TP1: Clasificación binaria - Predicción de la comestibilidad de hongos

Micaela Oliva, Camila Bernardez

2024-08-28

Ejercicio 1: Introducción al problema

El dataset elegido tiene como origen:

https://www.kaggle.com/datasets/vishalpnaik/mushroom-classification-edible-or-poisonous?resource=download&select=mushroom.csv

Es un dataset que busca clasificar si un hongo es comestible (edible) o no (poisonous). Para considerarlo, el dataset esta compuesto de 16 variables, de las cuáles:

- `class` (variable categórica binaria): indica si un hongo es comestible o no, y es lo que buscamos predecir.
  - edible
  - poisonous
- `cap-diameter` (variable numérica): indica el diametro del sombrero del hongo en cm.
- `cap-shape` (variable categórica): indica el forma del sombrero del hongo.
  - 'bell'
  - 'conical'
  - 'convex'
  - 'flat'
  - 'sunken'
  - 'spherical'
  - 'others'
- `cap-surface` (variable categórica): indica la textura de la superficie del sombrero del hongo.
  - 'fibrous'
  - 'grooves'
  - 'scaly'
  - 'smooth'
  - 'dry'
  - 'shiny'
  - 'leathery'
  - 'silky'
  - 'sticky'
  - 'wrinkled'
  - 'fleshy'
  - '' #
- `cap-color` (variable categórica): indica el color del sombrero del hongo.
  - 'brown'

  - 'orange'
  - 'buff'
  - 'gray'
  - 'green'
  - 'pink'
  - 'purple'
  - 'red'
  - 'white'
  - 'yellow'
  - 'blue'
  - 'black'

- `does-bruise-or-bleed` (variable categórica binaria -> true/false): indica si el hongo al lesionarse presenta moratones o sangrado.
  - 'true'
  - 'false'

- `gill-attachment` (variable categórica): indica cómo las láminas del hongo se adhieren al pie.
  - 'adnate'
  - 'adnexed'
  - 'decurrent'
  - 'free'
  - 'sinuate'
  - 'pores'
  - 'none' #
  - '' #

- `gill-spacing` (variable categórica): indica la separación entre las láminas del hongo.
  - 'close'
  - 'distant'
  - 'none' #
  - '' #

- `stem-heigh` (variable numérica): indica la altura del pie del hongo en cm.
- `stem-width` (variable numérica): indica el ancho del pie del hongo en mm.
- `stem-root` (variable categórica): indica la estructura de la raíz del pie del hongo.
  - 'bulbous'
  - 'swollen'
  - 'club'
  - 'cup' #extra
  - 'equal' #extra
  - 'rhizomorphs'#extra
  - 'rooted'
  - '' #falta es f, supongo que es none
  - '' #

- `veil-type` (variable categórica): indica el tipo de velo que cubre las láminas del hongo.
  - 'partial' #extra
  - 'universal'
  - '' #

- `has-ring` (variable categórica binaria -> true/false): indica si esta presente un anillo en el hongo o no.
  - 'true'

  - 'false'

- `ring-type` (variable categórica): indica el tipo del anillo presente en el hongo.
  - 'cobwebby' #extra
  - 'evanescent'
  - 'flaring'
  - 'grooved'
  - 'large'
  - 'pendant'
  - 'sheathing', #extra
  - 'zone'
  - 'scaly' #extra
  - 'movable'
  - 'none'
  - 'unknown' #extra
  - ''

- `habitat` (variable categórica): indica el ambiente en el cual el hongo fue encontrado.
  - 'grasses'
  - 'leaves'
  - 'meadows'
  - 'paths'
  - 'heaths'
  - 'urban'
  - 'waste'
  - 'woods'

- `season` (variable categórica): indica la estación en la cual el hongo es comunmente observado.
  - 'spring'
  - 'summer'
  - 'autumn'
  - 'winter'

Para resumir, la variable class es una variable categórica binaria que indica si un hongo es comestible (edible) o venenoso (poisonous) pra poder hacer una clasificación binaria. Asimismo, el conjunto de datos tiene varias variables categóricas (cap-shape, cap-surface, cap-color, entre otras) y variables numéricas como cap-diameter, stem-heigh, y stem-width, cumpliendo así la inclusión de ambos tipos de atributos. Las variables categóricas están representadas como palabras (e.g., cap-shape con valores como 'bell', 'conical', etc.). Además, hicimos un preprocesamiento de los datos tal que tengamos menos de 50.000 observaciones, pues el dataset original consistía de más de 60K de datos, y que además no tengan valores nulos. Usamos árboles de decisión porque son capaces de manejar tanto variables categóricas como numéricas, permitiendo identificar de manera eficaz patrones complejos y no lineales en los datos. Además, los árboles de decisión pueden dividir los datos en función de múltiples atributos, lo que es útil cuando tenemos varias características relevantes, como en este caso, para clasificar hongos entre comestibles y venenosos.

Ejercicio 2: Preparación de los datos

Carga del conjunto de datos y realizamiento del preprocesamiento

```
mushrooms <- read.csv('mushrooms_small.csv')

table(is.na(mushrooms)) #verificamos que no hay valores nulos
```

```
##
##  FALSE
```

```
## 967449
```

```
nrow(mushrooms) #verificamos que hay menos de 50.000 observaciones
```

```
## [1] 46069
```

Como los datos ya se encuentran sampleados (menos de 50.000 observaciones) y sin valores null, para preprocesarlos encargaremos de: * cambiar los valores de las variables categóricas por palabras enteras en vez de letras. * quitar columnas que dependen de otras.

La fuente del dataset especifica que las columnas 'gill-color', 'stem-surface', 'stem-color', 'veil-color' y 'spore-print-color' dependen de 'cap-color' y 'cap-surface', así que decidimos eliminar estas columnas correlacionadas para disminuir al máximo posible el ruido, que podría distorsionar nuestra clasificación binaria.

```
mushrooms <- mushrooms[, !(names(mushrooms) %in% c("gill.color", "stem.surface", "stem.color", "veil.co
ncol(mushrooms)
```

```
## [1] 16
```

Las variables categóricas toman valores de letras que pueden ser confusos, ya que no siempre reflejan de manera clara la palabra que representan (por ejemplo 'k' se utiliza en lugar de 'black'), y pueden repetirse entre columnas sin necesariamente significar lo mismo. Para que el manejo de los datos y las leyendas de los gráficos sean más claras decidimos reemplazarlas por las palabras correspondientes.

```
head(mushrooms_renamed)
```

```
##       class cap.diameter cap.shape cap.surface cap.color does.bruise.or.bleed
## 1 poisonous        15.26    convex     grooves    orange                false
## 2 poisonous        16.60    convex     grooves    orange                false
## 3 poisonous        14.07    convex     grooves    orange                false
## 4 poisonous        14.17      flat       shiny       red                false
## 5 poisonous        15.34    convex     grooves    orange                false
## 6 poisonous        12.85      flat     grooves    orange                false
##   gill.attachment gill.spacing stem.height stem.width stem.root veil.type
## 1            free         none       16.95      17.09   swollen universal
## 2            free         none       17.99      18.19   swollen universal
## 3            free        close       17.80      17.74   swollen universal
## 4            free         none       15.77      15.98   swollen universal
## 5            free      distant       17.84      18.79   swollen universal
## 6            free         none       17.27      18.69   swollen universal
##   has.ring ring.type habitat season
## 1     true   grooved   woods winter
## 2     true   grooved   woods summer
## 3     true   grooved   woods winter
## 4     true   pendant   woods winter
## 5     true   pendant   woods summer
## 6     true   pendant   woods autumn
```

Análisis exploratorio de datos: Estadísticas descriptivas y visualizaciones de las variables principales

Inicialmente buscamos obtener estadísticas sencillas con el método `summary`, pero primero veamos de qué tipos de datos son nuestras variables. Encontramos solo dos tipos de datos en nuestro dataset: 3 variables de tipo `numeric` **cap.diameter**, **stem.height** y **stem.width** y 13 categóricas de tipo `character`. Además, 3 de las 13 categóricas son binarias, incluyendo a nuestra variable a predecir **class**, que especifica si el hongo en cuestión es o no comestible ('edible' o 'poisonous').

```
str(mushrooms_renamed)
```

```
## 'data.frame':    46069 obs. of  16 variables:
```

```
##  $ class                : chr  "poisonous" "poisonous" "poisonous" "poisonous" ...
##  $ cap.diameter         : num  15.3 16.6 14.1 14.2 15.3 ...
##  $ cap.shape            : chr  "convex" "convex" "convex" "flat" ...
##  $ cap.surface          : chr  "grooves" "grooves" "grooves" "shiny" ...
##  $ cap.color            : chr  "orange" "orange" "orange" "red" ...
##  $ does.bruise.or.bleed: chr  "false" "false" "false" "false" ...
##  $ gill.attachment      : chr  "free" "free" "free" "free" ...
##  $ gill.spacing         : chr  "none" "none" "close" "none" ...
##  $ stem.height          : num  16.9 18 17.8 15.8 17.8 ...
##  $ stem.width           : num  17.1 18.2 17.7 16 18.8 ...
##  $ stem.root            : chr  "swollen" "swollen" "swollen" "swollen" ...
##  $ veil.type            : chr  "universal" "universal" "universal" "universal" ...
##  $ has.ring             : chr  "true" "true" "true" "true" ...
##  $ ring.type            : chr  "grooved" "grooved" "grooved" "pendant" ...
##  $ habitat              : chr  "woods" "woods" "woods" "woods" ...
##  $ season               : chr  "winter" "summer" "winter" "winter" ...
```

Comenzamos por analizar la distribución de algunas variables. En primer lugar nos interesa análizar si tenemos una proporción más o menos pareja de los valores que toma **class**. Siendo que hay aproximadamente 44 observaciones de 'edible' para cada 55 observaciones de 'poisonous' nos aseguramos una buena distribución de valores para que el modelo tenga suficientes ejemplos de ambas clases para aprender y generalizar adecuadamente.

```
num_edible <- sum(mushrooms_renamed$class == 'edible')
num_poisonous <- sum(mushrooms_renamed$class == 'poisonous')

prop_edible <- num_edible/(nrow(mushrooms_renamed))
prop_poisonous <- num_poisonous/(nrow(mushrooms_renamed))

print(paste("Edible:", num_edible, "(",round(prop_edible * 100, 2), "%)"))
```

```
## [1] "Edible: 20516 ( 44.53 %)"
```

```
print(paste("Poisonous:", num_poisonous, "(",round(prop_poisonous * 100, 2), "%)"))
```

```
## [1] "Poisonous: 25553 ( 55.47 %)"
```

MAS VARIABLES CATEGORICAS (histogramas?)

```
selected_columns <- mushrooms_renamed[, c("cap.diameter", "stem.height", "stem.width")]
summary(selected_columns)
```

```
##   cap.diameter     stem.height      stem.width
##  Min.   : 0.380   Min.   : 0.00   Min.   :  0.00
##  1st Qu.: 3.480   1st Qu.: 4.63   1st Qu.:  5.19
##  Median : 5.860   Median : 5.95   Median : 10.16
##  Mean   : 6.709   Mean   : 6.57   Mean   : 12.14
##  3rd Qu.: 8.540   3rd Qu.: 7.73   3rd Qu.: 16.53
##  Max.   :62.340   Max.   :33.92   Max.   :103.91
```

BOXPLOT?

Por último nos interesa analizar la correlación entre las variables, principalmente para saber si hay algunas variables más correlacionadas con nuestra variable objetivo que otras. Esto significaría que es posible que un modelo menos complejo (de menos variables) sea suficiente para predecirla.

```
#install.packages("PerformanceAnalytics")
library("PerformanceAnalytics")
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

##
## ############################ Warning from 'xts' package ###########################
## #                                                                                 #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                            #
## #                                                                                 #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                                 #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                                 #
## ###################################################################################

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend
```

```
mushrooms_renamed[] <- lapply(mushrooms_renamed, function(x) if (is.character(x)) as.factor(x) else x)
mushrooms_numeric <- data.frame(lapply(mushrooms_renamed, function(x) if (is.factor(x)) as.numeric(x) el

#chart.Correlation(mushrooms_numeric, histogram=TRUE, pch=19)
```

Ejercicio 3: Construcción de un árbol de decisión básico

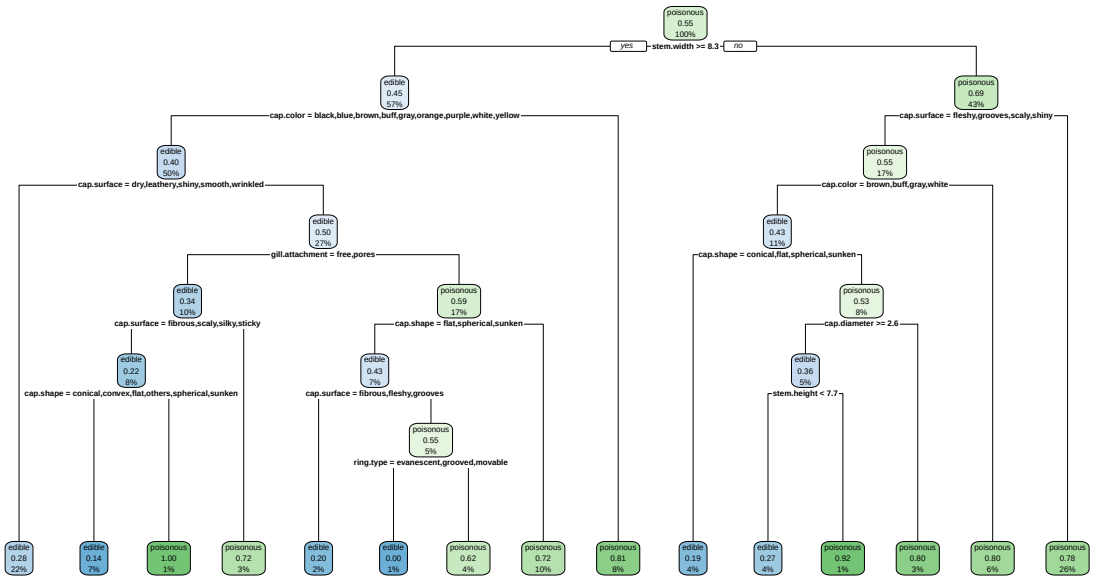
```
mushrooms_renamed$cap.shape <- as.factor(mushrooms_renamed$cap.shape)
mushrooms_renamed$cap.surface <- as.factor(mushrooms_renamed$cap.surface)
mushrooms_renamed$cap.color <- as.factor(mushrooms_renamed$cap.color)
mushrooms_renamed$gill.attachment <- as.factor(mushrooms_renamed$gill.attachment)
mushrooms_renamed$gill.spacing <- as.factor(mushrooms_renamed$gill.spacing)
mushrooms_renamed$stem.root <- as.factor(mushrooms_renamed$stem.root)
mushrooms_renamed$ring.type <- as.factor(mushrooms_renamed$ring.type)
mushrooms_renamed$veil.type <- as.factor(mushrooms_renamed$veil.type)
mushrooms_renamed$habitat <- as.factor(mushrooms_renamed$habitat)
mushrooms_renamed$season <- as.factor(mushrooms_renamed$season)
```

```r
mushrooms_renamed$class <- as.factor(mushrooms_renamed$class)
mushrooms_renamed$does.bruise.or.bleed <- as.factor(mushrooms_renamed$does.bruise.or.bleed)
mushrooms_renamed$has.ring <- as.factor(mushrooms_renamed$has.ring)
```

```r
library(dplyr)
set.seed(42) #semilla para replicabilidad

n <- nrow(mushrooms_renamed)
indices <- sample(1:n) #'mezclamos' primero el orden de las filas, para luego solo tomar de la 1:70%, 7
train_size <- floor(0.7 * n)
valid_size <- floor(0.15 * n)

train_indices <- indices[1:train_size]
valid_indices <- indices[(train_size + 1):(train_size + valid_size)]
test_indices <- indices[(train_size + valid_size + 1):n]

#creamos los subsets de datos
train_set <- mushrooms_renamed[train_indices, ]
valid_set <- mushrooms_renamed[valid_indices, ]
test_set <- mushrooms_renamed[test_indices, ]
```

```r
library(rpart)

tree_model <- rpart(class ~ .,
                    data = train_set,
                    method = "class")
```

```r
library(rpart.plot)

rpart.plot(tree_model)
```



Ejercicio 4: Evaluación del árbol de decisión básico

Matriz de confusión

```
# Instalacion y/o importe de las librerias
# install.packages('caret')
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
# Predicciones de probabilidades y de clases
pred_prob <- predict(tree_model, test_set, type = "prob")  # Probabilidades predichas
pred_class <- predict(tree_model, test_set, type = "class")  # Clases predichas
```

Hacemos la Matriz de Confusión:

```
conf_matrix <- confusionMatrix(pred_class, test_set$class)
print("Matriz de Confusión:")
```

```
## [1] "Matriz de Confusión:"
```

```
print(conf_matrix$table)
```

```
##            Reference
## Prediction  edible poisonous
##   edible      2075       613
##   poisonous   1008      3215
```

```r
print(t(conf_matrix$table)) # traspongo para que sea como visto en clase
```

```
##            Prediction
## Reference   edible poisonous
##   edible      2075      1008
##   poisonous    613      3215
```

Lo que obtenemos es:

- $TP = 2075$: Es un True Positive que dice la cantidad de instancias donde el modelo predijo `edible` y la clase verdaderamente valia edible.
- $FP = 613$: Es un False Positive que dice la cantidad de instancias donde el modelo predijo edible, pero la clase verdaderamente valia poisonous.
- $FN = 1008$: Es un False Negative que dice la cantidad de instancias donde el modelo predijo poisonous, pero la clase verdaderamente valia edible.
- $TN = 3215$: Es un True Negative que dice la cantidad de instancias donde el modelo predijo poisonous y la clase clase verdaderamente valia poisonous.

Por lo tanto, usamos Accuracy como una metrica de evaluacion si el resultado fue bueno:

Accuracy

Buscamos ver la proporcion de predicciones correctas (tanto positivas como negativas) de las predicciones totales.

$$
\begin{aligned}
Accuracy = \frac{TP + TN}{P + N} = \\
= \frac{TP + TN}{TP + FN + FP + TN} = \\
= \frac{TP + TN}{TP + FN + FP + TN} = \\
= \frac{2075 + 3215}{2075 + 1008 + 613 + 3215} = \\
= \frac{5290}{6911} = 0.7654464
\end{aligned}
$$

En R:

```r
#
TP <- conf_matrix$table[1, 1]
FP <- conf_matrix$table[1, 2]
FN <- conf_matrix$table[2, 1]
TN <- conf_matrix$table[2, 2]

# Calculamos el accuracy
Accuracy <- (TP + TN) / (TP + FN + FP + TN)
Accuracy
```

```
## [1] 0.7654464
```

```r
# Validacion
conf_matrix$overall['Accuracy']
```

```
##  Accuracy
## 0.7654464
```

Esto significa que el modelo clasifica correctamente alrededor de 76.5% de las instancias.

Lo que significa que en este caso $Errorrate = 1 - Accuracy = 0.2345536$.

Pero para considerar que las clases pueden llegar a estar desbalanceadas, usaremos Precision-Recall:

Precision and Recall

- Precision:

  Buscamos ver la proporcion de las instancias predichas de edible que verdaderamente valen edible.

$$Precision_{edible} = \frac{TP}{TP + FP} =$$

$$= \frac{2075}{2075 + 613} = 0.7719494$$

  En R:

```
Precision <- TP/(TP + FP)
Precision
```

```
## [1] 0.7719494
```

```
# Validacion
conf_matrix$byClass['Pos Pred Value']
```

```
## Pos Pred Value
##      0.7719494
```

  Esto significa que alrededor de 77.2% del tiempo cuando el modelo predice edible acierta.

- Recall:

  Buscamos ver la proporcion de las instancias que verdaderamente valen edible que fueron correctamente predichas como edible.

$$Recall_{edible} = \frac{TP}{TP + FN} =$$

$$= \frac{2075}{2075 + 1008} = 0.6730457$$

  En R:

```
Recall <- TP/(TP + FN)
Recall
```

```
## [1] 0.6730457
```

```
# Validacion
conf_matrix$byClass['Sensitivity']
```

```
## Sensitivity
##   0.6730457
```

  Esto significa que alrededor de 67.3% el modelo identifica correctamente si los hongos son comestibles.

Pero, queremos una ponderacion conjunta de Precision y de Recall, por lo cual usaremos F1-Score.

F1-Score

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} =$$

$$= \frac{2 * 0.7719494 * 0.6730457}{0.7719494 + 0.6730457} = 0.7191128$$

En R:

```
F1_score <- (2 * Precision * Recall)/(Precision + Recall)
F1_score
```

```
## [1] 0.7191128
```

Esto significa que el accuracy del modelo en predicir la clase edible es igual a 0.7191128.

ROC-AUC Score

Buscamos ver que tan bien el modelo tiene la capacidad de separar las clases.

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
# Calculamos ROC curve
roc_curve <- roc(test_set$class, pred_prob[, "edible"])
```

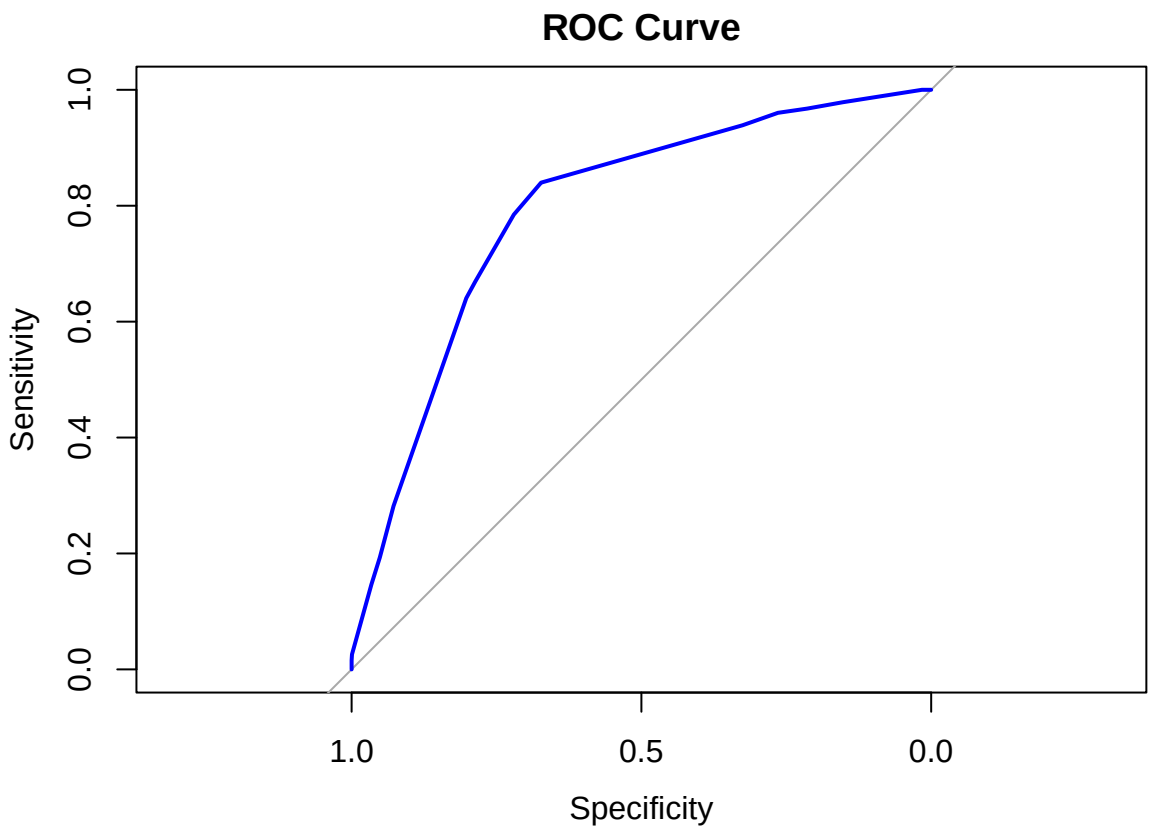
```
## Setting levels: control = edible, case = poisonous
```

```
## Setting direction: controls > cases
```

```
#  AUC score
auc_value <- auc(roc_curve)
print(paste("AUC-ROC Score:", auc_value))
```

```
## [1] "AUC-ROC Score: 0.79219154760779"
```

```
# Plot de ROC curve
plot(roc_curve, main = "ROC Curve", col = "blue", lwd = 2)
```



$$AUC - ROC = 0.7921915$$

Esto significa que como $0.7921915 > 0.5$, el modelo se ejecuta mejor que hacer random guessing.