TFM: Análisis predictivo de incidentes navales en EEUU, 2002 - 2015

Anexo 5.2. Modelado: MergedActivity

Oscar Antón

diciembre de 2023

Carga de librerías, funciones y datos

```
# Librería                          # Propósito
library(MASS)                       # Regresión ordinal
library(nnet)                       # Regresión multinomial
library(sampling)                   # Equilibrado de muestra. Método del cubo.
library(DMwR)                       # Equilibrado de muestra. Método Smote.
library(mice)                       # Imputación de valores ausentes.
library(arulesCBA)                  # Discretización de variables (Redes bayesianas)
library(fastDummies)                # Variables Dummy (One hot encoding)

library(caret)                      # Modelos de machine Learning
library(keras)                      # API para redes neuronales
library(MLmetrics)                  # Metricas en caret para variables multiclase (>2)
library(gbm)                         # Manejo de modelos Gradient Boosting. Debido a error va
rImp()
library(pROC)                       # Performance de modelos (curva ROC)
library(h2o)                        # Machine learning framework (Java)
library(doParallel)                 # Cómputo multihilo
library(tictoc)                     # Benchmarking (tiempo de cómputo)

library(DALEX)                      # Interpretabilidad de modelos ML
library(iBreakDown)                 # Explicatividad local
library(modelStudio)                # Análisis interactivo de explicabilidad

library(gridExtra)                  # Manejo de gráficos
library(kableExtra)                 # Formato de tablas
library(formattable)                # Formato de tablas
library(ggpubr)                     # Visualización de datos (ggarrange)
library(data.table)
library(tidyverse)                   # Sintaxis para el manejo de datos. Incluye dplyr, ggplo
t2, etc.

source("../4.Functions/myCustomFunctions.R")
```

```
# Cargar el dataframe MergedActivity (100% incidentes)
# Se lee como dataframe en vez de como datatable para evitar errores
MergedActivity <- as.data.frame(readRDS("../1.DataPreprocess/DataMergedActivity/MergedActi
vity.rds"))
```

Switches

```
# Guardar datos o no
save_switch <- 0
```

---

1. Creación de datasets para los modelos

1.1. Dataset con variables numéricas y factor (General)

· Criba de variables

· Creación de la variable objetivo: “y” (categoría de incidente). Será la última variable del dataset

· Reducción de variabilidad en variables categóricas

· Escalado para variables numéricas

```
# Adaptación de variables:
#  Obviar variables identificativas y de localización
#  Obviar otras variables con información no relevante para el análisis predictivo
#  Renombrado
#  Conversión de variables de fecha, hora y año a valores continuos
#  Reducir variabilidad de ciertas variables discretas (lump_factorials)
#  Convertir a factor el resto de variables discretas
#  Se escalarán las variables numéricas después del equilibrado e imputación de NAs, no ah
ora.

MergedActivity <- MergedActivity %>%
  select(-vessel_id, -imo_number, -vessel_name) %>%
  select(-event_type, -build_year, -wave_hgt, -visibility, -casualty, -pollution) %>%
  select(-flag_abbr, -classification_society, -solas_desc) %>%

  rename(vessel_length = length) %>%
  rename(y = event_class) %>%

  mutate(date = yday(as.Date(date))) %>%
  mutate(hour = round(as.numeric(sub(":.*", "", hour)) + (as.numeric(sub(".*:", "", hour))
/ 60), 2)) %>%

  mutate_at(vars(vessel_class), lump_factorials) %>%

  mutate_at(vars(region, watertype, damage_status, y), factor)

# Visualización de la estructura
str(MergedActivity)
```

```
## 'data.frame':    68000 obs. of  16 variables:
##  $ activity_id      : int  1475897 1475897 1477008 1477373 1477402 1484262 1485352 1485
352 1598058 1475186 ...
##  $ date             : num  1 1 1 1 1 1 1 1 1 2 ...
##  $ hour             : num  3.75 3.75 13.88 18.17 10 ...
##  $ region           : Factor w/ 6 levels "Alaska","Canada",..: 5 5 3 5 4 5 5 5 4 4 ...
##  $ latitude         : num  37 37 39.3 31.5 30.6 ...
##  $ longitude        : num  -88.3 -88.3 -76.4 -88 -88 ...
##  $ watertype        : Factor w/ 2 levels "ocean","river": 2 2 1 2 1 2 2 2 1 1 ...
##  $ damage_status    : Factor w/ 5 levels "Actual Total Loss",..: 5 5 5 2 5 5 2 2 5 2
...
##  $ vessel_class     : Factor w/ 11 levels "Barge","Bulk Carrier",..: 1 10 3 10 1 1 1 1
2 8 ...
##  $ age              : int  21 21 29 20 6 5 21 21 25 39 ...
##  $ gross_ton        : int  1065 932 19 227 764 823 888 888 38412 13 ...
##  $ vessel_length    : num  200 135.6 38.2 78.8 200 ...
##  $ air_temp         : num  -38.5 -38.5 -17.1 11 41.2 ...
##  $ wind_speed       : num  NA NA 66.2 NA 85.7 ...
##  $ damage_assessment: int  100 100 5600 5600 1000 95000 95000 10000 10000 40000 ...
##  $ y                : Factor w/ 5 levels "Critical Events",..: 2 2 5 3 2 2 2 4 2 5 ...
```

1.1.1. Equilibrado de variable objetivo

```r
# Verificación del equilibrado de la muestra
table(MergedActivity$y)
```

```
##
##     Critical Events  Maritime Accidents     Material Issues Onboard Emergencies
##               16938               18467               17158                6630
## Third-party Damages
##                8807
```

A efectos del análisis predictivo, se van a balancear los niveles de la variable objetivo a 9000 observaciones por nivel.

· Submuestreos (Cube) para: Critical Events, Maritime Accidents, Material Issues

· Sobremuestreos (smote) para: Onboard Emergencies, Third-party Damages

```r
# Tamaño de la muestra a la que queremos llegar en cada nivel
n = 9000
```

Submuestreos: Método del cubo

Variables significativas

```r
# También para una regresión ordinal
if (save_switch  == 1) {
# Establecemos un modelo de regresión ordinal para "y" puesto que se puede establecer un o
rden en sus valores
multi_model <- multinom(y ~ ., na.omit(MergedActivity), Hess = TRUE)
# Modelo de selección de variables por pasos con el criterio de Información de Akaike
multi_model_stp <- stepAIC(multi_model, direction = "both")
# Guardado
saveRDS(multi_model , "Models/multi_model.RDS")
saveRDS(multi_model_stp , "Models/multi_model_stp.RDS")
}else{
  multi_model_stp <- readRDS("Models/multi_model_stp.rds")
}
# Resultados. Variables relevantes en el modelo
multi_model_stp$terms[[3]]
```

```
## activity_id + date + hour + region + latitude + watertype + damage_status +
##     vessel_class + age + gross_ton + vessel_length + air_temp +
##     wind_speed + damage_assessment
```

Submuestreo en Critical Events

```r
# Se eliminarán observaciones con NA
CriticalEvents <- MergedActivity %>%
  filter(y == "Critical Events") %>%
  filter(complete.cases(.)) %>%
  mutate(across(where(is.factor), droplevels))
```

```r
# Para comprobar la estimación del tamaño poblacional,
# creamos un vector de "1" de dimensión = número de observaciones del nivel predominante
UNO = rep(1, nrow(CriticalEvents))

# Se necesita que todas las variables sean numéricas
# Variables cuantitativas
X1 <- CriticalEvents %>%
   select(activity_id, hour, longitude, age, gross_ton, vessel_length, air_temp, wind_spee
d)

# Variables cualitativas: One hot encoding
X2 <- disjunctive(CriticalEvents$region)
colnames(X2) <- levels(CriticalEvents$region)

X3 <- disjunctive(CriticalEvents$watertype)
colnames(X3) <- levels(CriticalEvents$watertype)

X4 <- disjunctive(CriticalEvents$vessel_class)
colnames(X4) <- levels(CriticalEvents$vessel_class)

X5 <- disjunctive(CriticalEvents$damage_status)
colnames(X5) <- levels(CriticalEvents$damage_status)

# Juntamos todo para formar la matriz de diseño
X = as.matrix(cbind(UNO, X1, X2, X3, X4, X5))

# Probabilidades de inclusión
pik = rep(n / nrow(CriticalEvents), nrow(CriticalEvents))

# Obtención de los índices de la nueva muestra con ayuda de la librería sampling
# method = 2 para fase de aterrizaje mediante supresión de variables
# order = 1 para que los datos sean ordenados aleatoriamente
set.seed(123)
indicemuestreo = samplecube(X, pik, method = 2, order = 1, comment = FALSE )

# Obtención de la submuestra
CriticalEvents_cube <- CriticalEvents[which(indicemuestreo == 1),]

# Comprobación del tamaño
cat('El tamaño de la submuestra con y = Critical Events, es:', dim(CriticalEvents_cube))
```

```
## El tamaño de la submuestra con y = Critical Events, es: 9000 16
```

Submuestreo en Maritime Accidents

```r
# Se eliminarán observaciones con NA
MaritimeAccidents <- MergedActivity %>%
  filter(y == "Maritime Accidents") %>%
  filter(complete.cases(.)) %>%
  mutate(across(where(is.factor), droplevels))
```

```r
# Para comprobar la estimación del tamaño poblacional,
# creamos un vector de "1" de dimensión = número de observaciones del nivel predominante
UNO = rep(1, nrow(MaritimeAccidents))

# Se necesita que todas las variables sean numéricas
# Variables cuantitativas
X1 <- MaritimeAccidents %>%
   select(activity_id, hour, longitude, age, gross_ton, vessel_length, air_temp, wind_spee
d)

# Variables cualitativas: One hot encoding
X2 <- disjunctive(MaritimeAccidents$region)
colnames(X2) <- levels(MaritimeAccidents$region)

X3 <- disjunctive(MaritimeAccidents$watertype)
colnames(X3) <- levels(MaritimeAccidents$watertype)

X4 <- disjunctive(MaritimeAccidents$vessel_class)
colnames(X4) <- levels(MaritimeAccidents$vessel_class)

X5 <- disjunctive(MaritimeAccidents$damage_status)
colnames(X5) <- levels(MaritimeAccidents$damage_status)

# Juntamos todo para formar la matriz de diseño
X = as.matrix(cbind(UNO, X1, X2, X3, X4, X5))

# Probabilidades de inclusión
pik = rep(n / nrow(MaritimeAccidents), nrow(MaritimeAccidents))

# Obtención de los índices de la nueva muestra con ayuda de la librería sampling
# method = 2 para fase de aterrizaje mediante supresión de variables
# order = 1 para que los datos sean ordenados aleatoriamente
set.seed(123)
indicemuestreo = samplecube(X, pik, method = 2, order = 1, comment = FALSE )

# Obtención de la submuestra
MaritimeAccidents_cube <- MaritimeAccidents[which(indicemuestreo == 1),]

# Comprobación del tamaño
cat('El tamaño de la submuestra con y = Maritime Accidents, es:', dim(MaritimeAccidents_cu
be))
```

```
## El tamaño de la submuestra con y = Maritime Accidents, es: 9000 16
```

Material Issues

```r
# Se eliminarán observaciones con NA
MaterialIssues <- MergedActivity %>%
  filter(y == "Material Issues") %>%
  filter(complete.cases(.)) %>%
  mutate(across(where(is.factor), droplevels))
```

```r
# Para comprobar la estimación del tamaño poblacional,
# creamos un vector de "1" de dimensión = número de observaciones del nivel predominante
UNO = rep(1, nrow(MaterialIssues))

# Se necesita que todas las variables sean numéricas
# Variables cuantitativas
X1 <- MaterialIssues %>%
   select(activity_id, hour, longitude, age, gross_ton, vessel_length, air_temp, wind_spee
d)

# Variables cualitativas: One hot encoding
X2 <- disjunctive(MaterialIssues$region)
colnames(X2) <- levels(MaterialIssues$region)

X3 <- disjunctive(MaterialIssues$watertype)
colnames(X3) <- levels(MaterialIssues$watertype)

X4 <- disjunctive(MaterialIssues$vessel_class)
colnames(X4) <- levels(MaterialIssues$vessel_class)

X5 <- disjunctive(MaterialIssues$damage_status)
colnames(X5) <- levels(MaterialIssues$damage_status)

# Juntamos todo para formar la matriz de diseño
X = as.matrix(cbind(UNO, X1, X2, X3, X4, X5))

# Probabilidades de inclusión
pik = rep(n / nrow(MaterialIssues), nrow(MaterialIssues))

# Obtención de los índices de la nueva muestra con ayuda de la librería sampling
# method = 2 para fase de aterrizaje mediante supresión de variables
# order = 1 para que los datos sean ordenados aleatoriamente
set.seed(123)
indicemuestreo = samplecube(X, pik, method = 2, order = 1, comment = FALSE )

# Obtención de la submuestra
MaterialIssues_cube <- MaterialIssues[which(indicemuestreo == 1),]

# Comprobación del tamaño
cat('El tamaño de la submuestra con y = Material Issues, es:', dim(MaterialIssues_cube))
```

```
## El tamaño de la submuestra con y = Material Issues, es: 9000 16
```

Sobremuestreos: Smote

Onboard Emergencies

```r
OnboardEmergencies <- MergedActivity %>%
  filter(y == "Onboard Emergencies")
```

```
# Observaciones a generar
ngenerar <- n - nrow(OnboardEmergencies)

# Porcentaje de muestra sintética o sobremuestreada
p_over <- ngenerar / nrow(OnboardEmergencies) * 100

# Muestra provisional: Se juntan con el resto de subconjuntos anteriormente nivelados.
# Se eliminan niveles no presentes
# La variable objetivo debe ser de tipo factor
# Las variables numéricas deben estar sin atributos de escalado
muestra_provisional <- rbind(CriticalEvents, MaritimeAccidents_cube, MaterialIssues_cube,
OnboardEmergencies) %>%
  mutate(across(where(is.factor), droplevels))


# Obtención de la muestra sintética con la librería DMwR
OnboardEmergencies_smote <- SMOTE(y ~ ., data = muestra_provisional, perc.over = p_over, p
erc.under = 0)

# Comprobación del tamaño
cat('El tamaño de la submuestra con y = Onboard Emergencies, es:', dim(OnboardEmergencies_
smote))
```

```
## El tamaño de la submuestra con y = Onboard Emergencies, es: 9000 16
```

Third-party Damages

```
ThirdpartyDamages <- MergedActivity %>%
  filter(y == "Third-party Damages")
```

```
ngenerar <- n - nrow(ThirdpartyDamages)

# Porcentaje de muestra sintética o sobremuestreada
p_over <- ngenerar / nrow(ThirdpartyDamages) * 100

# Muestra provisional: Se juntan con el resto de subconjuntos anteriormente nivelados.
# Se eliminan niveles no presentes
# La variable objetivo debe ser de tipo factor
# Las variables numéricas deben estar sin atributos de escalado
muestra_provisional <- rbind(CriticalEvents, MaritimeAccidents_cube, MaterialIssues_cube,
ThirdpartyDamages) %>%
  mutate(across(where(is.factor), droplevels))


# Obtención de la muestra sintética con la librería DMwR
ThirdpartyDamages_smote <- SMOTE(y ~ ., data = muestra_provisional, perc.over = p_over, pe
rc.under = 0)

# Comprobación del tamaño
cat('El tamaño de la submuestra con y = Third-Party Damages, es:', dim(ThirdpartyDamages_s
mote))
```

```
## El tamaño de la submuestra con y = Third-Party Damages, es: 9000 16
```

Unión de datos

```
# Unión de los subconjuntos
MergedActivityBalanced <- bind_rows(
   CriticalEvents_cube,
   MaritimeAccidents_cube,
   MaterialIssues_cube,
   OnboardEmergencies_smote,
   ThirdpartyDamages_smote)

# Estructura
str(MergedActivityBalanced)
```

```
## 'data.frame':    45000 obs. of  16 variables:
##  $ activity_id      : num  1475186 1484669 1482821 1479220 1730323 ...
##  $ date             : num  2 3 4 5 5 6 8 8 8 8 ...
##  $ hour             : num  22 13.9 18.5 11.8 10 ...
##  $ region            : Factor w/ 5 levels "Alaska","East Coast",..: 3 5 5 3 3 2 3 5 5 3
...
##  $ latitude         : num  27.8 37.8 44.6 27.9 30 ...
##  $ longitude        : num  -82.8 -122.2 -124.1 -82.5 -93.8 ...
##  $ watertype        : Factor w/ 2 levels "ocean","river": 1 1 1 1 1 1 1 1 1 1 ...
##  $ damage_status     : Factor w/ 5 levels "Actual Total Loss",..: 2 2 2 2 5 2 5 5 2 5
...
##  $ vessel_class     : Factor w/ 11 levels "Barge","Bulk Carrier",..: 8 7 3 7 10 8 4 4 7
10 ...
##  $ age              : num  39 1 36 58 30 13 22 1 75 30 ...
##  $ gross_ton        : num  13 91 49 5 163 ...
##  $ vessel_length    : num  32 107.3 54.8 31 71 ...
##  $ air_temp         : num  149.8 112.4 75.5 89.4 105.3 ...
##  $ wind_speed       : num  48.5 41.6 43.3 30 73.8 ...
##  $ damage_assessment: num  40000 0 200 0 0 150 0 0 0 0 ...
##  $ y                : Factor w/ 5 levels "Critical Events",..: 1 1 1 1 1 1 1 1 1 1 ...
```

1.1.2. Imputación de valores ausentes

```r
# Con ayuda de la librería mice, se van a aplicar los métodos Cart y Random forest.
# Se aplican 5 x 5 iteraciones a las tres variables con NA: air_temp, wind_speed y damage_
assessment
if (save_switch  == 1) {
MergedActivityBalancedCart <- MergedActivityBalanced %>%
  mice(method = "cart", minbucket = 4) %>%
  complete() %>%
  as.data.frame()

MergedActivityBalancedRF <- MergedActivityBalanced %>%
  mice(method = "rf", ntree = 3) %>%
  complete() %>%
  as.data.frame()

# Guardado junto
loggedsave(MergedActivityBalancedCart, "Datasets")
loggedsave(MergedActivityBalancedRF, "Datasets")
}else{
    MergedActivityBalancedCart <- readRDS("Datasets/MergedActivityBalancedCart.rds")
    MergedActivityBalancedRF <- readRDS("Datasets/MergedActivityBalancedRF.rds")
}
```

Comparación

```r
# Tabla con las medias de las variables imputadas junto con una columna de diferencias en
valor absoluto
bind_rows(
    summarise(MergedActivityBalanced,
              Dataset = "MergedActivityBalanced (Original)",
              mean_air_temp = mean(air_temp, na.rm = TRUE),
              mean_wind_speed = mean(wind_speed, na.rm = TRUE),
              mean_damage_assessment = mean(damage_assessment, na.rm = TRUE)),
    summarise(MergedActivityBalancedCart,
              Dataset = "MergedActivityBalancedCart",
              mean_air_temp = mean(air_temp),
              mean_wind_speed = mean(wind_speed),
              mean_damage_assessment = mean(damage_assessment)),
    summarise(MergedActivityBalancedRF,
              Dataset = "MergedActivityBalancedRF",
              mean_air_temp = mean(air_temp),
              mean_wind_speed = mean(wind_speed),
              mean_damage_assessment = mean(damage_assessment)),
) %>%
  mutate(suma = mean_air_temp + mean_wind_speed + mean_damage_assessment) %>%
  mutate(dif_total = abs(suma - suma[1])) %>%
  mutate(dif_total = color_tile("lightgreen", "white")(dif_total)) %>%
  select(-suma) %>%
  kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c("", "Comparación de medias"= 4))
```

Comparación de medias

Dataset	mean_air_temp	mean_wind_speed	mean_damage_assessment	dif_
MergedActivityBalanced (Original)	151.2707	50.51718	104365.3	0.0
MergedActivityBalancedCart	151.6104	49.00846	104383.4	16.
MergedActivityBalancedRF	151.7052	49.09728	104420.8	54.

Teniendo en cuenta las medias, la opción que minimiza las diferencias es el método Cart

1.1.3. Consolidación de datos

```
# Elección del dataframe completo.
# Filtrado de variables estadisticamente irrelevantes según regresión
# Se escalan variables numéricas
# Se cambian las etiquetas de la variable objetivo para evitar problemas en caret
if (save_switch  == 1) {
 MergedActivityGeneral <- MergedActivityBalancedCart %>%
   select(-date, -latitude, -damage_assessment) %>%
   mutate_if(is.numeric, scale) %>%
   mutate(y = factor(y, labels = make.names(levels(y))))

loggedsave(MergedActivityGeneral, "Datasets")
}else{
 MergedActivityGeneral <- readRDS("Datasets/MergedActivityGeneral.rds")
}

str(MergedActivityGeneral)
```

```
## 'data.frame':    45000 obs. of  13 variables:
##  $ activity_id  : num [1:45000, 1] -1.83 -1.82 -1.82 -1.83 -1.56 ...
##   ..- attr(*, "scaled:center")= num 3223764
##   ..- attr(*, "scaled:scale")= num 955025
##  $ hour         : num [1:45000, 1] 1.681 0.367 1.112 0.027 -0.271 ...
##   ..- attr(*, "scaled:center")= num 11.7
##   ..- attr(*, "scaled:scale")= num 6.15
##  $ region       : Factor w/ 5 levels "Alaska","East Coast",..: 3 5 5 3 3 2 3 5 5 3 ...
##  $ longitude    : num [1:45000, 1] 0.5352 -1.0936 -1.1722 0.5473 0.0788 ...
##   ..- attr(*, "scaled:center")= num -95.7
##   ..- attr(*, "scaled:scale")= num 24.2
##  $ watertype    : Factor w/ 2 levels "ocean","river": 1 1 1 1 1 1 1 1 1 1 ...
##  $ damage_status: Factor w/ 5 levels "Actual Total Loss",..: 2 2 2 2 5 2 5 5 2 5 ...
##  $ vessel_class : Factor w/ 11 levels "Barge","Bulk Carrier",..: 8 7 3 7 10 8 4 4 7 10
...
##  $ age          : num [1:45000, 1] 0.753 -1.509 0.575 1.884 0.217 ...
##   ..- attr(*, "scaled:center")= num 26.3
##   ..- attr(*, "scaled:scale")= num 16.8
##  $ gross_ton    : num [1:45000, 1] -0.342 -0.337 -0.34 -0.343 -0.332 ...
##   ..- attr(*, "scaled:center")= num 5014
##   ..- attr(*, "scaled:scale")= num 14606
##  $ vessel_length: num [1:45000, 1] -0.755 -0.401 -0.647 -0.759 -0.571 ...
##   ..- attr(*, "scaled:center")= num 193
##   ..- attr(*, "scaled:scale")= num 213
##  $ air_temp     : num [1:45000, 1] -0.0205 -0.4315 -0.8383 -0.685 -0.5105 ...
##   ..- attr(*, "scaled:center")= num 152
##   ..- attr(*, "scaled:scale")= num 90.8
##  $ wind_speed   : num [1:45000, 1] -0.0183 -0.2458 -0.189 -0.6315 0.8264 ...
##   ..- attr(*, "scaled:center")= num 49
##   ..- attr(*, "scaled:scale")= num 30
##  $ y            : Factor w/ 5 levels "Critical.Events",..: 1 1 1 1 1 1 1 1 1 1 ...
```

1.2. Dataset con variables factor (Redes bayesianas)

```
# Aplicación del método mdlp con ayuda de la librería arulesCBA para discretizar las varia
bles factoriales
MergedActivityFactor <- discretizeDF.supervised(y ~ ., MergedActivityGeneral)
```

```
# Guardado de datos
if (save_switch  == 1) {
loggedsave(MergedActivityFactor, "Datasets")
}else{
   MergedActivityFactor <- readRDS("Datasets/MergedActivityFactor.rds")
}
```

```
# Verificación de estructura
str(MergedActivityFactor)
```

```
## 'data.frame':    45000 obs. of  13 variables:
##  $ activity_id  : Factor w/ 5 levels "[-Inf,-0.991)",..: 1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "discretized:breaks")= num [1:6] -Inf -0.991 0.9027 0.0633 1.514 ...
##   ..- attr(*, "discretized:method")= chr "mdlp"
##  $ hour         : Factor w/ 7 levels "[-Inf,-1.84)",..: 7 5 6 4 4 5 4 2 4 4 ...
##   ..- attr(*, "discretized:breaks")= num [1:8] -Inf -1.838 -0.764 -0.948 0.135 ...
##   ..- attr(*, "discretized:method")= chr "mdlp"
##  $ region       : Factor w/ 5 levels "Alaska","East Coast",..: 3 5 5 3 3 2 3 5 5 3 ...
##  $ longitude    : Factor w/ 73 levels "[-Inf,-2.9393)",..: 63 11 9 65 24 65 51 12 10 22
...
##   ..- attr(*, "discretized:breaks")= num [1:74] -Inf -2.939 -1.481 -1.117 -0.885 ...
##   ..- attr(*, "discretized:method")= chr "mdlp"
##  $ watertype    : Factor w/ 2 levels "ocean","river": 1 1 1 1 1 1 1 1 1 1 ...
##  $ damage_status: Factor w/ 5 levels "Actual Total Loss",..: 2 2 2 2 5 2 5 5 2 5 ...
##  $ vessel_class : Factor w/ 11 levels "Barge","Bulk Carrier",..: 8 7 3 7 10 8 4 4 7 10
...
##  $ age          : Factor w/ 9 levels "[-Inf,-1.33)",..: 8 1 8 9 6 3 6 1 9 6 ...
##   ..- attr(*, "discretized:breaks")= num [1:10] -Inf -1.33 -0.734 -1.271 -0.701 ...
##   ..- attr(*, "discretized:method")= chr "mdlp"
##  $ gross_ton    : Factor w/ 35 levels "[-Inf,-0.34205)",..: 1 4 3 1 6 3 33 32 20 4 ...
##   ..- attr(*, "discretized:breaks")= num [1:36] -Inf -0.342 -0.338 -0.342 -0.337 ...
##   ..- attr(*, "discretized:method")= chr "mdlp"
##  $ vessel_length: Factor w/ 21 levels "[-Inf,-0.686)",..: 1 6 2 1 4 1 21 20 15 2 ...
##   ..- attr(*, "discretized:breaks")= num [1:22] -Inf -0.6856 -0.6175 0.0114 -0.5757 ...
##   ..- attr(*, "discretized:method")= chr "mdlp"
##  $ air_temp     : Factor w/ 3 levels "[-Inf,-1.06)",..: 3 2 2 2 2 3 2 3 2 2 ...
##   ..- attr(*, "discretized:breaks")= num [1:4] -Inf -1.058 -0.393 Inf
##   ..- attr(*, "discretized:method")= chr "mdlp"
##  $ wind_speed   : Factor w/ 4 levels "[-Inf,-0.591)",..: 2 2 2 1 3 4 2 3 1 2 ...
##   ..- attr(*, "discretized:breaks")= num [1:5] -Inf -0.591 0.334 2.129 Inf
##   ..- attr(*, "discretized:method")= chr "mdlp"
##  $ y            : Factor w/ 5 levels "Critical.Events",..: 1 1 1 1 1 1 1 1 1 1 ...
```

1.3. Dataset con variables numéricas (Gradient Boosting)

```r
# Creamos variables dummy con la ayuda de la librería fastDummies y juntamos con las varia
bles numéricas
# Pero la  variable objetivo se queda como factor para utilizarse en modelos de clasificac
ión
MergedActivityNum <- cbind(
  dummy_cols(MergedActivityGeneral[ , c(3, 5,6,7)], remove_selected_columns = TRUE),
  MergedActivityGeneral[ ,c(1,2, 4, 8,9,10,11,12,13)]
)
```

```r
# Guardado de datos
if (save_switch  == 1) {
loggedsave(MergedActivityNum, "Datasets")
}else{
   MergedActivityNum <- readRDS("Datasets/MergedActivityNum.rds")
}
```

```r
# Verificación de estructura
str(MergedActivityNum)
```

```
## 'data.frame':	45000 obs. of  32 variables:
##  $ region_Alaska                                  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ region_East Coast                              : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ region_Gulf of Mexico                          : int  1 0 0 1 1 0 1 0 0 1 ...
##  $ region_Mississippi                             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ region_West Coast                              : int  0 1 1 0 0 0 0 1 1 0 ...
##  $ watertype_ocean                                : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ watertype_river                                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ damage_status_Actual Total Loss                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ damage_status_Damaged                          : int  1 1 1 1 0 1 0 0 1 0 ...
##  $ damage_status_Total Constructive Loss: Salvaged  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ damage_status_Total Constructive Loss: Unsalvaged: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ damage_status_Undamaged                        : int  0 0 0 0 1 0 1 1 0 1 ...
##  $ vessel_class_Barge                             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vessel_class_Bulk Carrier                      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vessel_class_Fishing Vessel                    : int  0 0 1 0 0 0 0 0 0 0 ...
##  $ vessel_class_General Dry Cargo Ship            : int  0 0 0 0 0 0 1 1 0 0 ...
##  $ vessel_class_Miscellaneous Vessel              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vessel_class_Offshore                          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vessel_class_Passenger Ship                    : int  0 1 0 1 0 0 0 0 1 0 ...
##  $ vessel_class_Recreational                      : int  1 0 0 0 0 1 0 0 0 0 ...
##  $ vessel_class_Tank Ship                         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ vessel_class_Towing Vessel                     : int  0 0 0 0 1 0 0 0 0 1 ...
##  $ vessel_class_other value                       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ activity_id                                     : num [1:45000, 1] -1.83 -1.82 -1.8
2 -1.83 -1.56 ...
##   ..- attr(*, "scaled:center")= num 3223764
##   ..- attr(*, "scaled:scale")= num 955025
##  $ hour                                            : num [1:45000, 1] 1.681 0.367 1.11
2 0.027 -0.271 ...
##   ..- attr(*, "scaled:center")= num 11.7
##   ..- attr(*, "scaled:scale")= num 6.15
##  $ longitude                                       : num [1:45000, 1] 0.5352 -1.0936 -
1.1722 0.5473 0.0788 ...
##   ..- attr(*, "scaled:center")= num -95.7
##   ..- attr(*, "scaled:scale")= num 24.2
##  $ age                                             : num [1:45000, 1] 0.753 -1.509 0.5
75 1.884 0.217 ...
##   ..- attr(*, "scaled:center")= num 26.3
##   ..- attr(*, "scaled:scale")= num 16.8
##  $ gross_ton                                        : num [1:45000, 1] -0.342 -0.337 -
0.34 -0.343 -0.332 ...
##   ..- attr(*, "scaled:center")= num 5014
##   ..- attr(*, "scaled:scale")= num 14606
##  $ vessel_length                                    : num [1:45000, 1] -0.755 -0.401 -
0.647 -0.759 -0.571 ...
##   ..- attr(*, "scaled:center")= num 193
##   ..- attr(*, "scaled:scale")= num 213
##  $ air_temp                                         : num [1:45000, 1] -0.0205 -0.4315
-0.8383 -0.685 -0.5105 ...
##   ..- attr(*, "scaled:center")= num 152
##   ..- attr(*, "scaled:scale")= num 90.8
##  $ wind_speed                                       : num [1:45000, 1] -0.0183 -0.2458
-0.189 -0.6315 0.8264 ...
```

```
##   ..- attr(*, "scaled:center")= num 49
##   ..- attr(*, "scaled:scale")= num 30
##  $ y                                                : Factor w/ 5 levels "Critical.Even
ts",..: 1 1 1 1 1 1 1 1 1 1 ...
```

1.4. Particionado de datos

```
# Índice de partición
Indice_Particion <- createDataPartition(MergedActivityGeneral$y, p = 0.80, list = FALSE )

# Muestras de entrenamiento y test para propósito general
train_MA_general <- MergedActivityGeneral[Indice_Particion, ]
test_MA_general <- MergedActivityGeneral[-Indice_Particion, ]

# Muestras de entrenamiento y test para redes bayesanas
train_MA_factor <- MergedActivityFactor[Indice_Particion, ]
test_MA_factor <- MergedActivityFactor[-Indice_Particion, ]

# Muestras de entrenamiento y test para Gradient Boosting
train_MA_num <- MergedActivityNum[ Indice_Particion, ]
test_MA_num <- MergedActivityNum[ -Indice_Particion, ]
```

```
# Guardado de datos
if (save_switch  == 1) {
datasets_MA_particionados <- list(train_MA_general = train_MA_general,
                               test_MA_general = test_MA_general,
                               train_MA_factor = train_MA_factor,
                               test_MA_factor = test_MA_factor,
                               train_MA_num = train_MA_num,
                               test_MA_num = test_MA_num)

loggedsave(datasets_MA_particionados, "Datasets")
}
```

2. Entrenamiento de los modelos

```r
# Reset
rm(list = ls())
source("../4.Functions/myCustomFunctions.R")
train_switch <- 0
if (train_switch == 0){
    nb_MA_train <- readRDS("Models/nb_MA_train.RDS")
    GBM_MA_train <- readRDS("Models/GBM_MA_train.RDS")
    rf_MA_train <- readRDS("Models/rf_MA_train.RDS")
    nnet_MA_train <- readRDS("Models/nnet_MA_train.RDS")
    C5_MA_train <- readRDS("Models/C5_MA_train.RDS")
}
list2env(readRDS("Datasets/datasets_MA_particionados.rds"), envir = .GlobalEnv)
```

```
## <environment: R_GlobalEnv>
```

Método de validación cruzada

```r
fiveStats = function(...) c (multiClassSummary(...), defaultSummary(...))
control <- trainControl( method = "repeatedcv",
                         number = 8,
                         repeats = 2,
                         classProbs = TRUE,
                         summaryFunction = fiveStats,
                         returnResamp = "final",
                         verboseIter = TRUE,
                         allowParallel = TRUE)
metrica <- "logLoss"
```

2.1. Modelos de redes bayesianas

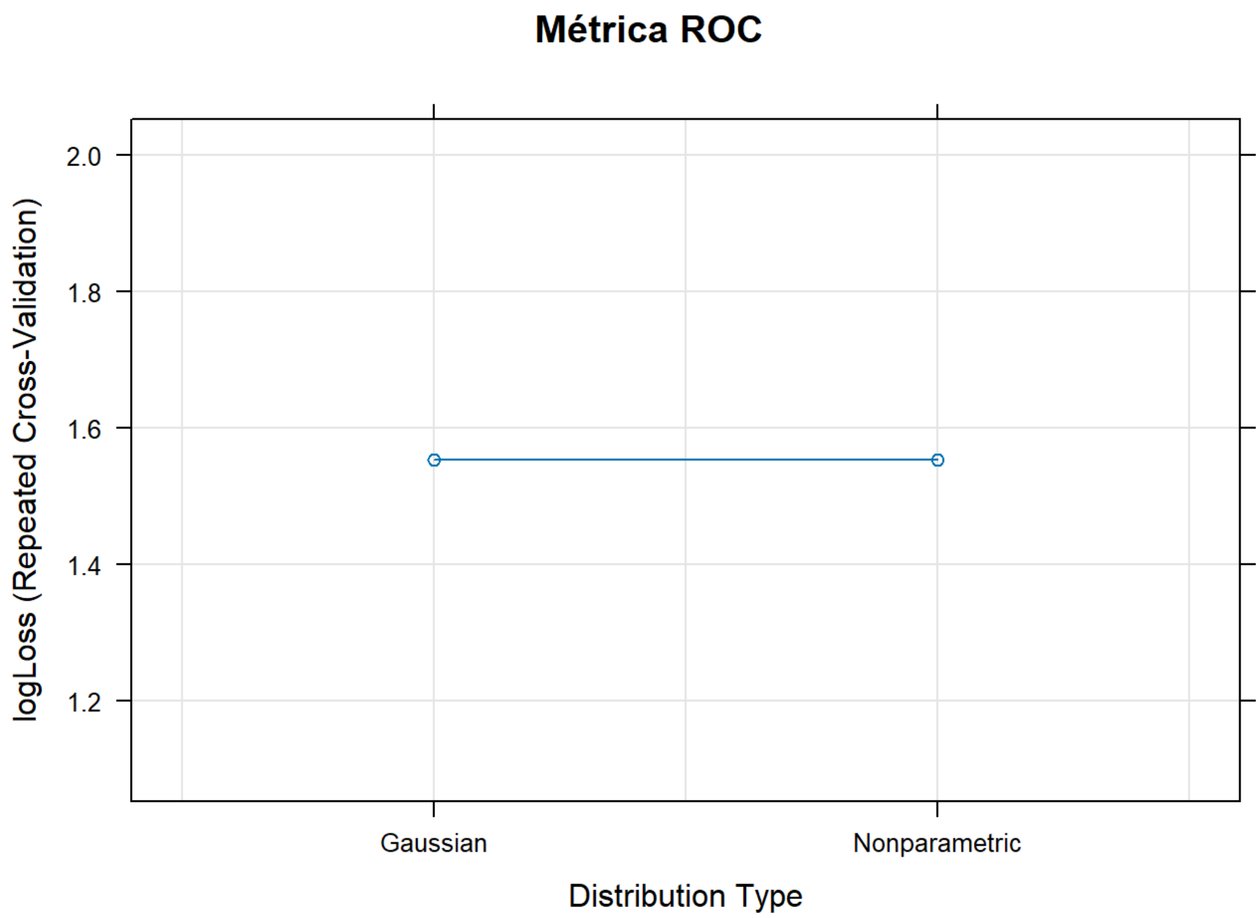
2.1.1. Naïve Bayes

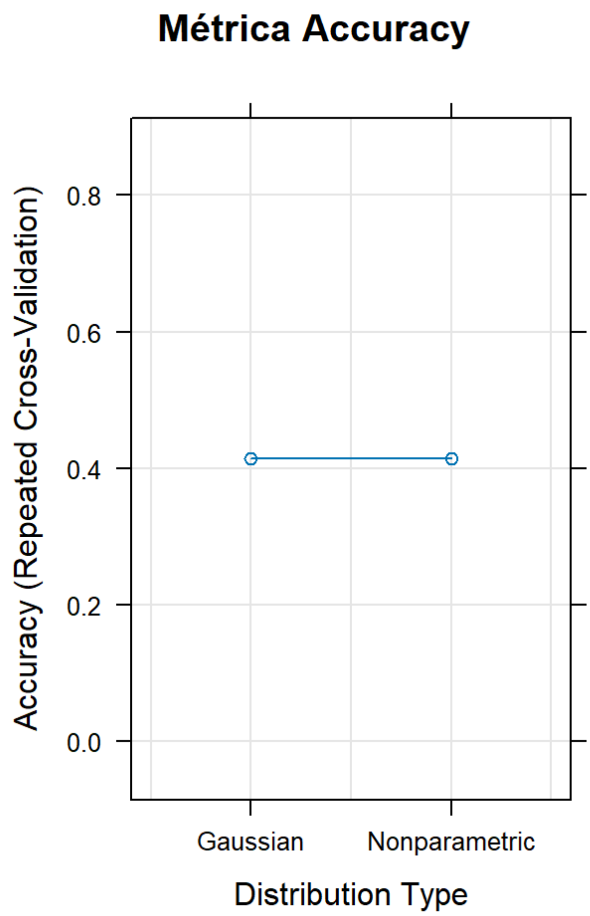
```r
if (train_switch  == 1) {
set.seed(7)

tic()

  clusterCPU <- makePSOCKcluster(detectCores() - 1)
  registerDoParallel(clusterCPU)

  nb_MA_train <- train(train_MA_factor[, !names(train_MA_factor) %in% "y"],
                  train_MA_factor$y,
                  method = 'nb',
                  metric = metrica,
                  # preProc = c('center', 'scale'),
                  trControl = control)


  stopCluster(clusterCPU)
  clusterCPU <- NULL

  saveRDS(nb_MA_train, "Models/nb_MA_train.RDS")

toc()

}else{
  nb_MA_train <- readRDS("Models/nb_MA_train.RDS")
}
```

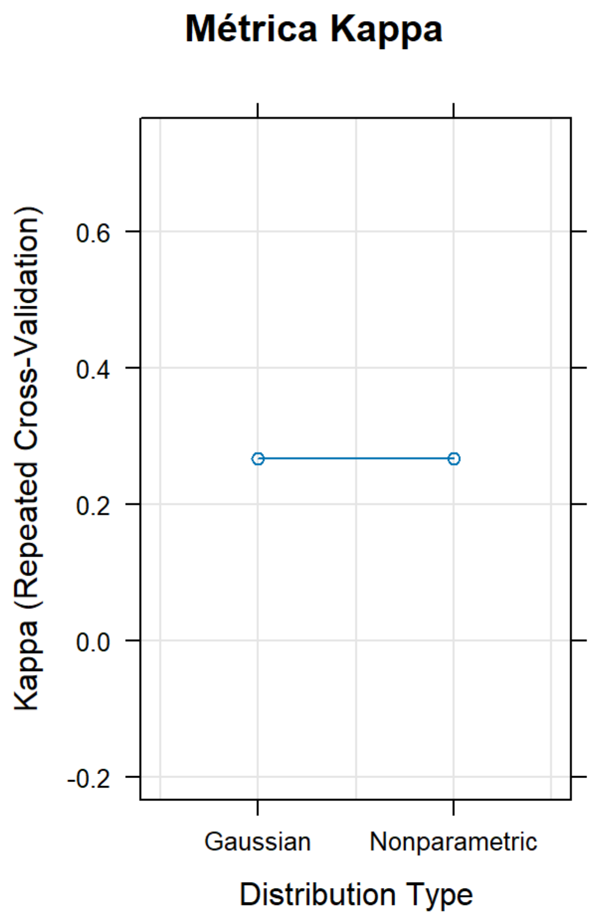
```r
# Resultados
nb_MA_train
```

```
## Naive Bayes
##
## 36000 samples
##    12 predictor
##     5 classes: 'Critical.Events', 'Maritime.Accidents', 'Material.Issues', 'Onboard.Eme
rgencies', 'Third.party.Damages'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 31500, 31500, 31500, 31500, 31500, 31500, ...
## Resampling results across tuning parameters:
##
##   usekernel  logLoss   AUC        prAUC      Accuracy  Kappa      Mean_F1
##   FALSE      1.552391  0.7211618  0.4272545  0.41325   0.2665625  0.4111456
##    TRUE      1.552391  0.7211618  0.4272545  0.41325   0.2665625  0.4111456
##   Mean_Sensitivity  Mean_Specificity  Mean_Pos_Pred_Value  Mean_Neg_Pred_Value
##   0.41325           0.8533125         0.4148792            0.8536251
##   0.41325           0.8533125         0.4148792            0.8536251
##   Mean_Precision  Mean_Recall  Mean_Detection_Rate  Mean_Balanced_Accuracy
##   0.4148792       0.41325      0.08265              0.6332813
##   0.4148792       0.41325      0.08265              0.6332813
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## logLoss was used to select the optimal model using the smallest value.
## The final values used for the model were fL = 0, usekernel = FALSE and adjust
##  = 1.
```

```
# Métricas
grafico_metricas(nb_MA_train)
```








```
# Resultados
resultados(nb_MA_train, "Naive Bayes")
```

RESULTADOS DEL MODELO Naive Bayes

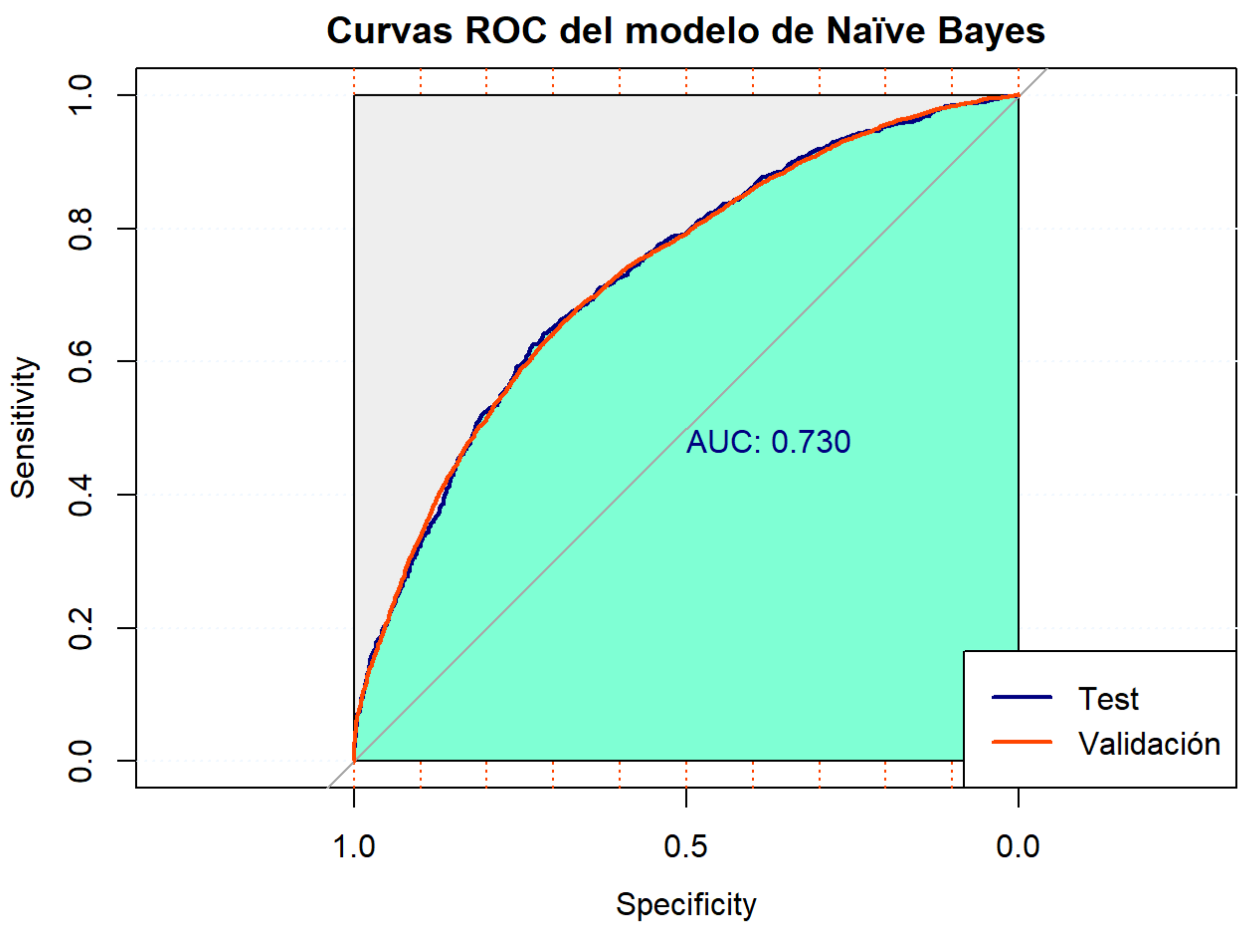
usekernel	fL	adjust	logLoss	AUC	prAUC	Accuracy	Kappa	Mean_F1	Mea
FALSE	0	1	1.552391	0.7211618	0.4272545	0.41325	0.2665625	0.4111456	
TRUE	0	1	1.552391	0.7211618	0.4272545	0.41325	0.2665625	0.4111456	

```
# Mejor modelo
mejor_modelo(nb_MA_train)
```

```
## [1] "El mejor módelo es el que muestra los siguientes hiperparámetros:"
```

fL	usekernel	adjust
0	FALSE	1

```
# Curvas ROC y AUC
curvas_ROC(nb_MA_train, "de Naïve Bayes", train_MA_factor, test_MA_factor)
```




```
## [1] "ROC del modelo con el fichero de test: 0.729759722222222"
```

```
# Validación: Matriz de confusión
validation(nb_MA_train, "de Naïve Bayes", train_MA_factor, test_MA_factor)
```

```
## [1] "Modelo de Naïve Bayes - Tabla de confusión para los datos de entrenamiento"
## Confusion Matrix and Statistics
##
##                      Reference
## Prediction            Critical.Events Maritime.Accidents Material.Issues
##   Critical.Events                1668                817            1231
##   Maritime.Accidents             1242               3378            1119
##   Material.Issues                2461               1349            3304
##   Onboard.Emergencies            1036                880             862
##   Third.party.Damages             793                776             684
##                      Reference
## Prediction            Onboard.Emergencies Third.party.Damages
##   Critical.Events                    1062                 719
##   Maritime.Accidents                  888                 883
##   Material.Issues                    1337                1061
##   Onboard.Emergencies                3185                1054
##   Third.party.Damages                 728                3483
##
## Overall Statistics
##
##                Accuracy : 0.4172
##                  95% CI : (0.4121, 0.4223)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2715
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Critical.Events Class: Maritime.Accidents
## Sensitivity                         0.23167                   0.46917
## Specificity                         0.86705                   0.85653
## Pos Pred Value                      0.30344                   0.44980
## Neg Pred Value                      0.81864                   0.86585
## Prevalence                          0.20000                   0.20000
## Detection Rate                      0.04633                   0.09383
## Detection Prevalence                0.15269                   0.20861
## Balanced Accuracy                   0.54936                   0.66285
##                      Class: Material.Issues Class: Onboard.Emergencies
## Sensitivity                         0.45889                    0.44236
## Specificity                         0.78444                    0.86694
## Pos Pred Value                      0.34735                    0.45390
## Neg Pred Value                      0.85291                    0.86147
## Prevalence                          0.20000                    0.20000
## Detection Rate                      0.09178                    0.08847
## Detection Prevalence                0.26422                    0.19492
## Balanced Accuracy                   0.62167                    0.65465
##                      Class: Third.party.Damages
## Sensitivity                             0.48375
## Specificity                             0.89649
## Pos Pred Value                          0.53883
## Neg Pred Value                          0.87415
```

```
## Prevalence                                   0.20000
## Detection Rate                               0.09675
## Detection Prevalence                         0.17956
## Balanced Accuracy                            0.69012
## [1] "Modelo de Naïve Bayes - Tabla de confusión para los datos de validación"
```

```
## Confusion Matrix and Statistics
##
##                       Reference
## Prediction             Critical.Events Maritime.Accidents Material.Issues
##   Critical.Events                  428                199             321
##   Maritime.Accidents               313                851             276
##   Material.Issues                  606                364             839
##   Onboard.Emergencies              241                206             185
##   Third.party.Damages              212                180             179
##                       Reference
## Prediction             Onboard.Emergencies Third.party.Damages
##   Critical.Events                      250                 185
##   Maritime.Accidents                   233                 196
##   Material.Issues                      336                 286
##   Onboard.Emergencies                  803                 275
##   Third.party.Damages                  178                 858
##
## Overall Statistics
##
##                Accuracy : 0.4199
##                  95% CI : (0.4097, 0.4302)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2749
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                       Class: Critical.Events Class: Maritime.Accidents
## Sensitivity                          0.23778                   0.47278
## Specificity                          0.86736                   0.85861
## Pos Pred Value                       0.30947                   0.45532
## Neg Pred Value                       0.81988                   0.86692
## Prevalence                           0.20000                   0.20000
## Detection Rate                       0.04756                   0.09456
## Detection Prevalence                 0.15367                   0.20767
## Balanced Accuracy                    0.55257                   0.66569
##                       Class: Material.Issues Class: Onboard.Emergencies
## Sensitivity                          0.46611                    0.44611
## Specificity                          0.77889                    0.87403
## Pos Pred Value                       0.34513                    0.46959
## Neg Pred Value                       0.85371                    0.86324
## Prevalence                           0.20000                    0.20000
## Detection Rate                       0.09322                    0.08922
## Detection Prevalence                 0.27011                    0.19000
## Balanced Accuracy                    0.62250                    0.66007
##                       Class: Third.party.Damages
## Sensitivity                              0.47667
## Specificity                              0.89597
## Pos Pred Value                           0.53391
## Neg Pred Value                           0.87258
## Prevalence                               0.20000
```

```
## Detection Rate                      0.09533
## Detection Prevalence                0.17856
## Balanced Accuracy                   0.68632
```

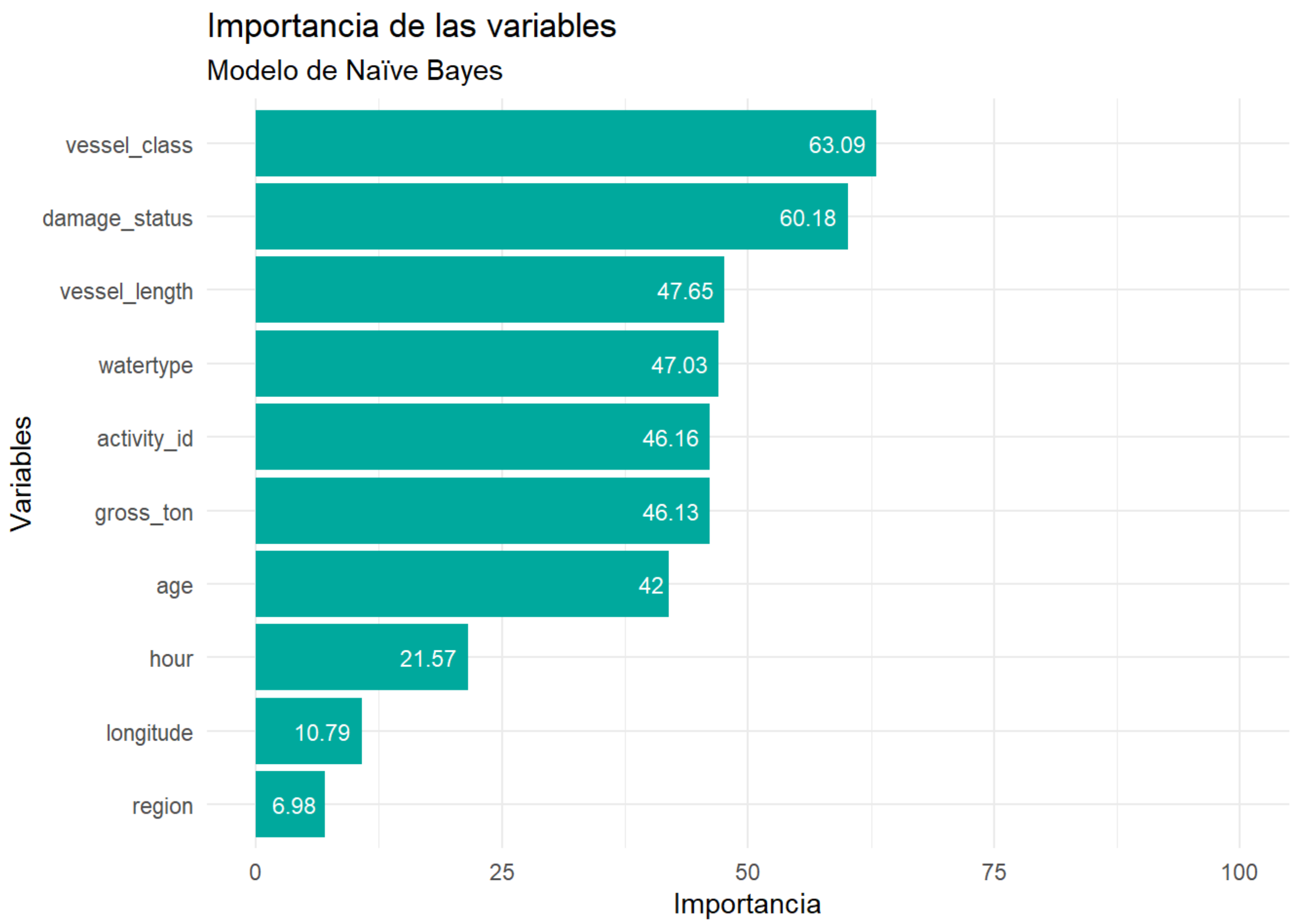
Resumen de métricas

```
# Resumen de métricas
resumen_MA_nb <- resumen_multiclass(nb_MA_train, train_MA_factor, test_MA_factor)

# Presentación
resumen_MA_nb %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ", "Naïve Bayes Classifier" = 5))
```

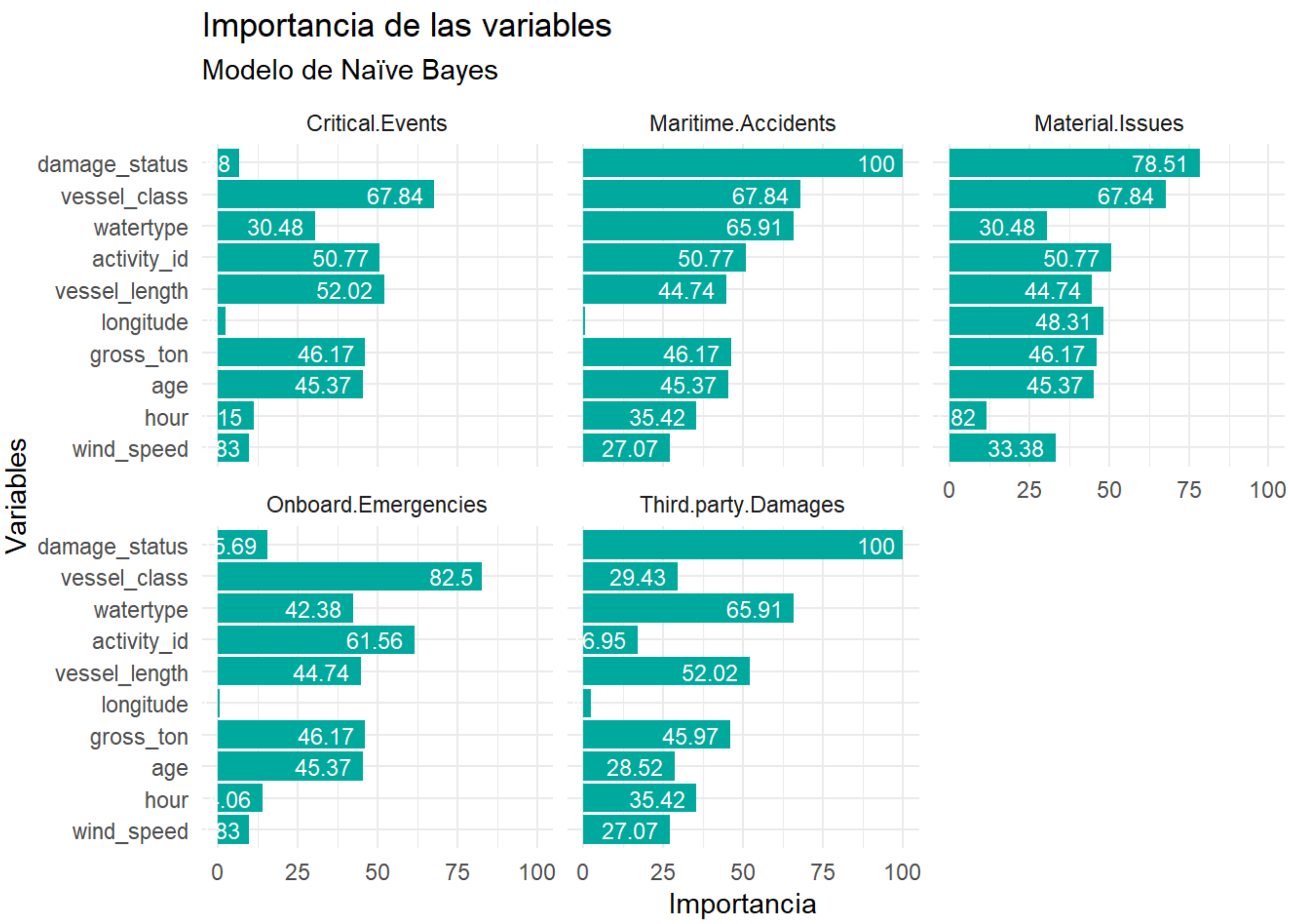
	Naïve Bayes Classifier				
	AUC	**Accuracy**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.725	0.417	0.271	0.417	0.854
Datos Validación	0.726	0.420	0.275	0.420	0.855

Importancia de las variables

```
# Importancia general de las variables
importancia_var_overall(nb_MA_train, "de Naïve Bayes")
```



```r
# Importancia de variables por cada valor de predicción
importancia_var(nb_MA_train, "de Naïve Bayes")
```



2.2. Modelos Gradient Boosting

2.2.1. Modelo GBM

```r
# Entrenamiento
if (train_switch  == 1) {
set.seed(7)
tic()

clusterCPU <- makePSOCKcluster( detectCores()-1 )
registerDoParallel(clusterCPU)

tune_grid <- expand.grid(n.trees = seq(from = 100, to = 500, by = 25),
                         interaction.depth = c(1, 2, 3, 4, 5),
                         shrinkage = 0.1,
                         n.minobsinnode = 10)

GBM_MA_train <- train(train_MA_num[ , -length(train_MA_num)],
                 train_MA_num$y,
                 method = "gbm",
                 metric = metrica,
                 trControl = control,
                 tuneGrid = tune_grid)

stopCluster(clusterCPU)

saveRDS(GBM_MA_train, "Models/GBM_MA_train.RDS")
toc()

}else{
  GBM_MA_train <- readRDS("Models/GBM_MA_train.RDS")
}
```

```r
# Resultados
GBM_MA_train
```

```
## Stochastic Gradient Boosting
##
## 36000 samples
##    31 predictor
##     5 classes: 'Critical.Events', 'Maritime.Accidents', 'Material.Issues', 'Onboard.Eme
rgencies', 'Third.party.Damages'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 31500, 31500, 31500, 31500, 31500, 31500, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  logLoss   AUC        prAUC      Accuracy
##   1                  100      1.421799  0.7104177  0.4074903  0.4012639
##   1                  125      1.415076  0.7133146  0.4113238  0.4045972
##   1                  150      1.409956  0.7154615  0.4141833  0.4074028
##   1                  175      1.405948  0.7173134  0.4164173  0.4082778
##   1                  200      1.402865  0.7186031  0.4180661  0.4081111
##   1                  225      1.400130  0.7198719  0.4196619  0.4089306
##   1                  250      1.397609  0.7209901  0.4209062  0.4101389
##   1                  275      1.395478  0.7219672  0.4221593  0.4101806
##   1                  300      1.393616  0.7228167  0.4231857  0.4110972
##   1                  325      1.391878  0.7235979  0.4241142  0.4105694
##   1                  350      1.390251  0.7243895  0.4251142  0.4115278
##   1                  375      1.388788  0.7250509  0.4258912  0.4125139
##   1                  400      1.387409  0.7256973  0.4267028  0.4123889
##   1                  425      1.386290  0.7262027  0.4274507  0.4131806
##   1                  450      1.385241  0.7267019  0.4280281  0.4133472
##   1                  475      1.384167  0.7272175  0.4286178  0.4144028
##   1                  500      1.383186  0.7276770  0.4292198  0.4149306
##   2                  100      1.378019  0.7310384  0.4380791  0.4188056
##   2                  125      1.369057  0.7349097  0.4432524  0.4226667
##   2                  150      1.361603  0.7381611  0.4475211  0.4241667
##   2                  175      1.355282  0.7407652  0.4510734  0.4272361
##   2                  200      1.350330  0.7428412  0.4537321  0.4279167
##   2                  225      1.346199  0.7444966  0.4557991  0.4292361
##   2                  250      1.342053  0.7461290  0.4579814  0.4306806
##   2                  275      1.339083  0.7471988  0.4595801  0.4319028
##   2                  300      1.336816  0.7480383  0.4607709  0.4315000
##   2                  325      1.334773  0.7487271  0.4618919  0.4321944
##   2                  350      1.333059  0.7493354  0.4625454  0.4324861
##   2                  375      1.331362  0.7498966  0.4635425  0.4334167
##   2                  400      1.329933  0.7504245  0.4642843  0.4336528
##   2                  425      1.328379  0.7509697  0.4651207  0.4345694
##   2                  450      1.327274  0.7513514  0.4655934  0.4353889
##   2                  475      1.326295  0.7516543  0.4659920  0.4349583
##   2                  500      1.325343  0.7519641  0.4665756  0.4355417
##   3                  100      1.353801  0.7419725  0.4536293  0.4269028
##   3                  125      1.345573  0.7451882  0.4577480  0.4297639
##   3                  150      1.338883  0.7476799  0.4609927  0.4313194
##   3                  175      1.333635  0.7496772  0.4638145  0.4322222
##   3                  200      1.329588  0.7511628  0.4658818  0.4337361
##   3                  225      1.326348  0.7521653  0.4674075  0.4340139
##   3                  250      1.323609  0.7530989  0.4684791  0.4348750
```

```
##   3              275       1.321356  0.7538947  0.4696158  0.4362917
##   3              300       1.319950  0.7542248  0.4700407  0.4363056
##   3              325       1.318212  0.7546953  0.4707831  0.4369861
##   3              350       1.316835  0.7550978  0.4712729  0.4370417
##   3              375       1.315735  0.7554160  0.4716288  0.4358611
##   3              400       1.314767  0.7556558  0.4719951  0.4363611
##   3              425       1.314018  0.7558778  0.4723211  0.4364306
##   3              450       1.313438  0.7559660  0.4725956  0.4366528
##   3              475       1.312668  0.7561717  0.4729624  0.4369444
##   3              500       1.311955  0.7563827  0.4732209  0.4368472
##   4              100       1.340799  0.7469502  0.4609868  0.4307639
##   4              125       1.333286  0.7496843  0.4645747  0.4337361
##   4              150       1.327647  0.7516810  0.4669337  0.4353889
##   4              175       1.324032  0.7527457  0.4683581  0.4354167
##   4              200       1.320874  0.7536796  0.4695436  0.4350000
##   4              225       1.318887  0.7541705  0.4700324  0.4351111
##   4              250       1.316778  0.7548176  0.4710103  0.4354167
##   4              275       1.315099  0.7553756  0.4714053  0.4357222
##   4              300       1.313932  0.7556535  0.4718179  0.4358333
##   4              325       1.313060  0.7558386  0.4720151  0.4359444
##   4              350       1.312593  0.7558391  0.4720426  0.4359306
##   4              375       1.311944  0.7559724  0.4724338  0.4366667
##   4              400       1.311542  0.7560016  0.4724530  0.4364444
##   4              425       1.311272  0.7560057  0.4726567  0.4362361
##   4              450       1.311036  0.7559903  0.4727301  0.4357917
##   4              475       1.310665  0.7560592  0.4728300  0.4357222
##   4              500       1.310523  0.7560021  0.4729357  0.4356250
##   5              100       1.331034  0.7510332  0.4664366  0.4341250
##   5              125       1.324844  0.7529935  0.4689662  0.4352500
##   5              150       1.320249  0.7544621  0.4706374  0.4362778
##   5              175       1.317313  0.7551641  0.4717544  0.4356250
##   5              200       1.314972  0.7556787  0.4723556  0.4356528
##   5              225       1.313310  0.7559960  0.4730165  0.4360417
##   5              250       1.312131  0.7562457  0.4734188  0.4360972
##   5              275       1.311219  0.7563096  0.4736295  0.4361944
##   5              300       1.311015  0.7562160  0.4734417  0.4347778
##   5              325       1.310787  0.7561085  0.4734252  0.4345972
##   5              350       1.310059  0.7562578  0.4736816  0.4349028
##   5              375       1.310068  0.7561652  0.4735240  0.4348056
##   5              400       1.310553  0.7558287  0.4731793  0.4350694
##   5              425       1.310803  0.7556284  0.4730491  0.4345833
##   5              450       1.310611  0.7556765  0.4733142  0.4340694
##   5              475       1.310779  0.7555445  0.4732981  0.4340833
##   5              500       1.311570  0.7551041  0.4728771  0.4334861
##   Kappa      Mean_F1    Mean_Sensitivity  Mean_Specificity  Mean_Pos_Pred_Value
##   0.2515799  0.3890900  0.4012639         0.8503160         0.3927028
##   0.2557465  0.3928106  0.4045972         0.8511493         0.3960388
##   0.2592535  0.3964689  0.4074028         0.8518507         0.3991196
##   0.2603472  0.3972535  0.4082778         0.8520694         0.3997583
##   0.2601389  0.3975688  0.4081111         0.8520278         0.3993471
##   0.2611632  0.3987747  0.4089306         0.8522326         0.4004735
##   0.2626736  0.4001359  0.4101389         0.8525347         0.4016410
##   0.2627257  0.4004952  0.4101806         0.8525451         0.4019097
##   0.2638715  0.4013311  0.4110972         0.8527743         0.4028735
##   0.2632118  0.4009011  0.4105694         0.8526424         0.4021071
```

```
##    0.2644097  0.4022694  0.4115278          0.8528819          0.4031628
##    0.2656424  0.4030958  0.4125139          0.8531285          0.4039788
##    0.2654861  0.4029481  0.4123889          0.8530972          0.4036084
##    0.2664757  0.4040265  0.4131806          0.8532951          0.4045800
##    0.2666840  0.4042635  0.4133472          0.8533368          0.4046831
##    0.2680035  0.4055390  0.4144028          0.8536007          0.4059576
##    0.2686632  0.4060198  0.4149306          0.8537326          0.4063206
##    0.2735069  0.4108844  0.4188056          0.8547014          0.4138034
##    0.2783333  0.4154980  0.4226667          0.8556667          0.4178866
##    0.2802083  0.4175638  0.4241667          0.8560417          0.4191474
##    0.2840451  0.4208558  0.4272361          0.8568090          0.4220413
##    0.2848958  0.4218846  0.4279167          0.8569792          0.4226199
##    0.2865451  0.4234332  0.4292361          0.8573090          0.4239022
##    0.2883507  0.4251657  0.4306806          0.8576701          0.4253399
##    0.2898785  0.4267873  0.4319028          0.8579757          0.4266442
##    0.2893750  0.4264556  0.4315000          0.8578750          0.4263258
##    0.2902431  0.4272559  0.4321944          0.8580486          0.4269126
##    0.2906076  0.4275235  0.4324861          0.8581215          0.4269200
##    0.2917708  0.4285529  0.4334167          0.8583542          0.4280586
##    0.2920660  0.4288972  0.4336528          0.8584132          0.4283725
##    0.2932118  0.4298976  0.4345694          0.8586424          0.4292295
##    0.2942361  0.4307605  0.4353889          0.8588472          0.4302341
##    0.2936979  0.4303980  0.4349583          0.8587396          0.4297642
##    0.2944271  0.4309013  0.4355417          0.8588854          0.4301558
##    0.2836285  0.4206684  0.4269028          0.8567257          0.4218615
##    0.2872049  0.4243941  0.4297639          0.8574410          0.4249856
##    0.2891493  0.4263073  0.4313194          0.8578299          0.4264688
##    0.2902778  0.4275815  0.4322222          0.8580556          0.4276451
##    0.2921701  0.4290575  0.4337361          0.8584340          0.4289883
##    0.2925174  0.4295952  0.4340139          0.8585035          0.4294040
##    0.2935937  0.4307858  0.4348750          0.8587187          0.4305344
##    0.2953646  0.4321680  0.4362917          0.8590729          0.4318702
##    0.2953819  0.4324228  0.4363056          0.8590764          0.4319835
##    0.2962326  0.4331212  0.4369861          0.8592465          0.4325762
##    0.2963021  0.4332892  0.4370417          0.8592604          0.4326216
##    0.2948264  0.4322331  0.4358611          0.8589653          0.4315735
##    0.2954514  0.4327910  0.4363611          0.8590903          0.4320583
##    0.2955382  0.4328757  0.4364306          0.8591076          0.4321156
##    0.2958160  0.4332827  0.4366528          0.8591632          0.4325442
##    0.2961806  0.4334098  0.4369444          0.8592361          0.4326404
##    0.2960590  0.4334754  0.4368472          0.8592118          0.4326536
##    0.2884549  0.4256477  0.4307639          0.8576910          0.4261704
##    0.2921701  0.4290566  0.4337361          0.8584340          0.4292855
##    0.2942361  0.4308813  0.4353889          0.8588472          0.4308010
##    0.2942708  0.4313089  0.4354167          0.8588542          0.4311696
##    0.2937500  0.4310607  0.4350000          0.8587500          0.4306731
##    0.2938889  0.4314470  0.4351111          0.8587778          0.4308453
##    0.2942708  0.4317796  0.4354167          0.8588542          0.4310664
##    0.2946528  0.4322033  0.4357222          0.8589306          0.4313224
##    0.2947917  0.4322968  0.4358333          0.8589583          0.4315182
##    0.2949306  0.4325614  0.4359444          0.8589861          0.4318544
##    0.2949132  0.4325447  0.4359306          0.8589826          0.4317114
##    0.2958333  0.4333376  0.4366667          0.8591667          0.4325402
##    0.2955556  0.4332412  0.4364444          0.8591111          0.4323905
##    0.2952951  0.4331743  0.4362361          0.8590590          0.4323977
```

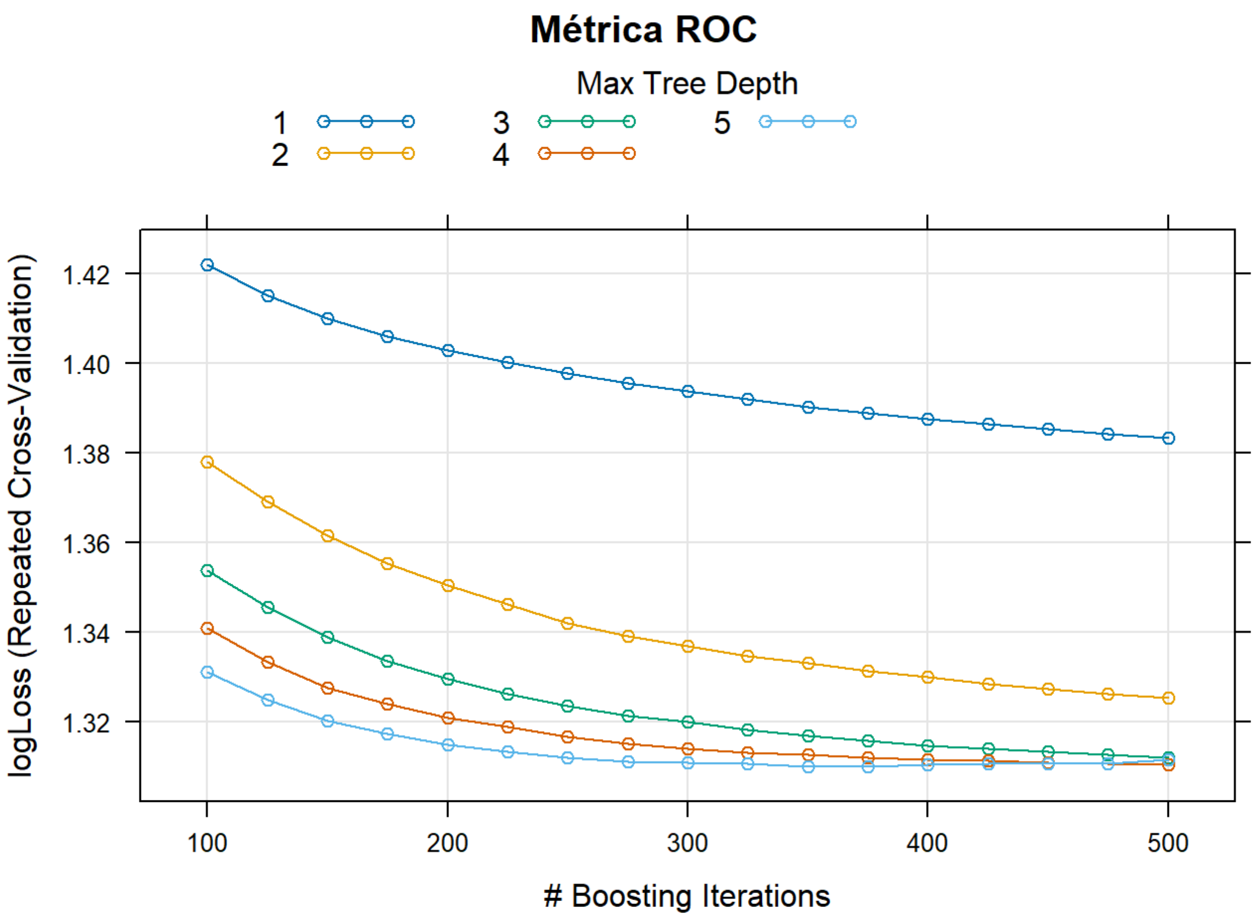
```
##    0.2947396  0.4328870  0.4357917         0.8589479         0.4321198
##    0.2946528  0.4327608  0.4357222         0.8589306         0.4319556
##    0.2945312  0.4328509  0.4356250         0.8589062         0.4321326
##    0.2926562  0.4299471  0.4341250         0.8585312         0.4304263
##    0.2940625  0.4312221  0.4352500         0.8588125         0.4314325
##    0.2953472  0.4324321  0.4362778         0.8590694         0.4324403
##    0.2945312  0.4319189  0.4356250         0.8589062         0.4315975
##    0.2945660  0.4322421  0.4356528         0.8589132         0.4319007
##    0.2950521  0.4326929  0.4360417         0.8590104         0.4323123
##    0.2951215  0.4329436  0.4360972         0.8590243         0.4325969
##    0.2952431  0.4330634  0.4361944         0.8590486         0.4326247
##    0.2934722  0.4318762  0.4347778         0.8586944         0.4314372
##    0.2932465  0.4316891  0.4345972         0.8586493         0.4312954
##    0.2936285  0.4322107  0.4349028         0.8587257         0.4317596
##    0.2935069  0.4322112  0.4348056         0.8587014         0.4317349
##    0.2938368  0.4325712  0.4350694         0.8587674         0.4320189
##    0.2932292  0.4322079  0.4345833         0.8586458         0.4316538
##    0.2925868  0.4316610  0.4340694         0.8585174         0.4311077
##    0.2926042  0.4317713  0.4340833         0.8585208         0.4312461
##    0.2918576  0.4311115  0.4334861         0.8583715         0.4305151
##    Mean_Neg_Pred_Value  Mean_Precision  Mean_Recall  Mean_Detection_Rate
##    0.8516946            0.3927028       0.4012639    0.08025278
##    0.8524995            0.3960388       0.4045972    0.08091944
##    0.8531201            0.3991196       0.4074028    0.08148056
##    0.8533463            0.3997583       0.4082778    0.08165556
##    0.8532582            0.3993471       0.4081111    0.08162222
##    0.8534220            0.4004735       0.4089306    0.08178611
##    0.8537103            0.4016410       0.4101389    0.08202778
##    0.8536905            0.4019097       0.4101806    0.08203611
##    0.8539284            0.4028735       0.4110972    0.08221944
##    0.8537882            0.4021071       0.4105694    0.08211389
##    0.8539903            0.4031628       0.4115278    0.08230556
##    0.8542578            0.4039788       0.4125139    0.08250278
##    0.8542304            0.4036084       0.4123889    0.08247778
##    0.8543981            0.4045800       0.4131806    0.08263611
##    0.8544319            0.4046831       0.4133472    0.08266944
##    0.8546762            0.4059576       0.4144028    0.08288056
##    0.8548119            0.4063206       0.4149306    0.08298611
##    0.8557241            0.4138034       0.4188056    0.08376111
##    0.8565992            0.4178866       0.4226667    0.08453333
##    0.8568934            0.4191474       0.4241667    0.08483333
##    0.8576263            0.4220413       0.4272361    0.08544722
##    0.8577470            0.4226199       0.4279167    0.08558333
##    0.8580481            0.4239022       0.4292361    0.08584722
##    0.8583738            0.4253399       0.4306806    0.08613611
##    0.8586325            0.4266442       0.4319028    0.08638056
##    0.8585189            0.4263258       0.4315000    0.08630000
##    0.8586798            0.4269126       0.4321944    0.08643889
##    0.8587520            0.4269200       0.4324861    0.08649722
##    0.8589739            0.4280586       0.4334167    0.08668333
##    0.8590178            0.4283725       0.4336528    0.08673056
##    0.8592378            0.4292295       0.4345694    0.08691389
##    0.8594369            0.4302341       0.4353889    0.08707778
##    0.8593187            0.4297642       0.4349583    0.08699167
##    0.8594764            0.4301558       0.4355417    0.08710833
```

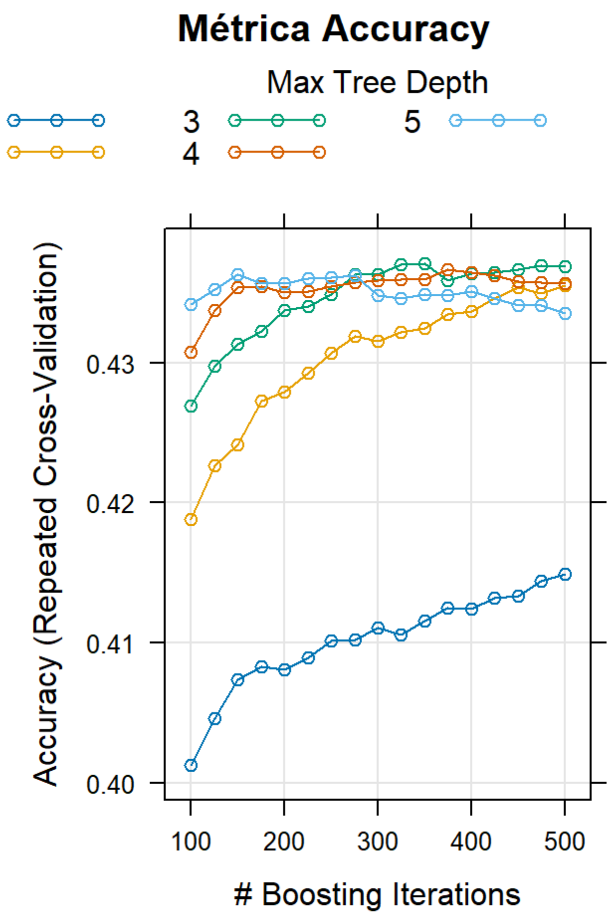
```
##    0.8575174       0.4218615     0.4269028   0.08538056
##    0.8581260       0.4249856     0.4297639   0.08595278
##    0.8584635       0.4264688     0.4313194   0.08626389
##    0.8586469       0.4276451     0.4322222   0.08644444
##    0.8590271       0.4289883     0.4337361   0.08674722
##    0.8590644       0.4294040     0.4340139   0.08680278
##    0.8592390       0.4305344     0.4348750   0.08697500
##    0.8595976       0.4318702     0.4362917   0.08725833
##    0.8595716       0.4319835     0.4363056   0.08726111
##    0.8597384       0.4325762     0.4369861   0.08739722
##    0.8597383       0.4326216     0.4370417   0.08740833
##    0.8594269       0.4315735     0.4358611   0.08717222
##    0.8595447       0.4320583     0.4363611   0.08727222
##    0.8595618       0.4321156     0.4364306   0.08728611
##    0.8595956       0.4325442     0.4366528   0.08733056
##    0.8596873       0.4326404     0.4369444   0.08738889
##    0.8596417       0.4326536     0.4368472   0.08736944
##    0.8583405       0.4261704     0.4307639   0.08615278
##    0.8590267       0.4292855     0.4337361   0.08674722
##    0.8594160       0.4308010     0.4353889   0.08707778
##    0.8593742       0.4311696     0.4354167   0.08708333
##    0.8592427       0.4306731     0.4350000   0.08700000
##    0.8592375       0.4308453     0.4351111   0.08702222
##    0.8593112       0.4310664     0.4354167   0.08708333
##    0.8593766       0.4313224     0.4357222   0.08714444
##    0.8594038       0.4315182     0.4358333   0.08716667
##    0.8594148       0.4318544     0.4359444   0.08718889
##    0.8594137       0.4317114     0.4359306   0.08718611
##    0.8595909       0.4325402     0.4366667   0.08733333
##    0.8595209       0.4323905     0.4364444   0.08728889
##    0.8594505       0.4323977     0.4362361   0.08724722
##    0.8593192       0.4321198     0.4357917   0.08715833
##    0.8593114       0.4319556     0.4357222   0.08714444
##    0.8592640       0.4321326     0.4356250   0.08712500
##    0.8590641       0.4304263     0.4341250   0.08682500
##    0.8593217       0.4314325     0.4352500   0.08705000
##    0.8595551       0.4324403     0.4362778   0.08725556
##    0.8593720       0.4315975     0.4356250   0.08712500
##    0.8593428       0.4319007     0.4356528   0.08713056
##    0.8594338       0.4323123     0.4360417   0.08720833
##    0.8594227       0.4325969     0.4360972   0.08721944
##    0.8594455       0.4326247     0.4361944   0.08723889
##    0.8590648       0.4314372     0.4347778   0.08695556
##    0.8590177       0.4312954     0.4345972   0.08691944
##    0.8590701       0.4317596     0.4349028   0.08698056
##    0.8590342       0.4317349     0.4348056   0.08696111
##    0.8590882       0.4320189     0.4350694   0.08701389
##    0.8589516       0.4316538     0.4345833   0.08691667
##    0.8588256       0.4311077     0.4340694   0.08681389
##    0.8588184       0.4312461     0.4340833   0.08681667
##    0.8586773       0.4305151     0.4334861   0.08669722
##    Mean_Balanced_Accuracy
##    0.6257899
##    0.6278733
##    0.6296267
```

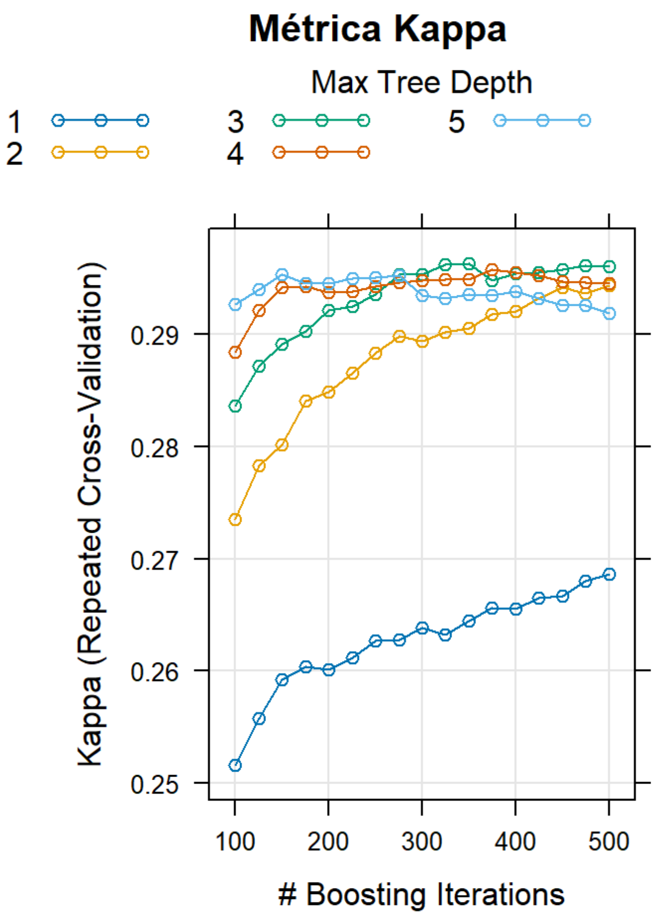
```
## 0.6301736
## 0.6300694
## 0.6305816
## 0.6313368
## 0.6313628
## 0.6319358
## 0.6316059
## 0.6322049
## 0.6328212
## 0.6327431
## 0.6332378
## 0.6333420
## 0.6340017
## 0.6343316
## 0.6367535
## 0.6391667
## 0.6401042
## 0.6420226
## 0.6424479
## 0.6432726
## 0.6441753
## 0.6449392
## 0.6446875
## 0.6451215
## 0.6453038
## 0.6458854
## 0.6460330
## 0.6466059
## 0.6471181
## 0.6468490
## 0.6472135
## 0.6418142
## 0.6436024
## 0.6445747
## 0.6451389
## 0.6460851
## 0.6462587
## 0.6467969
## 0.6476823
## 0.6476910
## 0.6481163
## 0.6481510
## 0.6474132
## 0.6477257
## 0.6477691
## 0.6479080
## 0.6480903
## 0.6480295
## 0.6442274
## 0.6460851
## 0.6471181
## 0.6471354
## 0.6468750
## 0.6469444
## 0.6471354
```

```
##   0.6473264
##   0.6473958
##   0.6474653
##   0.6474566
##   0.6479167
##   0.6477778
##   0.6476476
##   0.6473698
##   0.6473264
##   0.6472656
##   0.6463281
##   0.6470313
##   0.6476736
##   0.6472656
##   0.6472830
##   0.6475260
##   0.6475608
##   0.6476215
##   0.6467361
##   0.6466233
##   0.6468142
##   0.6467535
##   0.6469184
##   0.6466146
##   0.6462934
##   0.6463021
##   0.6459288
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## logLoss was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 350, interaction.depth =
##  5, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
# Metricas
grafico_metricas(GBM_MA_train)
```








```
# Resultados
resultados(GBM_MA_train, "Stochastic Gradient Boosting")
```

RESULTADOS DEL MODELO Stochastic Gradient Boosting

shrinkage	interaction.depth	n.minobsinnode	n.trees	logLoss	AUC	prAUC	Accu
0.1	1	10	100	1.421799	0.7104177	0.4074903	0.4012
0.1	2	10	100	1.378019	0.7310384	0.4380791	0.4188
0.1	3	10	100	1.353801	0.7419725	0.4536293	0.4269
0.1	4	10	100	1.340799	0.7469502	0.4609868	0.4307
0.1	5	10	100	1.331033	0.7510332	0.4664366	0.4341
0.1	1	10	125	1.415076	0.7133146	0.4113238	0.4045
0.1	2	10	125	1.369057	0.7349097	0.4432524	0.4226
0.1	3	10	125	1.345573	0.7451882	0.4577480	0.4297
0.1	4	10	125	1.333286	0.7496843	0.4645747	0.4337
0.1	5	10	125	1.324844	0.7529935	0.4689662	0.4352
0.1	1	10	150	1.409956	0.7154615	0.4141833	0.4074
0.1	2	10	150	1.361603	0.7381611	0.4475211	0.4241

```
# Mejore modelo
mejor_modelo(GBM_MA_train)
```

```
## [1] "El mejor módelo es el que muestra los siguientes hiperparámetros:"
```

	n.trees	interaction.depth	shrinkage	n.minobsinnode
79	350	5	0.1	10

```
# Curvas ROC y AUC
curvas_ROC(GBM_MA_train, "de Stochastic Gradient Boosting", train_MA_num, test_MA_num)
```

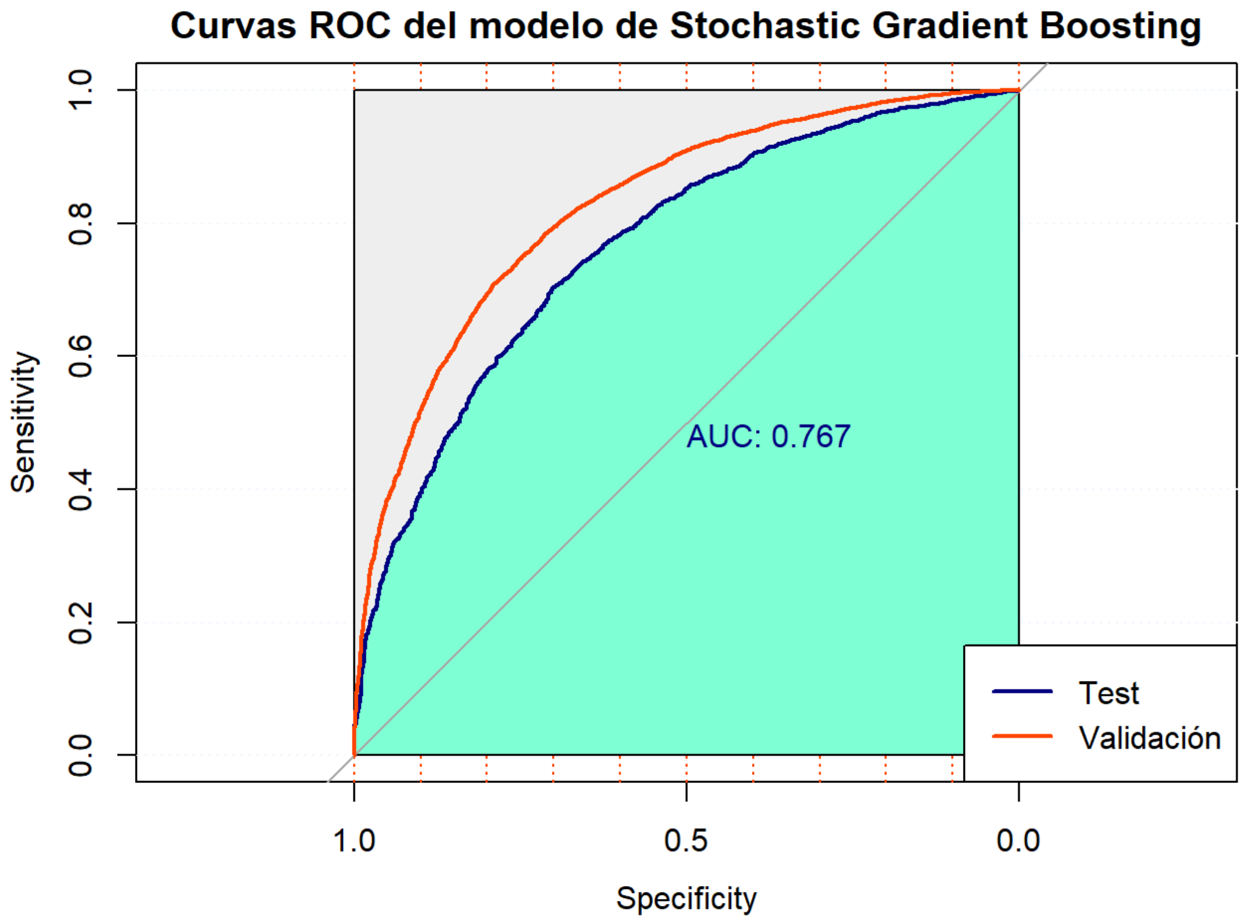
```
## Warning in roc.default(train[, c(length(train))], pred_train[, clase]):
## 'response' has more than two levels. Consider setting 'levels' explicitly or
## using 'multiclass.roc' instead
```

```
## Setting levels: control = Critical.Events, case = Maritime.Accidents
```

```
## Setting direction: controls < cases
```

```
## Warning in roc.default(test[, c(length(test))], pred_test[, clase]): 'response'
## has more than two levels. Consider setting 'levels' explicitly or using
## 'multiclass.roc' instead
```

```
## Setting levels: control = Critical.Events, case = Maritime.Accidents
## Setting direction: controls < cases
```




```
## [1] "ROC del modelo con el fichero de test: 0.767169290123457"
```

```
# Validación, Matriz de confusión
validation(GBM_MA_train, "Stochastic Gradient Boosting", train_MA_num, test_MA_num)
```

```
## [1] "Modelo Stochastic Gradient Boosting - Tabla de confusión para los datos de entrena
miento"
## Confusion Matrix and Statistics
##
##                      Reference
## Prediction            Critical.Events Maritime.Accidents Material.Issues
##   Critical.Events                2650                610            1253
##   Maritime.Accidents              976               4262             932
##   Material.Issues                1948                924            3532
##   Onboard.Emergencies             847                616             773
##   Third.party.Damages             779                788             710
##                      Reference
## Prediction            Onboard.Emergencies Third.party.Damages
##   Critical.Events                     843                 635
##   Maritime.Accidents                  674                 684
##   Material.Issues                    1055                 641
##   Onboard.Emergencies                3878                 680
##   Third.party.Damages                 750                4560
##
## Overall Statistics
##
##                Accuracy : 0.5245
##                  95% CI : (0.5193, 0.5297)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4056
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Critical.Events Class: Maritime.Accidents
## Sensitivity                         0.36806                    0.5919
## Specificity                         0.88399                    0.8866
## Pos Pred Value                      0.44233                    0.5662
## Neg Pred Value                      0.84838                    0.8968
## Prevalence                          0.20000                    0.2000
## Detection Rate                      0.07361                    0.1184
## Detection Prevalence                0.16642                    0.2091
## Balanced Accuracy                   0.62602                    0.7393
##                      Class: Material.Issues Class: Onboard.Emergencies
## Sensitivity                         0.49056                     0.5386
## Specificity                         0.84139                     0.8988
## Pos Pred Value                      0.43605                     0.5708
## Neg Pred Value                      0.86853                     0.8863
## Prevalence                          0.20000                     0.2000
## Detection Rate                      0.09811                     0.1077
## Detection Prevalence                0.22500                     0.1887
## Balanced Accuracy                   0.66597                     0.7187
##                      Class: Third.party.Damages
## Sensitivity                              0.6333
## Specificity                              0.8949
## Pos Pred Value                           0.6010
```

```
## Neg Pred Value                            0.9071
## Prevalence                                0.2000
## Detection Rate                            0.1267
## Detection Prevalence                      0.2107
## Balanced Accuracy                         0.7641
## [1] "Modelo Stochastic Gradient Boosting - Tabla de confusión para los datos de validac
ión"
```

```
## Confusion Matrix and Statistics
##
##                       Reference
## Prediction             Critical.Events Maritime.Accidents Material.Issues
##   Critical.Events                  444                173             452
##   Maritime.Accidents               318                939             278
##   Material.Issues                  595                290             667
##   Onboard.Emergencies              226                183             201
##   Third.party.Damages              217                215             202
##                       Reference
## Prediction             Onboard.Emergencies Third.party.Damages
##   Critical.Events                      258                 155
##   Maritime.Accidents                   188                 190
##   Material.Issues                      297                 198
##   Onboard.Emergencies                  848                 213
##   Third.party.Damages                  209                1044
##
## Overall Statistics
##
##                Accuracy : 0.438
##                  95% CI : (0.4277, 0.4483)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2975
##
##  Mcnemar's Test P-Value : 4.845e-16
##
## Statistics by Class:
##
##                       Class: Critical.Events Class: Maritime.Accidents
## Sensitivity                          0.24667                    0.5217
## Specificity                          0.85583                    0.8647
## Pos Pred Value                       0.29960                    0.4909
## Neg Pred Value                       0.81963                    0.8785
## Prevalence                           0.20000                    0.2000
## Detection Rate                       0.04933                    0.1043
## Detection Prevalence                 0.16467                    0.2126
## Balanced Accuracy                    0.55125                    0.6932
##                       Class: Material.Issues Class: Onboard.Emergencies
## Sensitivity                          0.37056                    0.47111
## Specificity                          0.80833                    0.88569
## Pos Pred Value                       0.32584                    0.50748
## Neg Pred Value                       0.83705                    0.87011
## Prevalence                           0.20000                    0.20000
## Detection Rate                       0.07411                    0.09422
## Detection Prevalence                 0.22744                    0.18567
## Balanced Accuracy                    0.58944                    0.67840
##                       Class: Third.party.Damages
## Sensitivity                               0.5800
## Specificity                               0.8829
## Pos Pred Value                            0.5533
## Neg Pred Value                            0.8937
## Prevalence                                0.2000
```

```
## Detection Rate                              0.1160
## Detection Prevalence                        0.2097
## Balanced Accuracy                           0.7315
```

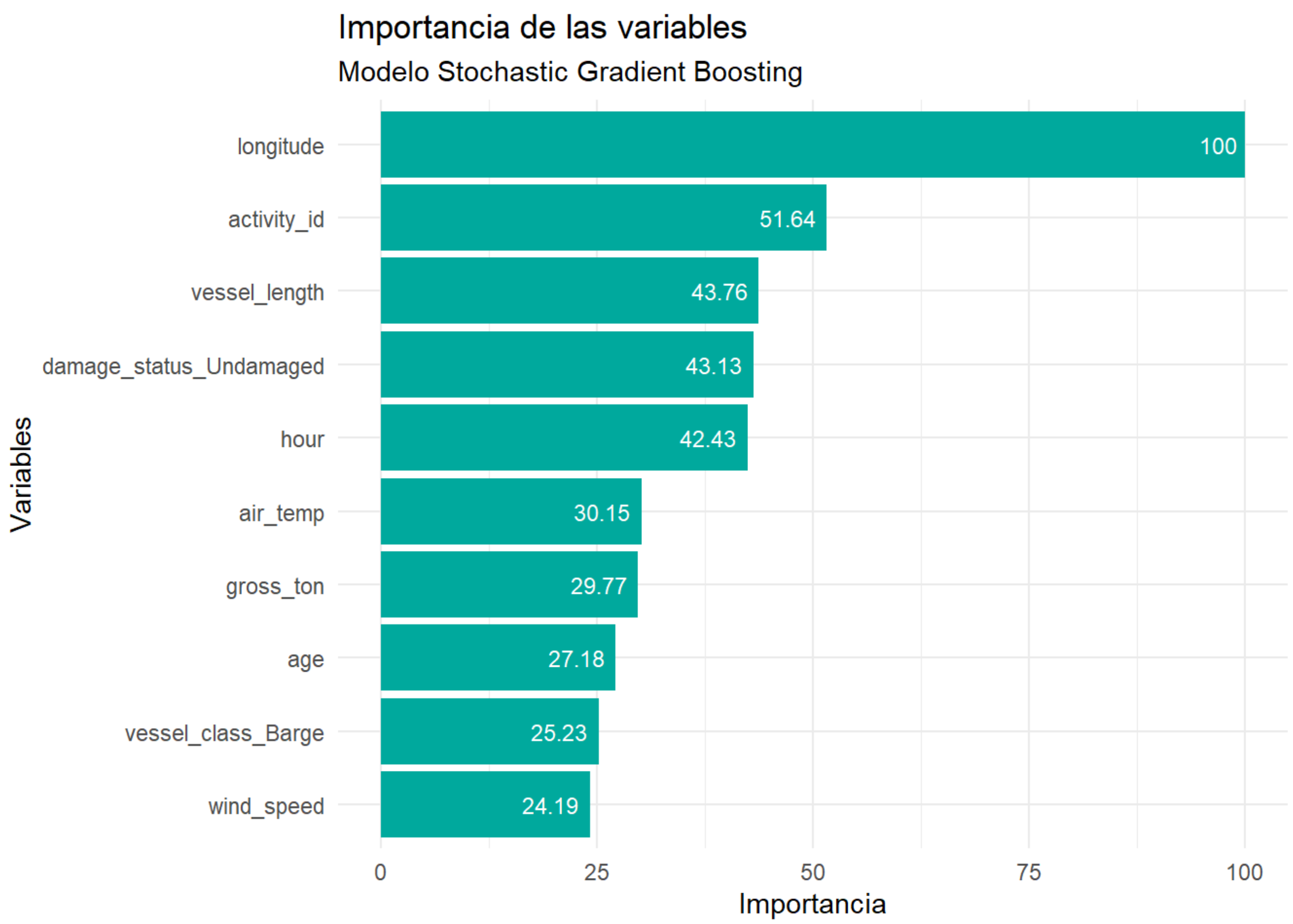
Resumen de métricas

```
# Resumen
resumen_MA_GBM <- resumen_multiclass(GBM_MA_train, train_MA_num, test_MA_num)

# Presentación
resumen_MA_GBM %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ",
            "Stochastic Gradient Boosting " = 5))
```

	Stochastic Gradient Boosting				
	AUC	**Accuracy**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.820	0.524	0.406	0.524	0.881
Datos Validación	0.759	0.438	0.298	0.438	0.860

Importancia de las variables

```
# Importancia de variables por cada valor de predicción
#GBM_MA_train$modelInfo$varImp <- NULL   # Anular la función para solucionar bug
importancia_var(GBM_MA_train, "Stochastic Gradient Boosting")
```



2.3. Otros modelos

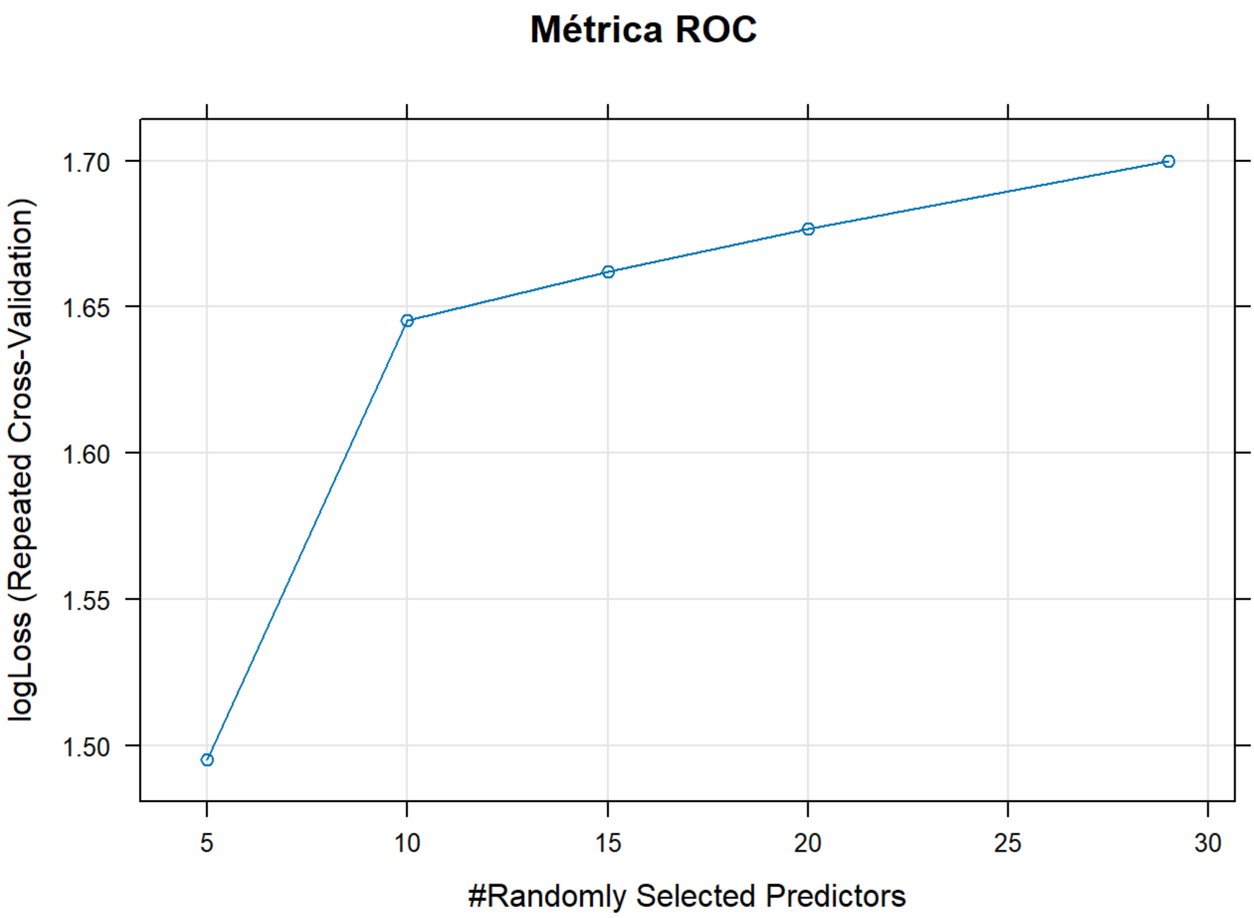
2.3.1. Random Forest

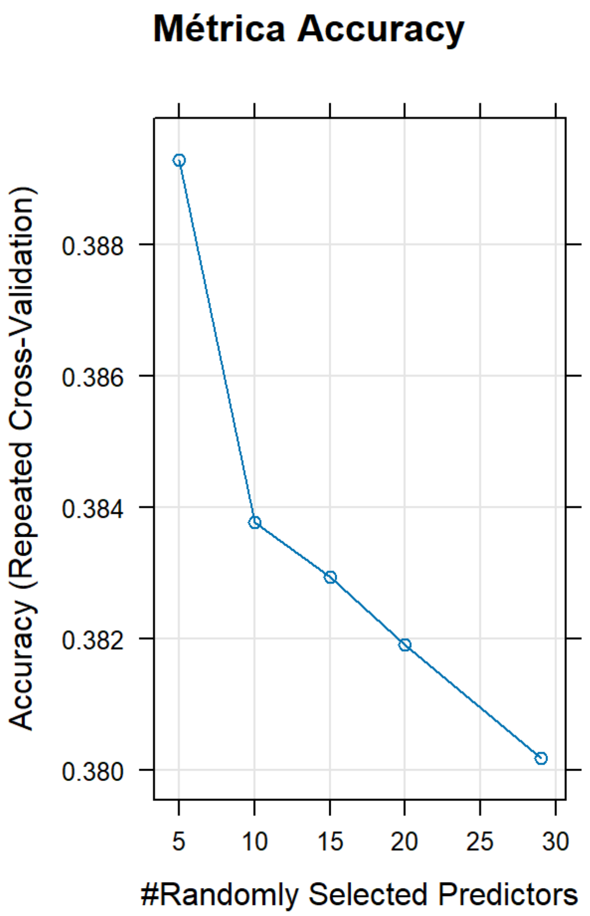
```r
# Entrenamiento
if (train_switch  == 1) {
set.seed(7)
tic()

clusterCPU <- makePSOCKcluster(detectCores()-1)
registerDoParallel(clusterCPU)

rfGrid <-  expand.grid(mtry = c(5,10,15,20,29))

rf_MA_train <- train(y ~ ., data = train_MA_general,
                  method = "rf", metric = metrica,
                  #preProc = c("center", "scale"),
                  trControl = control,
                  tuneGrid = rfGrid)

stopCluster(clusterCPU)

saveRDS(rf_MA_train, "Models/rf_MA_train.RDS")
toc()

}else{
  rf_MA_train <- readRDS("Models/rf_MA_train.RDS")
}
```

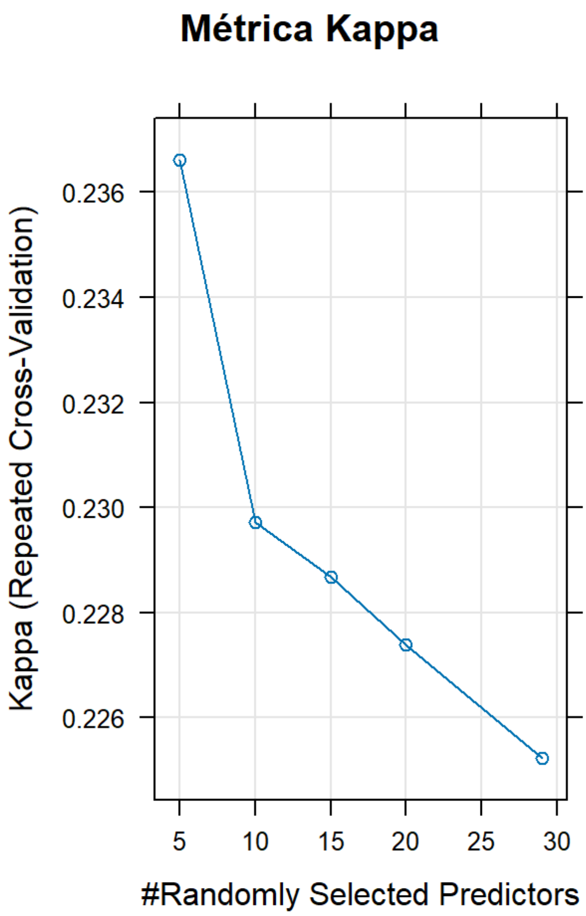
```r
# Resultados
rf_MA_train
```

```
## Random Forest
##
## 36000 samples
##    12 predictor
##     5 classes: 'Critical.Events', 'Maritime.Accidents', 'Material.Issues', 'Onboard.Eme
rgencies', 'Third.party.Damages'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 31500, 31500, 31500, 31500, 31500, 31500, ...
## Resampling results across tuning parameters:
##
##   mtry  logLoss   AUC        prAUC      Accuracy   Kappa      Mean_F1
##    5    1.495059  0.7145125  0.4276375  0.3892917  0.2366146  0.3901783
##   10    1.645300  0.6841079  0.4026863  0.3837778  0.2297222  0.3857173
##   15    1.661983  0.6836756  0.4022318  0.3829444  0.2286806  0.3849418
##   20    1.676596  0.6847599  0.4020952  0.3819167  0.2273958  0.3843439
##   29    1.699858  0.6854478  0.4000137  0.3801806  0.2252257  0.3830582
##   Mean_Sensitivity  Mean_Specificity  Mean_Pos_Pred_Value  Mean_Neg_Pred_Value
##   0.3892917         0.8473229         0.3924425            0.8471848
##   0.3837778         0.8459444         0.3882125            0.8457048
##   0.3829444         0.8457361         0.3875009            0.8454887
##   0.3819167         0.8454792         0.3873449            0.8451738
##   0.3801806         0.8450451         0.3865740            0.8446818
##   Mean_Precision  Mean_Recall  Mean_Detection_Rate  Mean_Balanced_Accuracy
##   0.3924425       0.3892917    0.07785833           0.6183073
##   0.3882125       0.3837778    0.07675556           0.6148611
##   0.3875009       0.3829444    0.07658889           0.6143403
##   0.3873449       0.3819167    0.07638333           0.6136979
##   0.3865740       0.3801806    0.07603611           0.6126128
##
## logLoss was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 5.
```

```
# Métricas
grafico_metricas(rf_MA_train)
```








```
# Resultados
resultados(rf_MA_train, "Random Forest")
```

RESULTADOS DEL MODELO Random Forest

mtry	logLoss	AUC	prAUC	Accuracy	Kappa	Mean_F1	Mean_Sensitivity	Me
5	1.495059	0.7145125	0.4276375	0.3892917	0.2366146	0.3901783	0.3892917	
10	1.645301	0.6841079	0.4026863	0.3837778	0.2297222	0.3857173	0.3837778	
15	1.661983	0.6836756	0.4022318	0.3829444	0.2286806	0.3849418	0.3829444	
20	1.676596	0.6847599	0.4020952	0.3819167	0.2273958	0.3843439	0.3819167	
29	1.699858	0.6854478	0.4000137	0.3801806	0.2252257	0.3830582	0.3801806	

```
# Mejor modelo
mejor_modelo(rf_MA_train)
```

```
## [1] "El mejor módelo es el que muestra los siguientes hiperparámetros:"
```

mtry
5

```
# Curvas ROC y AUC
curvas_ROC(rf_MA_train, "de Random Forest", train_MA_general, test_MA_general)
```

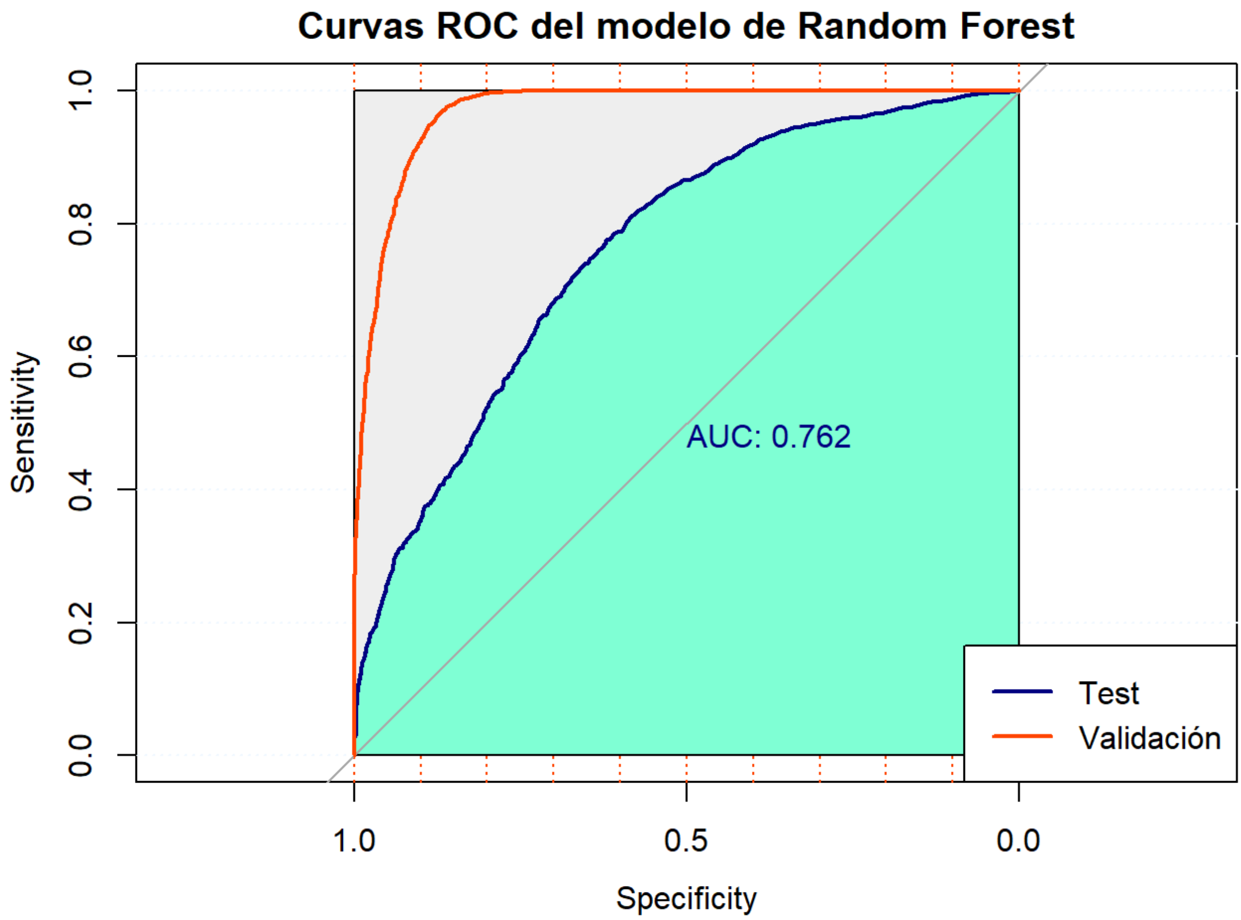
```
## Warning in roc.default(train[, c(length(train))], pred_train[, clase]):
## 'response' has more than two levels. Consider setting 'levels' explicitly or
## using 'multiclass.roc' instead
```

```
## Setting levels: control = Critical.Events, case = Maritime.Accidents
```

```
## Setting direction: controls < cases
```

```
## Warning in roc.default(test[, c(length(test))], pred_test[, clase]): 'response'
## has more than two levels. Consider setting 'levels' explicitly or using
## 'multiclass.roc' instead
```

```
## Setting levels: control = Critical.Events, case = Maritime.Accidents
## Setting direction: controls < cases
```




```
## [1] "ROC del modelo con el fichero de test: 0.761791203703704"
```

```
# Validación, Matriz de confusión
validation(rf_MA_train, "RF", train_MA_general, test_MA_general)
```

```
## [1] "Modelo RF - Tabla de confusión para los datos de entrenamiento"
## Confusion Matrix and Statistics
##
##                      Reference
## Prediction            Critical.Events Maritime.Accidents Material.Issues
##   Critical.Events                5148                211             731
##   Maritime.Accidents              462               6253             446
##   Material.Issues                1065                308            5430
##   Onboard.Emergencies             292                227             356
##   Third.party.Damages             233                201             237
##                      Reference
## Prediction            Onboard.Emergencies Third.party.Damages
##   Critical.Events                     414                 305
##   Maritime.Accidents                  205                 209
##   Material.Issues                     543                 195
##   Onboard.Emergencies                5760                 275
##   Third.party.Damages                 278                6216
##
## Overall Statistics
##
##                Accuracy : 0.8002
##                  95% CI : (0.796, 0.8043)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7502
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Critical.Events Class: Maritime.Accidents
## Sensitivity                          0.7150                    0.8685
## Specificity                          0.9423                    0.9541
## Pos Pred Value                       0.7561                    0.8255
## Neg Pred Value                       0.9297                    0.9667
## Prevalence                           0.2000                    0.2000
## Detection Rate                       0.1430                    0.1737
## Detection Prevalence                 0.1891                    0.2104
## Balanced Accuracy                    0.8287                    0.9113
##                      Class: Material.Issues Class: Onboard.Emergencies
## Sensitivity                          0.7542                     0.8000
## Specificity                          0.9267                     0.9601
## Pos Pred Value                       0.7201                     0.8336
## Neg Pred Value                       0.9378                     0.9505
## Prevalence                           0.2000                     0.2000
## Detection Rate                       0.1508                     0.1600
## Detection Prevalence                 0.2095                     0.1919
## Balanced Accuracy                    0.8404                     0.8800
##                      Class: Third.party.Damages
## Sensitivity                              0.8633
## Specificity                              0.9670
## Pos Pred Value                           0.8676
## Neg Pred Value                           0.9659
```

```
## Prevalence                                 0.2000
## Detection Rate                             0.1727
## Detection Prevalence                       0.1990
## Balanced Accuracy                          0.9152
## [1] "Modelo RF - Tabla de confusión para los datos de validación"
```

```
## Confusion Matrix and Statistics
##
##                       Reference
## Prediction             Critical.Events Maritime.Accidents Material.Issues
##   Critical.Events                  282                178             707
##   Maritime.Accidents               299                868             272
##   Material.Issues                  804                352             442
##   Onboard.Emergencies              235                168             220
##   Third.party.Damages              180                234             159
##                       Reference
## Prediction             Onboard.Emergencies Third.party.Damages
##   Critical.Events                      271                 198
##   Maritime.Accidents                   167                 201
##   Material.Issues                      347                 188
##   Onboard.Emergencies                  803                 189
##   Third.party.Damages                  212                1024
##
## Overall Statistics
##
##                Accuracy : 0.3799
##                  95% CI : (0.3698, 0.39)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2249
##
##  Mcnemar's Test P-Value : 4.573e-14
##
## Statistics by Class:
##
##                       Class: Critical.Events Class: Maritime.Accidents
## Sensitivity                          0.15667                   0.48222
## Specificity                          0.81194                   0.86958
## Pos Pred Value                       0.17237                   0.48035
## Neg Pred Value                       0.79386                   0.87043
## Prevalence                           0.20000                   0.20000
## Detection Rate                       0.03133                   0.09644
## Detection Prevalence                 0.18178                   0.20078
## Balanced Accuracy                    0.48431                   0.67590
##                       Class: Material.Issues Class: Onboard.Emergencies
## Sensitivity                          0.24556                    0.44611
## Specificity                          0.76514                    0.88722
## Pos Pred Value                       0.20722                    0.49721
## Neg Pred Value                       0.80224                    0.86500
## Prevalence                           0.20000                    0.20000
## Detection Rate                       0.04911                    0.08922
## Detection Prevalence                 0.23700                    0.17944
## Balanced Accuracy                    0.50535                    0.66667
##                       Class: Third.party.Damages
## Sensitivity                               0.5689
## Specificity                               0.8910
## Pos Pred Value                            0.5661
## Neg Pred Value                            0.8921
## Prevalence                                0.2000
```

```
## Detection Rate                          0.1138
## Detection Prevalence                    0.2010
## Balanced Accuracy                       0.7299
```

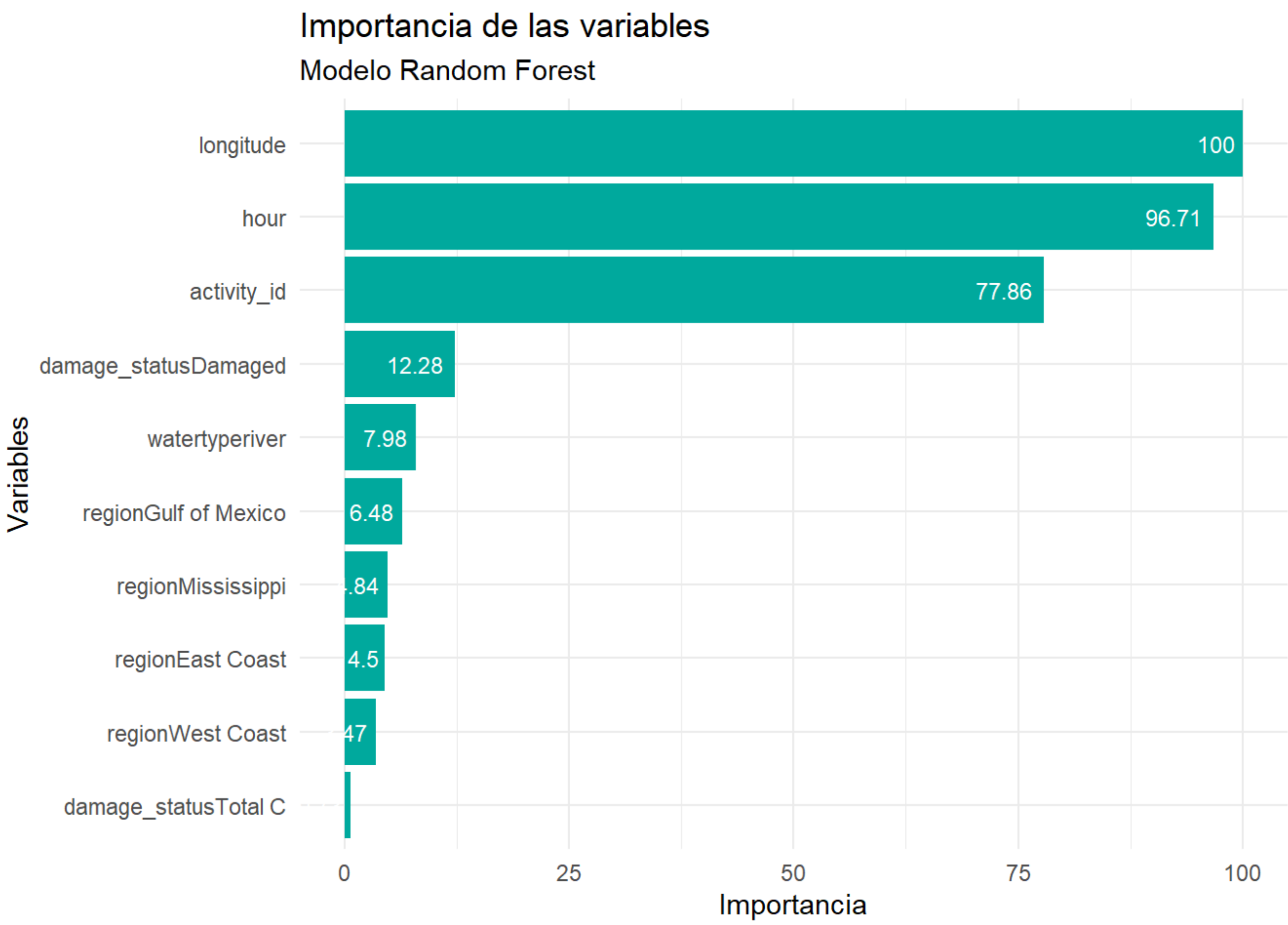
Resumen de métricas

```
# Resumen
resumen_MA_rf <- resumen_multiclass(rf_MA_train, train_MA_general, test_MA_general)

# Presentación
resumen_MA_rf %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ", "Random forest " = 5))
```

	Random forest				
	AUC	**Accuracy**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.969	0.80	0.750	0.80	0.950
Datos Validación	0.712	0.38	0.225	0.38	0.845

Importancia de las variables

```
# Importancia general de variables
importancia_var_overall(rf_MA_train, "Random Forest")
```



2.3.2. Perceptrón Multicapa

```
if (train_switch  == 1) {
set.seed(7)

tic()

clusterCPU <- makePSOCKcluster(detectCores()-1)
registerDoParallel(clusterCPU)

nnetGrid <-  expand.grid(size = c(1:10),
                         decay =c(0.01, 0.05, 0.5 ,0.1))

nnet_MA_train <- train(y ~ .,
                    data = train_MA_general,
                    method = "nnet",
                    metric = metrica,
                    #preProc = c("center", "scale"),
                    trControl = control,
                    tuneGrid = nnetGrid)

stopCluster(clusterCPU)

saveRDS(nnet_MA_train, "Models/nnet_MA_train.RDS")

toc()

}else{
  nnet_MA_train <- readRDS("Models/nnet_MA_train.RDS")
}
```

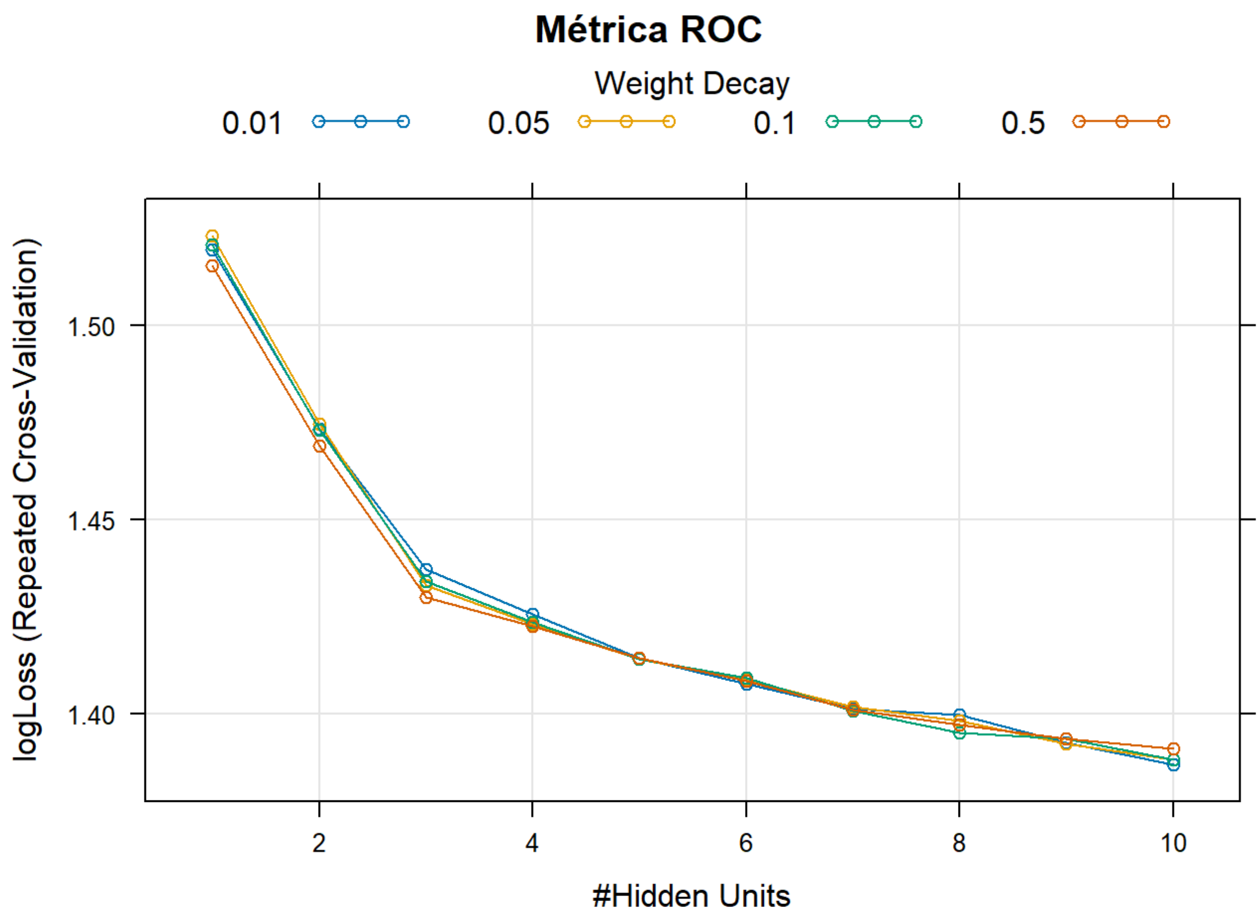
```
# Resultados
nnet_MA_train
```

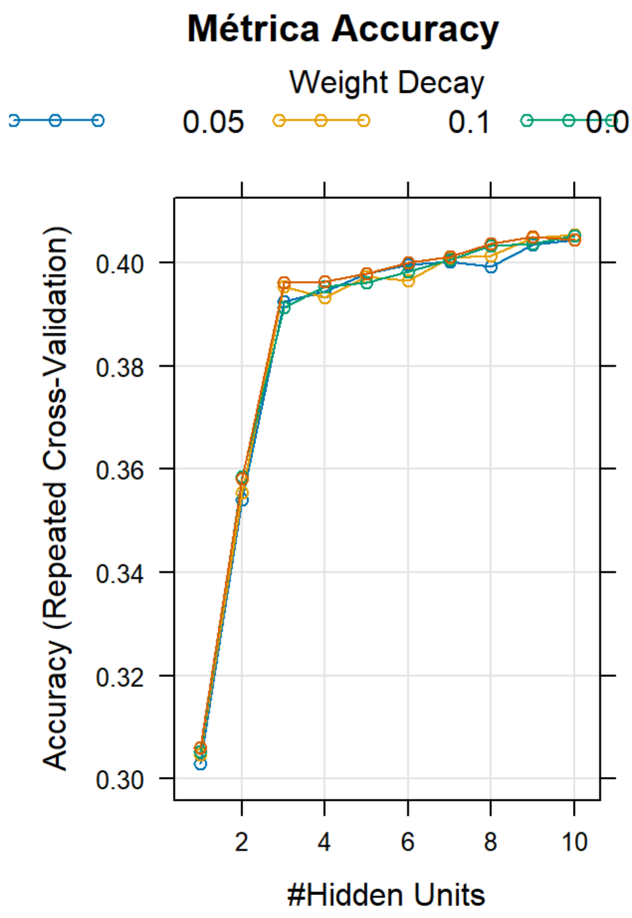
```
## Neural Network
##
## 36000 samples
##    12 predictor
##     5 classes: 'Critical.Events', 'Maritime.Accidents', 'Material.Issues', 'Onboard.Eme
rgencies', 'Third.party.Damages'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 31500, 31500, 31500, 31500, 31500, 31500, ...
## Resampling results across tuning parameters:
##
##   size  decay  logLoss   AUC        prAUC      Accuracy   Kappa      Mean_F1
##    1    0.01   1.519664  0.6221473  0.2804625  0.3029722  0.1287153  0.2454748
##    1    0.05   1.523253  0.6186305  0.2925513  0.3047222  0.1309028  0.2381025
##    1    0.10   1.520877  0.6199773  0.2966987  0.3052361  0.1315451  0.2393792
##    1    0.50   1.515592  0.6232340  0.2976701  0.3059722  0.1324653  0.2452584
##    2    0.01   1.473517  0.6682673  0.3498832  0.3540833  0.1926042  0.3243782
##    2    0.05   1.474719  0.6668120  0.3503621  0.3554028  0.1942535  0.3188136
##    2    0.10   1.473079  0.6696913  0.3505570  0.3585139  0.1981424  0.3269261
##    2    0.50   1.469200  0.6701568  0.3532544  0.3581944  0.1977431  0.3266217
##    3    0.01   1.437262  0.6956643  0.3816074  0.3924583  0.2405729  0.3632169
##    3    0.05   1.433127  0.6987220  0.3880050  0.3954167  0.2442708  0.3661203
##    3    0.10   1.434277  0.6972084  0.3869012  0.3913333  0.2391667  0.3630374
##    3    0.50   1.430214  0.7006509  0.3923930  0.3962083  0.2452604  0.3701908
##    4    0.01   1.425801  0.7033451  0.3945261  0.3944306  0.2430382  0.3745539
##    4    0.05   1.423084  0.7037890  0.3957770  0.3933194  0.2416493  0.3727188
##    4    0.10   1.423768  0.7042857  0.3958977  0.3953889  0.2442361  0.3745832
##    4    0.50   1.422541  0.7055508  0.3981398  0.3962917  0.2453646  0.3755291
##    5    0.01   1.414363  0.7089816  0.4027331  0.3977917  0.2472396  0.3818927
##    5    0.05   1.414279  0.7091442  0.4022228  0.3972500  0.2465625  0.3784312
##    5    0.10   1.414252  0.7088743  0.4020976  0.3961389  0.2451736  0.3768904
##    5    0.50   1.414337  0.7096713  0.4035351  0.3978750  0.2473437  0.3843152
##    6    0.01   1.407686  0.7128909  0.4073919  0.3995833  0.2494792  0.3848615
##    6    0.05   1.408870  0.7117183  0.4066203  0.3965556  0.2456944  0.3827887
##    6    0.10   1.409246  0.7119439  0.4066963  0.3981944  0.2477431  0.3840940
##    6    0.50   1.408543  0.7129201  0.4091744  0.3999583  0.2499479  0.3854345
##    7    0.01   1.401368  0.7159617  0.4127287  0.4002500  0.2503125  0.3895161
##    7    0.05   1.401892  0.7156687  0.4116978  0.4009028  0.2511285  0.3900581
##    7    0.10   1.400769  0.7164598  0.4157054  0.4006250  0.2507812  0.3868146
##    7    0.50   1.401184  0.7161484  0.4129168  0.4012083  0.2515104  0.3875725
##    8    0.01   1.399856  0.7167071  0.4142098  0.3993056  0.2491319  0.3894194
##    8    0.05   1.398312  0.7180932  0.4149825  0.4014167  0.2517708  0.3899967
##    8    0.10   1.395315  0.7190334  0.4176899  0.4033194  0.2541493  0.3929079
##    8    0.50   1.397300  0.7181131  0.4149385  0.4035972  0.2544965  0.3913921
##    9    0.01   1.392700  0.7203887  0.4194174  0.4034861  0.2543576  0.3943171
##    9    0.05   1.392283  0.7203482  0.4212082  0.4048611  0.2560764  0.3965004
##    9    0.10   1.393632  0.7200236  0.4199040  0.4036944  0.2546181  0.3936248
##    9    0.50   1.393728  0.7199400  0.4186684  0.4050139  0.2562674  0.3963877
##   10    0.01   1.387036  0.7229238  0.4230512  0.4044444  0.2555556  0.3966414
##   10    0.05   1.388362  0.7229946  0.4234236  0.4054444  0.2568056  0.3956124
##   10    0.10   1.388425  0.7221781  0.4216582  0.4052778  0.2565972  0.3949243
##   10    0.50   1.391190  0.7210525  0.4199281  0.4044028  0.2555035  0.3942856
##   Mean_Sensitivity  Mean_Specificity  Mean_Pos_Pred_Value  Mean_Neg_Pred_Value
```

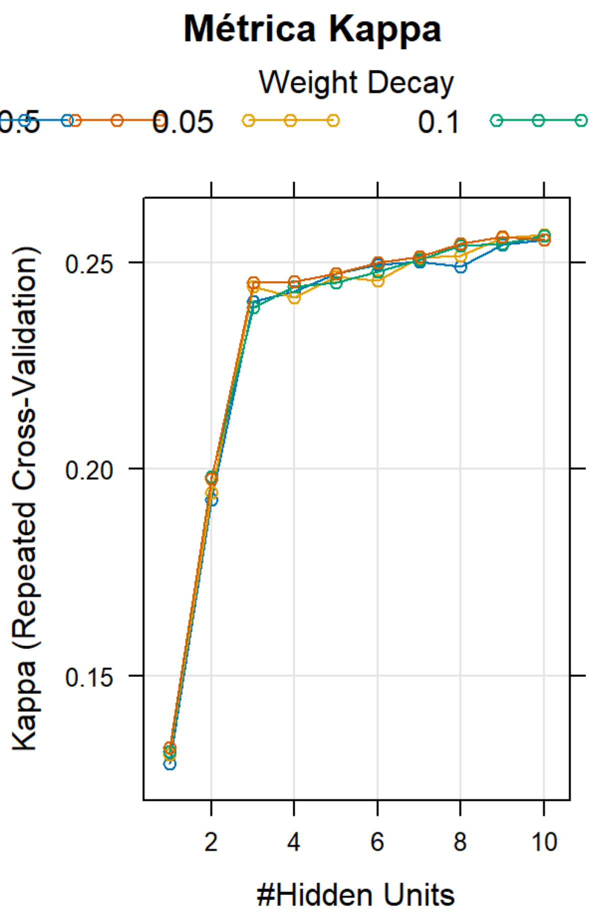
```
##    0.3029722         0.8257431         0.2898637          0.8335787
##    0.3047222         0.8261806         0.2922597          0.8349343
##    0.3052361         0.8263090         0.2935207          0.8347606
##    0.3059722         0.8264931         0.2930119          0.8341674
##    0.3540833         0.8385208         0.3365148          0.8416841
##    0.3554028         0.8388507         0.3378112          0.8424567
##    0.3585139         0.8396285         0.3427305          0.8427800
##    0.3581944         0.8395486         0.3409425          0.8426029
##    0.3924583         0.8481146         0.3770260          0.8507548
##    0.3954167         0.8488542         0.3808550          0.8514993
##    0.3913333         0.8478333         0.3801380          0.8504925
##    0.3962083         0.8490521         0.3857534          0.8515221
##    0.3944306         0.8486076         0.3841748          0.8505585
##    0.3933194         0.8483299         0.3806230          0.8503764
##    0.3953889         0.8488472         0.3848521          0.8508809
##    0.3962917         0.8490729         0.3834695          0.8510396
##    0.3977917         0.8494479         0.3891054          0.8511266
##    0.3972500         0.8493125         0.3849283          0.8511562
##    0.3961389         0.8490347         0.3848771          0.8509333
##    0.3978750         0.8494687         0.3890023          0.8508881
##    0.3995833         0.8498958         0.3905732          0.8514393
##    0.3965556         0.8491389         0.3862825          0.8505599
##    0.3981944         0.8495486         0.3904402          0.8510200
##    0.3999583         0.8499896         0.3909866          0.8515088
##    0.4002500         0.8500625         0.3919547          0.8512231
##    0.4009028         0.8502257         0.3928632          0.8513997
##    0.4006250         0.8501562         0.3941579          0.8516278
##    0.4012083         0.8503021         0.3913496          0.8517380
##    0.3993056         0.8498264         0.3918934          0.8508962
##    0.4014167         0.8503542         0.3924264          0.8515919
##    0.4033194         0.8508299         0.3963418          0.8519874
##    0.4035972         0.8508993         0.3959842          0.8522031
##    0.4034861         0.8508715         0.3969177          0.8519145
##    0.4048611         0.8512153         0.3998250          0.8521630
##    0.4036944         0.8509236         0.3963415          0.8520344
##    0.4050139         0.8512535         0.3989243          0.8522383
##    0.4044444         0.8511111         0.3998573          0.8520292
##    0.4054444         0.8513611         0.3997604          0.8524583
##    0.4052778         0.8513194         0.3990848          0.8524732
##    0.4044028         0.8511007         0.3964928          0.8522358
##    Mean_Precision  Mean_Recall  Mean_Detection_Rate  Mean_Balanced_Accuracy
##    0.2898637       0.3029722    0.06059444           0.5643576
##    0.2922597       0.3047222    0.06094444           0.5654514
##    0.2935207       0.3052361    0.06104722           0.5657726
##    0.2930119       0.3059722    0.06119444           0.5662326
##    0.3365148       0.3540833    0.07081667           0.5963021
##    0.3378112       0.3554028    0.07108056           0.5971267
##    0.3427305       0.3585139    0.07170278           0.5990712
##    0.3409425       0.3581944    0.07163889           0.5988715
##    0.3770260       0.3924583    0.07849167           0.6202865
##    0.3808550       0.3954167    0.07908333           0.6221354
##    0.3801380       0.3913333    0.07826667           0.6195833
##    0.3857534       0.3962083    0.07924167           0.6226302
##    0.3841748       0.3944306    0.07888611           0.6215191
##    0.3806230       0.3933194    0.07866389           0.6208247
```

```
##    0.3848521       0.3953889    0.07907778           0.6221181
##    0.3834695       0.3962917    0.07925833           0.6226823
##    0.3891054       0.3977917    0.07955833           0.6236198
##    0.3849283       0.3972500    0.07945000           0.6232813
##    0.3848771       0.3961389    0.07922778           0.6225868
##    0.3890023       0.3978750    0.07957500           0.6236719
##    0.3905732       0.3995833    0.07991667           0.6247396
##    0.3862825       0.3965556    0.07931111           0.6228472
##    0.3904402       0.3981944    0.07963889           0.6238715
##    0.3909866       0.3999583    0.07999167           0.6249740
##    0.3919547       0.4002500    0.08005000           0.6251562
##    0.3928632       0.4009028    0.08018056           0.6255642
##    0.3941579       0.4006250    0.08012500           0.6253906
##    0.3913496       0.4012083    0.08024167           0.6257552
##    0.3918934       0.3993056    0.07986111           0.6245660
##    0.3924264       0.4014167    0.08028333           0.6258854
##    0.3963418       0.4033194    0.08066389           0.6270747
##    0.3959842       0.4035972    0.08071944           0.6272483
##    0.3969177       0.4034861    0.08069722           0.6271788
##    0.3998250       0.4048611    0.08097222           0.6280382
##    0.3963415       0.4036944    0.08073889           0.6273090
##    0.3989243       0.4050139    0.08100278           0.6281337
##    0.3998573       0.4044444    0.08088889           0.6277778
##    0.3997604       0.4054444    0.08108889           0.6284028
##    0.3990848       0.4052778    0.08105556           0.6282986
##    0.3964928       0.4044028    0.08088056           0.6277517
## 
## logLoss was used to select the optimal model using the smallest value.
## The final values used for the model were size = 10 and decay = 0.01.
```

```
# Métricas
grafico_metricas(nnet_MA_train)
```








```
# Resultados
resultados(nnet_MA_train, "Perceptrón multicapa")
```

RESULTADOS DEL MODELO Perceptrón multicapa

size	decay	logLoss	AUC	prAUC	Accuracy	Kappa	Mean_F1	Mean_Sensiti
1	0.01	1.519664	0.6221473	0.2804625	0.3029722	0.1287153	0.2454748	0.3029
1	0.05	1.523253	0.6186305	0.2925513	0.3047222	0.1309028	0.2381025	0.3047
1	0.10	1.520877	0.6199773	0.2966987	0.3052361	0.1315451	0.2393792	0.3052
1	0.50	1.515592	0.6232340	0.2976701	0.3059722	0.1324653	0.2452584	0.3059
2	0.01	1.473517	0.6682673	0.3498832	0.3540833	0.1926042	0.3243782	0.3540
2	0.05	1.474719	0.6668120	0.3503621	0.3554028	0.1942535	0.3188136	0.3554
2	0.10	1.473078	0.6696913	0.3505570	0.3585139	0.1981424	0.3269261	0.3585
2	0.50	1.469200	0.6701568	0.3532544	0.3581944	0.1977431	0.3266217	0.3581
3	0.01	1.437262	0.6956643	0.3816074	0.3924583	0.2405729	0.3632169	0.3924
3	0.05	1.433128	0.6987220	0.3880050	0.3954167	0.2442708	0.3661203	0.3954
3	0.10	1.434277	0.6972084	0.3869012	0.3913333	0.2391667	0.3630374	0.3913
3	0.50	1.430214	0.7006509	0.3923930	0.3962083	0.2452604	0.3701908	0.3962

```
# Mejor modelo
mejor_modelo(nnet_MA_train)
```

```
## [1] "El mejor módelo es el que muestra los siguientes hiperparámetros:"
```

	size	decay
37	10	0.01

```
# Curvas ROC y AUC
curvas_ROC(nnet_MA_train, "Perceptrón multicapa", train_MA_general, test_MA_general)
```

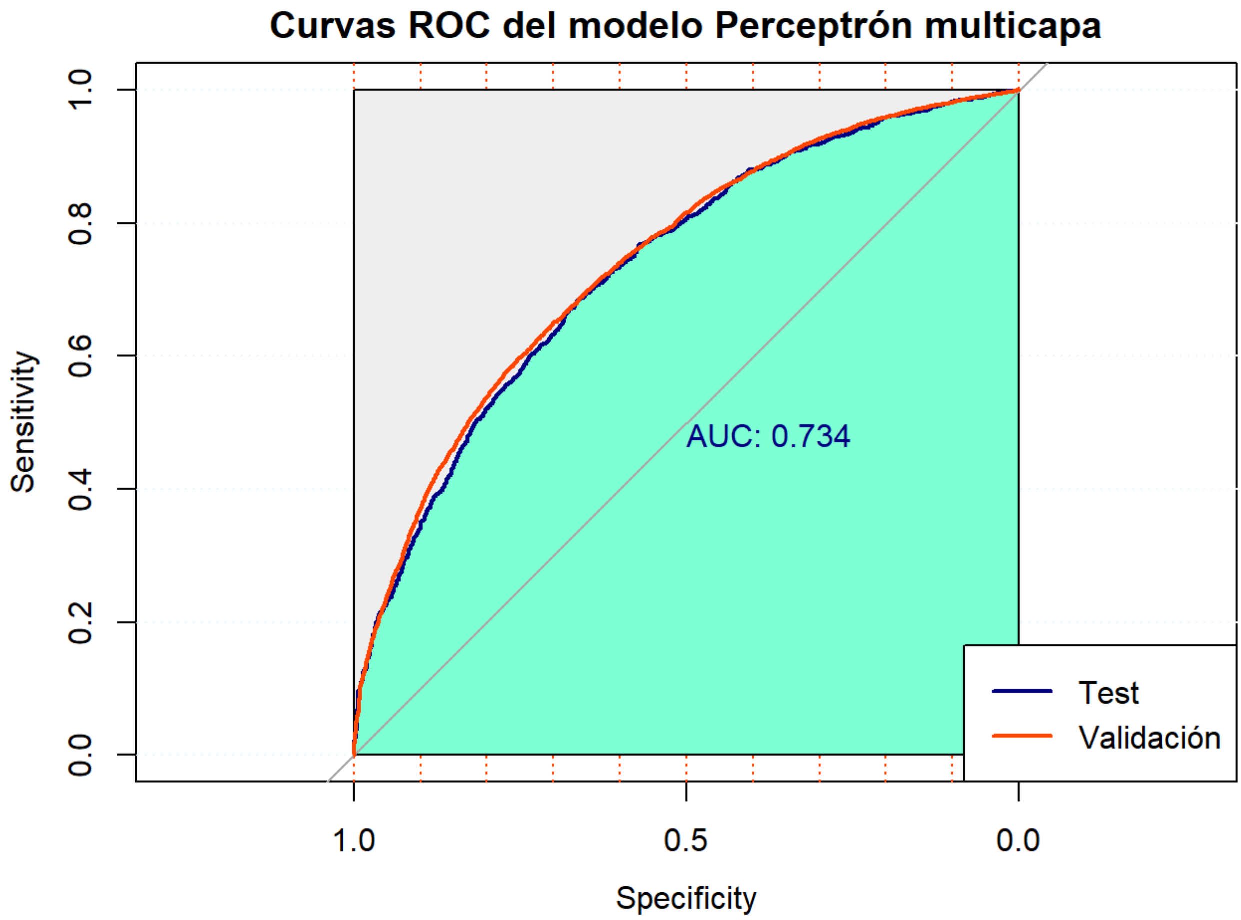
```
## Warning in roc.default(train[, c(length(train))], pred_train[, clase]):
## 'response' has more than two levels. Consider setting 'levels' explicitly or
## using 'multiclass.roc' instead
```

```
## Setting levels: control = Critical.Events, case = Maritime.Accidents
```

```
## Setting direction: controls < cases
```

```
## Warning in roc.default(test[, c(length(test))], pred_test[, clase]): 'response'
## has more than two levels. Consider setting 'levels' explicitly or using
## 'multiclass.roc' instead
```

```
## Setting levels: control = Critical.Events, case = Maritime.Accidents
## Setting direction: controls < cases
```




```
## [1] "ROC del modelo con el fichero de test: 0.734038117283951"
```

```
# Validación, Matriz de confusión
validation(nnet_MA_train, "Perceptrón multicapa", train_MA_general, test_MA_general)
```

```
## [1] "Modelo Perceptrón multicapa - Tabla de confusión para los datos de entrenamiento"
## Confusion Matrix and Statistics
##
##                      Reference
## Prediction            Critical.Events Maritime.Accidents Material.Issues
##   Critical.Events                1662                715            1238
##   Maritime.Accidents              986               3059             919
##   Material.Issues                2322               1405            3167
##   Onboard.Emergencies            1138                912             912
##   Third.party.Damages            1092               1109             964
##                      Reference
## Prediction            Onboard.Emergencies Third.party.Damages
##   Critical.Events                     936                 854
##   Maritime.Accidents                  739                 927
##   Material.Issues                    1447                 779
##   Onboard.Emergencies                3174                 766
##   Third.party.Damages                 904                3874
##
## Overall Statistics
##
##                Accuracy : 0.4149
##                  95% CI : (0.4098, 0.42)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2686
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Critical.Events Class: Maritime.Accidents
## Sensitivity                         0.23083                   0.42486
## Specificity                         0.87003                   0.87601
## Pos Pred Value                      0.30749                   0.46139
## Neg Pred Value                      0.81899                   0.85901
## Prevalence                          0.20000                   0.20000
## Detection Rate                      0.04617                   0.08497
## Detection Prevalence                0.15014                   0.18417
## Balanced Accuracy                   0.55043                   0.65043
##                      Class: Material.Issues Class: Onboard.Emergencies
## Sensitivity                         0.43986                    0.44083
## Specificity                         0.79330                    0.87056
## Pos Pred Value                      0.34726                    0.45987
## Neg Pred Value                      0.84996                    0.86164
## Prevalence                          0.20000                    0.20000
## Detection Rate                      0.08797                    0.08817
## Detection Prevalence                0.25333                    0.19172
## Balanced Accuracy                   0.61658                    0.65569
##                      Class: Third.party.Damages
## Sensitivity                              0.5381
## Specificity                              0.8587
## Pos Pred Value                           0.4877
## Neg Pred Value                           0.8815
```

```
## Prevalence                                        0.2000
## Detection Rate                                    0.1076
## Detection Prevalence                              0.2206
## Balanced Accuracy                                 0.6984
## [1] "Modelo Perceptrón multicapa - Tabla de confusión para los datos de validación"
```

```
## Confusion Matrix and Statistics
##
##                       Reference
## Prediction             Critical.Events Maritime.Accidents Material.Issues
##   Critical.Events                  397                186             339
##   Maritime.Accidents               272                776             229
##   Material.Issues                  576                365             792
##   Onboard.Emergencies              266                215             205
##   Third.party.Damages              289                258             235
##                       Reference
## Prediction             Onboard.Emergencies Third.party.Damages
##   Critical.Events                      213                 221
##   Maritime.Accidents                   196                 216
##   Material.Issues                      374                 211
##   Onboard.Emergencies                  777                 214
##   Third.party.Damages                  240                 938
##
## Overall Statistics
##
##                Accuracy : 0.4089
##                  95% CI : (0.3987, 0.4191)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2611
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Critical.Events Class: Maritime.Accidents
## Sensitivity                         0.22056                   0.43111
## Specificity                         0.86681                   0.87319
## Pos Pred Value                      0.29277                   0.45944
## Neg Pred Value                      0.81646                   0.85994
## Prevalence                          0.20000                   0.20000
## Detection Rate                      0.04411                   0.08622
## Detection Prevalence                0.15067                   0.18767
## Balanced Accuracy                   0.54368                   0.65215
##                      Class: Material.Issues Class: Onboard.Emergencies
## Sensitivity                          0.4400                    0.43167
## Specificity                          0.7881                    0.87500
## Pos Pred Value                       0.3417                    0.46333
## Neg Pred Value                       0.8491                    0.86030
## Prevalence                           0.2000                    0.20000
## Detection Rate                       0.0880                    0.08633
## Detection Prevalence                 0.2576                    0.18633
## Balanced Accuracy                    0.6140                    0.65333
##                      Class: Third.party.Damages
## Sensitivity                              0.5211
## Specificity                              0.8581
## Pos Pred Value                           0.4786
## Neg Pred Value                           0.8776
## Prevalence                               0.2000
```

```
## Detection Rate                             0.1042
## Detection Prevalence                       0.2178
## Balanced Accuracy                          0.6896
```

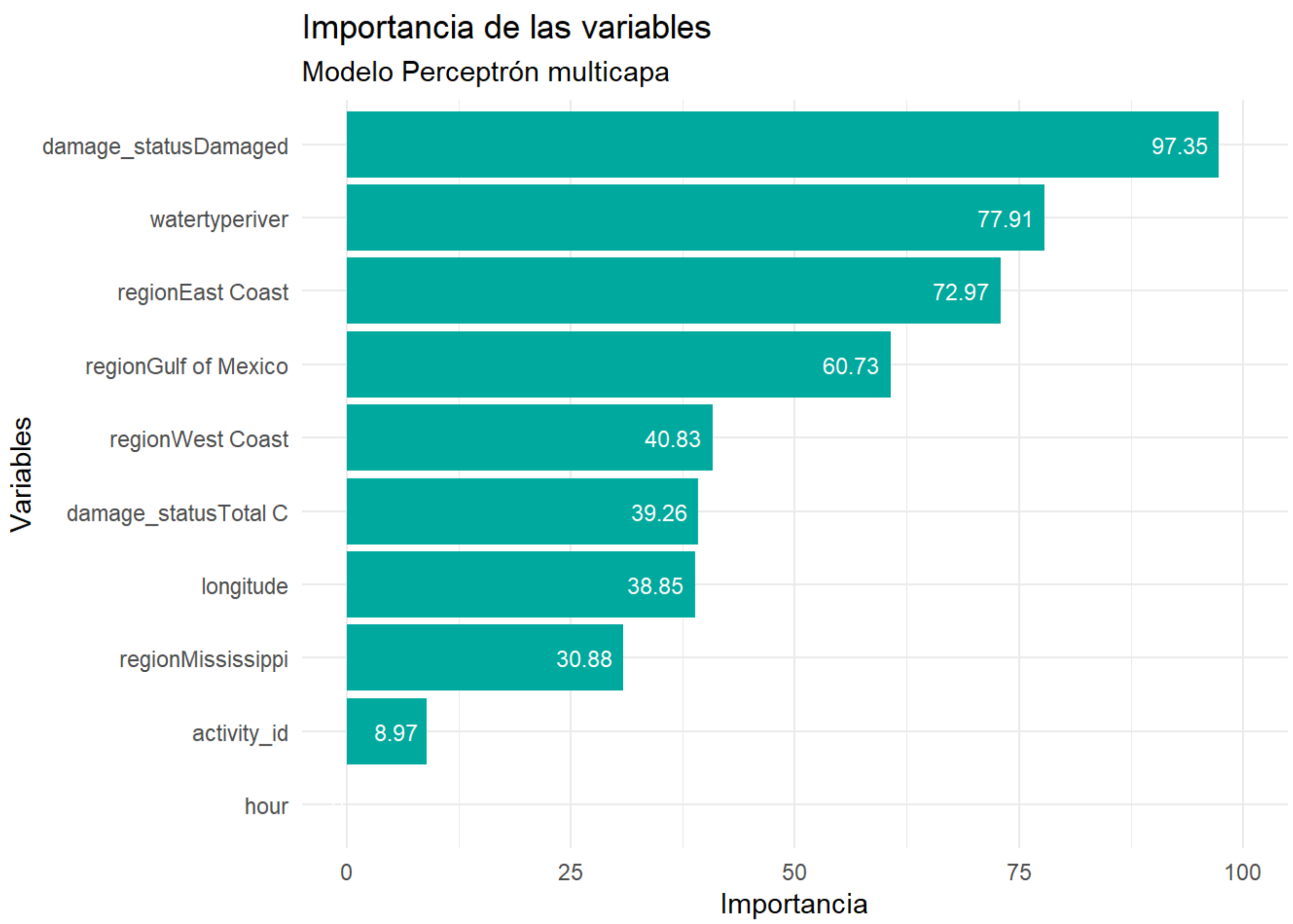
Resumen de métricas

```
# Resumen
resumen_MA_nnet <- resumen_multiclass(nnet_MA_train, train_MA_general, test_MA_general)

# Presentación
resumen_MA_nnet %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ",
            "Red Neuronal. Perceptrón Multicapa " = 5))
```

	Red Neuronal. Perceptrón Multicapa				
	AUC	**Accuracy**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.730	0.415	0.269	0.415	0.854
Datos Validación	0.724	0.409	0.261	0.409	0.852

Importancia de las variables

```
# Importancia general de las variables
importancia_var_overall(nnet_MA_train, "Perceptrón multicapa")
```


Importancia de las variables
Modelo Perceptrón multicapa
Variables
damage_statusDamaged
watertyperiver
regionEast Coast
regionGulf of Mexico
regionWest Coast
damage_statusTotal C
longitude
regionMississippi
activity_id
hour
97.35
77.91
72.97
60.73
40.83
39.26
38.85
30.88
8.97
0
25
50
75
100
Importancia

2.3.3. Árbol C5

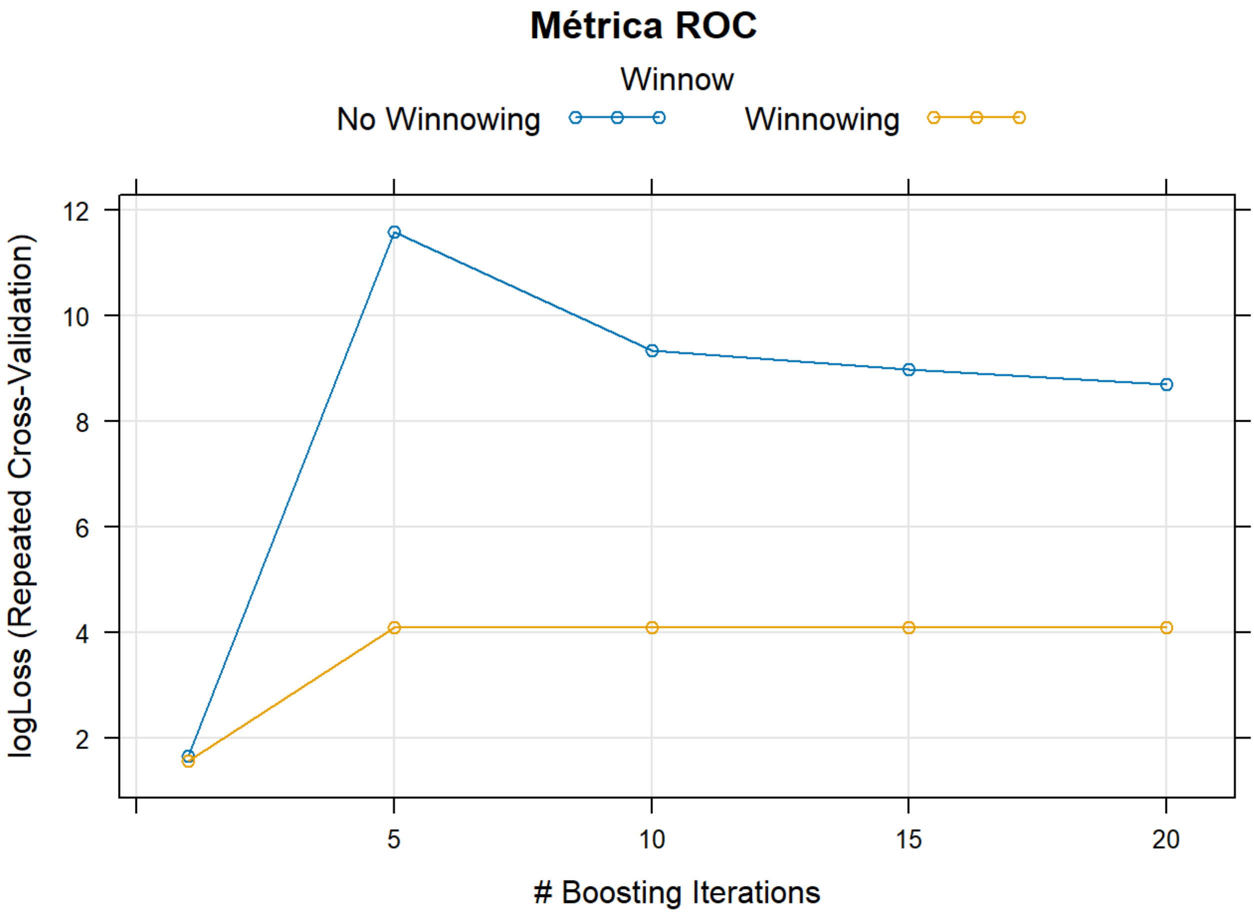
```r
# Entrenamiento
 if (train_switch  == 1) {
set.seed(7)

tic()

  clusterCPU <- makePSOCKcluster(detectCores() - 1)
  registerDoParallel(clusterCPU)


  grid_c50 <- expand.grid(winnow = c(T, F),
                        trials = c(1, 5, 10, 15, 20),
                        model = 'tree')

  tic()
  C5_MA_train <- train(y~.,
                  data = train_MA_general,
                  method = 'C5.0',
                  metric = metrica,
                  #preProc = c('center', 'scale'),
                  trControl = control,
                  tuneLength = 10,
                  tuneGrid = grid_c50)

  stopCluster(clusterCPU)
  clusterCPU <- NULL

  saveRDS(C5_MA_train, "Models/C5_MA_train.RDS")

toc()

}else{
  C5_MA_train <- readRDS("Models/C5_MA_train.RDS")
}
```

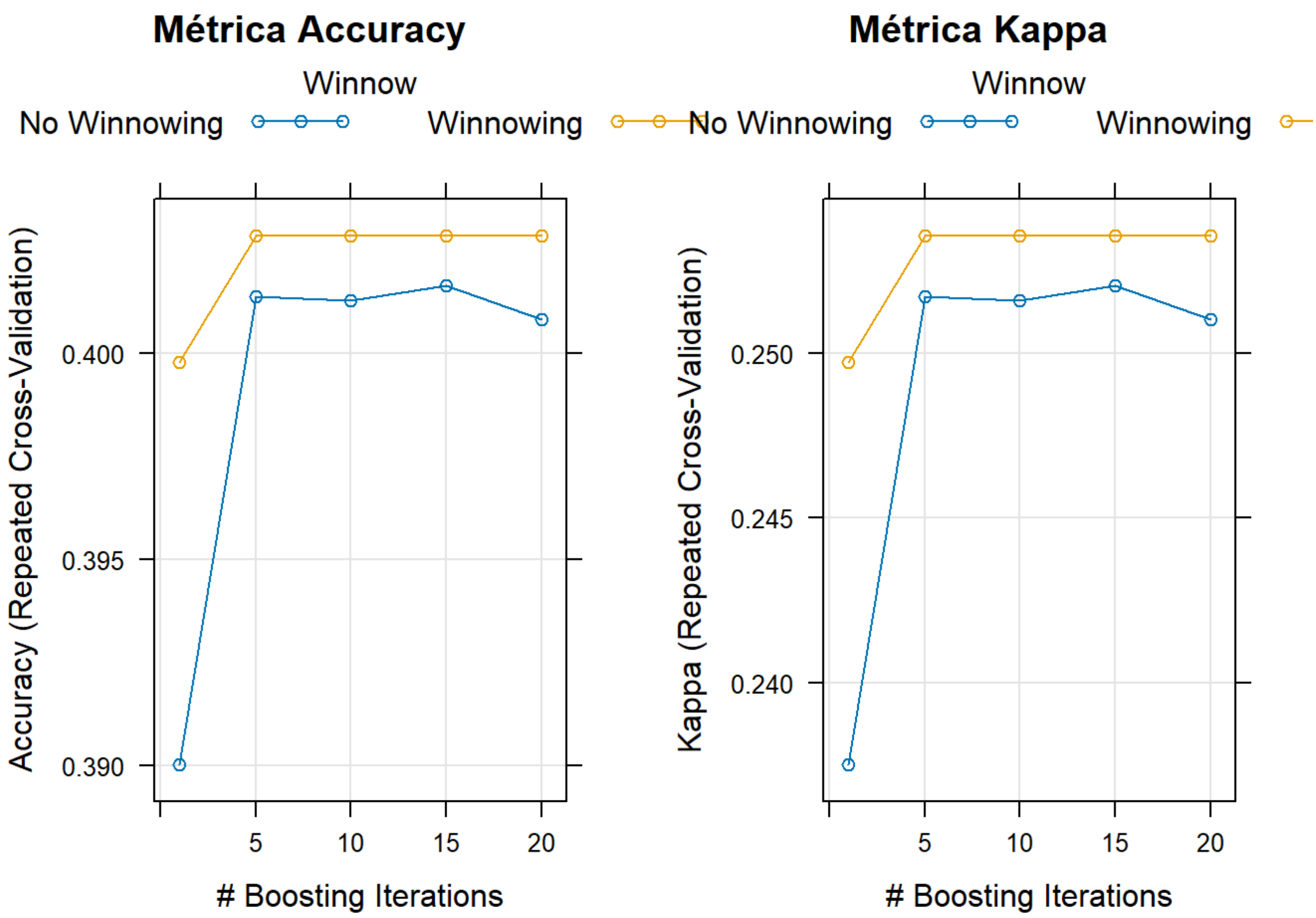
```r
# Resultado
C5_MA_train
```

```
## C5.0
##
## 36000 samples
##    12 predictor
##     5 classes: 'Critical.Events', 'Maritime.Accidents', 'Material.Issues', 'Onboard.Eme
rgencies', 'Third.party.Damages'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 31500, 31500, 31500, 31500, 31500, 31500, ...
## Resampling results across tuning parameters:
##
##   winnow  trials  logLoss    AUC        prAUC      Accuracy   Kappa
##   FALSE    1       1.659866  0.6921549  0.3930191  0.3900278  0.2375347
##   FALSE    5      11.598187  0.6879126  0.2911302  0.4013611  0.2517014
##   FALSE   10       9.353292  0.7000911  0.3169549  0.4012639  0.2515799
##   FALSE   15       8.991357  0.7016935  0.3211655  0.4016389  0.2520486
##   FALSE   20       8.711262  0.7029643  0.3245679  0.4008056  0.2510069
##    TRUE    1       1.560276  0.7045828  0.4078656  0.3997778  0.2497222
##    TRUE    5       4.084759  0.6989048  0.3791123  0.4028472  0.2535590
##    TRUE   10       4.084759  0.6989048  0.3791123  0.4028472  0.2535590
##    TRUE   15       4.084759  0.6989048  0.3791123  0.4028472  0.2535590
##    TRUE   20       4.084759  0.6989048  0.3791123  0.4028472  0.2535590
##   Mean_F1    Mean_Sensitivity  Mean_Specificity  Mean_Pos_Pred_Value
##   0.3917586  0.3900278         0.8475069         0.3947639
##   0.3982930  0.4013611         0.8503403         0.3961903
##   0.3979326  0.4012639         0.8503160         0.3955330
##   0.3982938  0.4016389         0.8504097         0.3958192
##   0.3973837  0.4008056         0.8502014         0.3948557
##   0.4016563  0.3997778         0.8499444         0.4051352
##   0.4038584  0.4028472         0.8507118         0.4064814
##   0.4038584  0.4028472         0.8507118         0.4064814
##   0.4038584  0.4028472         0.8507118         0.4064814
##   0.4038584  0.4028472         0.8507118         0.4064814
##   Mean_Neg_Pred_Value  Mean_Precision  Mean_Recall  Mean_Detection_Rate
##   0.8473113            0.3947639       0.3900278    0.07800556
##   0.8507216            0.3961903       0.4013611    0.08027222
##   0.8507289            0.3955330       0.4012639    0.08025278
##   0.8508249            0.3958192       0.4016389    0.08032778
##   0.8506273            0.3948557       0.4008056    0.08016111
##   0.8497363            0.4051352       0.3997778    0.07995556
##   0.8506058            0.4064814       0.4028472    0.08056944
##   0.8506058            0.4064814       0.4028472    0.08056944
##   0.8506058            0.4064814       0.4028472    0.08056944
##   0.8506058            0.4064814       0.4028472    0.08056944
##   Mean_Balanced_Accuracy
##   0.6187674
##   0.6258507
##   0.6257899
##   0.6260243
##   0.6255035
##   0.6248611
##   0.6267795
##   0.6267795
```

```
##    0.6267795
##    0.6267795
##
## Tuning parameter 'model' was held constant at a value of tree
## logLoss was used to select the optimal model using the smallest value.
## The final values used for the model were trials = 1, model = tree and winnow
##  = TRUE.
```

```
# Gráfico de métricas
grafico_metricas(C5_MA_train)
```






```
# Resultados
resultados(C5_MA_train, "Árbol C5")
```

RESULTADOS DEL MODELO Árbol C5

model	winnow	trials	logLoss	AUC	prAUC	Accuracy	Kappa	Mean_F1	M
tree	FALSE	1	1.659866	0.6921549	0.3930191	0.3900278	0.2375347	0.3917586	
tree	TRUE	1	1.560276	0.7045828	0.4078656	0.3997778	0.2497222	0.4016563	
tree	FALSE	5	11.598187	0.6879126	0.2911302	0.4013611	0.2517014	0.3982930	
tree	TRUE	5	4.084759	0.6989048	0.3791123	0.4028472	0.2535590	0.4038584	
tree	FALSE	10	9.353292	0.7000911	0.3169549	0.4012639	0.2515799	0.3979326	
tree	TRUE	10	4.084759	0.6989048	0.3791123	0.4028472	0.2535590	0.4038584	
tree	FALSE	15	8.991357	0.7016935	0.3211655	0.4016389	0.2520486	0.3982938	
tree	TRUE	15	4.084759	0.6989048	0.3791123	0.4028472	0.2535590	0.4038584	
tree	FALSE	20	8.711262	0.7029643	0.3245679	0.4008056	0.2510069	0.3973837	
tree	TRUE	20	4.084759	0.6989048	0.3791123	0.4028472	0.2535590	0.4038584	

```
# Mejor modelo
mejor_modelo(C5_MA_train)
```

```
## [1] "El mejor módelo es el que muestra los siguientes hiperparámetros:"
```

	trials	model	winnow
6	1	tree	TRUE

```
# Curvas ROC y AUC
curvas_ROC(C5_MA_train, "de Árbol C5", train_MA_general, test_MA_general)
```

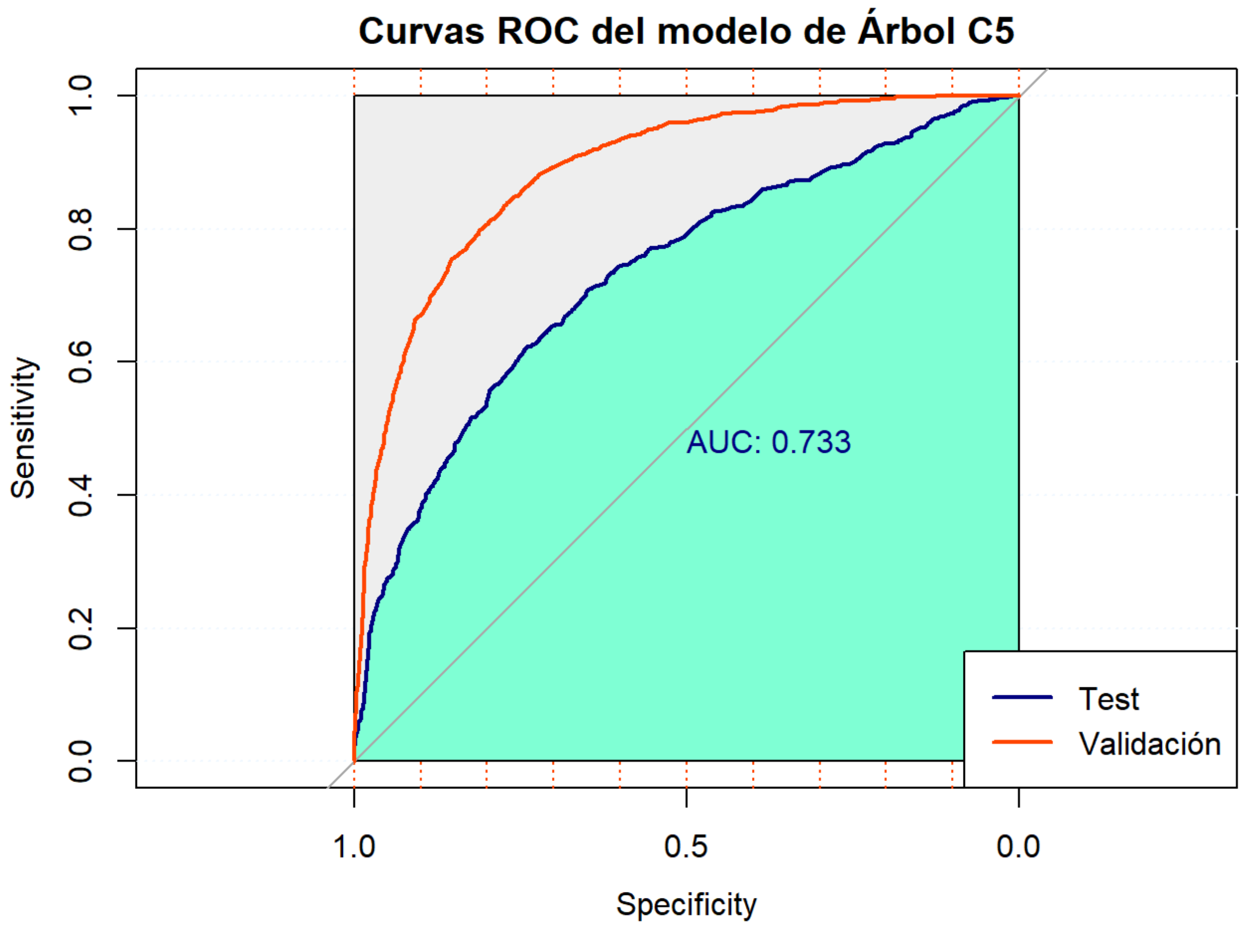
```
## Warning in roc.default(train[, c(length(train))], pred_train[, clase]):
## 'response' has more than two levels. Consider setting 'levels' explicitly or
## using 'multiclass.roc' instead
```

```
## Setting levels: control = Critical.Events, case = Maritime.Accidents
```

```
## Setting direction: controls < cases
```

```
## Warning in roc.default(test[, c(length(test))], pred_test[, clase]): 'response'
## has more than two levels. Consider setting 'levels' explicitly or using
## 'multiclass.roc' instead
```

```
## Setting levels: control = Critical.Events, case = Maritime.Accidents
## Setting direction: controls < cases
```




```
## [1] "ROC del modelo con el fichero de test: 0.732574845679012"
```

```
# Validación, Matriz de confusión
validation(C5_MA_train, "de Árbol C5", train_MA_general, test_MA_general)
```

```
## [1] "Modelo de Árbol C5 - Tabla de confusión para los datos de entrenamiento"
## Confusion Matrix and Statistics
##
##                      Reference
## Prediction            Critical.Events Maritime.Accidents Material.Issues
##   Critical.Events                3541                692            1644
##   Maritime.Accidents              761               4957             912
##   Material.Issues                1787                576            3631
##   Onboard.Emergencies             705                474             663
##   Third.party.Damages             406                501             350
##                      Reference
## Prediction            Onboard.Emergencies Third.party.Damages
##   Critical.Events                    1004                 735
##   Maritime.Accidents                  590                 494
##   Material.Issues                     782                 366
##   Onboard.Emergencies                4418                 642
##   Third.party.Damages                 406                4963
##
## Overall Statistics
##
##                Accuracy : 0.5975
##                  95% CI : (0.5924, 0.6026)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4969
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: Critical.Events Class: Maritime.Accidents
## Sensitivity                         0.49181                    0.6885
## Specificity                         0.85851                    0.9043
## Pos Pred Value                      0.46494                    0.6426
## Neg Pred Value                      0.87109                    0.9207
## Prevalence                          0.20000                    0.2000
## Detection Rate                      0.09836                    0.1377
## Detection Prevalence                0.21156                    0.2143
## Balanced Accuracy                   0.67516                    0.7964
##                      Class: Material.Issues Class: Onboard.Emergencies
## Sensitivity                          0.5043                     0.6136
## Specificity                          0.8781                     0.9137
## Pos Pred Value                       0.5084                     0.6401
## Neg Pred Value                       0.8763                     0.9044
## Prevalence                           0.2000                     0.2000
## Detection Rate                       0.1009                     0.1227
## Detection Prevalence                 0.1984                     0.1917
## Balanced Accuracy                    0.6912                     0.7637
##                      Class: Third.party.Damages
## Sensitivity                              0.6893
## Specificity                              0.9423
## Pos Pred Value                           0.7490
## Neg Pred Value                           0.9238
```

```
## Prevalence                                  0.2000
## Detection Rate                              0.1379
## Detection Prevalence                        0.1841
## Balanced Accuracy                           0.8158
## [1] "Modelo de Árbol C5 - Tabla de confusión para los datos de validación"
```

```
## Confusion Matrix and Statistics
##
##                       Reference
## Prediction             Critical.Events Maritime.Accidents Material.Issues
##   Critical.Events                  566                252             571
##   Maritime.Accidents               303                895             293
##   Material.Issues                  553                263             539
##   Onboard.Emergencies              225                167             234
##   Third.party.Damages              153                223             163
##                       Reference
## Prediction             Onboard.Emergencies Third.party.Damages
##   Critical.Events                      333                 235
##   Maritime.Accidents                   199                 223
##   Material.Issues                      303                 171
##   Onboard.Emergencies                  800                 254
##   Third.party.Damages                  165                 917
##
## Overall Statistics
##
##                Accuracy : 0.413
##                  95% CI : (0.4028, 0.4233)
##     No Information Rate : 0.2
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2662
##
##  Mcnemar's Test P-Value : 3.658e-12
##
## Statistics by Class:
##
##                       Class: Critical.Events Class: Maritime.Accidents
## Sensitivity                          0.31444                   0.49722
## Specificity                          0.80681                   0.85861
## Pos Pred Value                       0.28922                   0.46785
## Neg Pred Value                       0.82479                   0.87230
## Prevalence                           0.20000                   0.20000
## Detection Rate                       0.06289                   0.09944
## Detection Prevalence                 0.21744                   0.21256
## Balanced Accuracy                    0.56063                   0.67792
##                       Class: Material.Issues Class: Onboard.Emergencies
## Sensitivity                          0.29944                    0.44444
## Specificity                          0.82083                    0.87778
## Pos Pred Value                       0.29470                    0.47619
## Neg Pred Value                       0.82415                    0.86339
## Prevalence                           0.20000                    0.20000
## Detection Rate                       0.05989                    0.08889
## Detection Prevalence                 0.20322                    0.18667
## Balanced Accuracy                    0.56014                    0.66111
##                       Class: Third.party.Damages
## Sensitivity                               0.5094
## Specificity                               0.9022
## Pos Pred Value                            0.5657
## Neg Pred Value                            0.8803
## Prevalence                                0.2000
```

```
## Detection Rate                             0.1019
## Detection Prevalence                       0.1801
## Balanced Accuracy                          0.7058
```

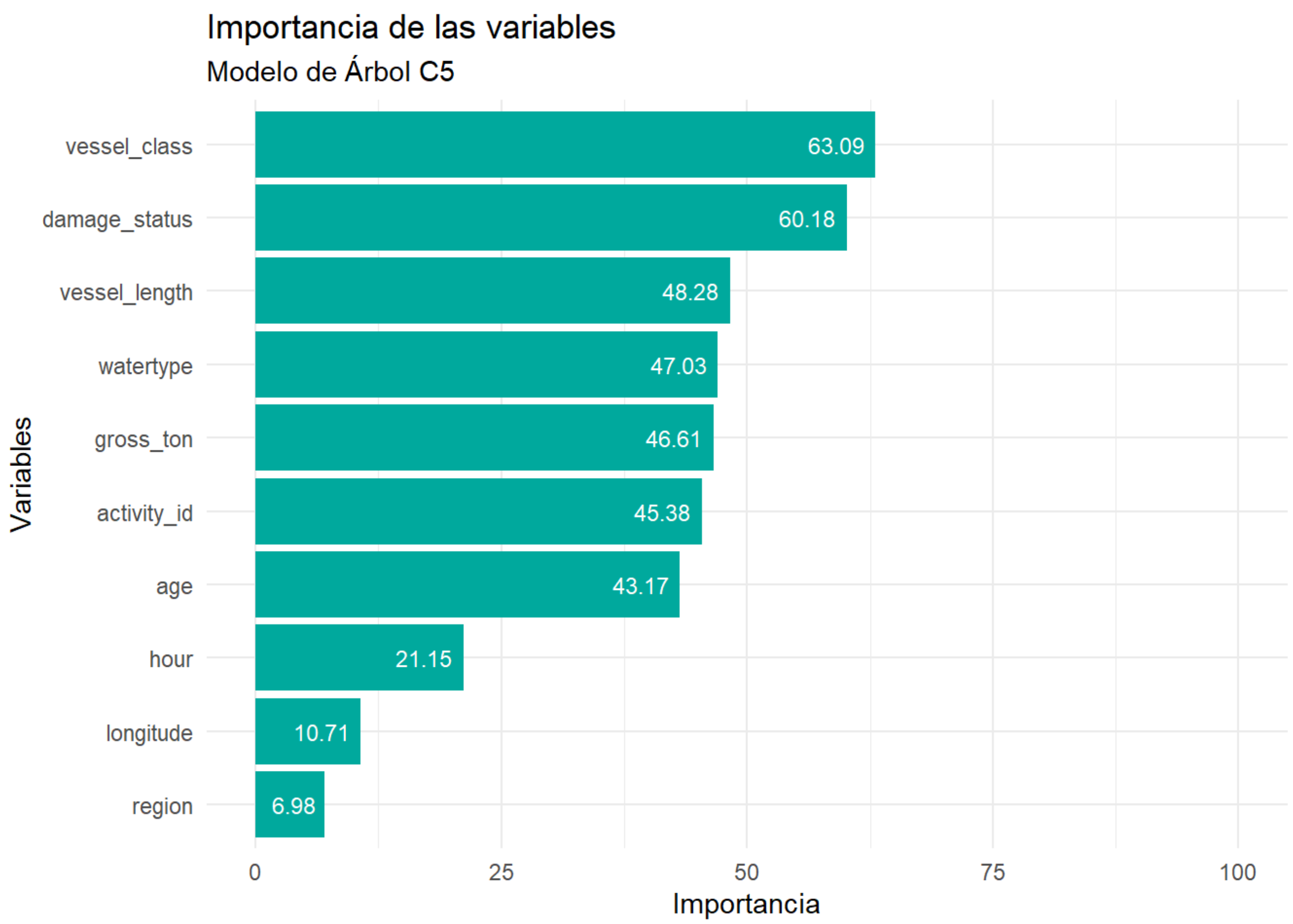
Resumen

```
# Resumen
resumen_MA_C5 <- resumen_multiclass(C5_MA_train, train_MA_general, test_MA_general)

# Presentación
resumen_MA_C5 %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ", "Árbol C5 " = 5))
```

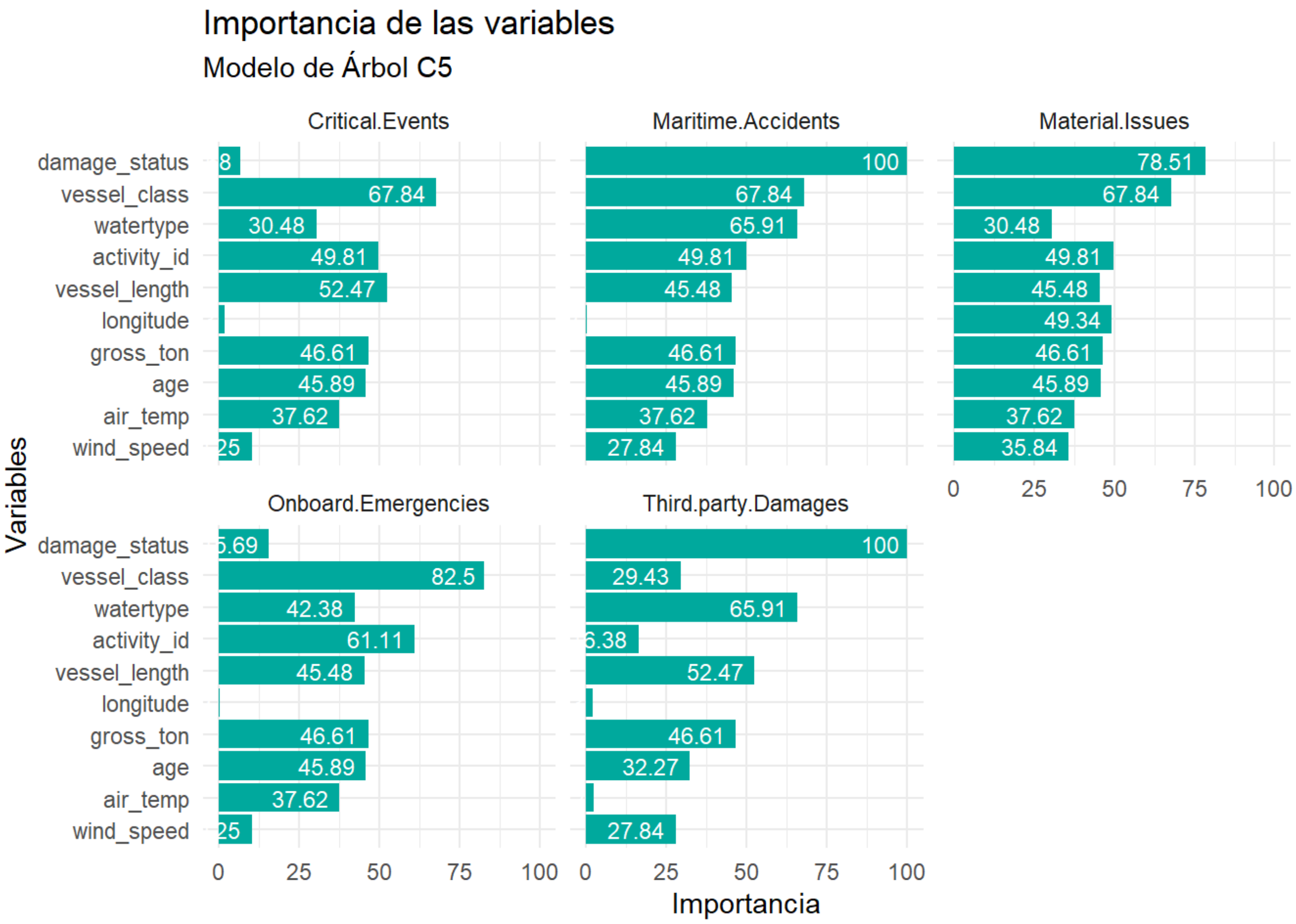
	Árbol C5				
	AUC	**Accuracy**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.874	0.598	0.497	0.598	0.899
Datos Validación	0.714	0.413	0.266	0.413	0.853

Importancia de las variables

```
# Importancia general de las variables
C5_MA_train$modelInfo$varImp <- NULL
importancia_var_overall(C5_MA_train, "de Árbol C5")
```




```
# Importancia de variables por cada valor de predicción

importancia_var(C5_MA_train, "de Árbol C5")
```



2.4. Redes neuronales con Keras

2.4.1. API Secuencial: Red densamente conectada

```
# Conversión a variables dummy para la variable objetivo con ayuda de la librería fastDumm
ies
y_train <- dummy_cols(train_MA_num, select_columns = "y", ) %>%
  select(starts_with("y_"))

y_test <- dummy_cols(test_MA_num, select_columns = "y") %>%
  select(starts_with("y_"))

# Selección de las variables explicativas en formato numérico (ya normalizadas)
x_train <- train_MA_num %>%
  select(-y)

x_test <- test_MA_num %>%
  select(-y)
```

```r
# Crear el modelo
keras_model_1 <- keras_model_sequential() %>%
  layer_dense(units = 128, activation = 'relu', input_shape = dim(x_train)[2]) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 64, activation = 'relu') %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = dim(y_train)[2], activation = 'softmax')

# Estructura
print(keras_model_1)
```

```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  dense_2 (Dense)                    (None, 128)                     4096
##  dropout_1 (Dropout)                (None, 128)                     0
##  dense_1 (Dense)                    (None, 64)                      8256
##  dropout (Dropout)                  (None, 64)                      0
##  dense (Dense)                      (None, 5)                       325
## ================================================================================
## Total params: 12,677
## Trainable params: 12,677
## Non-trainable params: 0
## ________________________________________________________________________________
```

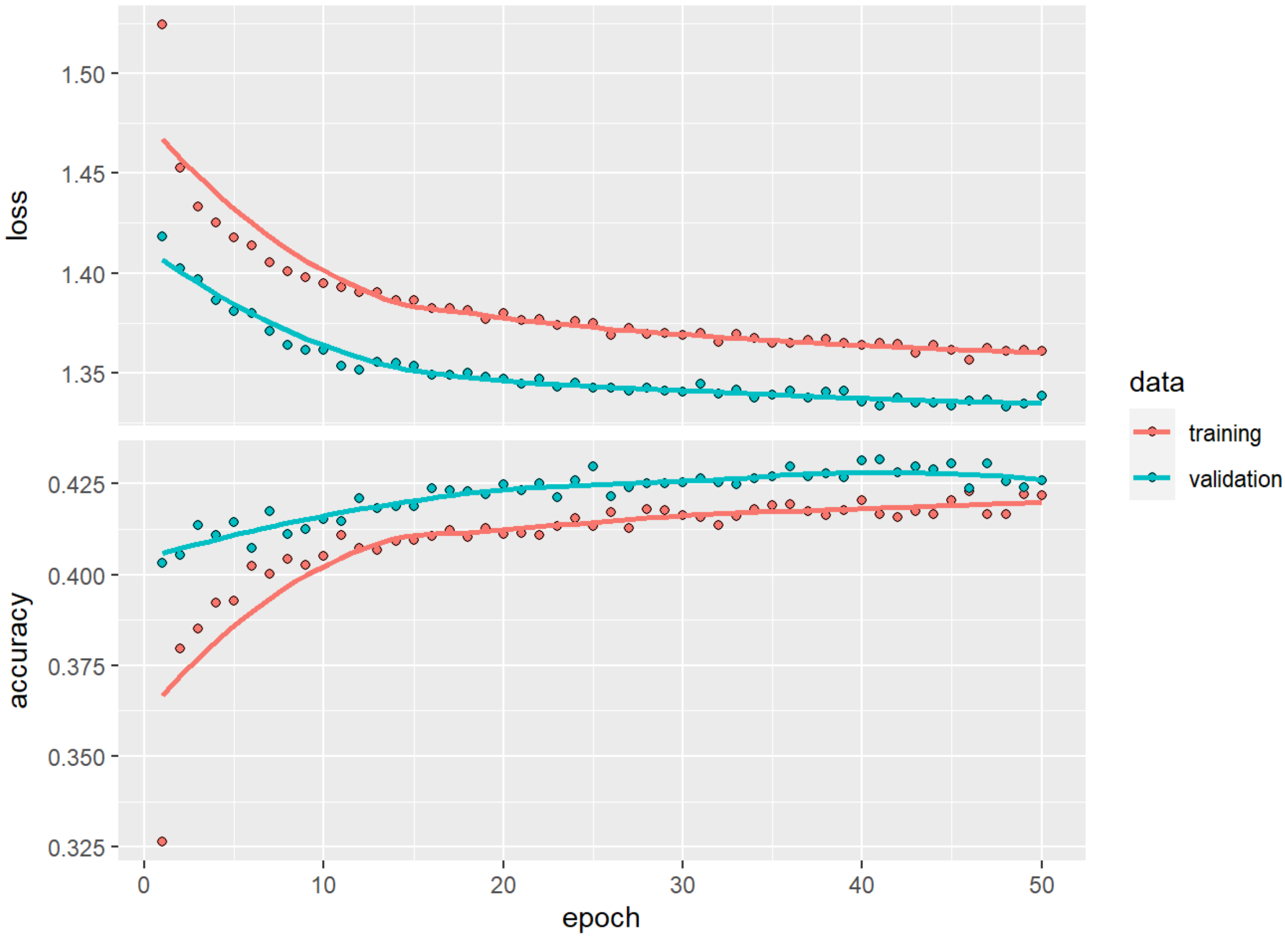
```r
# Compilar el modelo
keras_model_1 %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = 'adam',
  metrics = c('accuracy')
)
```

```r
# Entrenar el modelo
if (train_switch  == 1) {
keras_model_evolution <- keras_model_1 %>%
  fit(as.matrix(x_train), as.matrix(y_train),
      epochs = 50,
      batch_size = 32,
      callbacks = list(callback_early_stopping(monitor = 'val_loss', patience = 10, restor
e_best_weights = TRUE)),
      validation_data = list(as.matrix(x_test), as.matrix(y_test))
      )
  keras_model_1 %>% save_model_hdf5("Models/keras_model_1.hdf5")
  saveRDS(keras_model_evolution, "Models/keras_model_evolution.rds")
} else {
keras_model_1 <- load_model_hdf5("Models/keras_model_1.hdf5")
keras_model_evolution <- readRDS("Models/keras_model_evolution.rds")
}
```

```r
keras_model_evolution
```

```
##
## Final epoch (plot to see history):
##         loss: 1.361
##     accuracy: 0.4217
##     val_loss: 1.338
## val_accuracy: 0.4259
```

```
plot(keras_model_evolution)
```



Resumen de métricas

```
# Resumen de métricas
resumen_MA_keras_model_1 <- keras_resumen_multiclass(keras_model_1, x_train, x_test, y_tra
in, y_test)
```

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

```
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
```

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

```
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
```

```
# Presentación
resumen_MA_keras_model_1 %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ", "keras_model_1" = 5))
```

	keras_model_1				
	AUC	**Accuracy**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.624	0.445	0.306	0.456	0.862
Datos Validación	0.620	0.426	0.282	0.436	0.857

```
modelo <- keras_model_1

  # Entrenamiento
  predicciones <- predict(modelo, as.matrix(x_train))
  predicciones <- apply(predicciones, 1, which.max)
  Y_train <- max.col(y_train)

  curvaROC_train <- multiclass.roc(Y_train, predicciones)
```

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

```
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
## Setting direction: controls < cases
```

```
  AUC_train <- round(auc(curvaROC_train),digits=3)

  confusion_train <- confusionMatrix(as.factor(Y_train), as.factor(predicciones))

  Accuracy_train <- round(c(confusion_train[["overall"]][["Accuracy"]]), digits=3)

  Kappa_train <- round(c(confusion_train[["overall"]][["Kappa"]]), digits=3)
```

2.5. Extra: H2o Framework

Como extra, se va a comparar los modelos anteriores con uno automático para ver si se puede mejorar los resultados

```
# Arranque de h2o
h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         40 minutes 7 seconds
##     H2O cluster timezone:       Europe/Paris
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.40.0.4
##     H2O cluster version age:    7 months and 11 days
##     H2O cluster name:           H2O_started_from_R_0_okx249
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   5.38 GB
##     H2O cluster total cores:    12
##     H2O cluster allowed cores:  12
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     R Version:                  R version 4.3.0 (2023-04-21 ucrt)
```

```
## Warning in h2o.clusterInfo():
## Your H2O cluster version is (7 months and 11 days) old. There may be a newer version av
ailable.
## Please download and install the latest version from: https://h2o-release.s3.amazonaws.c
om/h2o/latest_stable.html
```

2.5.1. AutoML Procesado

```
# Conversión del dataframe general a un objeto H2o
h2o_df <- data.table(readRDS("Datasets/MergedActivityGeneral.rds")) %>%
  as.h2o()
```

```
##
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
# División de datos en entrenamiento y validación
splits <- h2o.splitFrame(h2o_df, ratios = c(0.8, 0.19999))
h2o_train <- splits[[1]]
h2o_test <- splits[[2]]

# Establecimiento de los nombres de las variables predictoras
predictoras <- colnames(h2o_train)[1:12]
# y el nombre de la variable objetivo
respuesta <- "y"
```

```
# Entrenamiento
if (train_switch  == 1) {

set.seed(123)

# Configuramos y ejecutamos el proceso de auto machine learning
mod_aml_h2o <- h2o.automl(
  x = predictoras,
  y = respuesta,
  training_frame = h2o_train,
  leaderboard_frame = h2o_test,
  max_runtime_secs = 2000  # Tiempo máximo de ejecución en segundos
)

# Guardamos el modelo y el objeto h2o
saveRDS(mod_aml_h2o, "Models/mod_aml_h2o.RDS")
h2o.saveModel(object= mod_aml_h2o@leader, path="Models/", force=TRUE)
}
# Leemos el modelo y el objeto h2o
mod_aml_h2o <- readRDS("Models/mod_aml_h2o.RDS")
h2o.loadModel(paste0("Models/", mod_aml_h2o@leader@model_id))
```

```
## Model Details:
## ==============
##
## H2OMultinomialModel: stackedensemble
## Model ID:  StackedEnsemble_BestOfFamily_4_AutoML_1_20231209_172047
## Model Summary for Stacked Ensemble:
##                                          key            value
## 1                          Stacking strategy cross_validation
## 2       Number of base models (used / total)              4/4
## 3           # GBM base models (used / total)              1/1
## 4           # DRF base models (used / total)              2/2
## 5  # DeepLearning base models (used / total)              1/1
## 6                      Metalearner algorithm              GBM
## 7         Metalearner fold assignment scheme           Random
## 8                         Metalearner nfolds                5
## 9                    Metalearner fold_column               NA
## 10        Custom metalearner hyperparameters             None
##
##
## H2OMultinomialMetrics: stackedensemble
## ** Reported on training data. **
##
## Training Set Metrics:
## =====================
##
## MSE: (Extract with `h2o.mse`) 0.510757
## RMSE: (Extract with `h2o.rmse`) 0.7146726
## Logloss: (Extract with `h2o.logloss`) 1.465757
## Mean Per-Class Error: 0.5957195
## AUC: (Extract with `h2o.auc`) NaN
## AUCPR: (Extract with `h2o.aucpr`) NaN
## Confusion Matrix: Extract with `h2o.confusionMatrix(<model>,train = TRUE)`)
## =========================================================================
## Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
##                     Critical.Events Maritime.Accidents Material.Issues
## Critical.Events                 424                189            1145
## Maritime.Accidents              206               1041             608
## Material.Issues                1205                261             274
## Onboard.Emergencies             278                 62             473
## Third.party.Damages             290                227             145
## Totals                         2403               1780            2645
##                     Onboard.Emergencies Third.party.Damages  Error
## Critical.Events                     226                  39 0.7904
## Maritime.Accidents                   64                 128 0.4915
## Material.Issues                     178                  31 0.8594
## Onboard.Emergencies                1147                  90 0.4405
## Third.party.Damages                 140                1219 0.3968
## Totals                             1755                1507 0.5932
##                                  Rate
## Critical.Events     =  1.599 / 2.023
## Maritime.Accidents  =  1.006 / 2.047
## Material.Issues     =  1.675 / 1.949
## Onboard.Emergencies =    903 / 2.050
## Third.party.Damages =    802 / 2.021
```

```
## Totals              = 5.985 / 10.090
##
## Hit Ratio Table: Extract with `h2o.hit_ratio_table(<model>,train = TRUE)`
## =======================================================================
## Top-5 Hit Ratios:
##   k hit_ratio
## 1 1  0.406838
## 2 2  0.586819
## 3 3  0.752230
## 4 4  0.888206
## 5 5  1.000000
##
##
##
##
##
## H2OMultinomialMetrics: stackedensemble
## ** Reported on cross-validation data. **
## ** 5-fold cross-validation on training data (Metrics computed for combined holdout pred
ictions) **
##
## Cross-Validation Set Metrics:
## =====================
##
## Extract cross-validation frame with `h2o.getFrame("levelone_training_StackedEnsemble_Be
stOfFamily_4_AutoML_1_20231209_172047")`
## MSE: (Extract with `h2o.mse`) 0.4281953
## RMSE: (Extract with `h2o.rmse`) 0.6543663
## Logloss: (Extract with `h2o.logloss`) 1.178701
## Mean Per-Class Error: 0.4772615
## AUC: (Extract with `h2o.auc`) NaN
## AUCPR: (Extract with `h2o.aucpr`) NaN
## Hit Ratio Table: Extract with `h2o.hit_ratio_table(<model>,xval = TRUE)`
## =======================================================================
## Top-5 Hit Ratios:
##   k hit_ratio
## 1 1  0.522976
## 2 2  0.733129
## 3 3  0.862225
## 4 4  0.948762
## 5 5  1.000000
##
##
##
##
## Cross-Validation Metrics Summary:
##                                mean        sd  cv_1_valid  cv_2_valid
## accuracy                   0.522983  0.003260    0.525721    0.517978
## auc                              NA  0.000000          NA          NA
## err                        0.477017  0.003260    0.474279    0.482022
## err_count               3429.800000 43.654324 3439.000000 3499.000000
## logloss                    1.178679  0.009542    1.170900    1.194422
## max_per_class_error        0.561557  0.010933    0.551382    0.554558
## mean_per_class_accuracy    0.522836  0.003776    0.525526    0.517734
## mean_per_class_error       0.477164  0.003776    0.474474    0.482266
```

```
## mse                       0.428193  0.002862    0.425300    0.432488
## pr_auc                          NA  0.000000          NA          NA
## r2                        0.786204  0.001790    0.788790    0.785716
## rmse                      0.654361  0.002185    0.652150    0.657638
##                         cv_3_valid  cv_4_valid  cv_5_valid
## accuracy                  0.523274    0.525963    0.521982
## auc                             NA          NA          NA
## err                       0.476726    0.474037    0.478018
## err_count              3390.000000 3396.000000 3425.000000
## logloss                   1.179102    1.171284    1.177687
## max_per_class_error       0.579592    0.561026    0.561224
## mean_per_class_accuracy   0.524978    0.526064    0.519879
## mean_per_class_error      0.475022    0.473936    0.480121
## mse                       0.429383    0.426130    0.427663
## pr_auc                          NA          NA          NA
## r2                        0.784412    0.787197    0.784903
## rmse                      0.655273    0.652787    0.653959
```

```
# Mejor modelo
mod_aml_h2o@leaderboard
```

```
##                                                   model_id mean_per_class_error
## 1 StackedEnsemble_BestOfFamily_4_AutoML_1_20231209_172047            0.4516766
## 2    StackedEnsemble_AllModels_5_AutoML_1_20231209_172047            0.4550481
## 3    StackedEnsemble_AllModels_3_AutoML_1_20231209_172047            0.4663601
## 4    StackedEnsemble_AllModels_2_AutoML_1_20231209_172047            0.4826991
## 5 StackedEnsemble_BestOfFamily_3_AutoML_1_20231209_172047            0.4865004
## 6 StackedEnsemble_BestOfFamily_2_AutoML_1_20231209_172047            0.4938496
##    logloss      rmse       mse
## 1 1.151665 0.6459388 0.4172370
## 2 1.275898 0.7047981 0.4967404
## 3 1.178963 0.6479503 0.4198396
## 4 1.223863 0.6698405 0.4486864
## 5 1.225977 0.6706256 0.4497387
## 6 1.244179 0.6768944 0.4581860
##
## [72 rows x 5 columns]
```

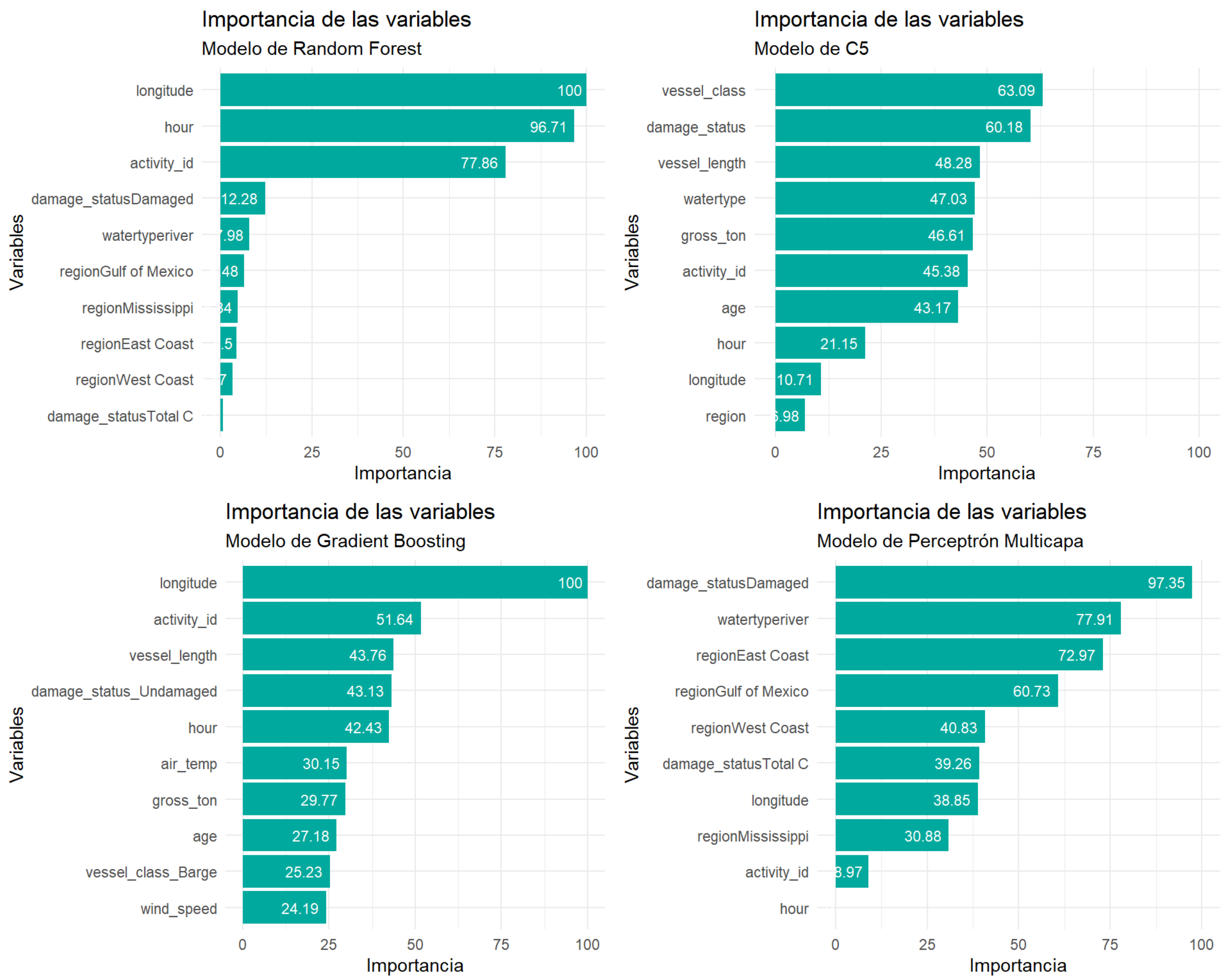
```
# Rendimiento del mejor modelo
h2o.performance(mod_aml_h2o@leader, newdata = h2o_test)
```

```
## H2OMultinomialMetrics: stackedensemble
##
## Test Set Metrics:
## =====================
##
## MSE: (Extract with `h2o.mse`) 0.4990596
## RMSE: (Extract with `h2o.rmse`) 0.7064415
## Logloss: (Extract with `h2o.logloss`) 1.429426
## Mean Per-Class Error: 0.5754344
## AUC: (Extract with `h2o.auc`) NaN
## AUCPR: (Extract with `h2o.aucpr`) NaN
## Confusion Matrix: Extract with `h2o.confusionMatrix(<model>, <data>)`)
## =========================================================================
## Confusion Matrix: Row labels: Actual class; Column labels: Predicted class
##                     Critical.Events Maritime.Accidents Material.Issues
## Critical.Events                 448                187             857
## Maritime.Accidents              191                969             459
## Material.Issues                 939                246             399
## Onboard.Emergencies             273                 91             359
## Third.party.Damages             247                218             124
## Totals                         2098               1711            2198
##                     Onboard.Emergencies Third.party.Damages  Error
## Critical.Events                     202                  73 0.7465
## Maritime.Accidents                   72                 149 0.4734
## Material.Issues                     158                  55 0.7780
## Onboard.Emergencies                 924                  89 0.4677
## Third.party.Damages                 154                1062 0.4116
## Totals                             1510                1428 0.5750
##                                Rate
## Critical.Events     = 1.319 / 1.767
## Maritime.Accidents  =   871 / 1.840
## Material.Issues     = 1.398 / 1.797
## Onboard.Emergencies =   812 / 1.736
## Third.party.Damages =   743 / 1.805
## Totals              = 5.143 / 8.945
##
## Hit Ratio Table: Extract with `h2o.hit_ratio_table(<model>, <data>)`
## =======================================================================
## Top-5 Hit Ratios:
##   k hit_ratio
## 1 1  0.425042
## 2 2  0.607937
## 3 3  0.764002
## 4 4  0.896926
## 5 5  1.000000
```

3. Comparación de los modelos

3.1. Importancia de las variables

```
ggarrange(importancia_var_overall(rf_MA_train, "de Random Forest"),
          importancia_var_overall(C5_MA_train, "de C5"),
          importancia_var(GBM_MA_train, "de Gradient Boosting"),
          importancia_var_overall(nnet_MA_train, "de Perceptrón Multicapa"),
          ncol=2,nrow=2)
```



3.2. Desempeño de los modelos

```r
# Los dos cuadros con los algoritmos utilizados los construimos uniendo la salida de la fu
nción resumen
Nombresmodelos <- c("NB", "GBM", "RF", "MLP","C5","Keras")

# Para los datos de entrenamiento
DatosEntrenamiento <- rbind(resumen_MA_nb[1,], resumen_MA_GBM[1,], resumen_MA_rf[1,], resu
men_MA_nnet[1,], resumen_MA_C5[1,], resumen_MA_keras_model_1[1,])

rownames(DatosEntrenamiento) <- Nombresmodelos

DatosEntrenamiento <- as.data.frame(DatosEntrenamiento)

DatosEntrenamiento %>% arrange(-AUC) %>%
    mutate(AUC = color_tile("white", "orange")(AUC),
    Accuracy = color_tile("white", "pink")(Accuracy),

    Kappa = color_tile("white", "pink")(Kappa),

    Sensitivity = color_tile("white", "purple")(Sensitivity),

    Specificity = color_tile("white", "green")(Specificity)

  ) %>%
  kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ", "Comparación con la Muestra de Entrenamiento" = 5))
```

	Comparación con la Muestra de Entrenamiento				
	AUC	**Accuracy**	**Kappa**	**Sensitivity**	**Specificity**
RF	0.969	0.800	0.750	0.800	0.950
C5	0.874	0.598	0.497	0.598	0.899
GBM	0.820	0.524	0.406	0.524	0.881
MLP	0.730	0.415	0.269	0.415	0.854
NB	0.725	0.417	0.271	0.417	0.854
Keras	0.624	0.445	0.306	0.456	0.862

```
# Los dos cuadros con los algoritmos utilizados los construimos uniendo la salida de la fu
nción resumen
Nombresmodelos <- c("NB", "GBM", "RF", "MLP","C5","Keras")

# Para los datos de Validacion
DatosValidacion <- rbind(resumen_MA_nb[2,], resumen_MA_GBM[2,], resumen_MA_rf[2,], resumen
_MA_nnet[2,], resumen_MA_C5[2,], resumen_MA_keras_model_1[2,])

rownames(DatosValidacion) <- Nombresmodelos

DatosValidacion <- as.data.frame(DatosValidacion)

DatosValidacion %>% arrange(-AUC) %>%
    mutate(AUC = color_tile("white", "orange")(AUC),
    Accuracy = color_tile("white", "pink")(Accuracy),

    Kappa = color_tile("white", "pink")(Kappa),

    Sensitivity = color_tile("white", "purple")(Sensitivity),

    Specificity = color_tile("white", "green")(Specificity)

  ) %>%
  kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ", "Comparación con la Muestra de Validación" = 5))
```

	Comparación con la Muestra de Validación				
	AUC	**Accuracy**	**Kappa**	**Sensitivity**	**Specificity**
GBM	0.759	0.438	0.298	0.438	0.860
NB	0.726	0.420	0.275	0.420	0.855
MLP	0.724	0.409	0.261	0.409	0.852
C5	0.714	0.413	0.266	0.413	0.853
RF	0.712	0.380	0.225	0.380	0.845
Keras	0.620	0.426	0.282	0.436	0.857

Comparativa de Logloss para todos los modelos:

```
# Tabla comparativa
Nombresmodelos <- c("NB", "GBM", "RF", "MLP","C5","Keras", "AutoML")

DatosEntrenamiento <- rbind(mean(nb_MA_train$results$logLoss),
                            mean(GBM_MA_train$results$logLoss),
                            mean(rf_MA_train$results$logLoss),
                            mean(nnet_MA_train$results$logLoss),
                            mean(C5_MA_train$results$logLoss),
                             (keras_model_1 %>% evaluate(as.matrix(x_test), as.matrix(y_tes
t)))[["loss"]],
                            h2o.performance(mod_aml_h2o@leader)@metrics[["logloss"]]
                            )

rownames(DatosEntrenamiento) <- Nombresmodelos

DatosEntrenamiento <- as.data.frame(DatosEntrenamiento) %>% rename(logloss = V1)

DatosEntrenamiento %>% arrange(logloss) %>%
    mutate(logloss = color_tile("lightyellow", "white")(logloss)) %>%
    kable(escape = F) %>%
    kable_styling("hover", full_width = F)
```

	logloss
Keras	1.338324
GBM	1.338944
MLP	1.425373
AutoML	1.465757
NB	1.552391
RF	1.635759
C5	5.821327

```
z <- h2o.performance(mod_aml_h2o@leader)@metrics
```

3.3. Contraste de hipótesis

```
modelos <- list(NB = nb_MA_train, GBM = GBM_MA_train, RF = rf_MA_train,
                MLP = nnet_MA_train, C5 = C5_MA_train)

comp_modelos <- resamples(modelos)
comp_modelos
```

```
##
## Call:
## resamples.default(x = modelos)
##
## Models: NB, GBM, RF, MLP, C5
## Number of resamples: 16
## Performance metrics: Accuracy, Accuracy.1, AUC, Kappa, Kappa.1, logLoss, Mean_Balanced_
Accuracy, Mean_Detection_Rate, Mean_F1, Mean_Neg_Pred_Value, Mean_Pos_Pred_Value, Mean_Pre
cision, Mean_Recall, Mean_Sensitivity, Mean_Specificity, prAUC
## Time estimates for: everything, final model fit
```

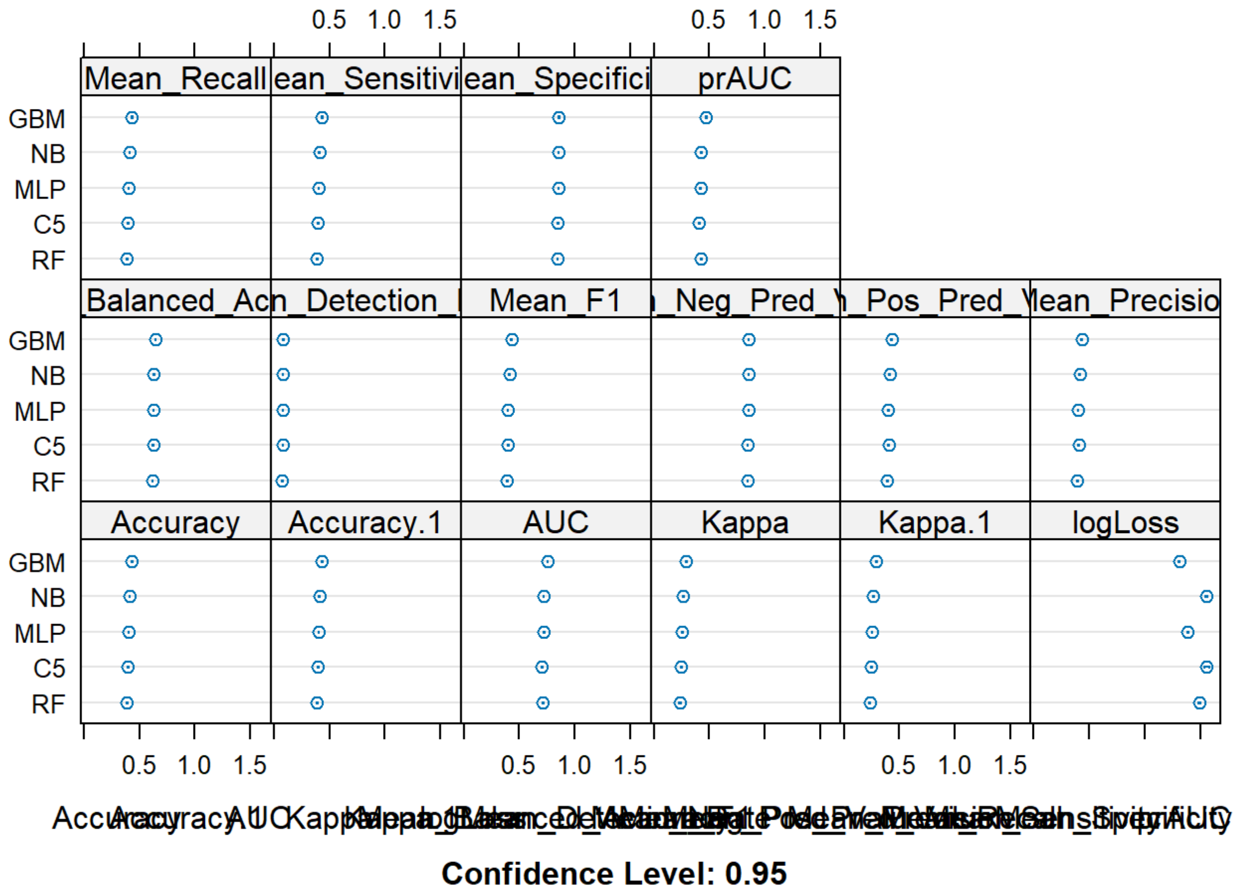
```
summary(comp_modelos)
```

```
##
## Call:
## summary.resamples(object = comp_modelos)
##
## Models: NB, GBM, RF, MLP, C5
## Number of resamples: 16
##
## Accuracy
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB    0.4026667 0.4086111 0.4131111 0.4132500 0.4166111 0.4242222    0
## NB.1  0.4026667 0.4086111 0.4131111 0.4132500 0.4166111 0.4242222    0
## GBM   0.4215556 0.4300000 0.4366667 0.4349028 0.4397222 0.4462222    0
## GBM.1 0.4215556 0.4300000 0.4366667 0.4349028 0.4397222 0.4462222    0
## RF    0.3748889 0.3830000 0.3920000 0.3892917 0.3945556 0.3971111    0
## RF.1  0.3748889 0.3830000 0.3920000 0.3892917 0.3945556 0.3971111    0
## MLP   0.3960000 0.3998889 0.4027778 0.4044444 0.4092778 0.4135556    0
## MLP.1 0.3960000 0.3998889 0.4027778 0.4044444 0.4092778 0.4135556    0
## C5    0.3860000 0.3930000 0.3987778 0.3997778 0.4037222 0.4211111    0
## C5.1  0.3860000 0.3930000 0.3987778 0.3997778 0.4037222 0.4211111    0
##
## Accuracy.1
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB  0.4026667 0.4086111 0.4131111 0.4132500 0.4166111 0.4242222    0
## GBM 0.4215556 0.4300000 0.4366667 0.4349028 0.4397222 0.4462222    0
## RF  0.3748889 0.3830000 0.3920000 0.3892917 0.3945556 0.3971111    0
## MLP 0.3960000 0.3998889 0.4027778 0.4044444 0.4092778 0.4135556    0
## C5  0.3860000 0.3930000 0.3987778 0.3997778 0.4037222 0.4211111    0
##
## AUC
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB  0.7126653 0.7194630 0.7214030 0.7211618 0.7248631 0.7269994    0
## GBM 0.7453859 0.7554500 0.7571335 0.7562578 0.7585071 0.7636082    0
## RF  0.7062244 0.7120075 0.7149972 0.7145125 0.7169910 0.7228908    0
## MLP 0.7164480 0.7201198 0.7220000 0.7229238 0.7270804 0.7311149    0
## C5  0.6953737 0.7003115 0.7052496 0.7045828 0.7075966 0.7194952    0
##
## Kappa
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB    0.2533333 0.2607639 0.2663889 0.2665625 0.2707639 0.2802778    0
## NB.1  0.2533333 0.2607639 0.2663889 0.2665625 0.2707639 0.2802778    0
## GBM   0.2769444 0.2875000 0.2958333 0.2936285 0.2996528 0.3077778    0
## GBM.1 0.2769444 0.2875000 0.2958333 0.2936285 0.2996528 0.3077778    0
## RF    0.2186111 0.2287500 0.2400000 0.2366146 0.2431944 0.2463889    0
## RF.1  0.2186111 0.2287500 0.2400000 0.2366146 0.2431944 0.2463889    0
## MLP   0.2450000 0.2498611 0.2534722 0.2555556 0.2615972 0.2669444    0
## MLP.1 0.2450000 0.2498611 0.2534722 0.2555556 0.2615972 0.2669444    0
## C5    0.2325000 0.2412500 0.2484722 0.2497222 0.2546528 0.2763889    0
## C5.1  0.2325000 0.2412500 0.2484722 0.2497222 0.2546528 0.2763889    0
##
## Kappa.1
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB  0.2533333 0.2607639 0.2663889 0.2665625 0.2707639 0.2802778    0
## GBM 0.2769444 0.2875000 0.2958333 0.2936285 0.2996528 0.3077778    0
## RF  0.2186111 0.2287500 0.2400000 0.2366146 0.2431944 0.2463889    0
```

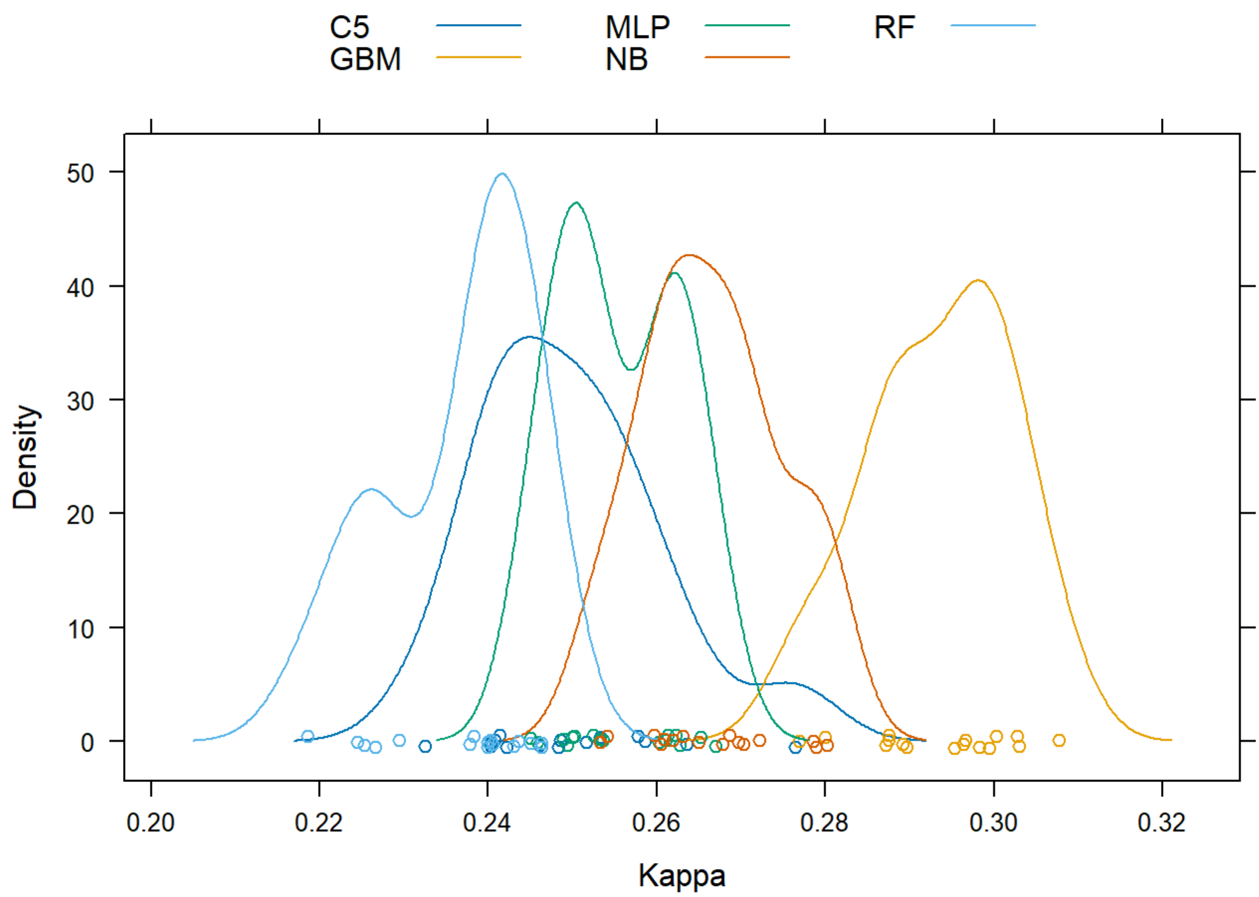
```
## MLP 0.2450000 0.2498611 0.2534722 0.2555556 0.2615972 0.2669444    0
## C5  0.2325000 0.2412500 0.2484722 0.2497222 0.2546528 0.2763889    0
##
## logLoss
##         Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## NB  1.526336 1.544231 1.548398 1.552391 1.565442 1.589366    0
## GBM 1.293248 1.306109 1.308582 1.310059 1.310104 1.337623    0
## RF  1.451035 1.485385 1.493904 1.495059 1.502951 1.531118    0
## MLP 1.370757 1.380793 1.388613 1.387036 1.395820 1.399624    0
## C5  1.477131 1.548544 1.564795 1.560276 1.582994 1.615533    0
##
## Mean_Balanced_Accuracy
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB  0.6266667 0.6303819 0.6331944 0.6332813 0.6353819 0.6401389    0
## GBM 0.6384722 0.6437500 0.6479167 0.6468142 0.6498264 0.6538889    0
## RF  0.6093056 0.6143750 0.6200000 0.6183073 0.6215972 0.6231944    0
## MLP 0.6225000 0.6249306 0.6267361 0.6277778 0.6307986 0.6334722    0
## C5  0.6162500 0.6206250 0.6242361 0.6248611 0.6273264 0.6381944    0
##
## Mean_Detection_Rate
##           Min.    1st Qu.     Median       Mean    3rd Qu.       Max. NA's
## NB  0.08053333 0.08172222 0.08262222 0.08265000 0.08332222 0.08484444    0
## GBM 0.08431111 0.08600000 0.08733333 0.08698056 0.08794444 0.08924444    0
## RF  0.07497778 0.07660000 0.07840000 0.07785833 0.07891111 0.07942222    0
## MLP 0.07920000 0.07997778 0.08055556 0.08088889 0.08185556 0.08271111    0
## C5  0.07720000 0.07860000 0.07975556 0.07995556 0.08074444 0.08422222    0
##
## Mean_F1
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB  0.4009419 0.4067160 0.4102325 0.4111456 0.4145803 0.4219515    0
## GBM 0.4162543 0.4281084 0.4336985 0.4322107 0.4368422 0.4436470    0
## RF  0.3742307 0.3855476 0.3925531 0.3901783 0.3955244 0.3983683    0
## MLP 0.3884735 0.3918767 0.3952359 0.3966414 0.4019253 0.4076257    0
## C5  0.3876638 0.3956343 0.4006359 0.4016563 0.4054051 0.4229917    0
##
## Mean_Neg_Pred_Value
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB  0.8508935 0.8524300 0.8536985 0.8536251 0.8544471 0.8565288    0
## GBM 0.8560227 0.8578432 0.8594715 0.8590701 0.8602851 0.8619187    0
## RF  0.8437675 0.8454122 0.8477863 0.8471848 0.8484515 0.8491002    0
## MLP 0.8494532 0.8508248 0.8519007 0.8520292 0.8531545 0.8541490    0
## C5  0.8463000 0.8481116 0.8494904 0.8497363 0.8507377 0.8550689    0
##
## Mean_Pos_Pred_Value
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB  0.4046612 0.4104087 0.4140718 0.4148792 0.4177292 0.4280659    0
## GBM 0.4153255 0.4276816 0.4332397 0.4317596 0.4363010 0.4433127    0
## RF  0.3754790 0.3897868 0.3942409 0.3924425 0.3979209 0.4011984    0
## MLP 0.3908149 0.3931473 0.4005744 0.3998573 0.4047983 0.4122558    0
## C5  0.3898278 0.3992120 0.4041697 0.4051352 0.4084073 0.4260283    0
##
## Mean_Precision
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB  0.4046612 0.4104087 0.4140718 0.4148792 0.4177292 0.4280659    0
## GBM 0.4153255 0.4276816 0.4332397 0.4317596 0.4363010 0.4433127    0
```

```
## RF  0.3754790 0.3897868 0.3942409 0.3924425 0.3979209 0.4011984    0
## MLP 0.3908149 0.3931473 0.4005744 0.3998573 0.4047983 0.4122558    0
## C5  0.3898278 0.3992120 0.4041697 0.4051352 0.4084073 0.4260283    0
##
## Mean_Recall
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB  0.4026667 0.4086111 0.4131111 0.4132500 0.4166111 0.4242222    0
## GBM 0.4215556 0.4300000 0.4366667 0.4349028 0.4397222 0.4462222    0
## RF  0.3748889 0.3830000 0.3920000 0.3892917 0.3945556 0.3971111    0
## MLP 0.3960000 0.3998889 0.4027778 0.4044444 0.4092778 0.4135556    0
## C5  0.3860000 0.3930000 0.3987778 0.3997778 0.4037222 0.4211111    0
##
## Mean_Sensitivity
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB  0.4026667 0.4086111 0.4131111 0.4132500 0.4166111 0.4242222    0
## GBM 0.4215556 0.4300000 0.4366667 0.4349028 0.4397222 0.4462222    0
## RF  0.3748889 0.3830000 0.3920000 0.3892917 0.3945556 0.3971111    0
## MLP 0.3960000 0.3998889 0.4027778 0.4044444 0.4092778 0.4135556    0
## C5  0.3860000 0.3930000 0.3987778 0.3997778 0.4037222 0.4211111    0
##
## Mean_Specificity
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB  0.8506667 0.8521528 0.8532778 0.8533125 0.8541528 0.8560556    0
## GBM 0.8553889 0.8575000 0.8591667 0.8587257 0.8599306 0.8615556    0
## RF  0.8437222 0.8457500 0.8480000 0.8473229 0.8486389 0.8492778    0
## MLP 0.8490000 0.8499722 0.8506944 0.8511111 0.8523194 0.8533889    0
## C5  0.8465000 0.8482500 0.8496944 0.8499444 0.8509306 0.8552778    0
##
## prAUC
##          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## NB  0.4179170 0.4211103 0.4269191 0.4272545 0.4319427 0.4386689    0
## GBM 0.4562260 0.4709588 0.4744424 0.4736816 0.4766839 0.4876280    0
## RF  0.4134681 0.4243376 0.4280715 0.4276375 0.4317999 0.4369321    0
## MLP 0.4151405 0.4193325 0.4211189 0.4230512 0.4263797 0.4351272    0
## C5  0.3936745 0.4047484 0.4081179 0.4078656 0.4131351 0.4204168    0
```

```
dotplot(comp_modelos)
```




```
densityplot(comp_modelos, metric = "Kappa" ,auto.key = list(columns = 3))
```



```
diferencias <- diff(comp_modelos)
summary(diferencias)
```

```
##
## Call:
## summary.diff.resamples(object = diferencias)
##
## p-value adjustment: bonferroni
## Upper diagonal: estimates of the difference
## Lower diagonal: p-value for H0: difference = 0
##
## Accuracy
##     NB        GBM       RF        MLP       C5
## NB            -0.021653  0.023958  0.008806  0.013472
## GBM 1.093e-10            0.045611  0.030458  0.035125
## RF  7.628e-12 < 2.2e-16           -0.015153 -0.010486
## MLP 0.001453  3.685e-10 1.784e-05            0.004667
## C5  6.868e-05 6.107e-10 0.001427  0.461759
##
## Accuracy.1
##     NB        GBM       RF        MLP       C5
## NB            -0.021653  0.023958  0.008806  0.013472
## GBM 1.093e-10            0.045611  0.030458  0.035125
## RF  7.628e-12 < 2.2e-16           -0.015153 -0.010486
## MLP 0.001453  3.685e-10 1.784e-05            0.004667
## C5  6.868e-05 6.107e-10 0.001427  0.461759
##
## AUC
##     NB        GBM       RF        MLP       C5
## NB            -0.035096  0.006649 -0.001762  0.016579
## GBM < 2.2e-16            0.041745  0.033334  0.051675
## RF  6.431e-05 < 2.2e-16           -0.008411  0.009930
## MLP 0.3993    7.677e-14 2.690e-05            0.018341
## C5  2.356e-08 5.680e-16 9.573e-05 7.073e-09
##
## Kappa
##     NB        GBM       RF        MLP       C5
## NB            -0.027066  0.029948  0.011007  0.016840
## GBM 1.093e-10            0.057014  0.038073  0.043906
## RF  7.628e-12 < 2.2e-16           -0.018941 -0.013108
## MLP 0.001453  3.685e-10 1.784e-05            0.005833
## C5  6.868e-05 6.107e-10 0.001427  0.461759
##
## Kappa.1
##     NB        GBM       RF        MLP       C5
## NB            -0.027066  0.029948  0.011007  0.016840
## GBM 1.093e-10            0.057014  0.038073  0.043906
## RF  7.628e-12 < 2.2e-16           -0.018941 -0.013108
## MLP 0.001453  3.685e-10 1.784e-05            0.005833
## C5  6.868e-05 6.107e-10 0.001427  0.461759
##
## logLoss
##     NB        GBM       RF        MLP       C5
## NB             0.242332  0.057332  0.165354 -0.007885
## GBM < 2.2e-16           -0.185000 -0.076977 -0.250217
## RF  2.736e-08 < 2.2e-16            0.108023 -0.065217
## MLP < 2.2e-16 1.465e-14 1.601e-12           -0.173240
```

```
## C5  1.0000000 9.002e-14 0.0001216 1.987e-11
##
## Mean_Balanced_Accuracy
##     NB        GBM       RF        MLP       C5
## NB            -0.013533  0.014974  0.005503  0.008420
## GBM 1.093e-10            0.028507  0.019036  0.021953
## RF  7.628e-12 < 2.2e-16           -0.009470 -0.006554
## MLP 0.001453  3.685e-10 1.784e-05            0.002917
## C5  6.868e-05 6.107e-10 0.001427  0.461759
##
## Mean_Detection_Rate
##     NB        GBM        RF         MLP        C5
## NB            -0.0043306  0.0047917  0.0017611  0.0026944
## GBM 1.093e-10             0.0091222  0.0060917  0.0070250
## RF  7.628e-12 < 2.2e-16             -0.0030306 -0.0020972
## MLP 0.001453  3.685e-10  1.784e-05             0.0009333
## C5  6.868e-05 6.107e-10  0.001427   0.461759
##
## Mean_F1
##     NB        GBM       RF        MLP       C5
## NB            -0.021065  0.020967  0.014504  0.009489
## GBM 8.092e-10            0.042032  0.035569  0.030554
## RF  3.801e-11 < 2.2e-16           -0.006463 -0.011478
## MLP 7.402e-06 9.673e-11 0.0665975           -0.005015
## C5  0.0028082 5.013e-09 0.0004242 0.4330947
##
## Mean_Neg_Pred_Value
##     NB        GBM       RF        MLP       C5
## NB            -0.005445  0.006440  0.001596  0.003889
## GBM 6.244e-11            0.011885  0.007041  0.009334
## RF  3.922e-12 < 2.2e-16           -0.004844 -0.002551
## MLP 0.033420  1.447e-09 1.352e-06            0.002293
## C5  1.356e-05 2.708e-10 0.002306  0.007742
##
## Mean_Pos_Pred_Value
##     NB        GBM       RF        MLP       C5
## NB            -0.016880  0.022437  0.015022  0.009744
## GBM 6.765e-08            0.039317  0.031902  0.026624
## RF  3.236e-11 1.338e-15           -0.007415 -0.012693
## MLP 7.339e-06 2.150e-09 0.0501308           -0.005278
## C5  0.0030415 3.630e-08 0.0001425 0.3288351
##
## Mean_Precision
##     NB        GBM       RF        MLP       C5
## NB            -0.016880  0.022437  0.015022  0.009744
## GBM 6.765e-08            0.039317  0.031902  0.026624
## RF  3.236e-11 1.338e-15           -0.007415 -0.012693
## MLP 7.339e-06 2.150e-09 0.0501308           -0.005278
## C5  0.0030415 3.630e-08 0.0001425 0.3288351
##
## Mean_Recall
##     NB        GBM       RF        MLP       C5
## NB            -0.021653  0.023958  0.008806  0.013472
## GBM 1.093e-10            0.045611  0.030458  0.035125
## RF  7.628e-12 < 2.2e-16           -0.015153 -0.010486
```

```
## MLP 0.001453  3.685e-10 1.784e-05            0.004667
## C5   6.868e-05 6.107e-10 0.001427  0.461759
##
## Mean_Sensitivity
##      NB        GBM       RF        MLP       C5
## NB             -0.021653  0.023958  0.008806  0.013472
## GBM 1.093e-10            0.045611  0.030458  0.035125
## RF   7.628e-12 < 2.2e-16           -0.015153 -0.010486
## MLP 0.001453  3.685e-10 1.784e-05            0.004667
## C5   6.868e-05 6.107e-10 0.001427  0.461759
##
## Mean_Specificity
##      NB        GBM       RF        MLP       C5
## NB             -0.005413  0.005990  0.002201  0.003368
## GBM 1.093e-10            0.011403  0.007615  0.008781
## RF   7.628e-12 < 2.2e-16           -0.003788 -0.002622
## MLP 0.001453  3.685e-10 1.784e-05            0.001167
## C5   6.868e-05 6.107e-10 0.001427  0.461759
##
## prAUC
##      NB        GBM        RF         MLP        C5
## NB             -0.0464271 -0.0003831  0.0042032  0.0193888
## GBM 2.936e-15             0.0460440  0.0506303  0.0658159
## RF   1.00000   < 2.2e-16             0.0045863  0.0197719
## MLP 0.19388   5.704e-14  0.02432               0.0151856
## C5   6.467e-08 5.458e-15  1.231e-07  2.798e-07
```