

Naive Bayes - Clasificadores Bayesianos (R)

Los *naive Bayes* asumen que la presencia de una variable particular en una clase **no** está relacionada con la presencia de ninguna otra variable.

Clasifican a la nueva información basándose en la probabilidad de pertenecer a cada una de ellas

Escenario del problema

Una empresa de coches ha sacado un nuevo modelo al mercado. Le ha preguntado a una red social quién ha comprado el producto, recaudando el sexo, la edad y el salario de cada uno de ellos.

Ahora queremos construir un modelo que nos permita determinar con estos atributos si la persona comprará el producto o no, para tomar medidas en función de la respuesta para que lo acabe comprando. ¡Vamos a ello!

```
# 1. Importar librerías
library(caTools)
library(ggplot2)
library(e1071)           # Nos va a permitir general el modelo de Bayes
library(ElemStatLearn)   # Nos va a permitir dibujar las clasificaciones
```

```
# 2. Importar datos
datos <- read.csv('../Datos/4.2.Compras.csv')
datos <- datos[3:5] # Eliminamos la columna del sexo
# 2.1. Codificar la respuesta como categórica (factores en R)
datos$Compra <- factor(datos$Compra,
                        levels = c(0, 1),
                        labels = c(0, 1))
head(datos, 5)
```

```
##   Edad Salario Compra
## 1   19   19000      0
## 2   35   20000      0
## 3   26   43000      0
## 4   27   57000      0
## 5   19   76000      0
```

Compra puede ser 0 (compró) o 1 (no compró) → Distribución binomial (Bernouilli)

```
# 3. Separar en Entrenamiento y Validación
set.seed(123)
split <- sample.split(datos$Compra, SplitRatio = 0.75)
train <- subset(datos, split==TRUE)
test  <- subset(datos, split==FALSE)
dim(train)/dim(test)
```

```
## [1] 3 1
```

```
# 4. Hacer las predicciones para el conjunto de Validación
train[-3] <- scale(train[-3])
test[-3]  <- scale(test[-3])
```

```
# 5. Construir el Modelo y Predecir (kNN lo hace todo junto)
bayesiano <- e1071::naiveBayes(x = train[-3],
                               y = train$Compra)
```

```

# 6. Construir el Modelo y Predecir (kNN lo hace todo junto)
y_pred <- predict(bayesiano, newdata = test)

# 7. Hacer la matriz de confusión
cm = table(test[, 3], y_pred)
cm

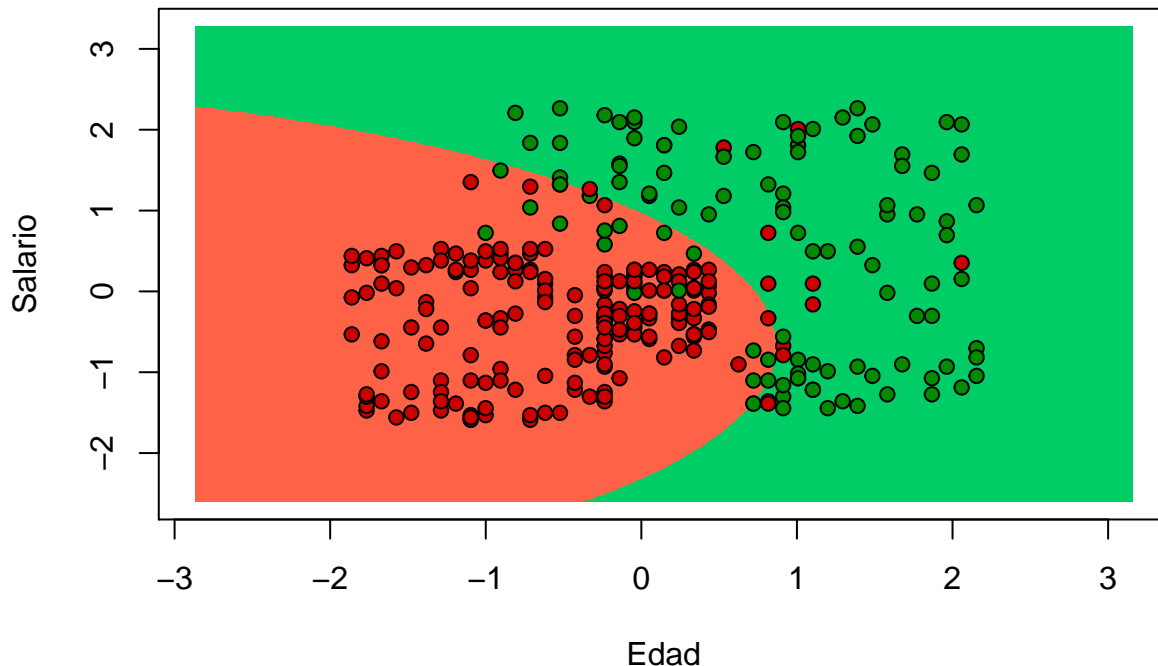
##      y_pred
##      0  1
## 0 57  7
## 1  7 29

# 8. Echemos un vistazo a la pinta que tienen las predicciones
# 8.1. Conjunto de entrenamiento
set = train
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Edad', 'Salario')

y_grid = predict(bayesiano, newdata = grid_set)
plot(set[, -3],
      main = 'Bayesiano (Conjunto de entrenamiento)',
      xlab = 'Edad', ylab = 'Salario',
      xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))

```

Bayesiano (Conjunto de entrenamiento)



```

# 8.2. Conjunto de validación
set = test

```

```

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.01)
grid_set = expand.grid(X1, X2)
colnames(grid_set) = c('Edad', 'Salario')

y_grid = predict(bayesiano, newdata = grid_set)
plot(set[, -3],
      main = 'Bayesiano (Conjunto de entrenamiento)',
      xlab = 'Edad', ylab = 'Salario',
      xlim = range(X1), ylim = range(X2))
contour(X1, X2, matrix(as.numeric(y_grid), length(X1), length(X2)), add = TRUE)
points(grid_set, pch = '.', col = ifelse(y_grid == 1, 'springgreen3', 'tomato'))
points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))

```

Bayesiano (Conjunto de entrenamiento)

