.642 |     0.358 |           |
-------------|-----------|-----------|-----------|
```

Mejora del Modelo

Como se observa, y si se considera la floración tardía como la variable positiva, se han obtenido 37 falsos negativos y 20 falsos positivos sobre un total de 232 sujetos analizados. 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.

```
Flower.Bayes2 <- naiveBayes(Genotypes.Train, Flower.Train$Flowering, laplace = 1)
Bayes2.Pred <- predict(Flower.Bayes2, Genotypes.Test)
Kross2<-CrossTable(Bayes2.Pred, Flower.Test$Flowering, prop.chisq = FALSE, prop.t = FALSE, dnn = c('Pred
```

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


Total Observations in Table:  232
```

```
             | Actual
   Predicted |      Fast |      Slow | Row Total |
-------------|-----------|-----------|-----------|
        Fast |       128 |        34 |       162 |
             |     0.790 |     0.210 |     0.698 |
             |     0.859 |     0.410 |           |
-------------|-----------|-----------|-----------|
        Slow |        21 |        49 |        70 |
             |     0.300 |     0.700 |     0.302 |
             |     0.141 |     0.590 |           |
-------------|-----------|-----------|-----------|
Column Total |       149 |        83 |       232 |
             |     0.642 |     0.358 |           |
-------------|-----------|-----------|-----------|
```

Se ha disminuido el número de falsos negativos de 37 a 34, pero el número de falsos positivos ha aumentado de 20 a 21. Se prueba con otro valor para el estimador de laplace, como 2, pero se observa que la diferencia respecto a aplicar un laplace de 1 es mínima, si bien en este caso no aumenta el número de falsos positivos, luego si hubiera de elegirse entre uno de los tres modelos, se escogería este tercero.

```
Flower.Bayes3 <- naiveBayes(Genotypes.Train, Flower.Train$Flowering, laplace = 2)
Bayes3.Pred <- predict(Flower.Bayes3, Genotypes.Test)
Kross3<-CrossTable(Bayes3.Pred, Flower.Test$Flowering, prop.chisq = FALSE, prop.t = FALSE, dnn = c('Pre
```

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


Total Observations in Table:  232


             | Actual
   Predicted |      Fast |      Slow | Row Total |
-------------|-----------|-----------|-----------|
        Fast |       129 |        35 |       164 |
             |     0.787 |     0.213 |     0.707 |
             |     0.866 |     0.422 |           |
-------------|-----------|-----------|-----------|
        Slow |        20 |        48 |        68 |
             |     0.294 |     0.706 |     0.293 |
             |     0.134 |     0.578 |           |
-------------|-----------|-----------|-----------|
Column Total |       149 |        83 |       232 |
             |     0.642 |     0.358 |           |
-------------|-----------|-----------|-----------|
```

Se pueden comparar los resultados obtenidos por *CrossTable()* mediante la siguiente función, *EvalNaiveBayes()*, que es de cosecha propia, la cual toma como parámetro un objeto devuelto por la función *CrossTable()* y calcula su precisión, tasa de error, valor kappa, sensibilidad, especificidad, recall, valor predictivo positivo y F-Measure. Es preciso recalcar que solamente es válida para un objeto creado mediante *CrossTable()*.

```
KrossA<-EvalNaiveBayes(Kross1)
```

```
En el modelo de Naive Bayes utilizado se ha encontrado que:
    > Los verdaderos positivos son: 46.
    > Los falsos negativos encontrados son: 37.
    > Los verdaderos negativos son: 129.
    > Los falsos positivos hallados son: 20.

Evaluación del modelo de Naive Bayes:
    > La precisión del modelo es 0.7543103.
    > La tasa de error del modelo es 0.2456897.
    > El valor de Kappa del modelo es 0.4399458.
    > La sensibilidad del modelo es 0.5542169.
    > La especificidad del modelo es 0.8657718.
    > El recall del modelo es 0.5542169.
    > El valor predictivo positivo es 0.6969697.
    > La medida F del modelo es 0.6174497.
```

```
KrossB<-EvalNaiveBayes(Kross2)
```

```
En el modelo de Naive Bayes utilizado se ha encontrado que:
    > Los verdaderos positivos son: 49.
    > Los falsos negativos encontrados son: 34.
    > Los verdaderos negativos son: 128.
    > Los falsos positivos hallados son: 21.

Evaluación del modelo de Naive Bayes:
    > La precisión del modelo es 0.762931.
    > La tasa de error del modelo es 0.237069.
    > El valor de Kappa del modelo es 0.4655721.
    > La sensibilidad del modelo es 0.5903614.
    > La especificidad del modelo es 0.8590604.
    > El recall del modelo es 0.5903614.
    > El valor predictivo positivo es 0.7.
    > La medida F del modelo es 0.6405229.
```

```
KrossC<-EvalNaiveBayes(Kross3)
```

```
En el modelo de Naive Bayes utilizado se ha encontrado que:
    > Los verdaderos positivos son: 48.
    > Los falsos negativos encontrados son: 35.
    > Los verdaderos negativos son: 129.
    > Los falsos positivos hallados son: 20.

Evaluación del modelo de Naive Bayes:
    > La precisión del modelo es 0.762931.
    > La tasa de error del modelo es 0.237069.
    > El valor de Kappa del modelo es 0.4626011.
    > La sensibilidad del modelo es 0.5783133.
```

```
> La especificidad del modelo es 0.8657718.
> El recall del modelo es 0.5783133.
> El valor predictivo positivo es 0.7058824.
> La medida F del modelo es 0.6357616.
```

```
BayesDF<-rbind(KrossA, KrossB, KrossC)
colnames(BayesDF)<-c("Accuracy","Error", "Falsos Positivos", "Falsos Negativos", "Valor Kappa","Sensibi
rownames(BayesDF)<-c("Sin Laplace", "Laplace 1", "Laplace 2")
BayesDF
```

Curvas ROC

Otra opción interesante para la evaluación de los modelos creados es la conocida como **curva ROC** (*Receiver Operating Characteristic*), la cual se utiliza comúnmente para examinar el equilibrio entre la detección de los verdaderos positivos y la evasión de los falsos positivos. Se utiliza para visualizar la eficacia de los modelos de aprendizaje de las máquinas. Las características de un diagrama ROC típico es presentar la proporción de verdaderos positivos en el eje vertical, y la proporción de falsos positivos en el eje horizontal. Estos valores son equivalentes a la sensibilidad y a (1 - especificidad), respectivamente. Por ello, el diagrama es también conocido como un gráfico de sensibilidad/especificidad.

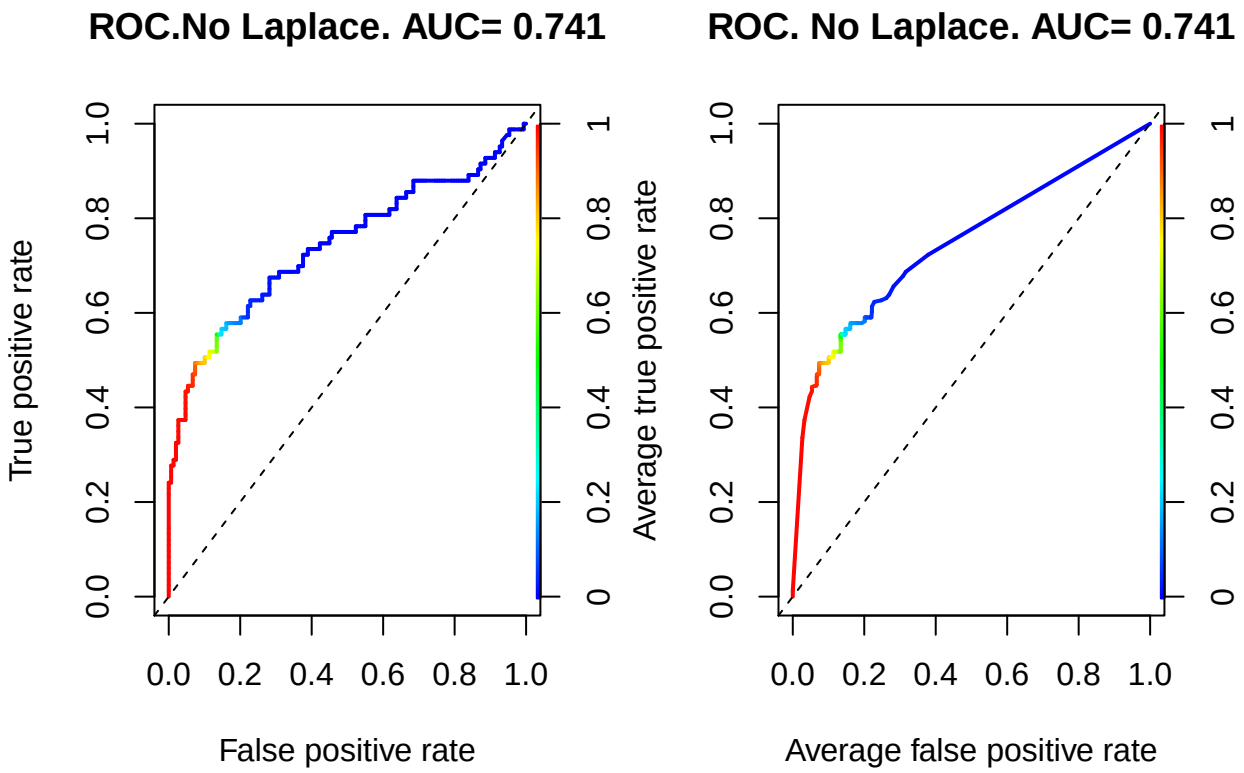
Los puntos que comprenden las curvas ROC indican la tasa positiva verdadera frente a un umbral variable de la tasa de falsos positivos. Para crear las curvas, las predicciones de un clasificador se ordenan por la probabilidad estimada del modelo de la clase positiva, con los valores más grandes primero. Partiendo del origen, el impacto de cada predicción en la tasa positiva verdadera y en la tasa de falsos positivos resultará en un trazado de la curva en forma vertical (para una predicción correcta), u horizontalmente (para una predicción incorrecta).

Para ilustrar este concepto, se contrastan tres hipotéticos clasificadores en la gráfica de la curva ROC. En primer lugar, la línea diagonal desde la esquina inferior izquierda a la superior derecha del diagrama representa un clasificador sin valor de predicción. Este tipo de clasificador detecta los verdaderos positivos y los falsos positivos exactamente al mismo ritmo, lo que implica que el clasificador no puede discriminar entre ambos. Esta es la línea de base por la que se puede juzgar a otros clasificadores; las curvas ROC que caen cerca de esta línea indican modelos que no son muy útiles. De manera similar, el clasificador perfecto tiene una curva que pasa por el punto a una tasa de 100% de verdaderos positivos y 0% de falsos positivos. Es capaz de identificar correctamente todos los verdaderos positivos antes de clasificar incorrectamente cualquier resultado negativo. La mayoría de los clasificadores del mundo real son similares al clasificador de prueba; caen en algún lugar de la zona entre perfecto e inútil. Así, cuanto más cerca esté la curva del clasificador perfecto, mejor será la identificación de los valores positivos. Esto puede medirse usando una estadística conocida como el área bajo la curva ROC (**AUC**). El AUC trata el diagrama ROC como un cuadrado bidimensional y mide el área total bajo la curva ROC, yendo así desde 0,5 (para un clasificador sin valor predictivo), hasta 1.0 (para un clasificador perfecto). Una convención para interpretar las puntuaciones de la AUC es el siguiente sistema:

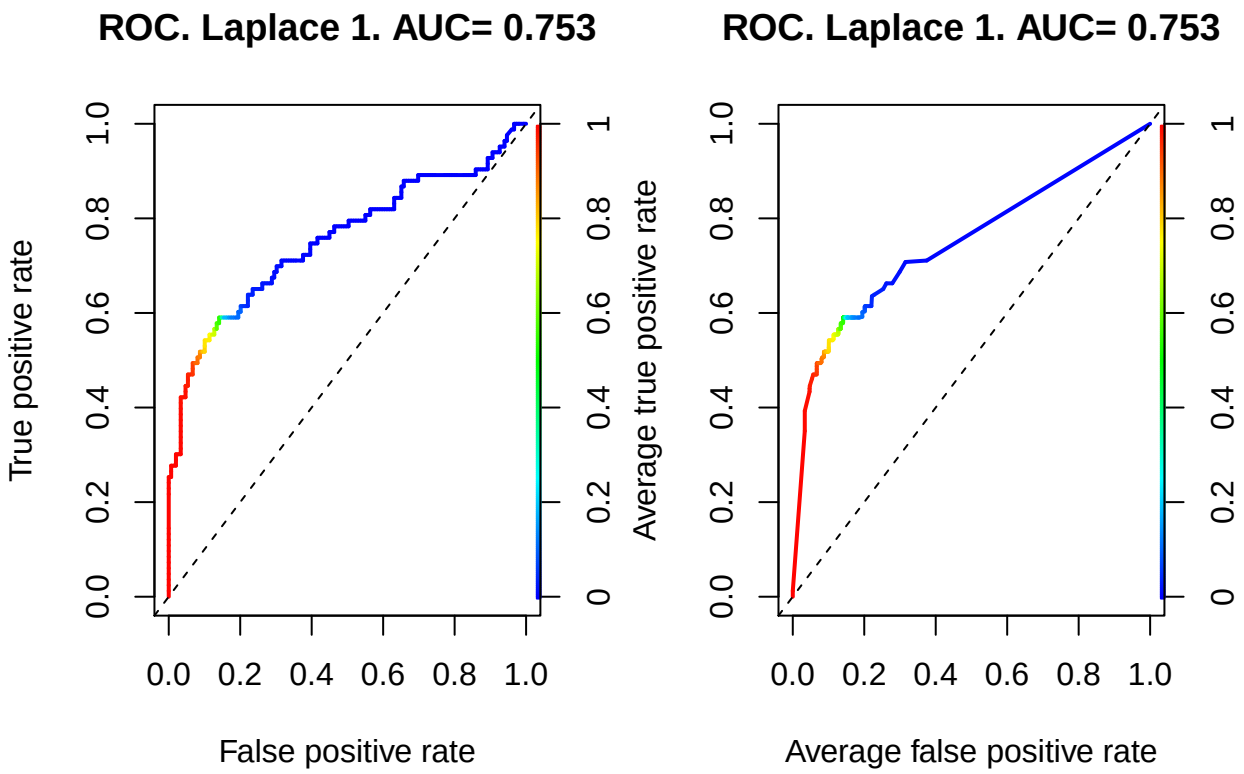
- 0.9 – 1.0 = A (Sobresaliente)
- 0.8 – 0.9 = B (Excelente/Bueno)
- 0.7 – 0.8 = C (Aceptable/Justo)
- 0.6 – 0.7 = D (Pobre)
- 0.5 – 0.6 = E (No hay discriminación)

Para cada modelo creado se realiza la curva ROC y se muestra además el AUC. Para la creación de curvas ROC se recurre al paquete **ROCR**, que permite la construcción de un objeto de rendimiento para el objeto de predicción calculado previamente mediante la función *prediction()*. Puesto que las curvas ROC trazan verdaderas tasas positivas frente a tasas falsas positivas, simplemente se llama a la función *performance()* especificando las medidas tpr y fpr. Con el objeto resultante de *performance()*, se visualiza la curva ROC con la función *plot()*. Se traza la diagonal mediante la función *abline()* fijando la intercepción en cero y considerando una pendiente de uno, y se especifica también que la recta sea discontinua.

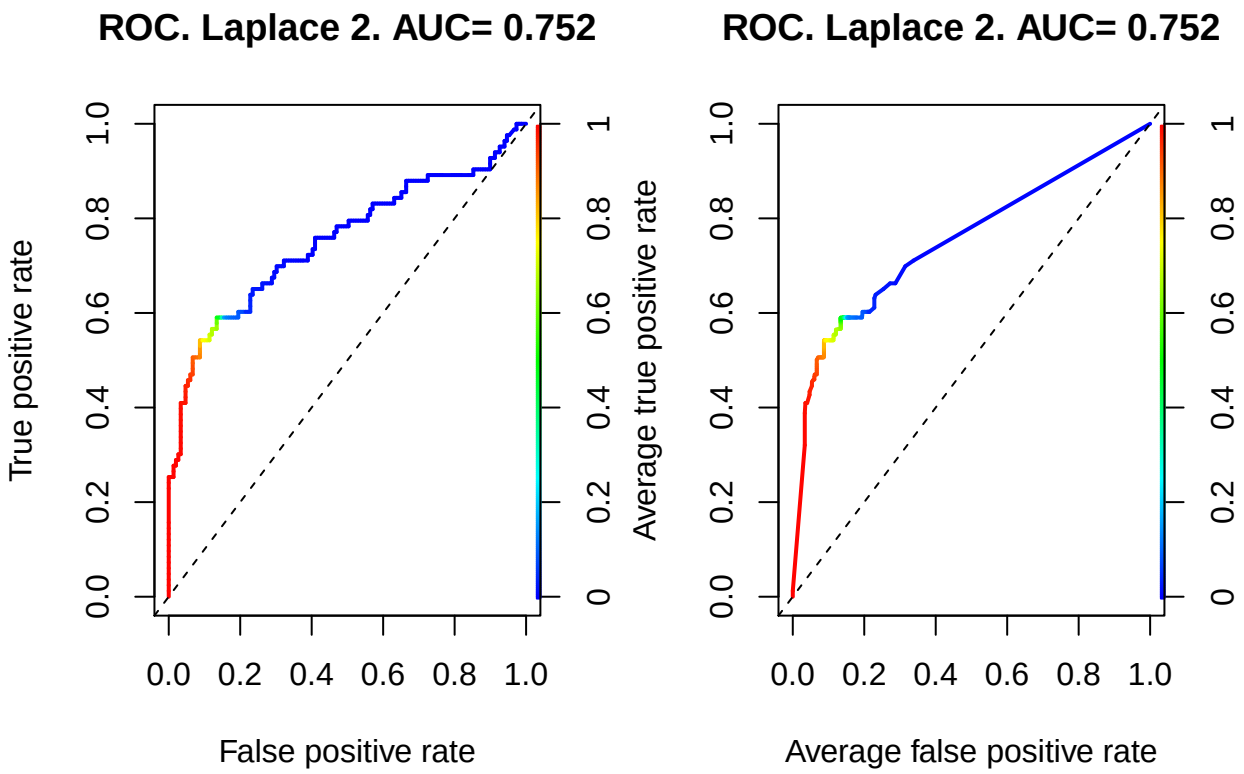
```
library("ROCR")
par(mfrow=c(1,2))
Pred.Prob<-predict(Flower.Bayes, Genotypes.Test, type = "raw")
Pred.Prob<-as.data.frame(Pred.Prob)
pred <- prediction(predictions= Pred.Prob$Slow, labels= Flower.Test$Flowering)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
perf.auc <- performance(pred, 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)
```




```
par(mfrow=c(1,2))
Pred.Prob<-predict(Flower.Bayes2, Genotypes.Test, type = "raw")
Pred.Prob<-as.data.frame(Pred.Prob)
pred <- prediction(predictions= Pred.Prob$Slow, labels= Flower.Test$Flowering)
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)
```




```
par(mfrow=c(1,2))
Pred.Prob<-predict(Flower.Bayes3, Genotypes.Test, type = "raw")
Pred.Prob<-as.data.frame(Pred.Prob)
pred <- prediction(predictions= Pred.Prob$Slow, labels= Flower.Test$Flowering)
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 2. 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 2. AUC=", round(perf.auc,3)))
abline(a = 0, b = 1, lwd = 1, lty = 2)
```



Como se puede observar, la ROC mejora al utilizar el estimador de Laplace, y esto no solamente se observa en la gráfica, sino que se manifiesta en el valor del AUC, que aumenta en una centésima aproximadamente.

Bibliografía

Lantz, Brett. 2015. *Machine Learning with R*. Packt Publishing Ltd. http://www.packtpub.com/books/content/machine-learning-r.