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

Anexo 5.1. Modelado: VesselBalancedSample

Oscar Antón

diciembre de 2023

Carga de librerías, funciones y datos

```
# Librería                        # Propósito
library(tidyverse)                 # Sintaxis para el manejo de datos. Incluye dplyr, ggplo
t2, etc.
library(data.table)               # Manejo eficiente de conjuntos de datos

library(arulesCBA)                # Discretización de variables (Redes bayesianas)
library(fastDummies)              # Variables Dummy (One hot encoding)

library(caret)                    # Modelos de machine Learning
library(gbm)                       # Manejo de modelos Gradient Boosting. Debido a error va
rImp()
library(pROC)                     # Performance de modelos (curva ROC)
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(tidyverse)                 # Sintaxis para el manejo de datos. Incluye dplyr, ggplo
t2, etc.

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

```
# Cargar el dataframe VesselBalancedSample (50% barcos con incidentes, 50% barcos sin inci
dentes)
# Se lee como dataframe en vez de como datatable para evitar errores
VesselBalancedSample <- as.data.frame(readRDS("../1.DataPreprocess/DataMergedActivity/Vess
elBalancedSample.rds"))
```

Switches

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

---

1. Creación de datasets para los modelos

· 1: Propósito general: Con variables factor y numéricas (normalizadas): DataSetGeneral

· 2: Para redes Bayesanas: Con variables factoriales: DataSetFactor

· 3: Para modelos de Gradient Boosting: Con variables numéricas: DataSetNum

1.1. Dataset con variables numéricas y factor (General)

· Creación de la variable objetivo: “y” (involucrado en incidente o no). Será la última variable del dataset

· Criba de variables

· Reducción de variabilidad en categóricas

· Escalado para variables numéricas

```
# Adaptación de variables:
DataSetGeneral <- VesselBalancedSample %>%
  mutate(y = factor(ifelse(event_type != "No event", "Yes", "No"))) %>%
  select(-vessel_id, -imo_number, -vessel_name, -event_type, -damage_status) %>%
  mutate(build_year = cut(as.integer(build_year),
                          breaks = c(-Inf, 1940, 1960, 1980, 2000, Inf),
                           labels = c("very Old", "old", "average", "new", "very new"))) %
>%
  rename(vessel_length = length) %>%
  mutate_at(vars(vessel_class, flag_abbr, classification_society, solas_desc), lump_factor
ials) %>%
  mutate_at(vars(gross_ton, vessel_length), scale)
```

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

```
## 'data.frame':    109836 obs. of  8 variables:
##  $ vessel_class         : Factor w/ 11 levels "Barge","Bulk Carrier",..: 8 3 8 7 8 7 8 8 8 10 ...
##  $ build_year            : Factor w/ 5 levels "very Old","old",..: 4 3 4 3 4 3 4 3 5 5 ...
##  $ gross_ton            : num [1:109836, 1] -0.255 -0.255 -0.256 -0.255 -0.256 ...
##   ..- attr(*, "scaled:center")= num 2946
##   ..- attr(*, "scaled:scale")= num 11477
##  $ vessel_length        : num [1:109836, 1] -0.594 -0.563 -0.61 -0.58 -0.627 ...
##   ..- attr(*, "scaled:center")= num 137
##   ..- attr(*, "scaled:scale")= num 175
##  $ flag_abbr             : Factor w/ 5 levels "CA","LR","PA",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ classification_society: Factor w/ 6 levels "AMERICAN BUREAU OF SHIPPING",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ solas_desc            : Factor w/ 3 levels "Active SOLAS",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ y                    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

Nota: En este caso, la muestra ya está balanceada y no hay valores ausentes

```
# Guardado de datos
if (save_switch  == 1) {
loggedsave(DataSetGeneral, "Datasets")
}
```

1.2. Dataset con variables factor (Redes bayesianas)

```
# Aplicación del método mdlp con ayuda de la librería arulesCBA para discretizar las variables factoriales
DataSetFactor <- discretizeDF.supervised(y ~ ., DataSetGeneral)
```

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

```
## 'data.frame':	109836 obs. of  8 variables:
##  $ vessel_class         : Factor w/ 11 levels "Barge","Bulk Carrier",..: 8 3 8 7 8 7 8
8 8 10 ...
##  $ build_year           : Factor w/ 5 levels "very Old","old",..: 4 3 4 3 4 3 4 3 5 5
...
##  $ gross_ton            : Factor w/ 36 levels "[-Inf,-0.2565)",..: 5 6 3 7 3 7 3 10 6
4 ...
##   ..- attr(*, "discretized:breaks")= num [1:37] -Inf -0.256 -0.256 -0.256 -0.255 ...
##   ..- attr(*, "discretized:method")= chr "mdlp"
##  $ vessel_length        : Factor w/ 45 levels "[-Inf,-0.634)",..: 5 6 4 5 2 6 2 9 5 1
...
##   ..- attr(*, "discretized:breaks")= num [1:46] -Inf -0.634 -0.606 -0.565 -0.502 ...
##   ..- attr(*, "discretized:method")= chr "mdlp"
##  $ flag_abbr             : Factor w/ 5 levels "CA","LR","PA",..: 4 4 4 4 4 4 4 4 4 4
...
##  $ classification_society: Factor w/ 6 levels "AMERICAN BUREAU OF SHIPPING",..: 5 5 5 5
5 5 5 5 5 5 ...
##  $ solas_desc             : Factor w/ 3 levels "Active SOLAS",..: 3 3 3 3 3 3 3 3 3 3
...
##  $ y                    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
```

```
# Guardado de datos
if (save_switch  == 1) {
loggedsave(DataSetFactor, "Datasets")
}
```

1.3. Dataset con variables numéricas (Gradient Boosting)

```
# 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
DataSetNum <- cbind(
  dummy_cols(DataSetGeneral[ , c(1, 2, 5, 6, 7)], remove_selected_columns = TRUE),
  DataSetGeneral[ ,c(3, 4, 8)]
)
```

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

```
## 'data.frame':	109836 obs. of  33 variables:
##  $ 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 1 0 0 0 0 0 0 0 0 ...
##  $ vessel_class_General Dry Cargo Ship              : int  0 0 0 0 0 0 0 0 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 0 0 1 0 1 0 0 0 0 ...
##  $ vessel_class_Recreational                        : int  1 0 1 0 1 0 1 1 1 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 0 0 0 0 0 1 ...
##  $ vessel_class_other value                         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ build_year_very Old                              : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ build_year_old                                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ build_year_average                               : int  0 1 0 1 0 1 0 1 0 0 ...
##  $ build_year_new                                   : int  1 0 1 0 1 0 1 0 0 0 ...
##  $ build_year_very new                              : int  0 0 0 0 0 0 0 0 1 1 ...
##  $ flag_abbr_CA                                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ flag_abbr_LR                                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ flag_abbr_PA                                     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ flag_abbr_US                                     : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ flag_abbr_other value                            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ classification_society_AMERICAN BUREAU OF SHIPPING : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ classification_society_DET NORSKE VERITAS        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ classification_society_LLOYD'S REGISTER OF SHIPPING: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ classification_society_NIPPON KAIJI KYOKAI       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ classification_society_UNSPECIFIED               : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ classification_society_other value               : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ solas_desc_Active SOLAS                          : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ solas_desc_Historical SOLAS                      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ solas_desc_Non SOLAS                             : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ gross_ton                                         : num [1:109836, 1] -0.255 -0.255
-0.256 -0.255 -0.256 ...
##   ..- attr(*, "scaled:center")= num 2946
##   ..- attr(*, "scaled:scale")= num 11477
##  $ vessel_length                                     : num [1:109836, 1] -0.594 -0.563
-0.61 -0.58 -0.627 ...
##   ..- attr(*, "scaled:center")= num 137
##   ..- attr(*, "scaled:scale")= num 175
##  $ y                                                  : Factor w/ 2 levels "No","Yes":
1 1 1 1 1 1 1 1 1 1 ...
```

```
# Guardado de datos
if (save_switch  == 1) {
loggedsave(DataSetNum, "Datasets")
}
```

1.4. Particionado de datos

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

# Muestras de entrenamiento y test para propósito general
train_general <- DataSetGeneral[Indice_Particion, ]
test_general <- DataSetGeneral[-Indice_Particion, ]

# Muestras de entrenamiento y test para redes bayesanas
train_factor <- DataSetFactor[Indice_Particion, ]
test_factor <- DataSetFactor[-Indice_Particion, ]

# Muestras de entrenamiento y test para Gradient Boosting
train_num <- DataSetNum[ Indice_Particion, ]
test_num <- DataSetNum[ -Indice_Particion, ]
```

```r
# Guardado de datos
if (save_switch  == 1) {
datasets_particionados <- list(train_general = train_general,
                               test_general = test_general,
                               train_factor = train_factor,
                               test_factor = test_factor,
                               train_num = train_num,
                               test_num = test_num)

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

---

2. Entrenamiento de los modelos

```r
# Reset
rm(list = ls())
source("../4.Functions/myCustomFunctions.R")
train_switch <- 0
list2env(readRDS("Datasets/datasets_particionados.rds"), envir = .GlobalEnv)
```

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

Método de validación cruzada

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

2.1. Modelos de redes bayesianas

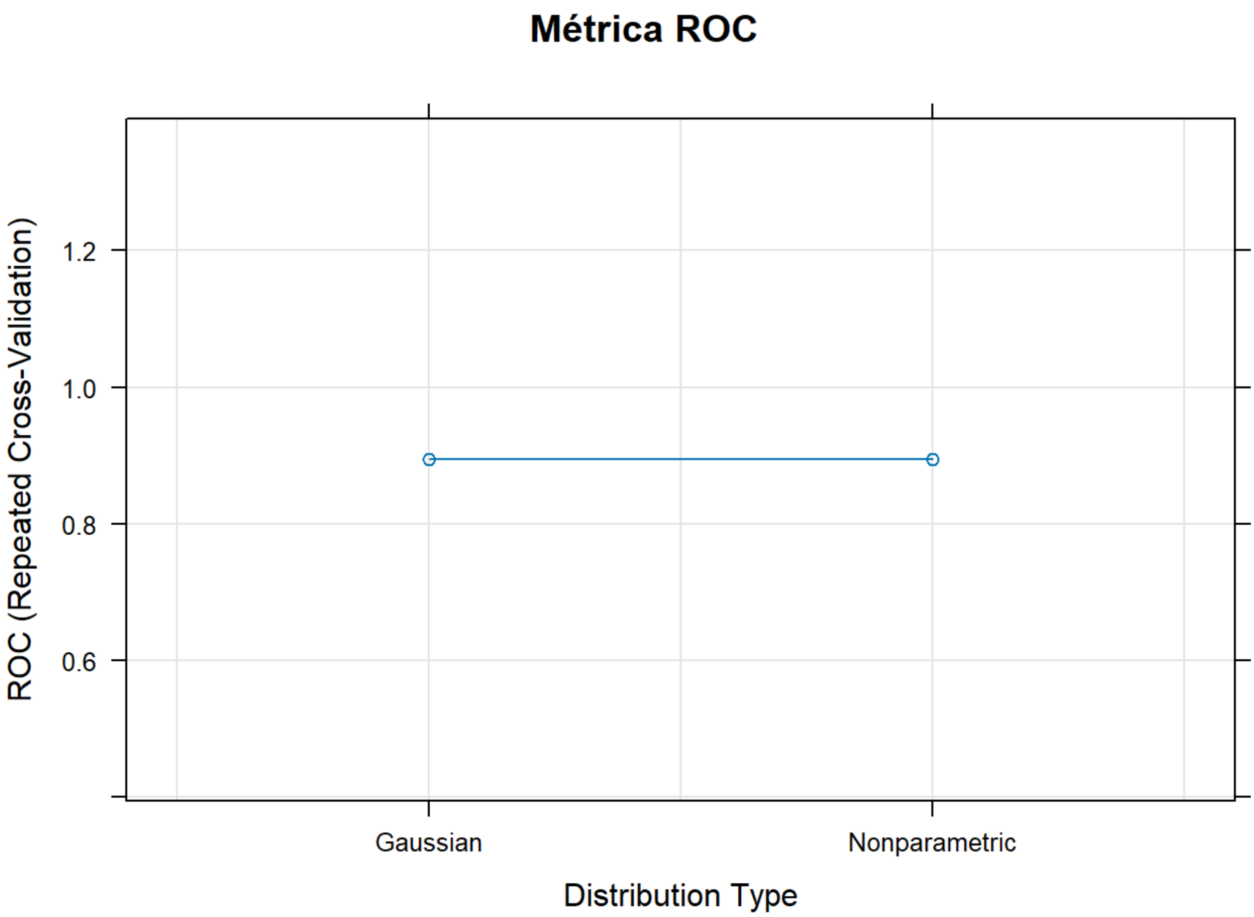
2.1.1. Naïve Bayes

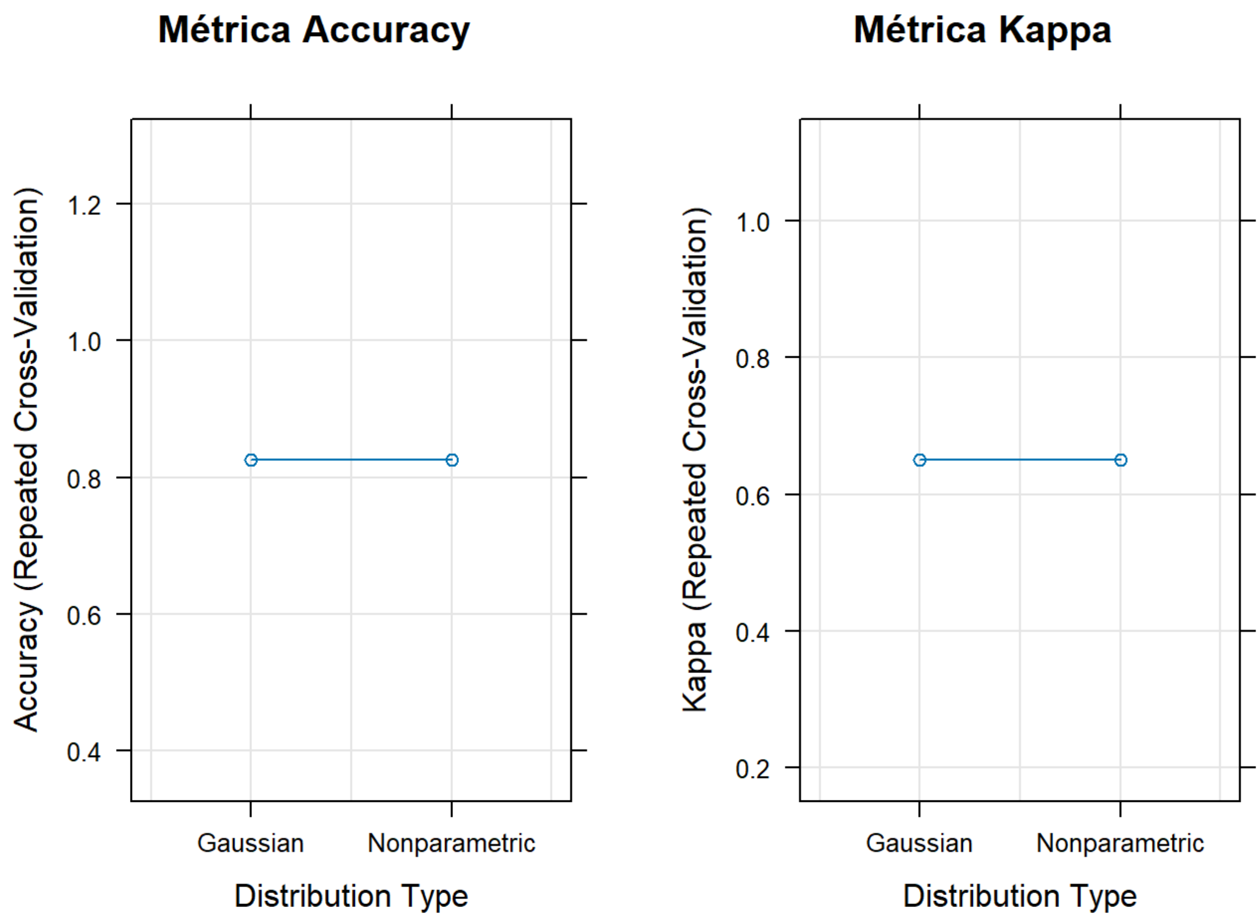
```r
if (train_switch  == 1) {
set.seed(7)

tic()

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

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


  stopCluster(clusterCPU)
  clusterCPU <- NULL

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

toc()

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

```r
# Resultado
nb_train
```

```
## Naive Bayes
##
## 87870 samples
##     7 predictor
##     2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 76887, 76886, 76886, 76887, 76886, 76886, ...
## Resampling results across tuning parameters:
##
##   usekernel  ROC        Sens       Spec       Accuracy   Kappa
##   FALSE      0.8934897  0.7932856  0.8563561  0.8248208  0.6496417
##    TRUE      0.8934897  0.7932856  0.8563561  0.8248208  0.6496417
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = FALSE and adjust
##  = 1.
```

```
# Métricas
grafico_metricas(nb_train)
```






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

RESULTADOS DEL MODELO Naive Bayes

usekernel	fL	adjust	ROC	Sens	Spec	Accuracy	Kappa	ROCSD	S
FALSE	0	1	0.8934897	0.7932856	0.8563561	0.8248208	0.6496417	0.0022782	0.(
TRUE	0	1	0.8934897	0.7932856	0.8563561	0.8248208	0.6496417	0.0022782	0.(

```
# Mejor modelo
mejor_modelo(nb_train)
```

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

fL	usekernel	adjust
0	FALSE	1

```
curvas_ROC(nb_train, "de Naïve Bayes", train_factor, test_factor)
```

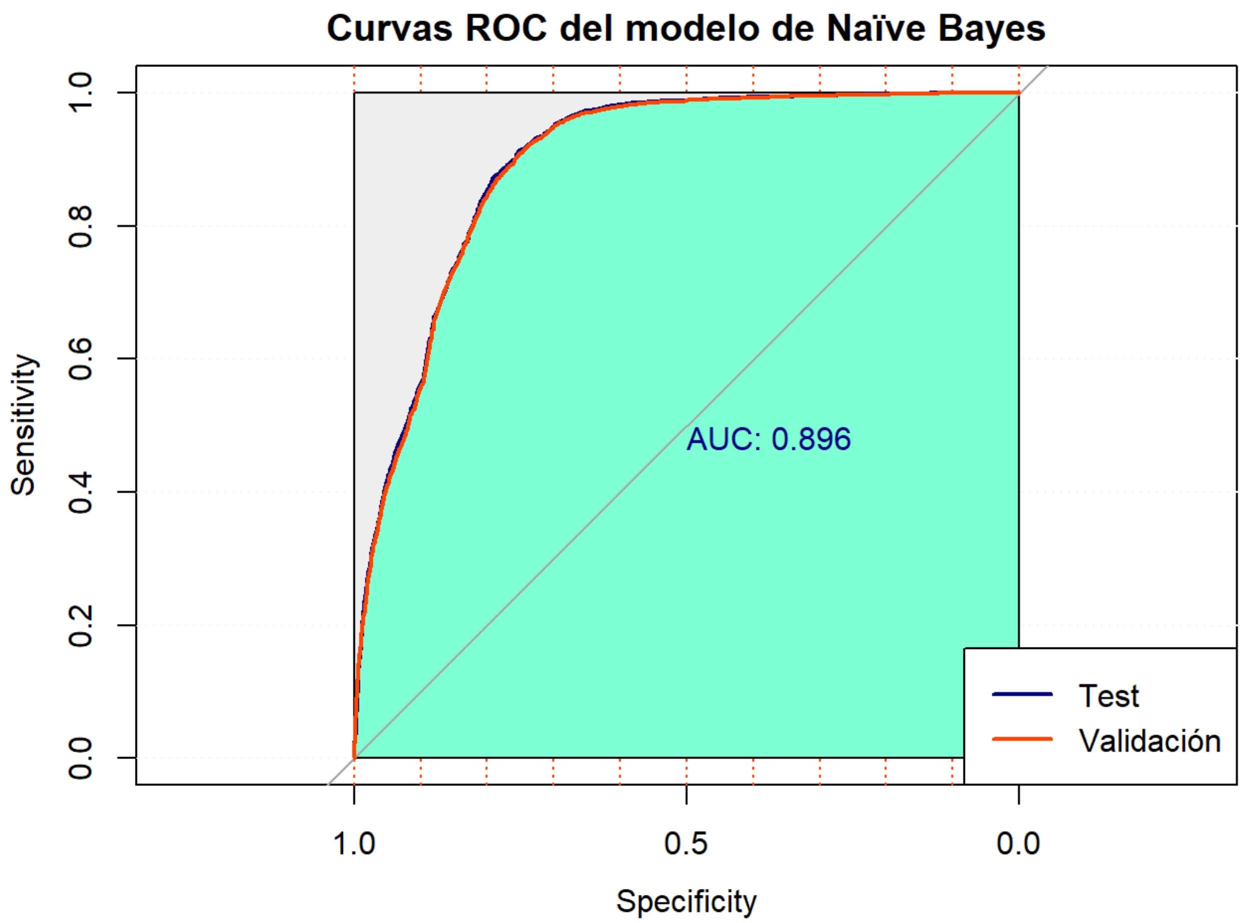
```
## Registered S3 method overwritten by 'questionr':
##   method            from
##   print.description DALEX
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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




```
## [1] "ROC del modelo con el fichero de test: 0.895760902501112"
```

```
validation(nb_train, "de Naïve Bayes", train_factor, test_factor)
```

```
## [1] "Modelo de Naïve Bayes - Tabla de confusión para los datos de entrenamiento"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  34851  6318
##        Yes  9084 37617
##
##                Accuracy : 0.8247
##                  95% CI : (0.8222, 0.8272)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6494
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7932
##             Specificity : 0.8562
##          Pos Pred Value : 0.8465
##          Neg Pred Value : 0.8055
##              Prevalence : 0.5000
##          Detection Rate : 0.3966
##    Detection Prevalence : 0.4685
##       Balanced Accuracy : 0.8247
##
##        'Positive' Class : No
##
## [1] "Modelo de Naïve Bayes - Tabla de confusión para los datos de validación"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  8733 1519
##        Yes 2250 9464
##
##                Accuracy : 0.8284
##                  95% CI : (0.8234, 0.8334)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6568
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7951
##             Specificity : 0.8617
##          Pos Pred Value : 0.8518
##          Neg Pred Value : 0.8079
##              Prevalence : 0.5000
##          Detection Rate : 0.3976
##    Detection Prevalence : 0.4667
##       Balanced Accuracy : 0.8284
##
##        'Positive' Class : No
##
```

```
resumen_nb <- resumen(nb_train, train_factor, test_factor)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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

```
resumen_nb %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ", "Naïve Bayes Classifier" = 7))
```

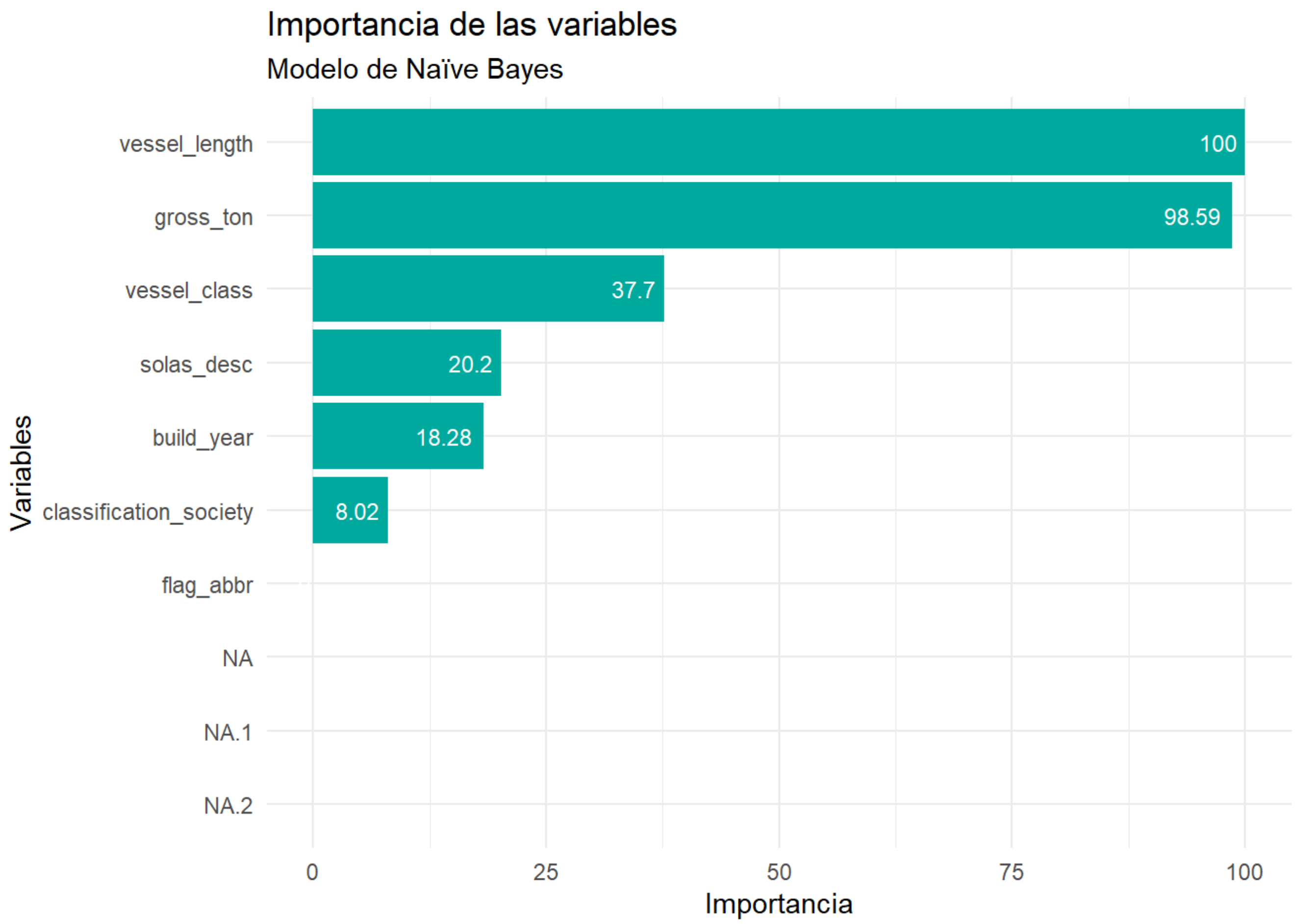
	Naïve Bayes Classifier						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.894	0.825	0.847	0.805	0.649	0.793	0.856

	Naïve Bayes Classifier						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Validación	0.896	0.828	0.852	0.808	0.657	0.795	0.862

```
importancia_var(nb_train, "de Naïve Bayes")
```

```
## Warning: Removed 3 rows containing missing values (`position_stack()`).
## Removed 3 rows containing missing values (`position_stack()`).
```

```
## Warning: Removed 3 rows containing missing values (`geom_text()`).
```



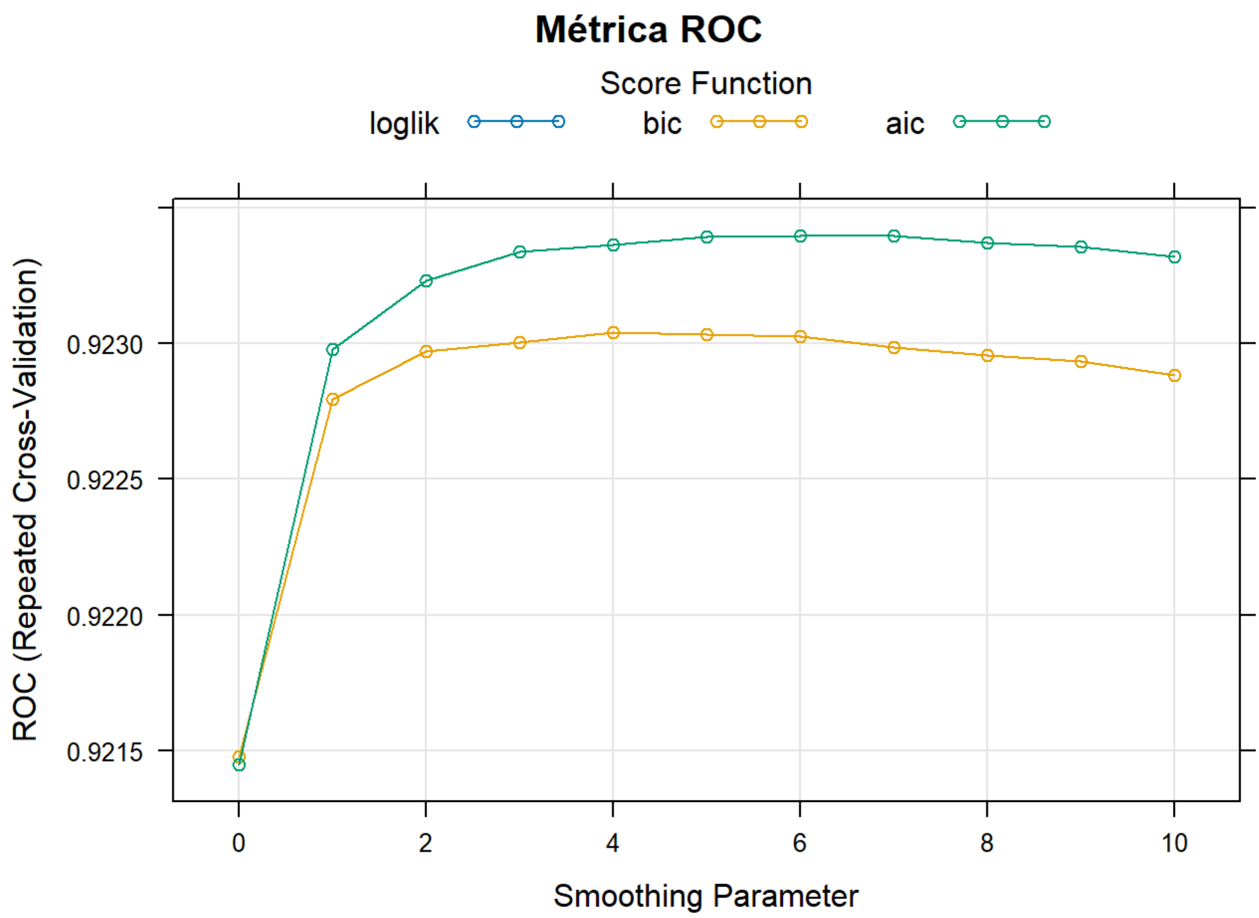
2.1.2. Modelo TAN

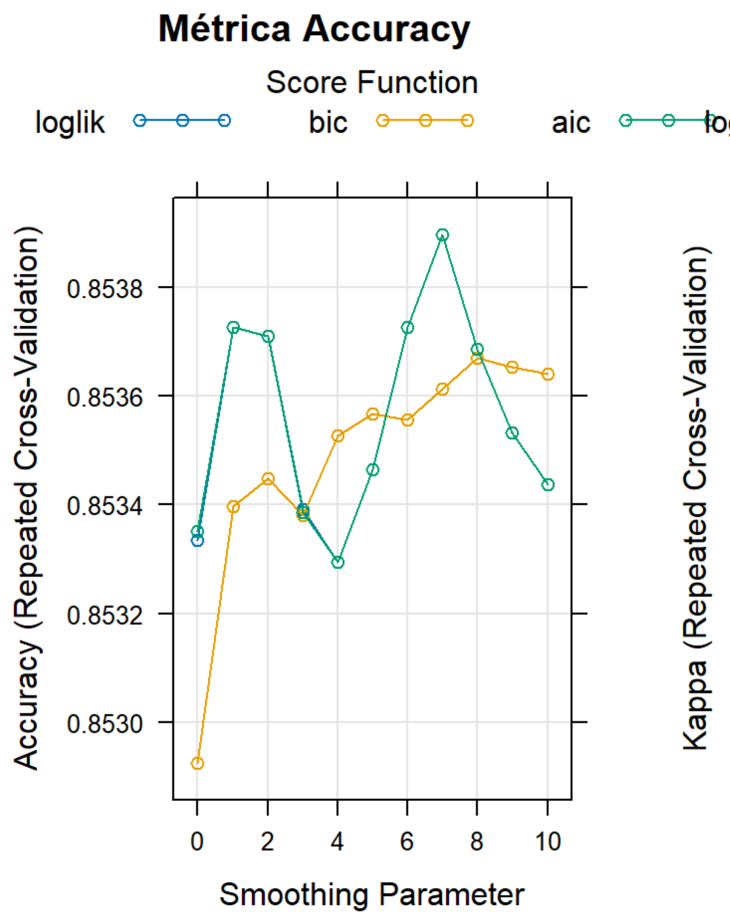
```r
# Entrenamiento
if (train_switch  == 1) {
set.seed(7)

tic()

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

TAN_Grid <- expand.grid(score = c( 'loglik', 'bic', 'aic' ),
                         smooth = seq(from = 0, to = 10, by = 1))
tan_train <- train(train_factor[,-length(train_factor)],
                   train_factor$y,
                 method = 'tan',
                 metric = metrica,
                 trControl = control,
                 tuneGrid = TAN_Grid)

stopCluster(clusterCPU)

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

toc()

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

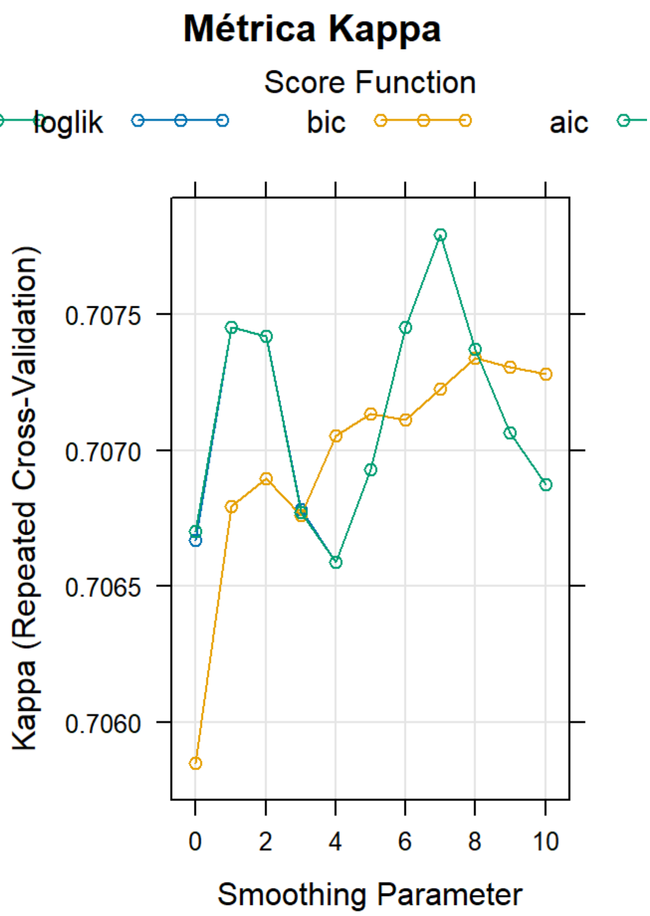
```r
# Resultado
tan_train
```

```
## Tree Augmented Naive Bayes Classifier
##
## 87870 samples
##     7 predictor
##     2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 76887, 76886, 76886, 76887, 76886, 76886, ...
## Resampling results across tuning parameters:
##
##   score   smooth  ROC        Sens       Spec       Accuracy   Kappa
##   loglik   0      0.9214476  0.8221806  0.8844884  0.8533344  0.7066689
##   loglik   1      0.9229789  0.8227609  0.8846933  0.8537271  0.7074541
##   loglik   2      0.9232324  0.8230454  0.8843746  0.8537100  0.7074200
##   loglik   3      0.9233388  0.8230796  0.8837032  0.8533913  0.7067827
##   loglik   4      0.9233630  0.8227609  0.8838284  0.8532946  0.7065892
##   loglik   5      0.9233953  0.8223057  0.8846250  0.8534653  0.7069307
##   loglik   6      0.9233969  0.8221578  0.8852964  0.8537271  0.7074542
##   loglik   7      0.9233983  0.8218278  0.8859679  0.8538978  0.7077956
##   loglik   8      0.9233705  0.8206215  0.8867531  0.8536872  0.7073745
##   loglik   9      0.9233558  0.8199728  0.8870945  0.8535336  0.7070672
##   loglik  10      0.9233195  0.8191875  0.8876863  0.8534369  0.7068738
##   bic      0      0.9214780  0.8209173  0.8849323  0.8529247  0.7058495
##   bic      1      0.9227936  0.8209742  0.8858199  0.8533970  0.7067941
##   bic      2      0.9229711  0.8213156  0.8855809  0.8534482  0.7068965
##   bic      3      0.9230050  0.8212132  0.8855468  0.8533800  0.7067599
##   bic      4      0.9230404  0.8211904  0.8858654  0.8535279  0.7070558
##   bic      5      0.9230340  0.8210311  0.8861044  0.8535677  0.7071355
##   bic      6      0.9230260  0.8206670  0.8864459  0.8535564  0.7071128
##   bic      7      0.9229855  0.8205987  0.8866280  0.8536133  0.7072266
##   bic      8      0.9229578  0.8202345  0.8871059  0.8536702  0.7073404
##   bic      9      0.9229329  0.8200524  0.8872539  0.8536531  0.7073062
##   bic     10      0.9228827  0.8193810  0.8879026  0.8536417  0.7072835
##   aic      0      0.9214476  0.8222375  0.8844657  0.8533515  0.7067031
##   aic      1      0.9229789  0.8227609  0.8846933  0.8537271  0.7074541
##   aic      2      0.9232324  0.8230454  0.8843746  0.8537100  0.7074200
##   aic      3      0.9233388  0.8230796  0.8836918  0.8533856  0.7067713
##   aic      4      0.9233630  0.8227609  0.8838284  0.8532946  0.7065892
##   aic      5      0.9233953  0.8223057  0.8846250  0.8534653  0.7069307
##   aic      6      0.9233969  0.8221578  0.8852964  0.8537271  0.7074542
##   aic      7      0.9233983  0.8218278  0.8859679  0.8538978  0.7077956
##   aic      8      0.9233705  0.8206215  0.8867531  0.8536872  0.7073745
##   aic      9      0.9233558  0.8199728  0.8870945  0.8535336  0.7070672
##   aic     10      0.9233195  0.8191875  0.8876863  0.8534369  0.7068738
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were score = loglik and smooth = 7.
```

```
grafico_metricas(tan_train)
```








```
resultados(tan_train, "Tree Augmented Naïve Bayes")
```

RESULTADOS DEL MODELO Tree Augmented Naïve Bayes

score	smooth	ROC	Sens	Spec	Accuracy	Kappa	ROCSD	SensSD
loglik	0	0.9214476	0.8221806	0.8844884	0.8533344	0.7066689	0.0018608	0.005289
loglik	1	0.9229789	0.8227609	0.8846933	0.8537271	0.7074541	0.0017163	0.005041
loglik	2	0.9232324	0.8230454	0.8843746	0.8537100	0.7074200	0.0017065	0.005087
loglik	3	0.9233388	0.8230796	0.8837032	0.8533913	0.7067827	0.0017266	0.005216
loglik	4	0.9233630	0.8227609	0.8838284	0.8532946	0.7065892	0.0017429	0.005362
loglik	5	0.9233953	0.8223057	0.8846250	0.8534653	0.7069307	0.0017133	0.005490
loglik	6	0.9233969	0.8221578	0.8852964	0.8537271	0.7074542	0.0017233	0.005530
loglik	7	0.9233983	0.8218278	0.8859679	0.8538978	0.7077956	0.0017149	0.005441
loglik	8	0.9233705	0.8206215	0.8867531	0.8536872	0.7073745	0.0017335	0.005936
loglik	9	0.9233558	0.8199728	0.8870945	0.8535336	0.7070672	0.0017343	0.006000
loglik	10	0.9233195	0.8191875	0.8876863	0.8534369	0.7068738	0.0017266	0.006025
bic	0	0.9214780	0.8209173	0.8849323	0.8529247	0.7058495	0.0019555	0.006132

```
mejor_modelo(tan_train)
```

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

	score	smooth
8	loglik	7

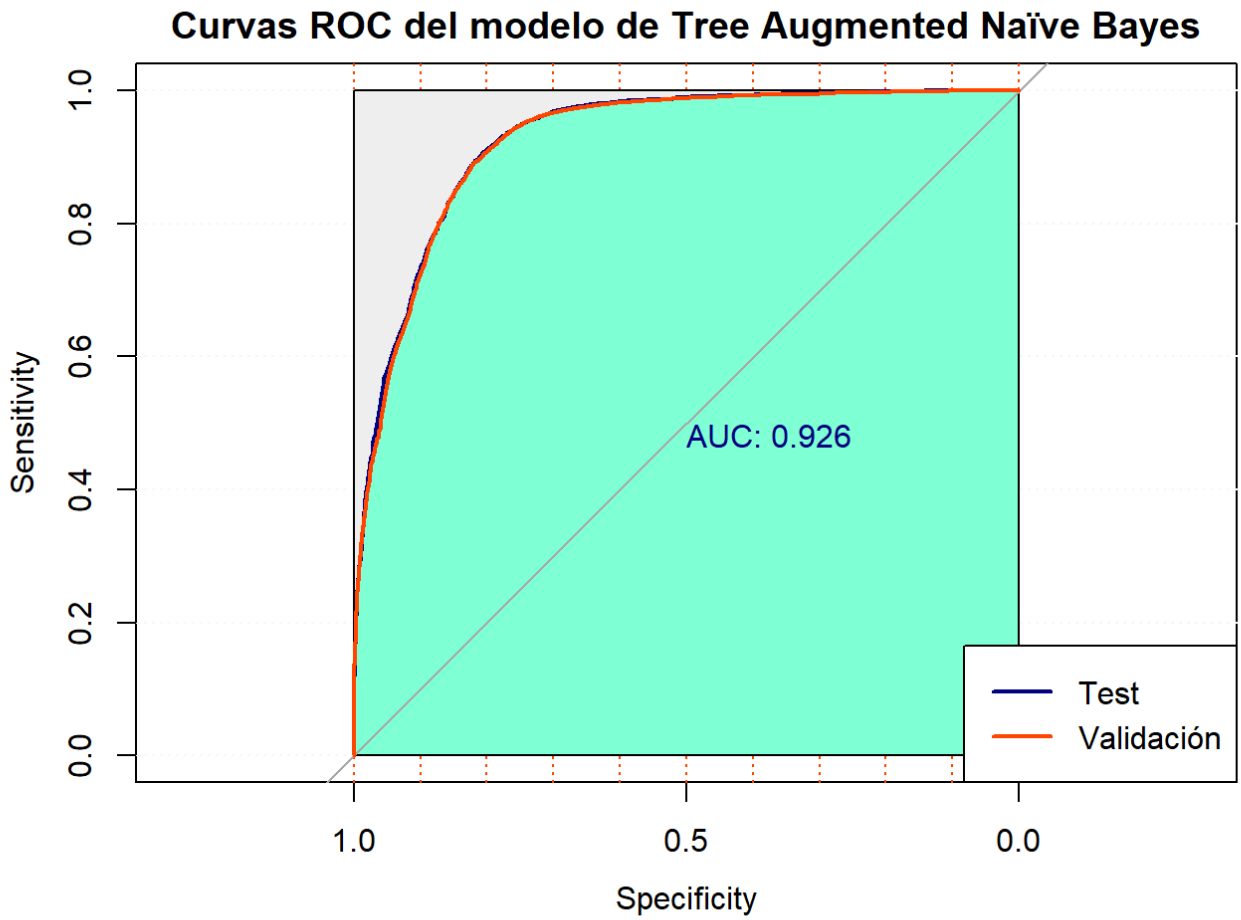
```
curvas_ROC(tan_train, "de Tree Augmented Naïve Bayes", train_factor, test_factor)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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




```
## [1] "ROC del modelo con el fichero de test: 0.926023339738156"
```

```
validation(tan_train, "de Tree Augmented Naïve Bayes", train_factor, test_factor)
```

```
## [1] "Modelo de Tree Augmented Naïve Bayes - Tabla de confusión para los datos de entren
amiento"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  36120  4949
##        Yes  7815 38986
##
##                Accuracy : 0.8547
##                  95% CI : (0.8524, 0.8571)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7095
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8221
##             Specificity : 0.8874
##          Pos Pred Value : 0.8795
##          Neg Pred Value : 0.8330
##              Prevalence : 0.5000
##          Detection Rate : 0.4111
##    Detection Prevalence : 0.4674
##       Balanced Accuracy : 0.8547
##
##        'Positive' Class : No
##
## [1] "Modelo de Tree Augmented Naïve Bayes - Tabla de confusión para los datos de valida
ción"
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   No  Yes
##        No  9031 1228
##        Yes 1952 9755
## 
##                Accuracy : 0.8552
##                  95% CI : (0.8505, 0.8599)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.7105
## 
##  Mcnemar's Test P-Value : < 2.2e-16
## 
##             Sensitivity : 0.8223
##             Specificity : 0.8882
##          Pos Pred Value : 0.8803
##          Neg Pred Value : 0.8333
##              Prevalence : 0.5000
##          Detection Rate : 0.4111
##    Detection Prevalence : 0.4670
##       Balanced Accuracy : 0.8552
## 
##        'Positive' Class : No
## 
```

```
resumen_tan <- resumen(tan_train, train_factor, test_factor)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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

```
resumen_tan %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ", "Tree Augmented Naïve Bayes" = 7))
```

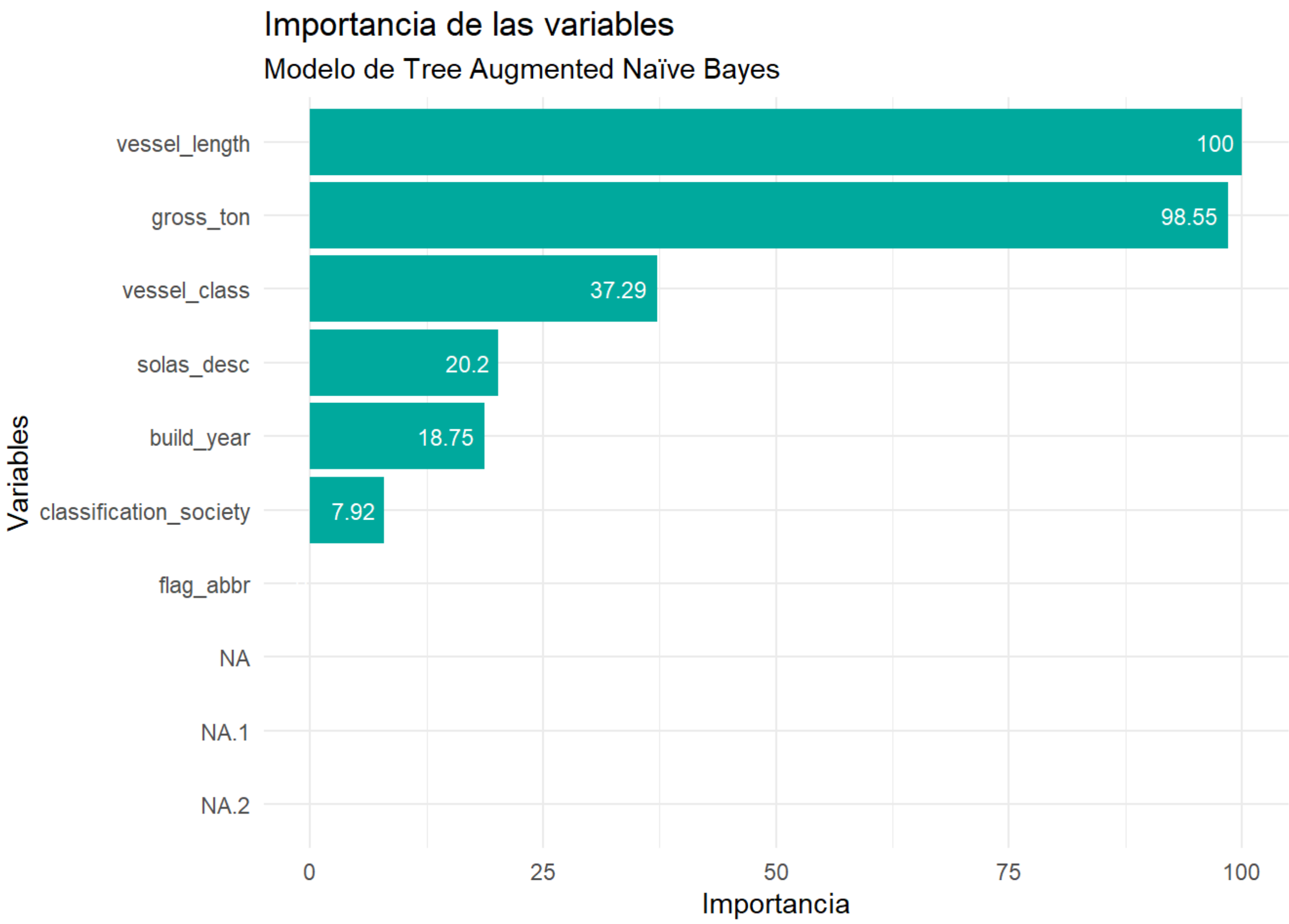
	Tree Augmented Naïve Bayes						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.924	0.855	0.879	0.833	0.709	0.822	0.887

	Tree Augmented Naïve Bayes						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Validación	0.926	0.855	0.880	0.833	0.710	0.822	0.888

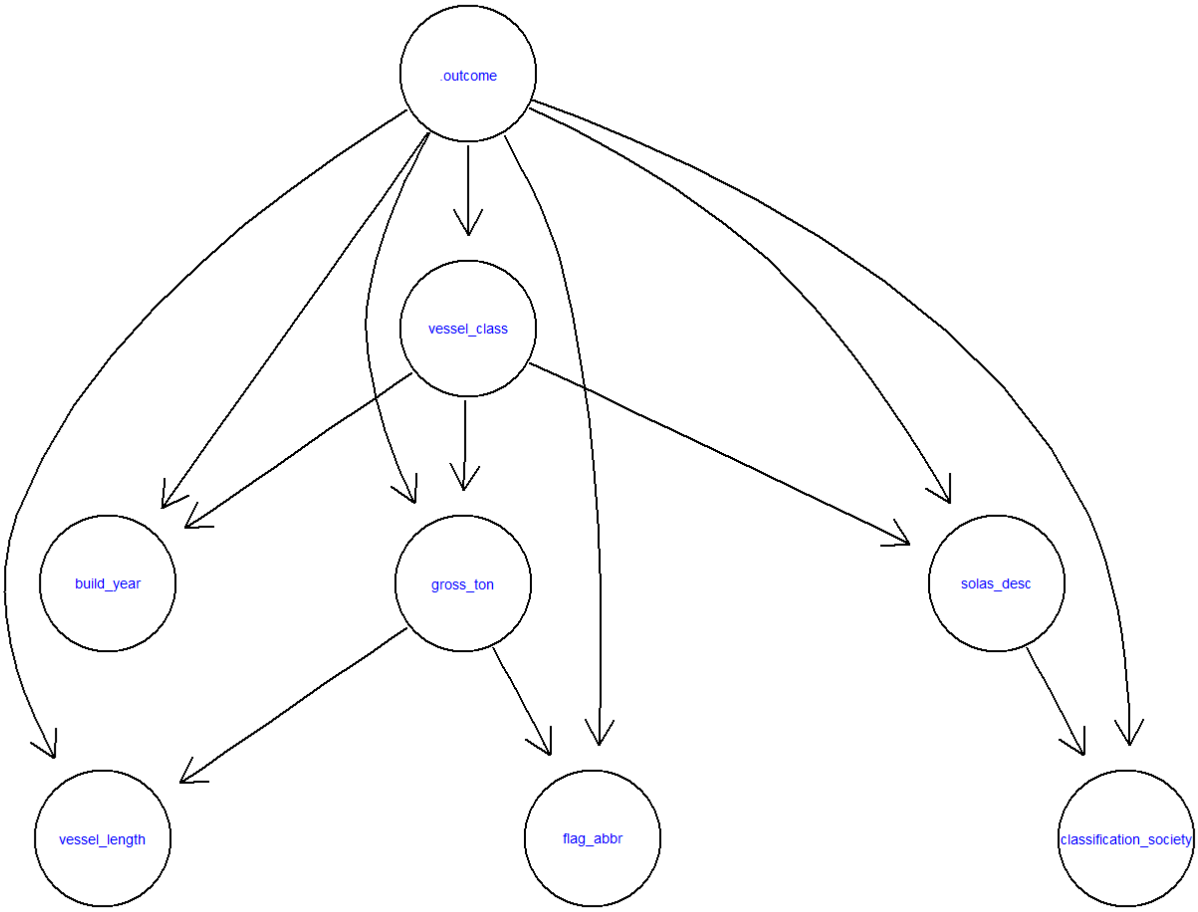
```
importancia_var(tan_train, "de Tree Augmented Naïve Bayes")
```

```
## Warning: Removed 3 rows containing missing values (`position_stack()`).
## Removed 3 rows containing missing values (`position_stack()`).
```

```
## Warning: Removed 3 rows containing missing values (`geom_text()`).
```




```
Rgraphviz::plot(tan_train$finalModel)
```



2.1.3. Modelo TAN Search

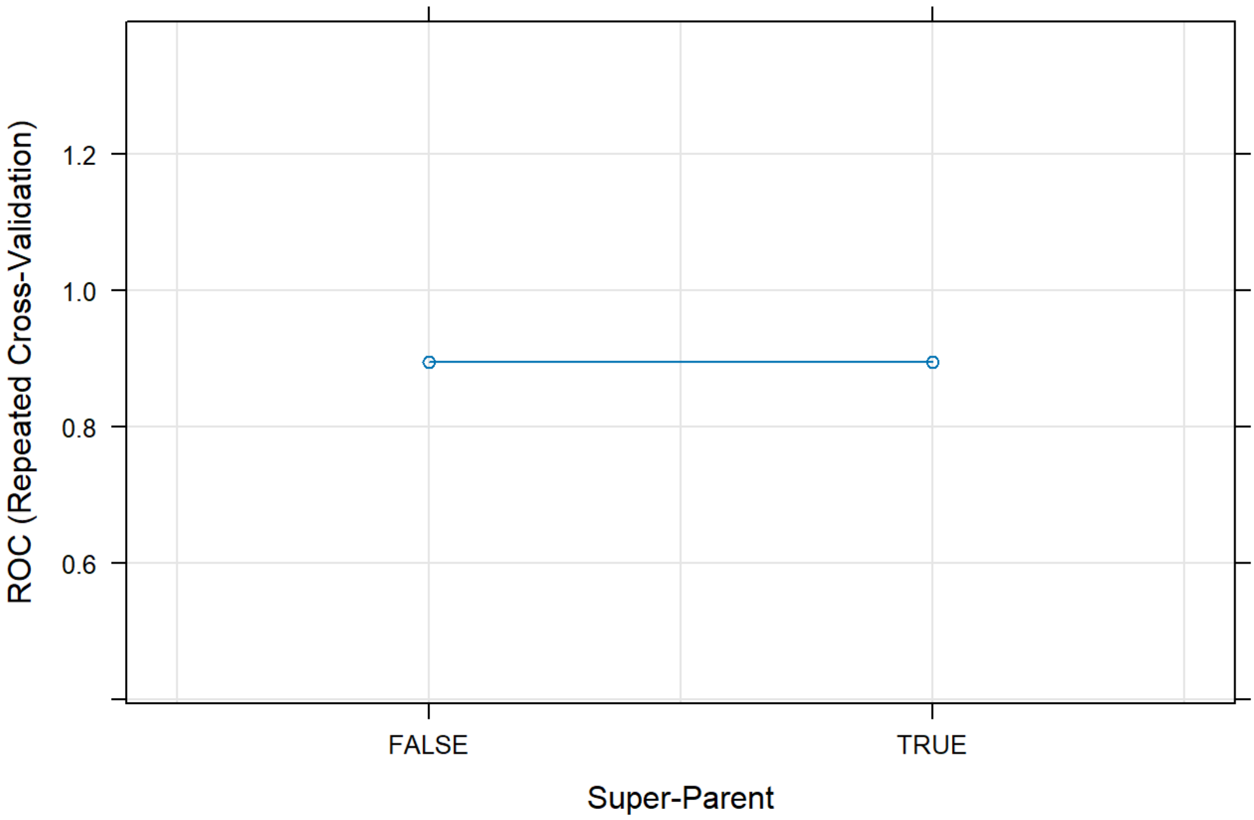
```
# Entrenamiento
if (train_switch  == 1) {
set.seed(7)

tic()

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

  tanse_train <- train(train_factor[,-length(train_factor)],
                      train_factor$y,
                      method = 'tanSearch',
                      metric=metrica,
                      trControl=control)
    stopCluster(clusterCPU)
  clusterCPU <- NULL

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

toc()

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

```r
# Resultados
tanse_train
```

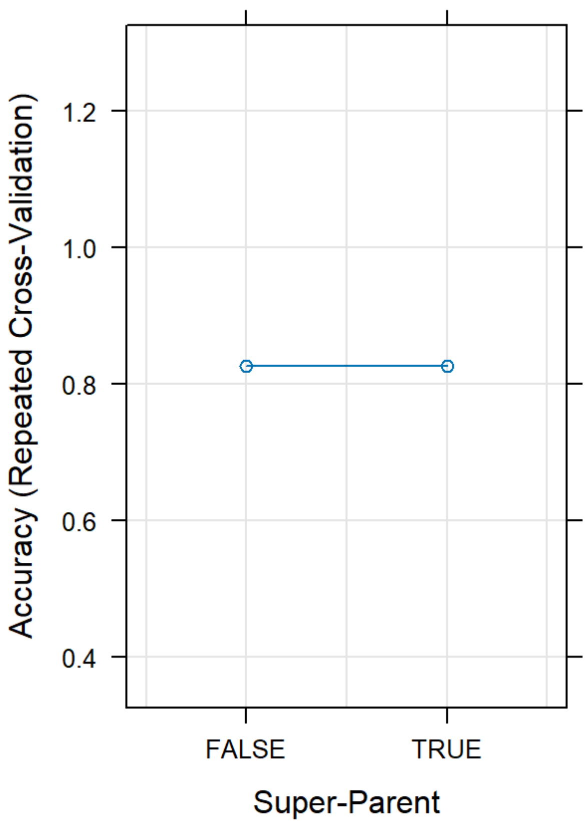
```
## Tree Augmented Naive Bayes Classifier Structure Learner Wrapper
## 
## 87870 samples
##     7 predictor
##     2 classes: 'No', 'Yes'
## 
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 76887, 76886, 76886, 76887, 76886, 76886, ... 
## Resampling results across tuning parameters:
## 
##   sp     ROC        Sens       Spec       Accuracy   Kappa    
##   FALSE  0.8942328  0.7946171  0.8574941  0.8260555  0.6521111
##    TRUE  0.8942328  0.7946171  0.8574941  0.8260555  0.6521111
## 
## Tuning parameter 'k' was held constant at a value of 10
## Tuning
##  parameter 'smooth' was held constant at a value of 0.01
## Tuning
##  parameter 'final_smooth' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were k = 10, epsilon = 0.01, smooth =
##  0.01, final_smooth = 1 and sp = FALSE.
```

```r
# Gráfico de métricas
grafico_metricas(tanse_train)
```

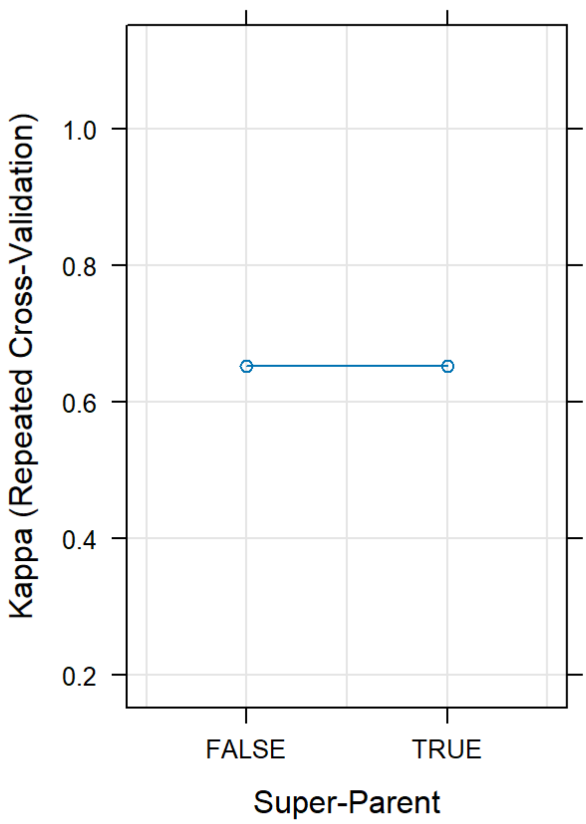
Métrica ROC



Métrica Accuracy



Métrica Kappa




```
resultados(tanse_train, "Search Tree Augmented Naïve Bayes")
```

RESULTADOS DEL MODELO Search Tree Augmented Naïve Bayes

k	epsilon	smooth	final_smooth	sp	ROC	Sens	Spec	Accuracy	
10	0.01	0.01	1	FALSE	0.8942328	0.7946171	0.8574941	0.8260555	0.6!
10	0.01	0.01	1	TRUE	0.8942328	0.7946171	0.8574941	0.8260555	0.6!

```
mejor_modelo(tanse_train)
```

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

k	epsilon	smooth	final_smooth	sp
10	0.01	0.01	1	FALSE

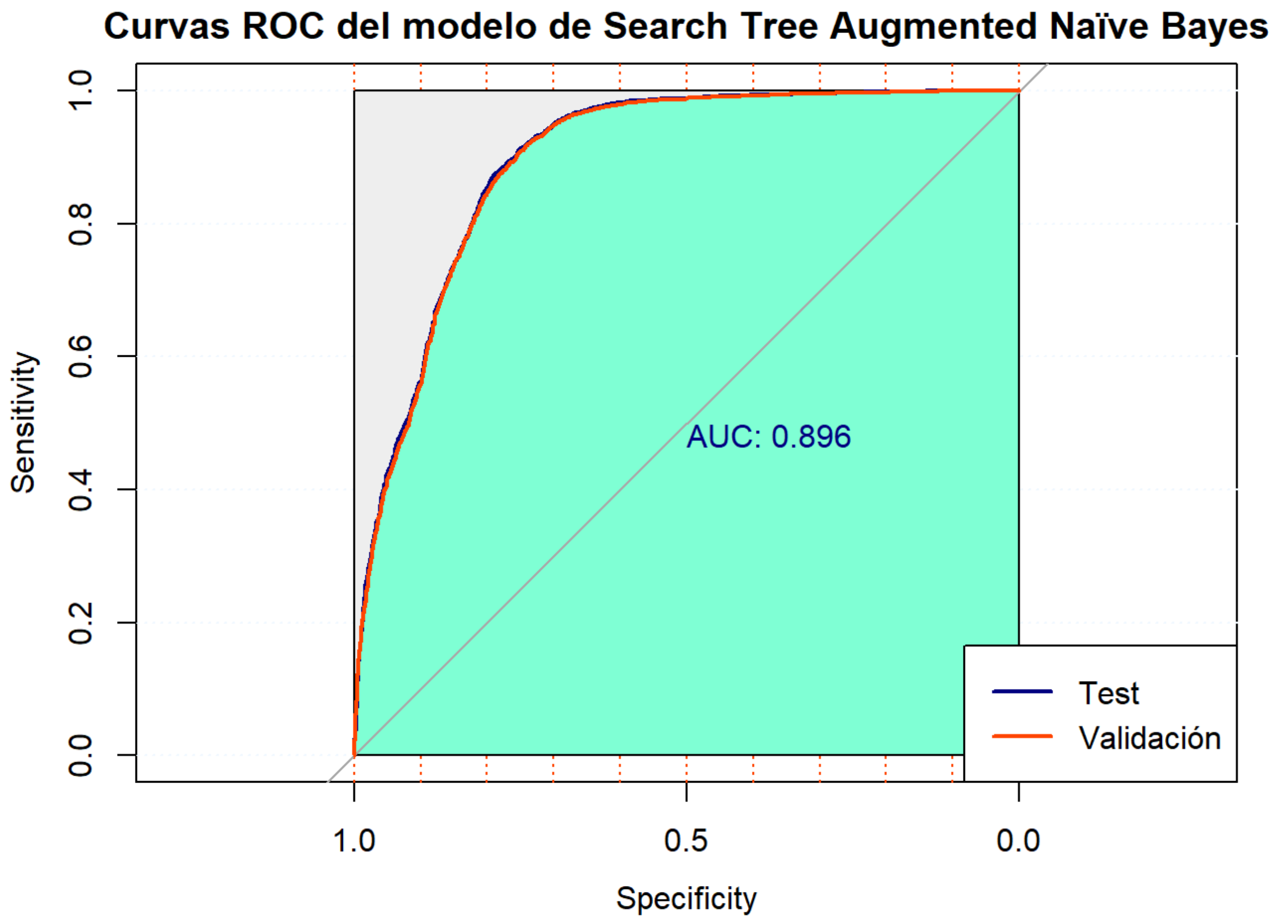
```
curvas_ROC(tanse_train, "de Search Tree Augmented Naïve Bayes", train_factor, test_factor)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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




```
## [1] "ROC del modelo con el fichero de test: 0.896123692406719"
```

```
validation(tanse_train, "de Search Tree Augmented Naïve Bayes", train_factor, test_factor)
```

```
## [1] "Modelo de Search Tree Augmented Naïve Bayes - Tabla de confusión para los datos de entrenamiento"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  34870  6344
##        Yes  9065 37591
##
##                Accuracy : 0.8246
##                  95% CI : (0.8221, 0.8271)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6493
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7937
##             Specificity : 0.8556
##          Pos Pred Value : 0.8461
##          Neg Pred Value : 0.8057
##              Prevalence : 0.5000
##          Detection Rate : 0.3968
##    Detection Prevalence : 0.4690
##       Balanced Accuracy : 0.8246
##
##        'Positive' Class : No
##
## [1] "Modelo de Search Tree Augmented Naïve Bayes - Tabla de confusión para los datos de validación"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  8742 1522
##        Yes 2241 9461
##
##                Accuracy : 0.8287
##                  95% CI : (0.8236, 0.8337)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6574
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7960
##             Specificity : 0.8614
##          Pos Pred Value : 0.8517
##          Neg Pred Value : 0.8085
##              Prevalence : 0.5000
##          Detection Rate : 0.3980
##    Detection Prevalence : 0.4673
##       Balanced Accuracy : 0.8287
##
##        'Positive' Class : No
##
```

```
resumen_tanse <- resumen(tanse_train, train_factor, test_factor)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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

```
resumen_tanse %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ", "Search Tree Augmented Naïve Bayes" = 7))
```

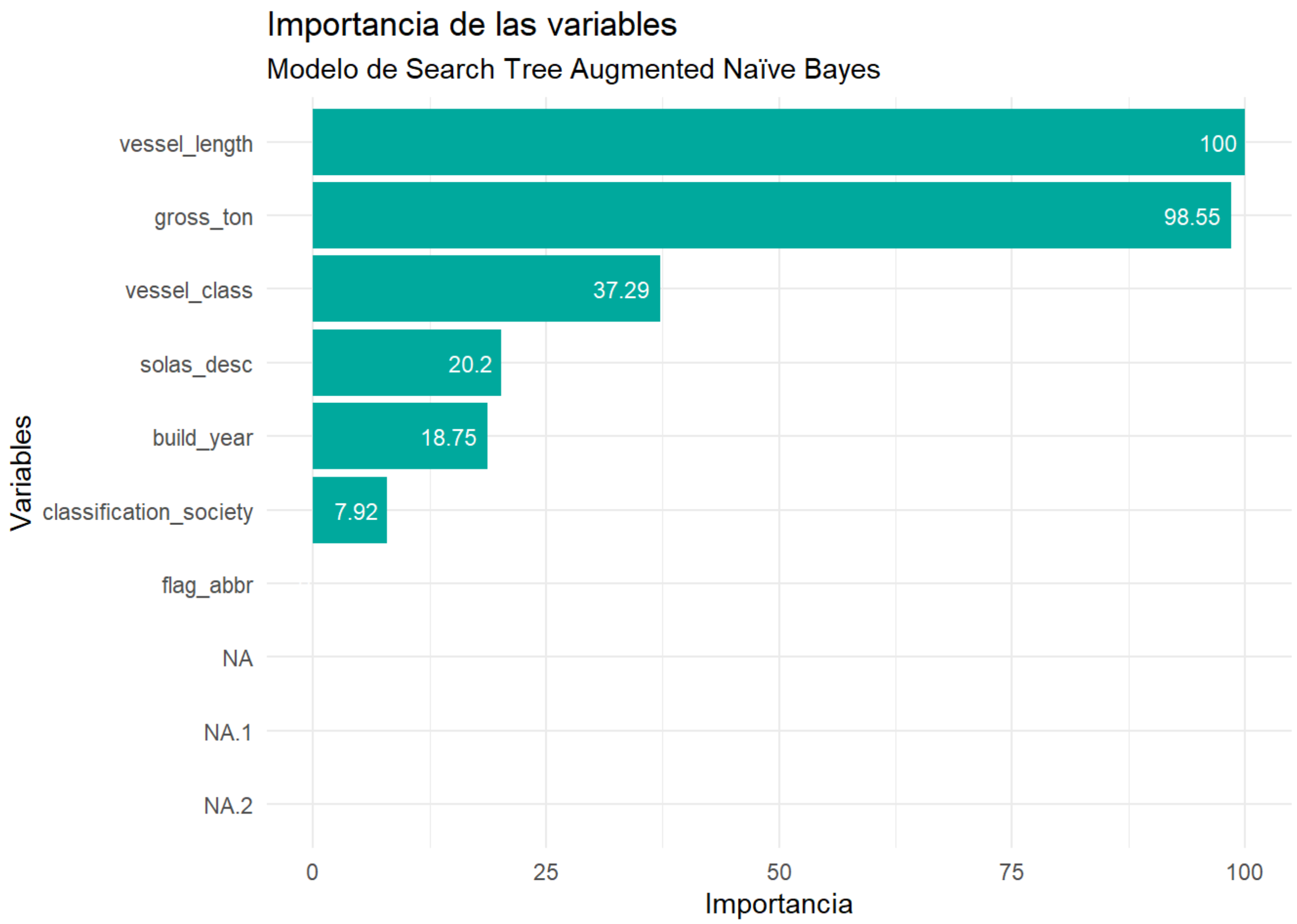
	Search Tree Augmented Naïve Bayes						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.894	0.825	0.846	0.806	0.649	0.794	0.856

Search Tree Augmented Naïve Bayes							
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Validación	0.896	0.829	0.852	0.808	0.657	0.796	0.861

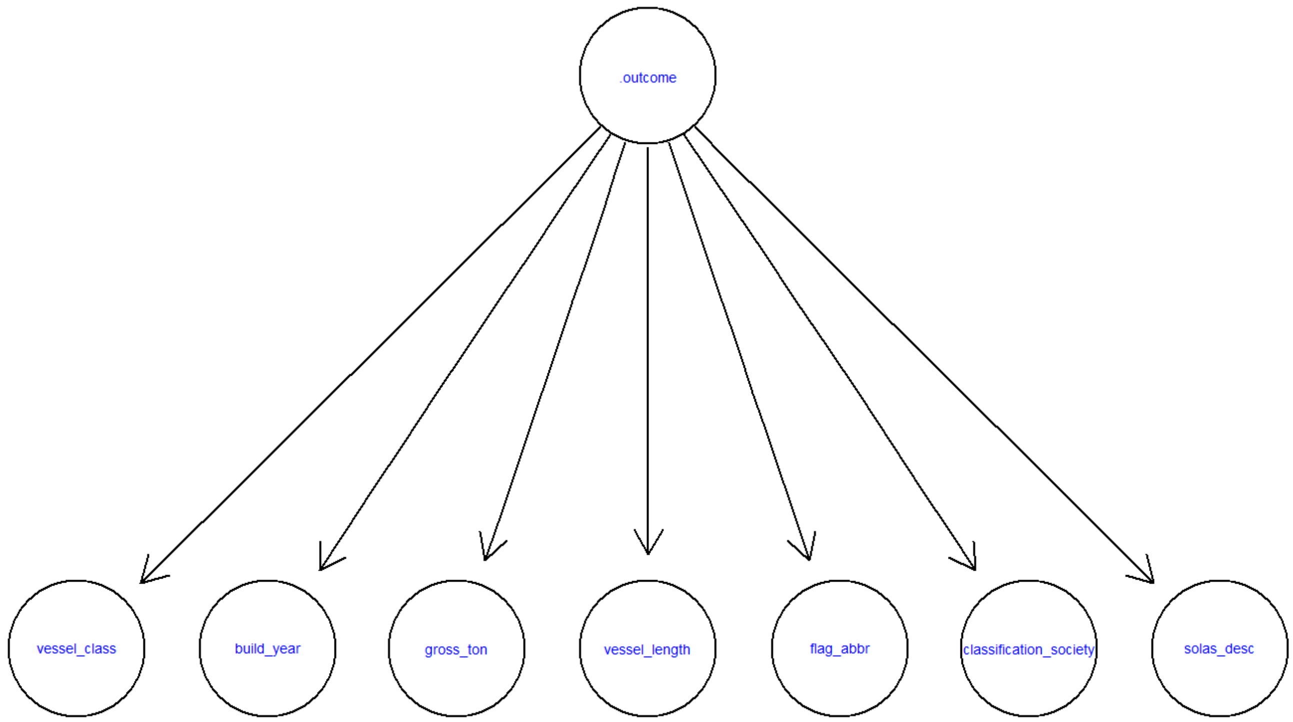
```
importancia_var(tanse_train, "de Search Tree Augmented Naïve Bayes")
```

```
## Warning: Removed 3 rows containing missing values (`position_stack()`).
## Removed 3 rows containing missing values (`position_stack()`).
```

```
## Warning: Removed 3 rows containing missing values (`geom_text()`).
```




```
Rgraphviz::plot(tanse_train$finalModel)
```



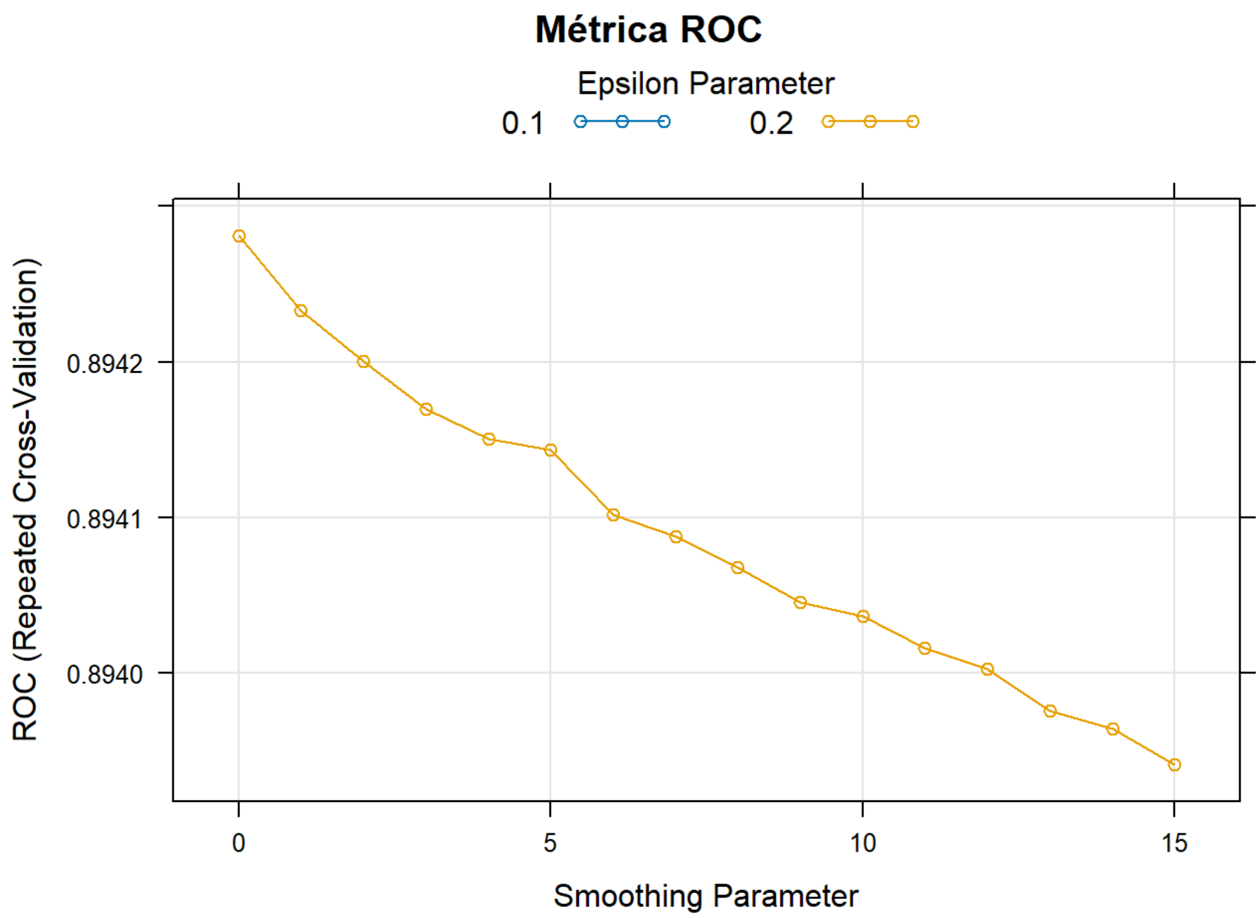
2.1.4. Modelo TAN Hill Climbing

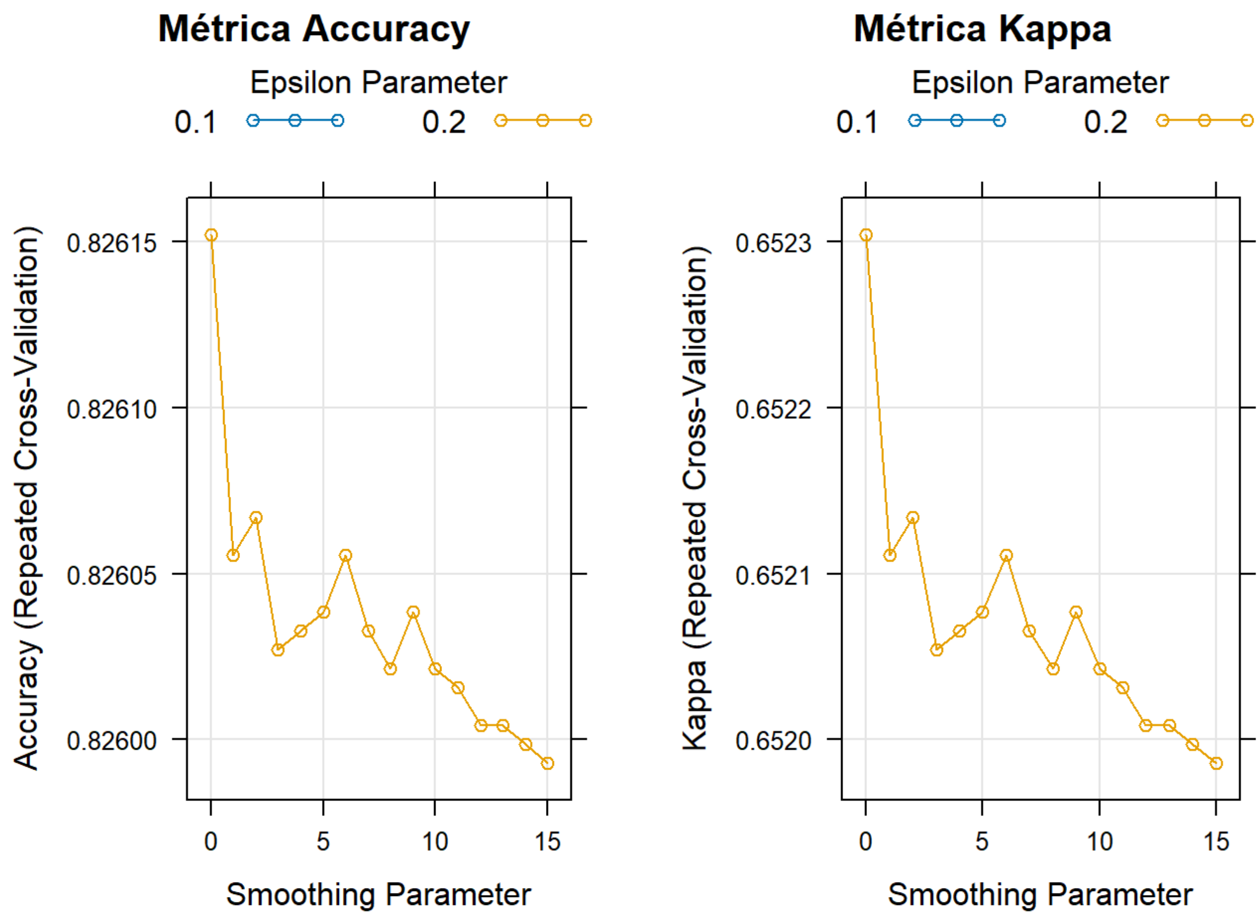
```r
# Entrenamiento
if (train_switch  == 1) {
set.seed(7)

tic()

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

TANHC_Grid <- expand.grid(smooth = seq(from = 0, to = 15, by=1),epsilon=c(0.1,0.2))

tanhc_train <- train(train_factor[,-length(train_factor)],
                   train_factor$y,
                   method = AlgTANHC,
                   metric = metrica,
                   trControl = control,
                   tuneGrid = TANHC_Grid)


stopCluster(clusterCPU)
saveRDS(tanhc_train, "Models/tanhc_train.RDS")

toc()

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

```r
# Resultados
tanhc_train
```

```
## Hill Climbing Tree Augmented Naive Bayes Classifier
##
## 87870 samples
##     7 predictor
##     2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 76887, 76886, 76886, 76887, 76886, 76886, ...
## Resampling results across tuning parameters:
##
##   smooth  epsilon  ROC        Sens       Spec       Accuracy   Kappa
##    0      0.1      0.8942812  0.7948219  0.8574827  0.8261522  0.6523045
##    0      0.2      0.8942812  0.7948219  0.8574827  0.8261522  0.6523045
##    1      0.1      0.8942328  0.7946171  0.8574941  0.8260555  0.6521111
##    1      0.2      0.8942328  0.7946171  0.8574941  0.8260555  0.6521111
##    2      0.1      0.8942004  0.7946171  0.8575168  0.8260669  0.6521338
##    2      0.2      0.8942004  0.7946171  0.8575168  0.8260669  0.6521338
##    3      0.1      0.8941697  0.7945374  0.8575168  0.8260271  0.6520542
##    3      0.2      0.8941697  0.7945374  0.8575168  0.8260271  0.6520542
##    4      0.1      0.8941507  0.7945488  0.8575168  0.8260327  0.6520655
##    4      0.2      0.8941507  0.7945488  0.8575168  0.8260327  0.6520655
##    5      0.1      0.8941435  0.7945033  0.8575737  0.8260384  0.6520769
##    5      0.2      0.8941435  0.7945033  0.8575737  0.8260384  0.6520769
##    6      0.1      0.8941022  0.7944464  0.8576648  0.8260555  0.6521111
##    6      0.2      0.8941022  0.7944464  0.8576648  0.8260555  0.6521111
##    7      0.1      0.8940880  0.7943895  0.8576761  0.8260327  0.6520655
##    7      0.2      0.8940880  0.7943895  0.8576761  0.8260327  0.6520655
##    8      0.1      0.8940678  0.7943781  0.8576648  0.8260214  0.6520428
##    8      0.2      0.8940678  0.7943781  0.8576648  0.8260214  0.6520428
##    9      0.1      0.8940454  0.7943326  0.8577444  0.8260384  0.6520769
##    9      0.2      0.8940454  0.7943326  0.8577444  0.8260384  0.6520769
##   10      0.1      0.8940365  0.7942757  0.8577672  0.8260214  0.6520428
##   10      0.2      0.8940365  0.7942757  0.8577672  0.8260214  0.6520428
##   11      0.1      0.8940162  0.7942415  0.8577899  0.8260157  0.6520314
##   11      0.2      0.8940162  0.7942415  0.8577899  0.8260157  0.6520314
##   12      0.1      0.8940028  0.7942188  0.8577899  0.8260043  0.6520086
##   12      0.2      0.8940028  0.7942188  0.8577899  0.8260043  0.6520086
##   13      0.1      0.8939760  0.7941960  0.8578127  0.8260043  0.6520086
##   13      0.2      0.8939760  0.7941960  0.8578127  0.8260043  0.6520086
##   14      0.1      0.8939642  0.7941164  0.8578810  0.8259986  0.6519973
##   14      0.2      0.8939642  0.7941164  0.8578810  0.8259986  0.6519973
##   15      0.1      0.8939414  0.7940936  0.8578924  0.8259929  0.6519859
##   15      0.2      0.8939414  0.7940936  0.8578924  0.8259929  0.6519859
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were epsilon = 0.1 and smooth = 0.
```

```
grafico_metricas(tanhc_train)
```






```
resultados(tanhc_train, "Hill Climbing Tree Augmented Naïve Bayes")
```

RESULTADOS DEL MODELO Hill Climbing Tree Augmented Naïve Bayes

smooth	epsilon	ROC	Sens	Spec	Accuracy	Kappa	ROCSD	SensS
0	0.1	0.8942812	0.7948219	0.8574827	0.8261522	0.6523045	0.0025142	0.00581
1	0.1	0.8942328	0.7946171	0.8574941	0.8260555	0.6521111	0.0025304	0.00583
2	0.1	0.8942004	0.7946171	0.8575168	0.8260669	0.6521338	0.0025421	0.00581
3	0.1	0.8941697	0.7945374	0.8575168	0.8260271	0.6520542	0.0025367	0.00588
4	0.1	0.8941507	0.7945488	0.8575168	0.8260327	0.6520655	0.0025366	0.00587
5	0.1	0.8941435	0.7945033	0.8575737	0.8260384	0.6520769	0.0025311	0.00588
6	0.1	0.8941022	0.7944464	0.8576648	0.8260555	0.6521111	0.0025497	0.00583
7	0.1	0.8940880	0.7943895	0.8576761	0.8260327	0.6520655	0.0025492	0.00580
8	0.1	0.8940678	0.7943781	0.8576648	0.8260214	0.6520428	0.0025436	0.00583
9	0.1	0.8940454	0.7943326	0.8577444	0.8260384	0.6520769	0.0025423	0.00582
10	0.1	0.8940365	0.7942757	0.8577672	0.8260214	0.6520428	0.0025481	0.00586
11	0.1	0.8940162	0.7942415	0.8577899	0.8260157	0.6520314	0.0025573	0.00585

```
mejor_modelo(tanhc_train)
```

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

epsilon	smooth
0.1	0

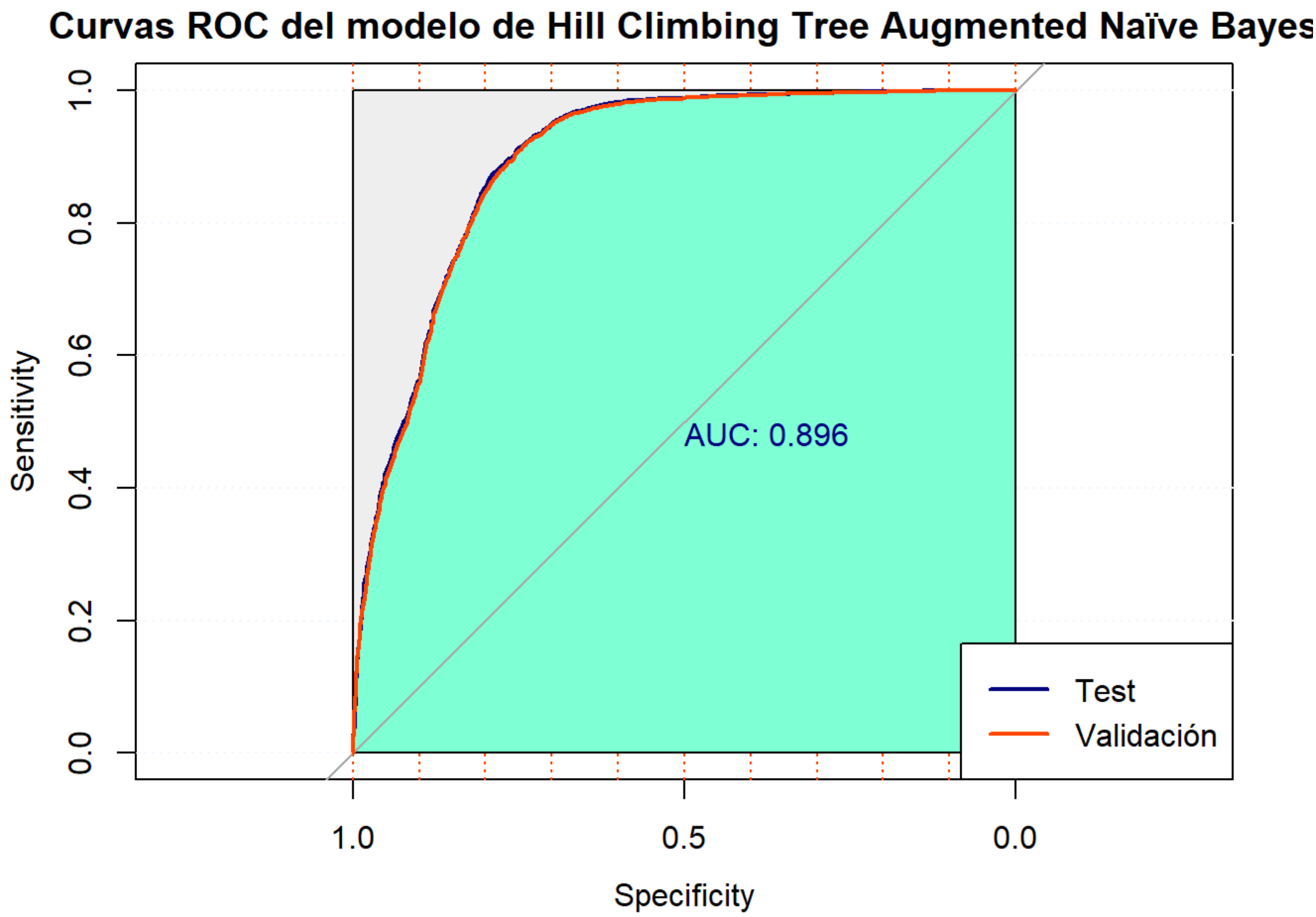
```
curvas_ROC(tanhc_train, "de Hill Climbing Tree Augmented Naïve Bayes", train_factor, test_
factor)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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




```
## [1] "ROC del modelo con el fichero de test: 0.896120243738908"
```

```
validation(tanhc_train, "de Hill Climbing Tree Augmented Naïve Bayes", train_factor, test_
factor)
```

```
## [1] "Modelo de Hill Climbing Tree Augmented Naïve Bayes - Tabla de confusión para los d
atos de entrenamiento"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  34877  6344
##        Yes  9058 37591
##
##                Accuracy : 0.8247
##                  95% CI : (0.8222, 0.8272)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6494
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7938
##             Specificity : 0.8556
##          Pos Pred Value : 0.8461
##          Neg Pred Value : 0.8058
##              Prevalence : 0.5000
##          Detection Rate : 0.3969
##    Detection Prevalence : 0.4691
##       Balanced Accuracy : 0.8247
##
##        'Positive' Class : No
##
## [1] "Modelo de Hill Climbing Tree Augmented Naïve Bayes - Tabla de confusión para los d
atos de validación"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  8742 1522
##        Yes 2241 9461
##
##                Accuracy : 0.8287
##                  95% CI : (0.8236, 0.8337)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6574
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7960
##             Specificity : 0.8614
##          Pos Pred Value : 0.8517
##          Neg Pred Value : 0.8085
##              Prevalence : 0.5000
##          Detection Rate : 0.3980
##    Detection Prevalence : 0.4673
##       Balanced Accuracy : 0.8287
##
##        'Positive' Class : No
##
```

```
resumen_tanhc <- resumen(tanhc_train, train_factor, test_factor)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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

```
resumen_tanhc %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ", "Hill Climbing Tree Augmented Naïve Bayes" = 7))
```

	Hill Climbing Tree Augmented Naïve Bayes						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.894	0.825	0.846	0.806	0.649	0.794	0.856

	Hill Climbing Tree Augmented Naïve Bayes						
	AUC	Accuracy	Aciertos Clase SI	Aciertos Clase NO	Kappa	Sensitivity	Specificity
Datos Validación	0.896	0.829	0.852	0.808	0.657	0.796	0.861

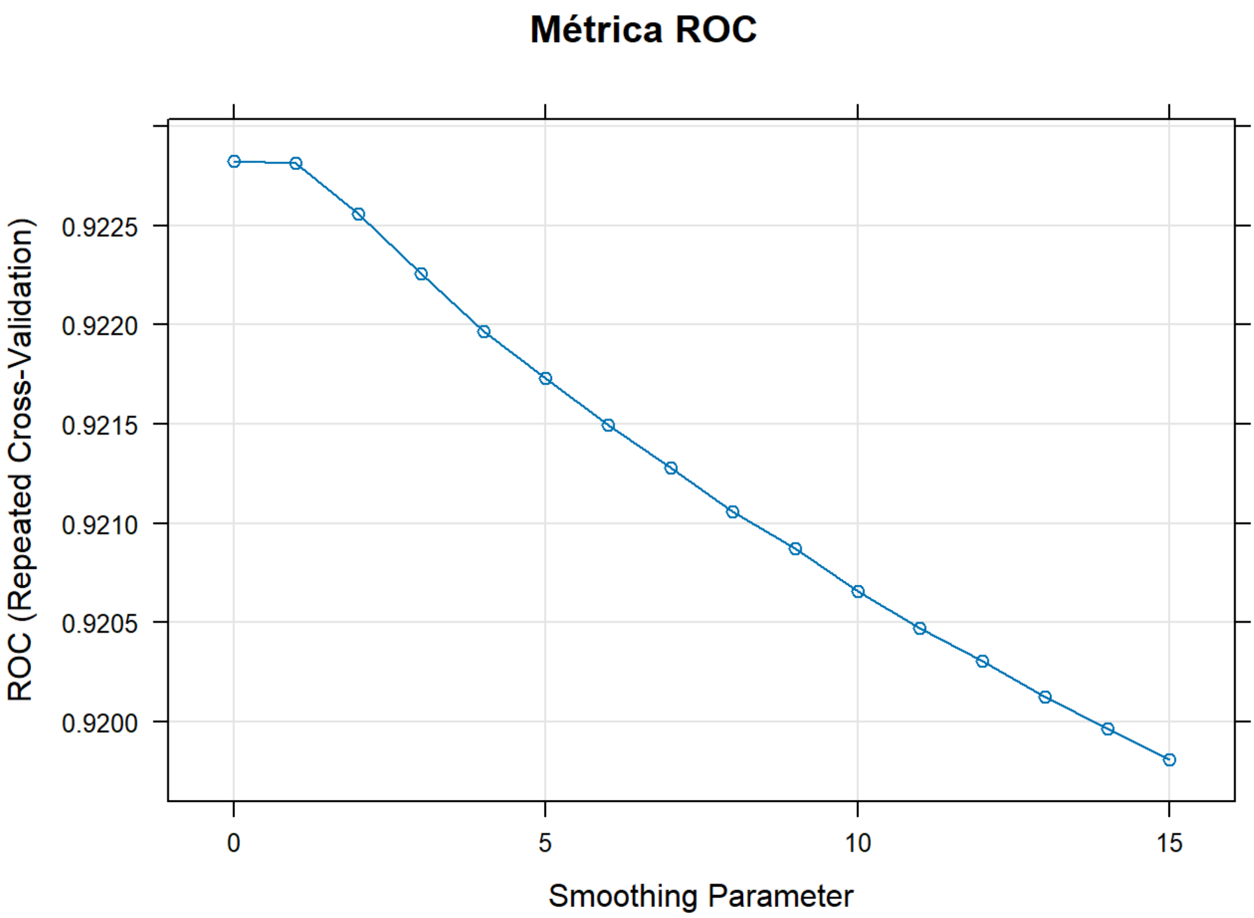
2.1.5. Modelo Aggregating One-Dependence Estimators (AODE)

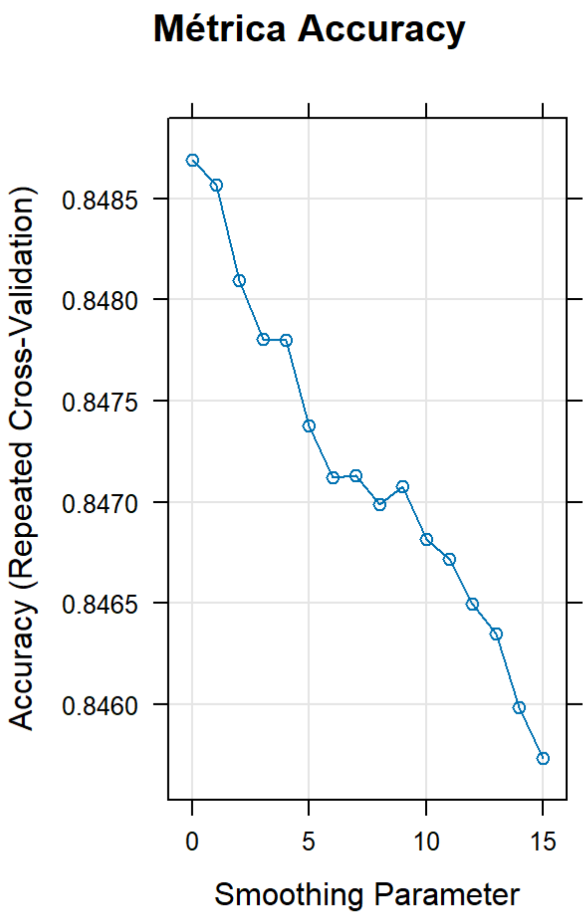
```
# Entrenamiento
if (train_switch  == 1) {
set.seed(7)

tic()

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

AODE_Grid <- expand.grid(smooth = seq(from = 0, to = 15, by = 1))

aode_train <- train(train_factor[,-length(train_factor)],
                  train_factor$y,
                  method = AODE,
                  metric = metrica,
                  trControl = control,
                  tuneGrid = AODE_Grid)

stopCluster(clusterCPU)

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

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

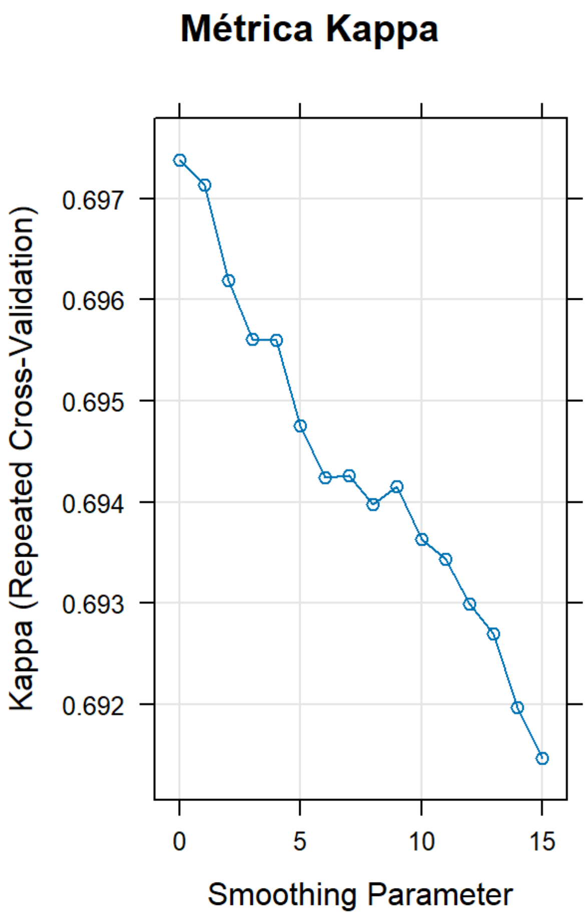
```
# Resultado
aode_train
```

```
## AODE Naive Bayes Classifier
##
## 87870 samples
##     7 predictor
##     2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 76887, 76886, 76886, 76887, 76886, 76886, ...
## Resampling results across tuning parameters:
##
##   smooth  ROC        Sens       Spec       Accuracy   Kappa
##    0      0.9228257  0.8189371  0.8784455  0.8486912  0.6973825
##    1      0.9228132  0.8172756  0.8798566  0.8485660  0.6971321
##    2      0.9225578  0.8161944  0.8799932  0.8480938  0.6961876
##    3      0.9222566  0.8153409  0.8802664  0.8478036  0.6956072
##    4      0.9219681  0.8146353  0.8809606  0.8477979  0.6955958
##    5      0.9217294  0.8138956  0.8808581  0.8473768  0.6947536
##    6      0.9214940  0.8133380  0.8809037  0.8471207  0.6942415
##    7      0.9212794  0.8132696  0.8809947  0.8471321  0.6942643
##    8      0.9210604  0.8130876  0.8808923  0.8469899  0.6939797
##    9      0.9208719  0.8131559  0.8809947  0.8470752  0.6941505
##   10      0.9206570  0.8125754  0.8810516  0.8468135  0.6936270
##   11      0.9204726  0.8121885  0.8812451  0.8467167  0.6934335
##   12      0.9203036  0.8116195  0.8813702  0.8464948  0.6929896
##   13      0.9201260  0.8111074  0.8815865  0.8463469  0.6926938
##   14      0.9199630  0.8101969  0.8817685  0.8459827  0.6919654
##   15      0.9198075  0.8098668  0.8815978  0.8457323  0.6914646
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was smooth = 0.
```

```
grafico_metricas(aode_train)
```








```
resultados(aode_train, "Aggregating One-Dependence Estimators")
```

RESULTADOS DEL MODELO Aggregating One-Dependence Estimators

smooth	ROC	Sens	Spec	Accuracy	Kappa	ROCSD	SensSD	Spe
0	0.9228257	0.8189371	0.8784455	0.8486912	0.6973825	0.0017796	0.0052997	0.004
1	0.9228132	0.8172756	0.8798566	0.8485660	0.6971321	0.0017714	0.0051400	0.004
2	0.9225578	0.8161944	0.8799932	0.8480938	0.6961876	0.0017874	0.0052365	0.004
3	0.9222566	0.8153409	0.8802664	0.8478036	0.6956072	0.0018016	0.0051597	0.004
4	0.9219681	0.8146353	0.8809606	0.8477979	0.6955958	0.0017909	0.0053831	0.004
5	0.9217294	0.8138956	0.8808581	0.8473768	0.6947536	0.0017989	0.0051418	0.004
6	0.9214940	0.8133380	0.8809037	0.8471207	0.6942415	0.0018069	0.0050992	0.005
7	0.9212794	0.8132696	0.8809947	0.8471321	0.6942643	0.0018034	0.0048728	0.004
8	0.9210604	0.8130876	0.8808923	0.8469899	0.6939797	0.0017841	0.0049266	0.004
9	0.9208719	0.8131559	0.8809947	0.8470752	0.6941505	0.0017676	0.0046591	0.004
10	0.9206570	0.8125754	0.8810516	0.8468135	0.6936270	0.0017732	0.0049221	0.004
11	0.9204726	0.8121885	0.8812451	0.8467167	0.6934335	0.0017786	0.0049555	0.004

```
mejor_modelo(aode_train)
```

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

smooth
0

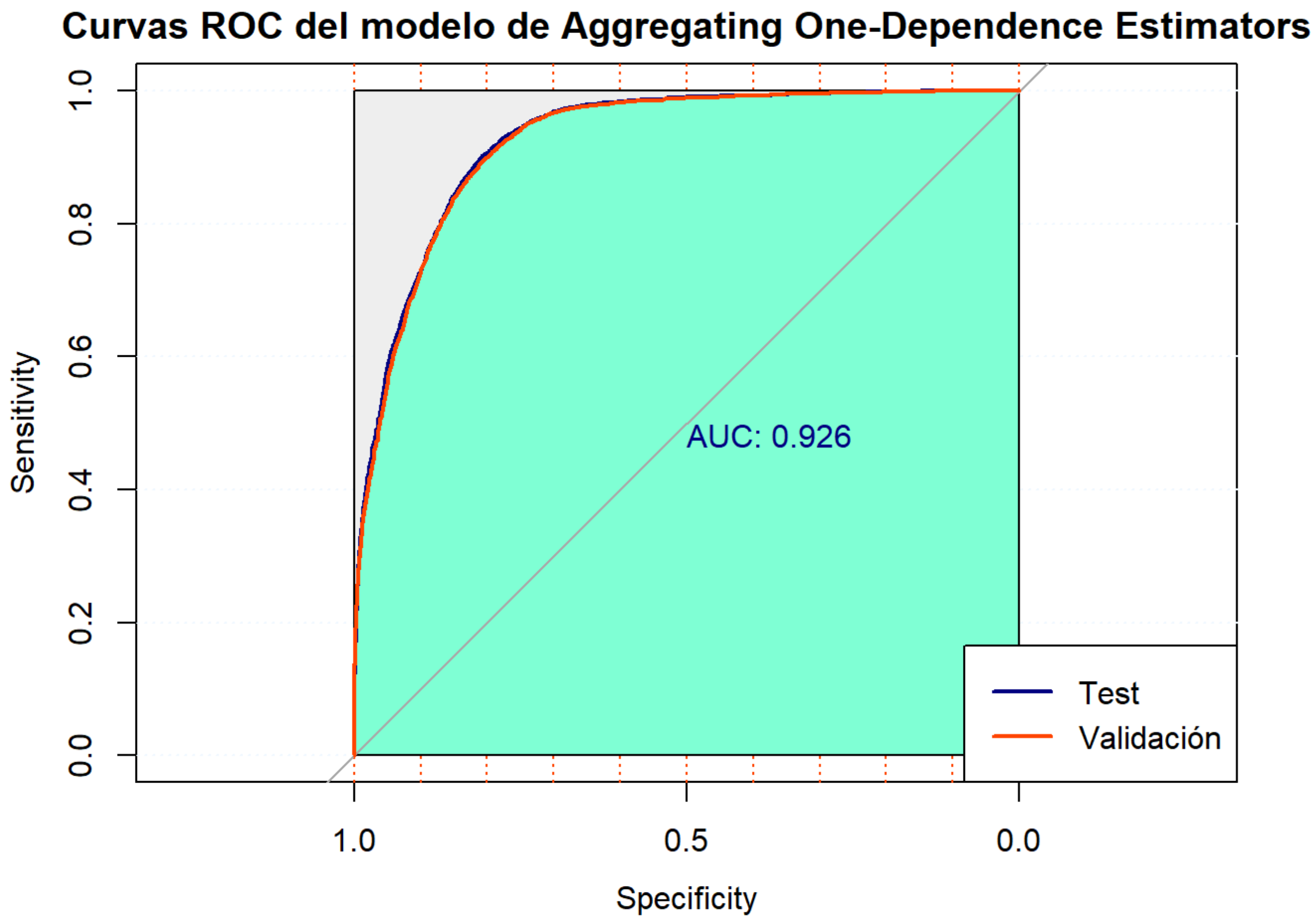
```
curvas_ROC(aode_train, "de Aggregating One-Dependence Estimators", train_factor, test_fact
or)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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




```
## [1] "ROC del modelo con el fichero de test: 0.926237177867587"
```

```
validation(aode_train, "de Aggregating One-Dependence Estimators", train_factor, test_fact
or)
```

```
## [1] "Modelo de Aggregating One-Dependence Estimators - Tabla de confusión para los dato
s de entrenamiento"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  36004  5338
##        Yes  7931 38597
##
##                Accuracy : 0.849
##                  95% CI : (0.8466, 0.8514)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.698
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8195
##             Specificity : 0.8785
##          Pos Pred Value : 0.8709
##          Neg Pred Value : 0.8295
##              Prevalence : 0.5000
##          Detection Rate : 0.4097
##    Detection Prevalence : 0.4705
##       Balanced Accuracy : 0.8490
##
##        'Positive' Class : No
##
## [1] "Modelo de Aggregating One-Dependence Estimators - Tabla de confusión para los dato
s de validación"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  9003 1248
##        Yes 1980 9735
##
##                Accuracy : 0.853
##                  95% CI : (0.8483, 0.8577)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7061
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8197
##             Specificity : 0.8864
##          Pos Pred Value : 0.8783
##          Neg Pred Value : 0.8310
##              Prevalence : 0.5000
##          Detection Rate : 0.4099
##    Detection Prevalence : 0.4667
##       Balanced Accuracy : 0.8530
##
##        'Positive' Class : No
##
```

```
resumen_AODE <- resumen(aode_train, train_factor, test_factor)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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

```
resumen_AODE %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ",
            "Aggregating One-Dependence Estimators " = 7))
```

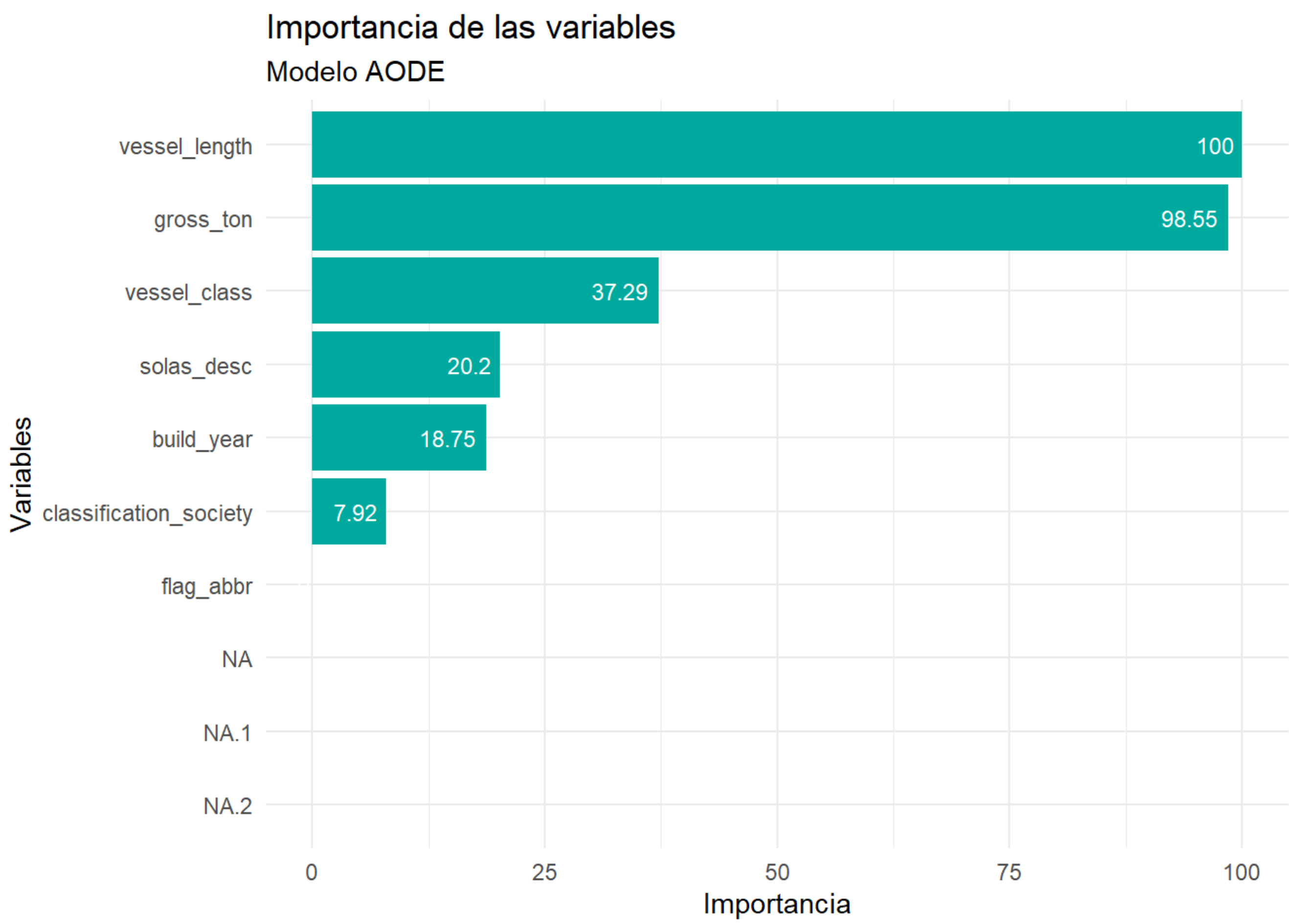
	Aggregating One-Dependence Estimators						
	AUC	Accuracy	Aciertos Clase SI	Aciertos Clase NO	Kappa	Sensitivity	Specificity

Aggregating One-Dependence Estimators							
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.924	0.849	0.871	0.830	0.698	0.819	0.879
Datos Validación	0.926	0.853	0.878	0.831	0.706	0.820	0.886

```
importancia_var(aode_train, "AODE")
```

```
## Warning: Removed 3 rows containing missing values (`position_stack()`).
## Removed 3 rows containing missing values (`position_stack()`).
```

```
## Warning: Removed 3 rows containing missing values (`geom_text()`).
```



2.2. Modelos Gradient Boosting

2.2.1. Modelo GBM

```
# 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_train <- train(train_num[ , -length(train_num)],
                 train_num$y,
                 method = "gbm",
                 metric = metrica,
                 trControl = control,
                 tuneGrid = tune_grid)

stopCluster(clusterCPU)

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

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

```
GBM_train
```

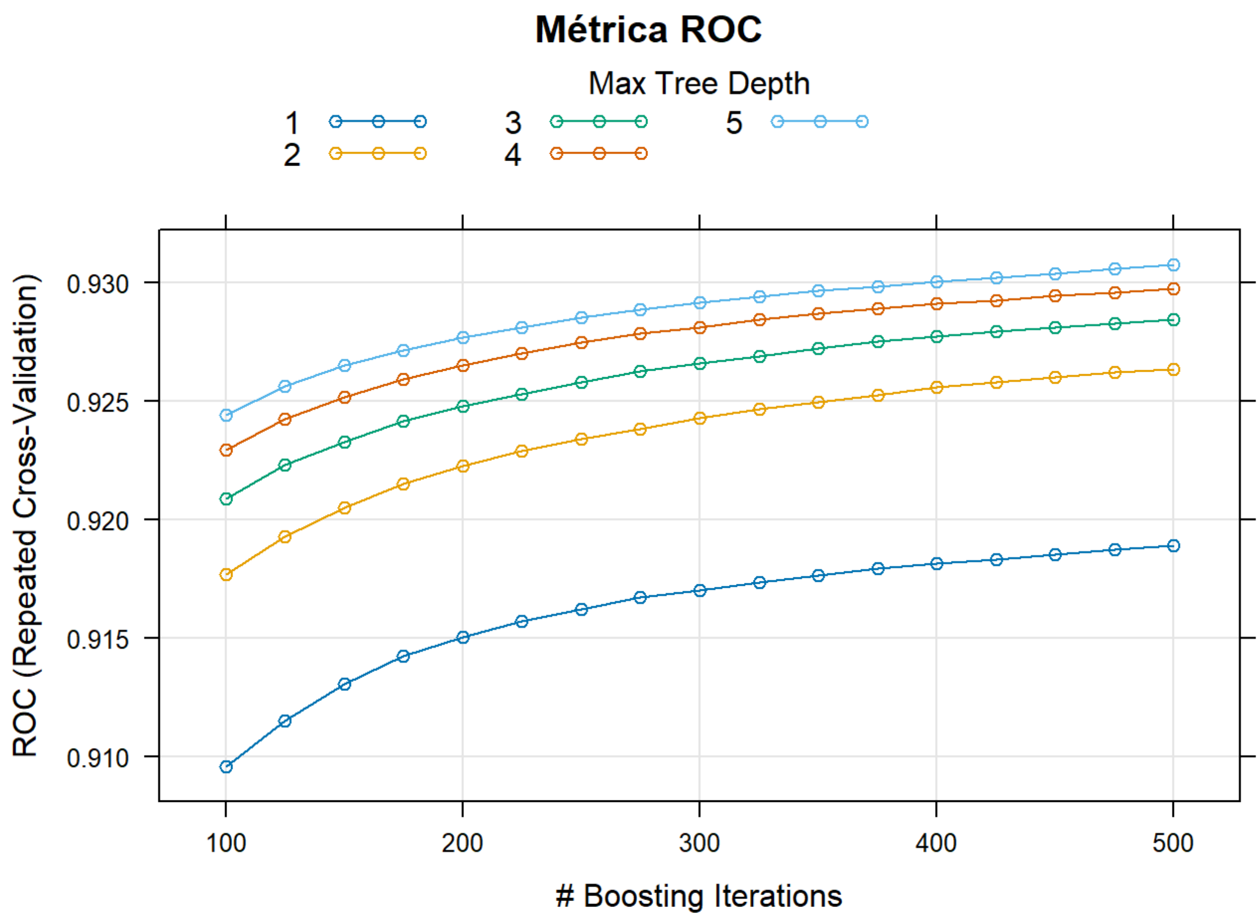
```
## Stochastic Gradient Boosting
##
## 87870 samples
##    32 predictor
##     2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 76887, 76886, 76886, 76887, 76886, 76886, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  ROC        Sens       Spec       Accuracy
##   1                  100      0.9095986  0.7506772  0.9197792  0.8352282
##   1                  125      0.9115356  0.7506886  0.9218732  0.8362808
##   1                  150      0.9130866  0.7504610  0.9230340  0.8367475
##   1                  175      0.9142570  0.7523160  0.9234095  0.8378627
##   1                  200      0.9150295  0.7527371  0.9242517  0.8384943
##   1                  225      0.9157087  0.7541938  0.9240810  0.8391373
##   1                  250      0.9162275  0.7545124  0.9242859  0.8393991
##   1                  275      0.9167168  0.7563218  0.9233754  0.8398486
##   1                  300      0.9170386  0.7579265  0.9227267  0.8403266
##   1                  325      0.9173387  0.7590987  0.9225333  0.8408159
##   1                  350      0.9176667  0.7597815  0.9220325  0.8409070
##   1                  375      0.9179254  0.7608513  0.9212700  0.8410606
##   1                  400      0.9181327  0.7616820  0.9203710  0.8410265
##   1                  425      0.9183413  0.7627291  0.9200068  0.8413679
##   1                  450      0.9185497  0.7632981  0.9196199  0.8414589
##   1                  475      0.9187296  0.7643792  0.9189370  0.8416581
##   1                  500      0.9188951  0.7648572  0.9185160  0.8416865
##   2                  100      0.9177048  0.7702516  0.9182428  0.8442471
##   2                  125      0.9192968  0.7750086  0.9173665  0.8461875
##   2                  150      0.9205046  0.7774895  0.9175714  0.8475304
##   2                  175      0.9215252  0.7814044  0.9161147  0.8487595
##   2                  200      0.9222467  0.7855697  0.9147376  0.8501536
##   2                  225      0.9228789  0.7882441  0.9134061  0.8508251
##   2                  250      0.9233840  0.7899853  0.9129851  0.8514851
##   2                  275      0.9238359  0.7911916  0.9121315  0.8516615
##   2                  300      0.9242646  0.7922499  0.9123364  0.8522931
##   2                  325      0.9246566  0.7934108  0.9114715  0.8524411
##   2                  350      0.9249599  0.7943212  0.9115739  0.8529475
##   2                  375      0.9252608  0.7951406  0.9109821  0.8530613
##   2                  400      0.9255637  0.7956982  0.9107431  0.8532206
##   2                  425      0.9257993  0.7961876  0.9109480  0.8535677
##   2                  450      0.9259922  0.7966542  0.9108455  0.8537498
##   2                  475      0.9261959  0.7968932  0.9109935  0.8539433
##   2                  500      0.9263452  0.7972915  0.9107431  0.8540173
##   3                  100      0.9208629  0.7786504  0.9169796  0.8478150
##   3                  125      0.9223261  0.7853080  0.9154091  0.8503585
##   3                  150      0.9232875  0.7898488  0.9131330  0.8514908
##   3                  175      0.9241376  0.7917947  0.9131103  0.8524525
##   3                  200      0.9247987  0.7931263  0.9127802  0.8529532
##   3                  225      0.9253027  0.7944009  0.9124389  0.8534198
##   3                  250      0.9257932  0.7956641  0.9120292  0.8538466
##   3                  275      0.9262497  0.7965404  0.9120747  0.8543075
```

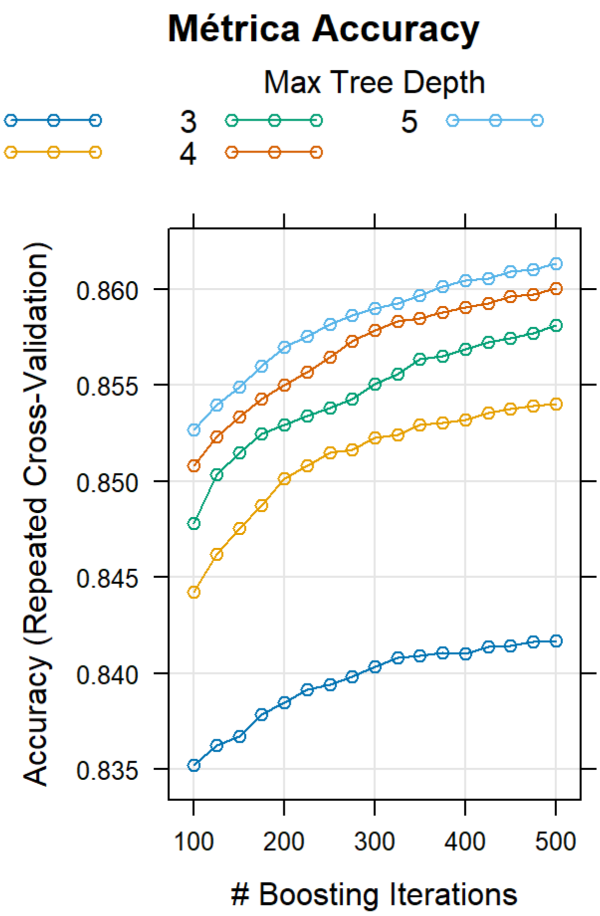
```
##   3                  300      0.9265975  0.7980426  0.9121202  0.8550813
##   3                  325      0.9268985  0.7992261  0.9119608  0.8555935
##   3                  350      0.9272088  0.8006715  0.9120405  0.8563559
##   3                  375      0.9274913  0.8009446  0.9120747  0.8565096
##   3                  400      0.9277275  0.8019802  0.9117674  0.8568737
##   3                  425      0.9279419  0.8022647  0.9122226  0.8572436
##   3                  450      0.9280959  0.8025378  0.9123705  0.8574542
##   3                  475      0.9282594  0.8034597  0.9119722  0.8577159
##   3                  500      0.9284353  0.8043132  0.9118925  0.8581028
##   4                  100      0.9229327  0.7885513  0.9130420  0.8507966
##   4                  125      0.9242223  0.7918972  0.9127688  0.8523330
##   4                  150      0.9251480  0.7934108  0.9132810  0.8533458
##   4                  175      0.9259043  0.7951748  0.9133606  0.8542676
##   4                  200      0.9264865  0.7968932  0.9131444  0.8550187
##   4                  225      0.9270064  0.7986799  0.9127233  0.8557016
##   4                  250      0.9274588  0.8002846  0.9125982  0.8564413
##   4                  275      0.9278277  0.8025607  0.9120519  0.8573062
##   4                  300      0.9281140  0.8033459  0.9123591  0.8578525
##   4                  325      0.9284306  0.8046319  0.9120063  0.8583191
##   4                  350      0.9286895  0.8051099  0.9119039  0.8585068
##   4                  375      0.9288837  0.8057813  0.9118470  0.8588141
##   4                  400      0.9290893  0.8062821  0.9118584  0.8590702
##   4                  425      0.9292487  0.8067145  0.9117673  0.8592409
##   4                  450      0.9294514  0.8072493  0.9120177  0.8596335
##   4                  475      0.9295791  0.8079663  0.9114828  0.8597245
##   4                  500      0.9297239  0.8086150  0.9115170  0.8600660
##   5                  100      0.9244184  0.7923865  0.9130192  0.8527028
##   5                  125      0.9256066  0.7947764  0.9132013  0.8539888
##   5                  150      0.9264802  0.7968135  0.9129851  0.8548992
##   5                  175      0.9271254  0.7989644  0.9130420  0.8560031
##   5                  200      0.9276890  0.8008991  0.9131103  0.8570046
##   5                  225      0.9281010  0.8026061  0.9125185  0.8575623
##   5                  250      0.9284991  0.8036532  0.9127461  0.8581996
##   5                  275      0.9288555  0.8050529  0.9122226  0.8586377
##   5                  300      0.9291584  0.8060658  0.9119722  0.8590190
##   5                  325      0.9294035  0.8068397  0.9116877  0.8592637
##   5                  350      0.9296457  0.8076022  0.9117787  0.8596904
##   5                  375      0.9298159  0.8083761  0.9118698  0.8601229
##   5                  400      0.9300413  0.8089337  0.9119608  0.8604472
##   5                  425      0.9301994  0.8093320  0.9117787  0.8605553
##   5                  450      0.9303757  0.8099124  0.9119608  0.8609366
##   5                  475      0.9305577  0.8100376  0.9120405  0.8610390
##   5                  500      0.9307190  0.8104018  0.9122226  0.8613121
##   Kappa
##   0.6704564
##   0.6725617
##   0.6734949
##   0.6757255
##   0.6769887
##   0.6782747
##   0.6787982
##   0.6796972
##   0.6806532
##   0.6816319
##   0.6818140
```

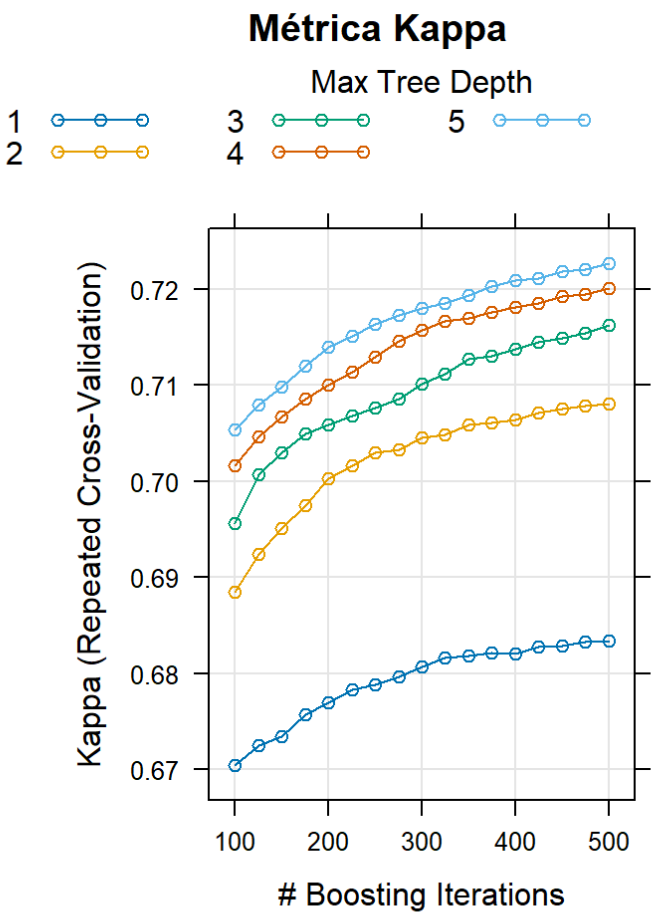
```
##    0.6821213
##    0.6820530
##    0.6827358
##    0.6829179
##    0.6833162
##    0.6833731
##    0.6884943
##    0.6923751
##    0.6950609
##    0.6975190
##    0.7003073
##    0.7016501
##    0.7029703
##    0.7033230
##    0.7045863
##    0.7048822
##    0.7058950
##    0.7061226
##    0.7064413
##    0.7071355
##    0.7074997
##    0.7078866
##    0.7080346
##    0.6956299
##    0.7007170
##    0.7029817
##    0.7049049
##    0.7059064
##    0.7068396
##    0.7076931
##    0.7086150
##    0.7101627
##    0.7111869
##    0.7127119
##    0.7130192
##    0.7137475
##    0.7144872
##    0.7149083
##    0.7154318
##    0.7162057
##    0.7015932
##    0.7046659
##    0.7066917
##    0.7085353
##    0.7100375
##    0.7114032
##    0.7128827
##    0.7146125
##    0.7157050
##    0.7166382
##    0.7170137
##    0.7176283
##    0.7181404
##    0.7184818
##    0.7192670
```

```
##   0.7194491
##   0.7201319
##   0.7054057
##   0.7079777
##   0.7097985
##   0.7120063
##   0.7140093
##   0.7151246
##   0.7163992
##   0.7172755
##   0.7180380
##   0.7185273
##   0.7193809
##   0.7202458
##   0.7208944
##   0.7211107
##   0.7218732
##   0.7220780
##   0.7226243
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 500, interaction.depth =
##  5, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
grafico_metricas(GBM_train)
```








```
resultados(GBM_train, "Stochastic Gradient Boosting")
```

RESULTADOS DEL MODELO Stochastic Gradient Boosting

shrinkage	interaction.depth	n.minobsinnode	n.trees	ROC	Sens	Spec	Accu
0.1	1	10	100	0.9095986	0.7506772	0.9197792	0.835
0.1	2	10	100	0.9177048	0.7702516	0.9182428	0.844
0.1	3	10	100	0.9208629	0.7786504	0.9169796	0.847
0.1	4	10	100	0.9229327	0.7885513	0.9130420	0.850
0.1	5	10	100	0.9244184	0.7923865	0.9130192	0.852
0.1	1	10	125	0.9115356	0.7506886	0.9218732	0.836
0.1	2	10	125	0.9192968	0.7750086	0.9173665	0.846
0.1	3	10	125	0.9223261	0.7853080	0.9154091	0.850
0.1	4	10	125	0.9242223	0.7918972	0.9127688	0.852
0.1	5	10	125	0.9256066	0.7947764	0.9132013	0.853
0.1	1	10	150	0.9130866	0.7504610	0.9230340	0.836
0.1	2	10	150	0.9205046	0.7774895	0.9175714	0.847

```
mejor_modelo(GBM_train)
```

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

	n.trees	interaction.depth	shrinkage	n.minobsinnode
85	500	5	0.1	10

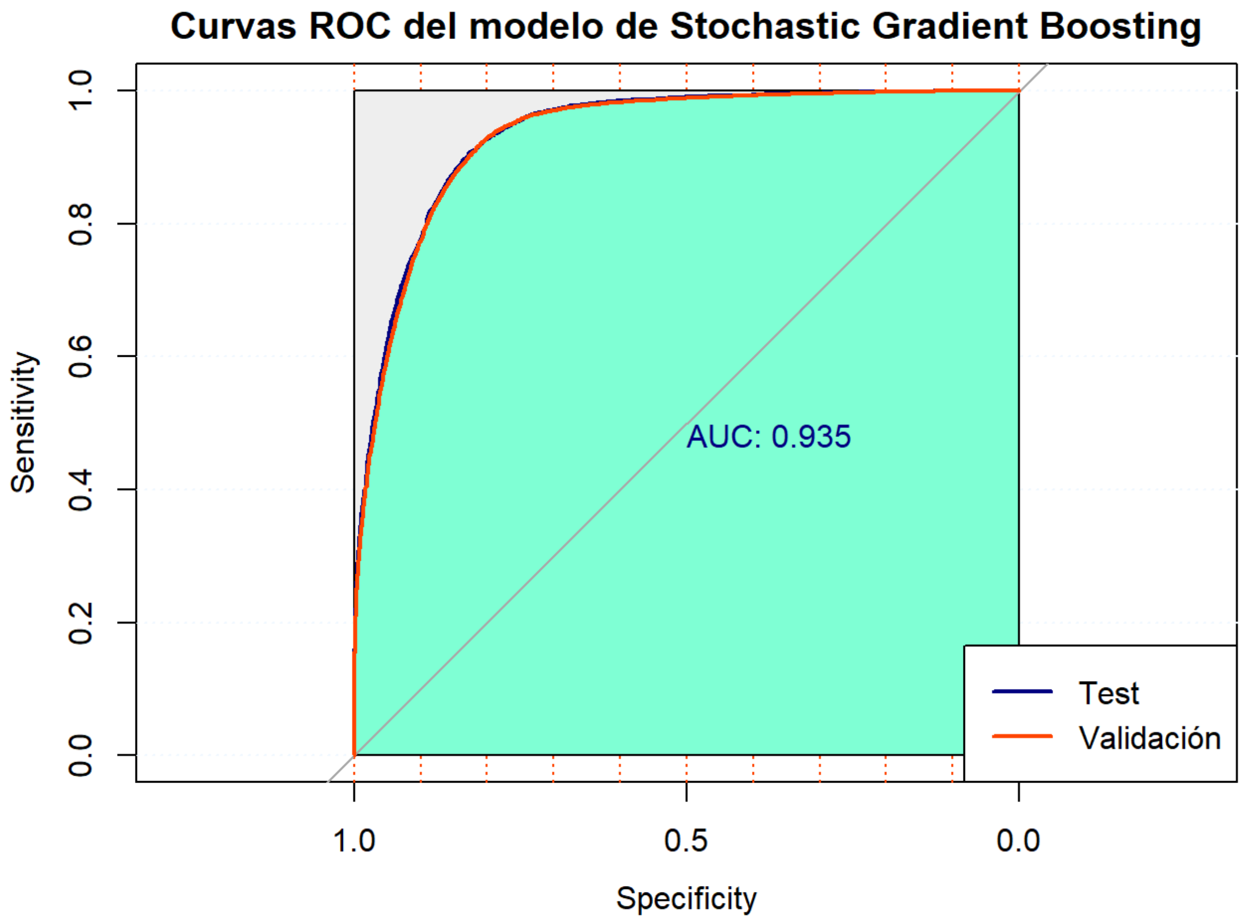
```
curvas_ROC(GBM_train, "de Stochastic Gradient Boosting", train_num, test_num)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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




```
## [1] "ROC del modelo con el fichero de test: 0.935285217967702"
```

```
validation(GBM_train, "Stochastic Gradient Boosting", train_num, test_num)
```

```
## [1] "Modelo Stochastic Gradient Boosting - Tabla de confusión para los datos de entrena
miento"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  35843  3803
##        Yes  8092 40132
##
##                Accuracy : 0.8646
##                  95% CI : (0.8623, 0.8669)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7293
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8158
##             Specificity : 0.9134
##          Pos Pred Value : 0.9041
##          Neg Pred Value : 0.8322
##              Prevalence : 0.5000
##          Detection Rate : 0.4079
##    Detection Prevalence : 0.4512
##       Balanced Accuracy : 0.8646
##
##        'Positive' Class : No
##
## [1] "Modelo Stochastic Gradient Boosting - Tabla de confusión para los datos de validac
ión"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No   8924   913
##        Yes  2059 10070
##
##                Accuracy : 0.8647
##                  95% CI : (0.8601, 0.8692)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7294
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8125
##             Specificity : 0.9169
##          Pos Pred Value : 0.9072
##          Neg Pred Value : 0.8302
##              Prevalence : 0.5000
##          Detection Rate : 0.4063
##    Detection Prevalence : 0.4478
##       Balanced Accuracy : 0.8647
##
##        'Positive' Class : No
##
```

```
resumen_GBM <- resumen(GBM_train, train_num, test_num)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

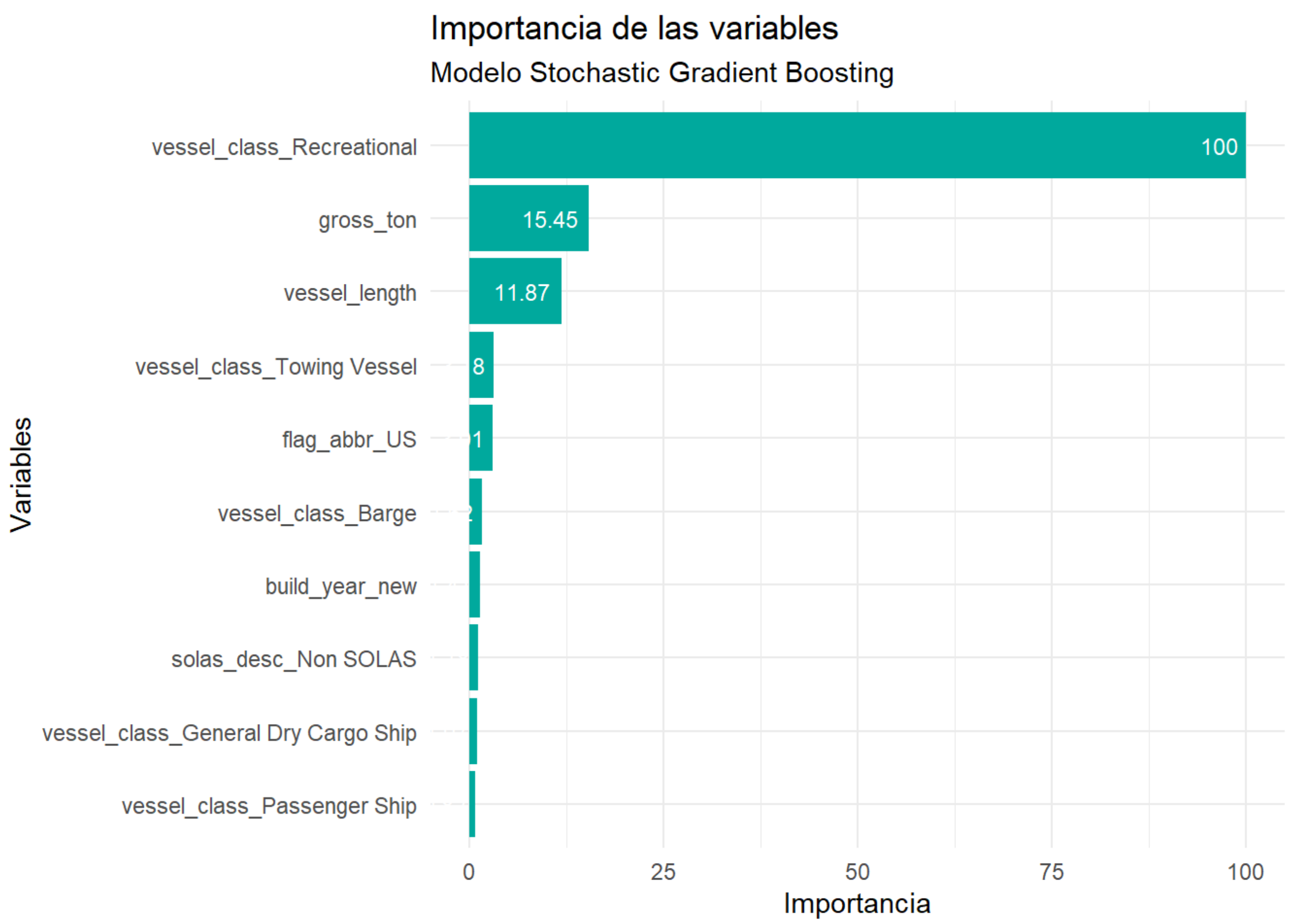
```
## Setting direction: controls < cases
```

```
resumen_GBM %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ",
             "Stochastic Gradient Boosting " = 7))
```

	Stochastic Gradient Boosting						
	AUC	Accuracy	Aciertos Clase SI	Aciertos Clase NO	Kappa	Sensitivity	Specificity

	Stochastic Gradient Boosting						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.933	0.865	0.904	0.832	0.729	0.816	0.913
Datos Validación	0.935	0.865	0.907	0.830	0.729	0.813	0.917

```
importancia_var(GBM_train, "Stochastic Gradient Boosting")
```



2.2.2. Modelo Extreme Gradient Boosting (XGBTree)

```
# Entrenamiento
if (train_switch  == 10) {
set.seed(7)
tic()

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

# Primera ronda

tune_grid <- expand.grid(nrounds = c(200, 250, 300, 350, 400, 500),
                         eta = c(0.01, 0.025, 0.05, 0.1, 0.2, 0.3, 0.5),
                         max_depth = c(2, 3, 4, 5),
                         gamma = 0,
                         colsample_bytree = 0.3,
                         min_child_weight = 1,
                         subsample = 1 )

XGB1_train <- train(train_num[ , -length(train_num)],
                   train_num$y,
                   method = "xgbTree",
                   metric = metrica,
                   trControl = control,
                   tuneGrid = tune_grid)

# Segunda ronda

tune_grid2 <- expand.grid(nrounds = XGB1_train$bestTune$nrounds,
                          eta = XGB1_train$bestTune$eta,
                          max_depth = c(3,4,5,6,7),
                          gamma = 0,
                          colsample_bytree = 0.3,
                          min_child_weight = c(1, 2, 3, 4, 5),
                          subsample = 1)


XGB2_train <- train(train_num[ , -length(train_num)],
                   train_num$y,
                   method = "xgbTree",
                   metric = metrica,
                   trControl = control,
                   tuneGrid = tune_grid2)

# Tercera ronda

tune_grid3 <- expand.grid(nrounds = XGB1_train$bestTune$nrounds,
                          eta = XGB1_train$bestTune$eta,
                          max_depth = XGB2_train$bestTune$max_depth,
                          gamma = 0,
                          colsample_bytree = seq(0.1, 0.9, 0.1),
                          min_child_weight = XGB2_train$bestTune$min_child_weight,
                          subsample = c(0.25, 0.5, 0.75, 1.0 ))
```

```r
XGB3_train <- train(train_num[ , -length(train_num)],
                   train_num$y,
                   method = "xgbTree",
                   metric = metrica,
                   trControl = control,
                   tuneGrid = tune_grid3)

# Cuarta ronda

tune_grid4 <- expand.grid(nrounds = XGB1_train$bestTune$nrounds,
                          eta = XGB1_train$bestTune$eta,
                          max_depth = XGB2_train$bestTune$max_depth,
                          gamma = seq(0, 1, 0.05),
                          colsample_bytree = XGB3_train$bestTune$colsample_bytree,
                          min_child_weight = XGB2_train$bestTune$min_child_weight,
                          subsample = XGB3_train$bestTune$subsample)

XGB4_train <- train(train_num[ , -length(train_num)],
                   train_num$y,
                   method = "xgbTree",
                   metric = metrica,
                   trControl = control,
                   tuneGrid = tune_grid4)

# Quinta ronda

tune_grid5 <- expand.grid(nrounds = seq(100, 600, 25),
                          eta =  c(0.01, 0.05, 0.005),
                          max_depth = XGB2_train$bestTune$max_depth,
                          gamma = XGB4_train$bestTune$gamma,
                          colsample_bytree = XGB3_train$bestTune$colsample_bytree,
                          min_child_weight = XGB2_train$bestTune$min_child_weight,
                          subsample = XGB3_train$bestTune$subsample)

XGB5_train <- train(train_num[ , -length(train_num)],
                   train_num$y,
                   method = "xgbTree",
                   metric = metrica,
                   trControl = control,
                   tuneGrid = tune_grid5)

# Ronda final

tune_gridFinal <- expand.grid(nrounds = XGB5_train$bestTune$nrounds,
                          eta = XGB5_train$bestTune$eta,
                          max_depth = XGB5_train$bestTune$max_depth,
                          gamma = XGB5_train$bestTune$gamma,
                          colsample_bytree = XGB5_train$bestTune$colsample_bytree,
                          min_child_weight = XGB5_train$bestTune$min_child_weight,
                          subsample = XGB5_train$bestTune$subsample)

XGBFinal_train <- train(train_num[ , -length(train_num)],
                   train_num$y,
                   method = "xgbTree",
                   metric = metrica,
```

```
                    trControl = control,
                    tuneGrid = tune_gridFinal)

stopCluster(clusterCPU)

toc()

saveRDS(XGB1_train, "Models/XGB1_train.RDS")
saveRDS(XGB2_train, "Models/XGB2_train.RDS")
saveRDS(XGB3_train, "Models/XGB3_train.RDS")
saveRDS(XGB4_train, "Models/XGB4_train.RDS")
saveRDS(XGB5_train, "Models/XGB5_train.RDS")
saveRDS(XGBFinal_train, "Models/XGBFinal_train.RDS")

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

```
resultados(XGBFinal_train , "Extreme Gradient Boosting")
```

RESULTADOS DEL MODELO Extreme Gradient Boosting

nrounds	eta	max_depth	gamma	colsample_bytree	min_child_weight	subsample	R(
600	0.05	7	0	0.9	1	1	0.93634

```
mejor_modelo(XGBFinal_train)
```

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

nrounds	max_depth	eta	gamma	colsample_bytree	min_child_weight	subsample
600	7	0.05	0	0.9	1	1

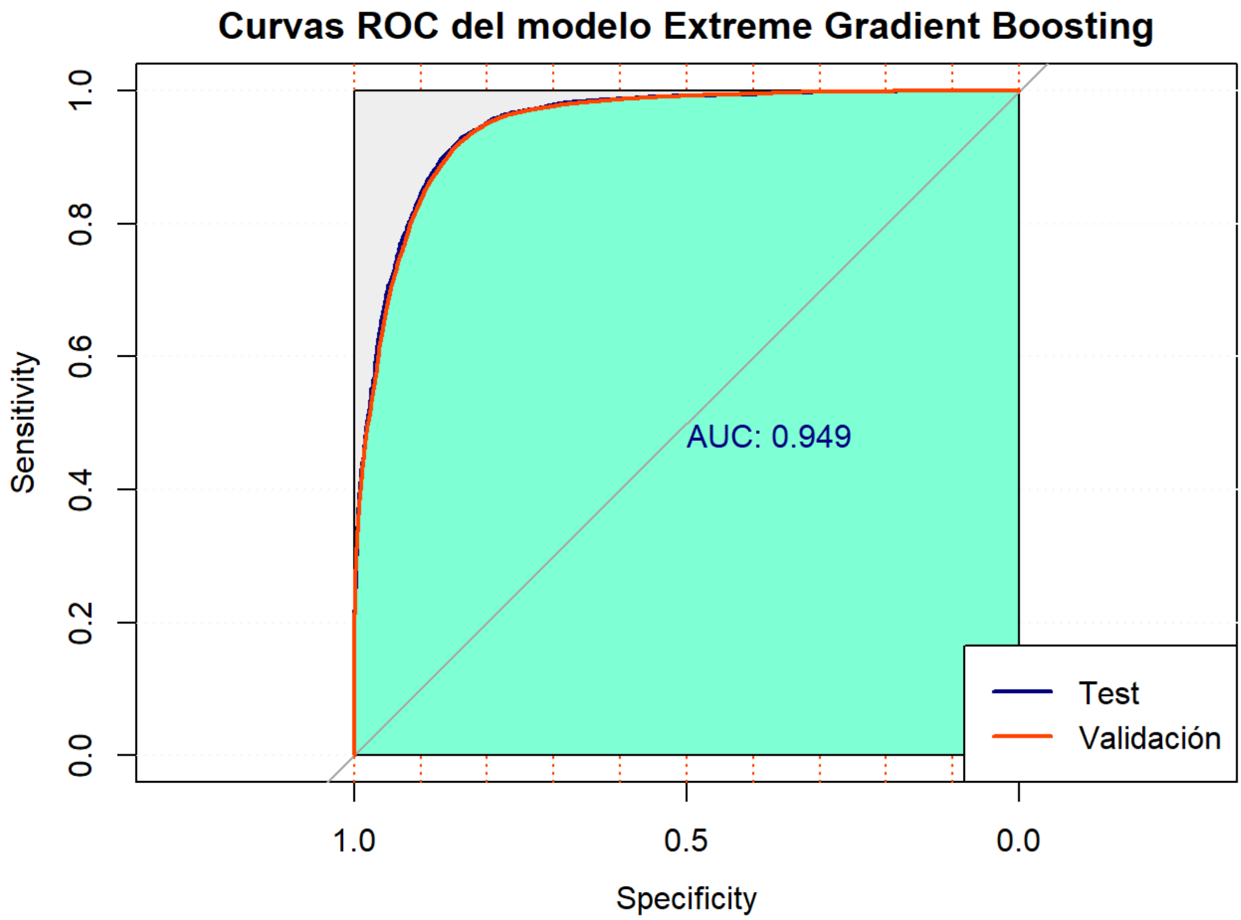
```
curvas_ROC(XGBFinal_train , "Extreme Gradient Boosting", train_num, test_num)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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




```
## [1] "ROC del modelo con el fichero de test: 0.948741745673698"
```

```
validation(XGBFinal_train , "Extreme Gradient Boosting", train_num, test_num)
```

```
## [1] "Modelo Extreme Gradient Boosting - Tabla de confusión para los datos de entrenamie
nto"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  36685  3201
##        Yes  7250 40734
##
##                Accuracy : 0.8811
##                  95% CI : (0.8789, 0.8832)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7621
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8350
##             Specificity : 0.9271
##          Pos Pred Value : 0.9197
##          Neg Pred Value : 0.8489
##              Prevalence : 0.5000
##          Detection Rate : 0.4175
##    Detection Prevalence : 0.4539
##       Balanced Accuracy : 0.8811
##
##        'Positive' Class : No
##
## [1] "Modelo Extreme Gradient Boosting - Tabla de confusión para los datos de validació
n"
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    No   Yes
##        No   9174   763
##        Yes  1809 10220
##                                          
##                Accuracy : 0.8829         
##                  95% CI : (0.8786, 0.8871)
##     No Information Rate : 0.5            
##     P-Value [Acc > NIR] : < 2.2e-16      
##                                          
##                   Kappa : 0.7658         
##                                          
##  Mcnemar's Test P-Value : < 2.2e-16      
##                                          
##             Sensitivity : 0.8353         
##             Specificity : 0.9305         
##          Pos Pred Value : 0.9232         
##          Neg Pred Value : 0.8496         
##              Prevalence : 0.5000         
##          Detection Rate : 0.4176         
##    Detection Prevalence : 0.4524         
##       Balanced Accuracy : 0.8829         
##                                          
##        'Positive' Class : No             
## 
```

```
resumen_XGBFinal <- resumen(XGBFinal_train, train_num, test_num)
```

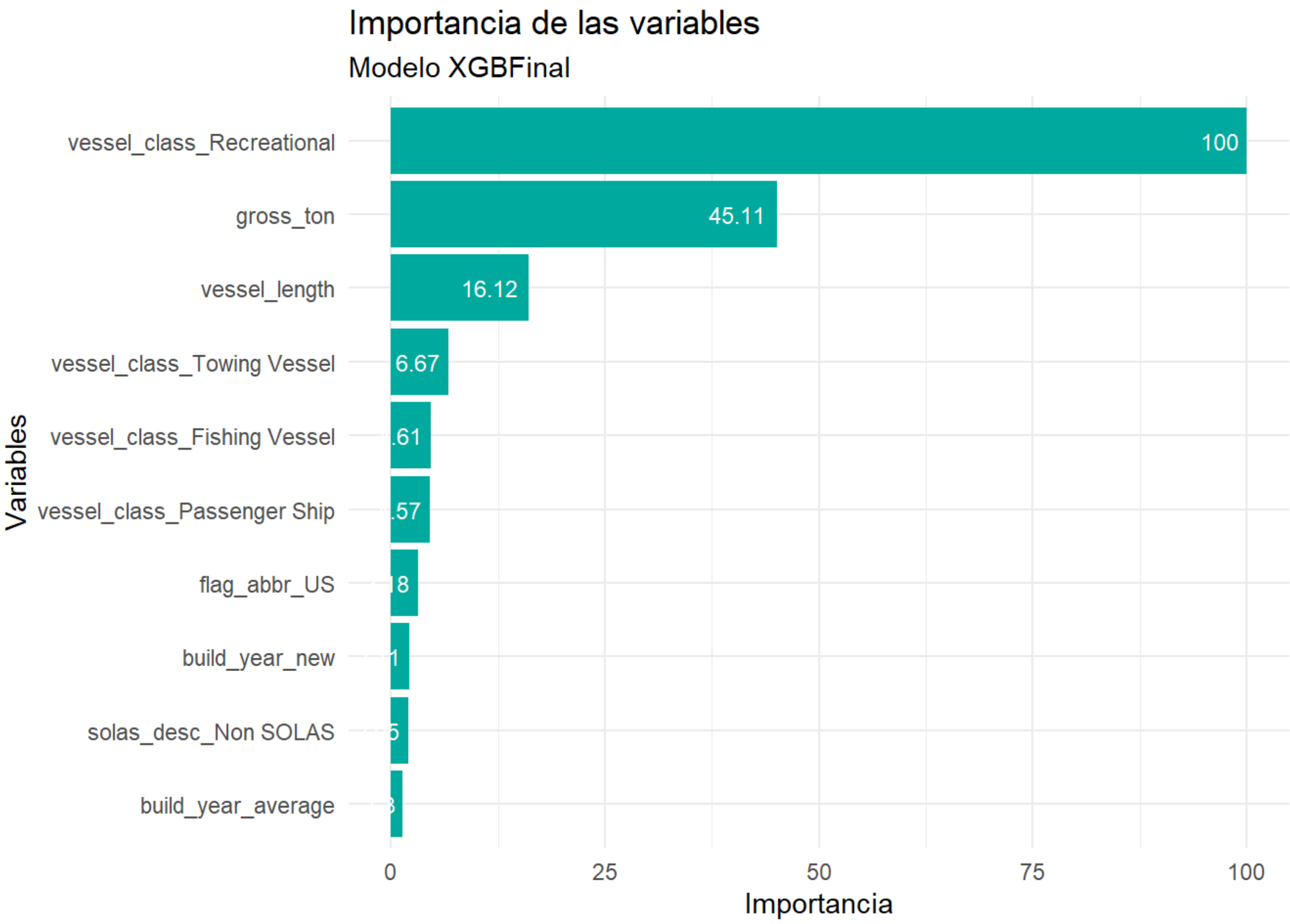
```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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

```
importancia_var(XGBFinal_train, "XGBFinal")
```



2.3. Otros modelos

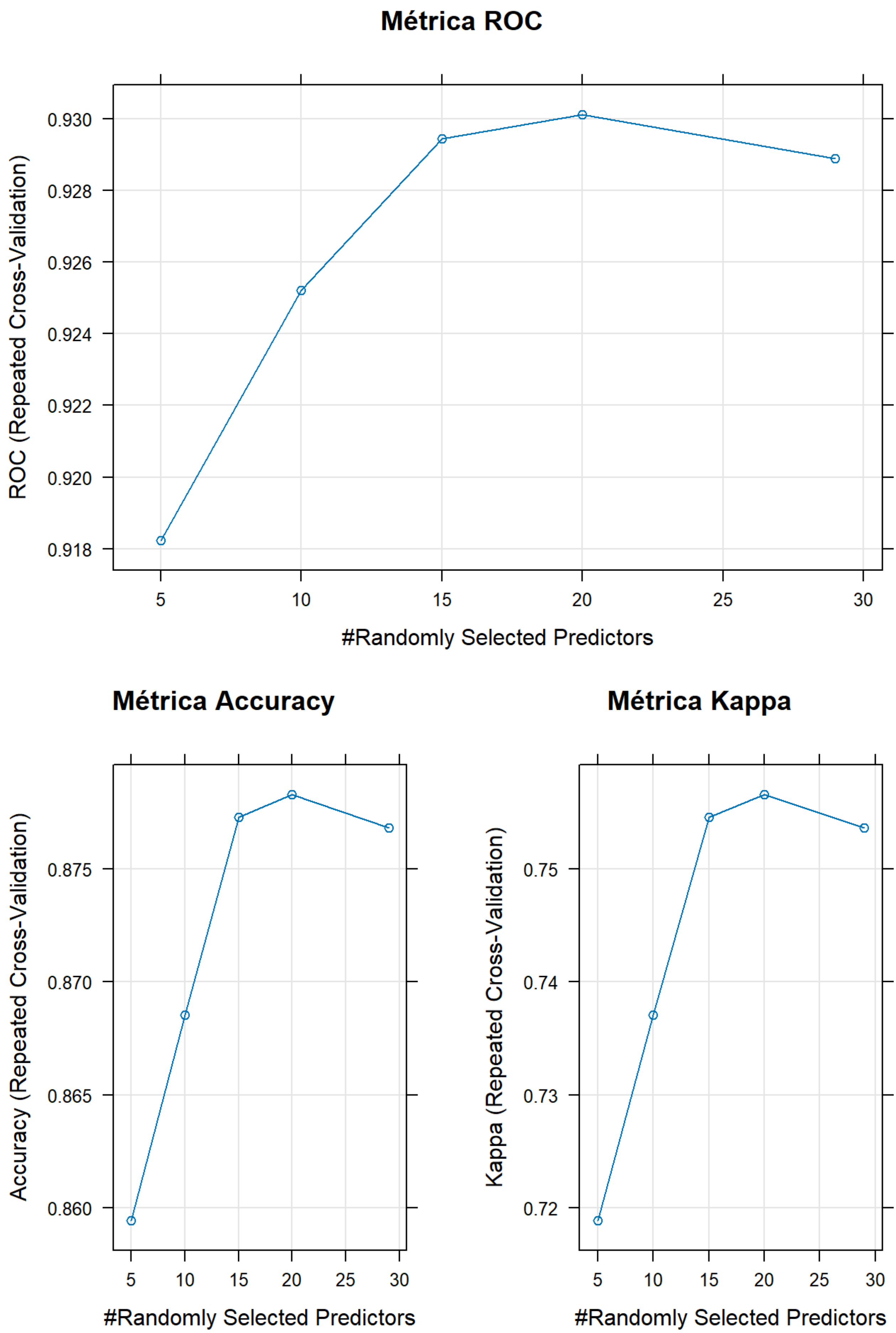
2.3.1. Random Forest

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

stopCluster(clusterCPU)

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

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

```r
rf_train
```

```
## Random Forest
##
## 87870 samples
##     7 predictor
##     2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 76887, 76886, 76886, 76887, 76886, 76886, ...
## Resampling results across tuning parameters:
##
##   mtry  ROC        Sens       Spec       Accuracy   Kappa
##    5    0.9182289  0.7953796  0.9235006  0.8594401  0.7188802
##   10    0.9252101  0.8138501  0.9232389  0.8685444  0.7370889
##   15    0.9294534  0.8302493  0.9242745  0.8772619  0.7545237
##   20    0.9301246  0.8357005  0.9208603  0.8782804  0.7565608
##   29    0.9288872  0.8384205  0.9152042  0.8768123  0.7536247
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 20.
```

```r
grafico_metricas(rf_train)
```




```
resultados(rf_train, "Random Forest")
```

RESULTADOS DEL MODELO Random Forest

mtry	ROC	Sens	Spec	Accuracy	Kappa	ROCSD	SensSD	SpecS
5	0.9182289	0.7953796	0.9235006	0.8594401	0.7188802	0.0038079	0.0054153	0.004944
10	0.9252101	0.8138501	0.9232389	0.8685444	0.7370889	0.0028879	0.0056582	0.004523
15	0.9294534	0.8302493	0.9242745	0.8772619	0.7545237	0.0028553	0.0058838	0.003855
20	0.9301246	0.8357005	0.9208603	0.8782804	0.7565608	0.0026459	0.0058301	0.003086
29	0.9288872	0.8384205	0.9152042	0.8768123	0.7536247	0.0026490	0.0053659	0.003604

```
mejor_modelo(rf_train)
```

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

	mtry
4	20

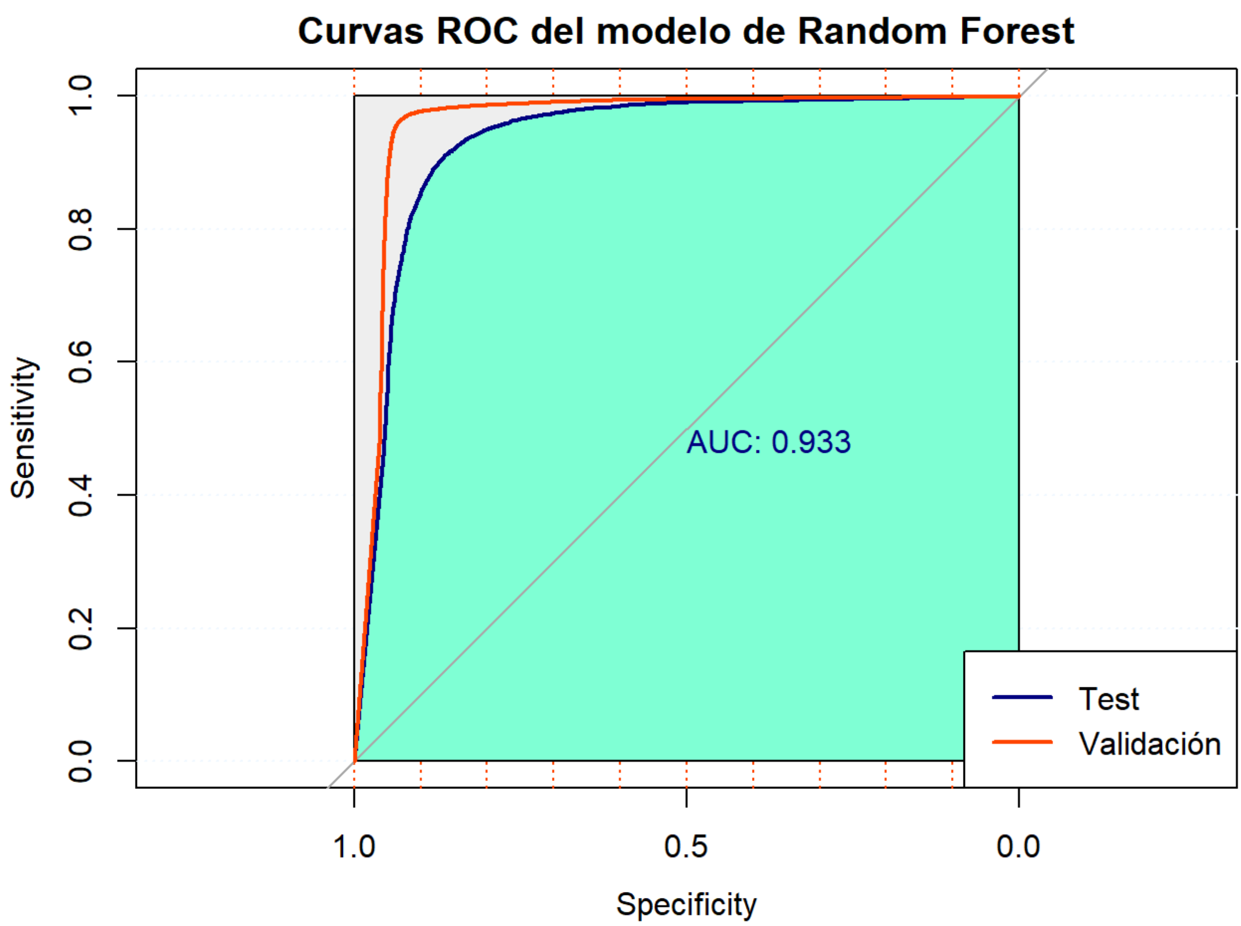
```
curvas_ROC(rf_train, "de Random Forest", train_general, test_general)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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




```
## [1] "ROC del modelo con el fichero de test: 0.933254516351738"
```

```
validation(rf_train, "RF", train_general, test_general)
```

```
## [1] "Modelo RF - Tabla de confusión para los datos de entrenamiento"
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    No   Yes
##        No  40580  1409
##        Yes  3355 42526
## 
##                Accuracy : 0.9458
##                  95% CI : (0.9443, 0.9473)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.8916
## 
##  Mcnemar's Test P-Value : < 2.2e-16
## 
##             Sensitivity : 0.9236
##             Specificity : 0.9679
##          Pos Pred Value : 0.9664
##          Neg Pred Value : 0.9269
##              Prevalence : 0.5000
##          Detection Rate : 0.4618
##    Detection Prevalence : 0.4779
##       Balanced Accuracy : 0.9458
## 
##        'Positive' Class : No
## 
## [1] "Modelo RF - Tabla de confusión para los datos de validación"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No   9219   790
##        Yes  1764 10193
##
##                Accuracy : 0.8837
##                  95% CI : (0.8794, 0.8879)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7675
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8394
##             Specificity : 0.9281
##          Pos Pred Value : 0.9211
##          Neg Pred Value : 0.8525
##              Prevalence : 0.5000
##          Detection Rate : 0.4197
##    Detection Prevalence : 0.4557
##       Balanced Accuracy : 0.8837
##
##        'Positive' Class : No
##
```

```
resumen_rf <- resumen(rf_train, train_general, test_general)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

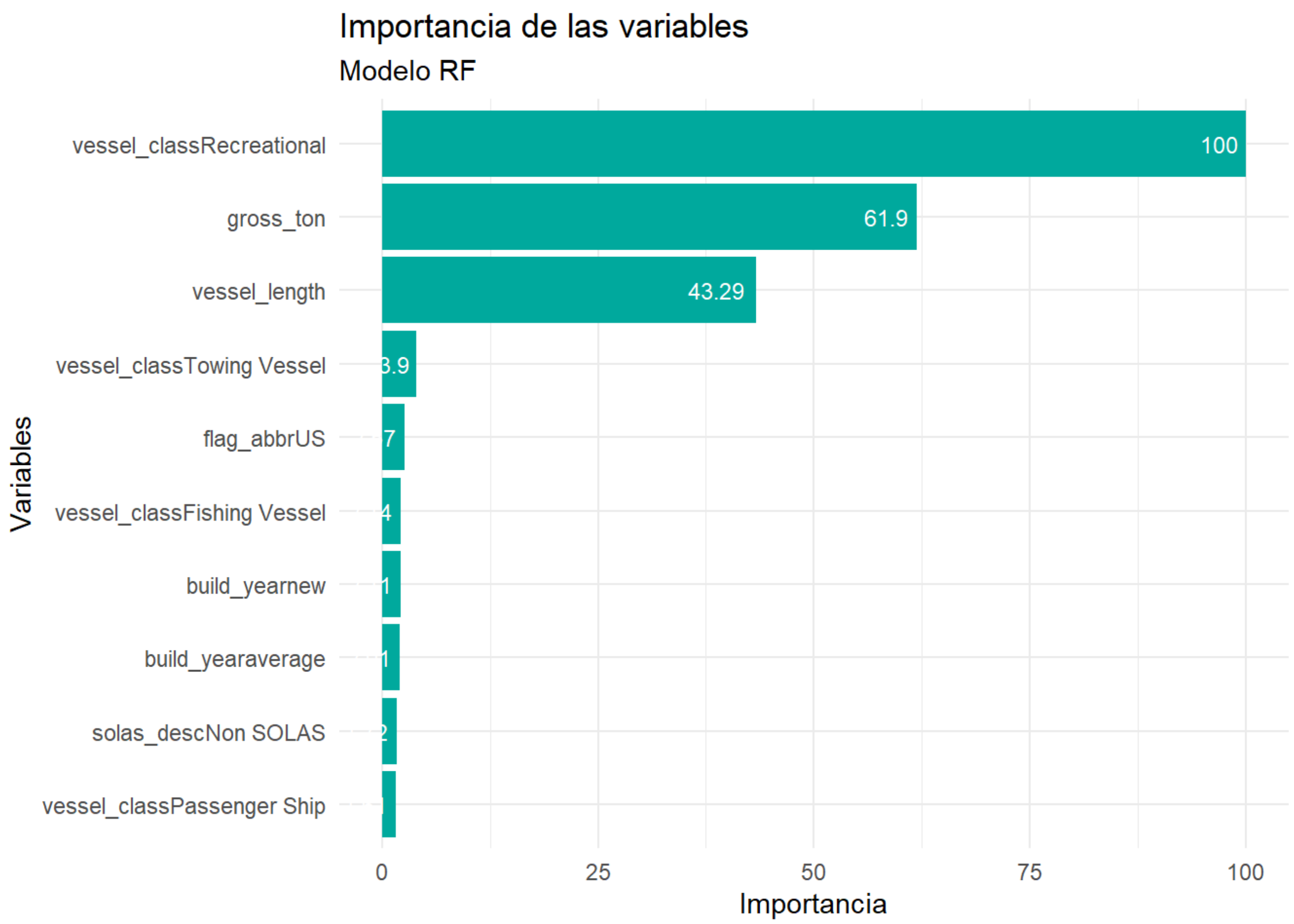
```
## Setting direction: controls < cases
```

```
resumen_rf %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ", "Random forest " = 7))
```

	Random forest						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.960	0.946	0.966	0.927	0.892	0.924	0.968

	Random forest						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Validación	0.933	0.884	0.921	0.852	0.768	0.839	0.928

```
importancia_var(rf_train, "RF")
```



2.3.2. Máquinas de vector soporte

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

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

svmGrid <-  expand.grid(sigma =seq (0.015, 0.045, by = 0.002), C = seq (0.15, 0.35, by =
0.02))

svm_train <- train(y ~ .,
                   data = train_general,
                   method= "svmRadial",
                   metric = metrica,
                   #preProc = c("center", "scale"),
                   trControl = control,
                   tuneGrid = svmGrid)

stopCluster(clusterCPU)

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

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

```
svm_train
```

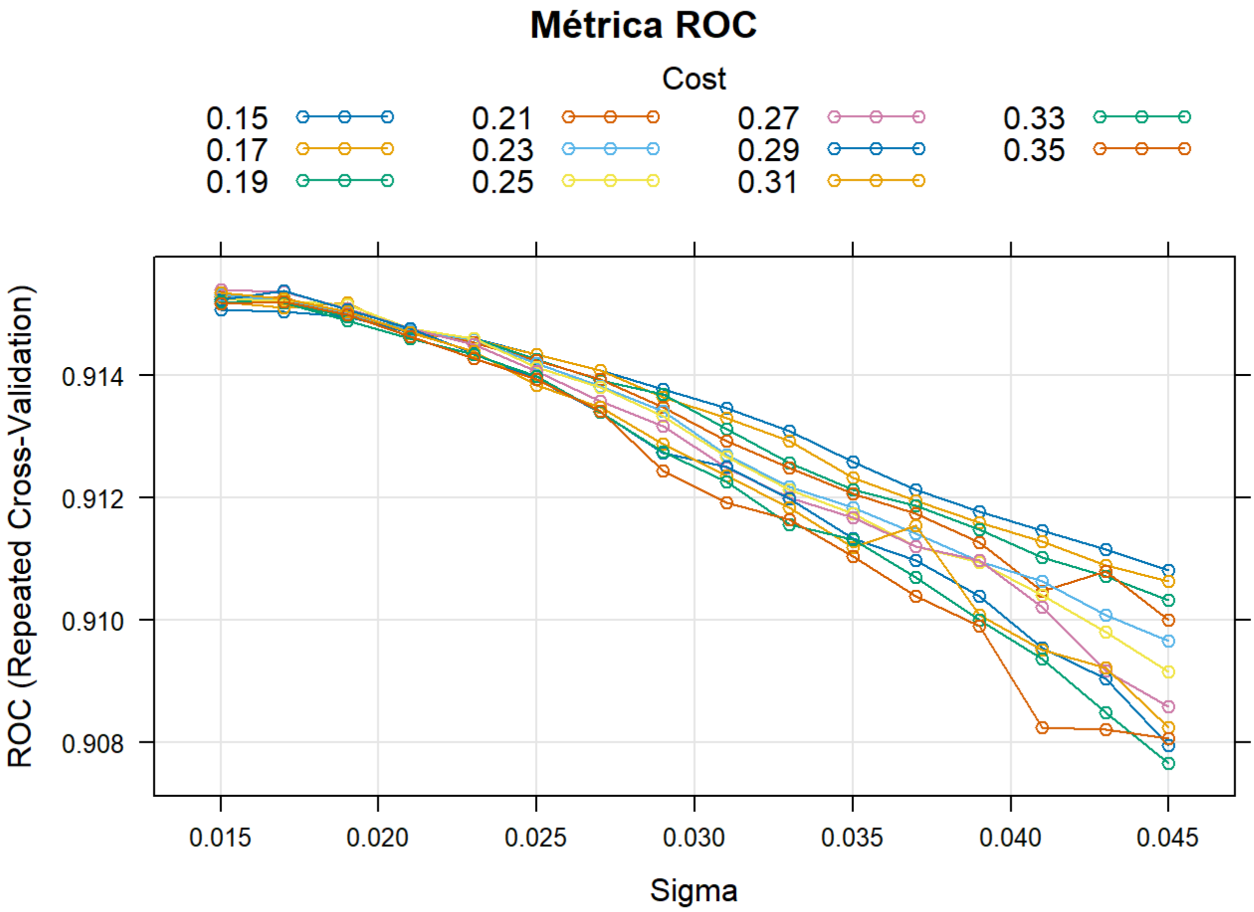
```
## Support Vector Machines with Radial Basis Function Kernel
##
## 17574 samples
##     7 predictor
##     2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 15377, 15378, 15377, 15378, 15377, 15376, ...
## Resampling results across tuning parameters:
##
##   sigma  C     ROC        Sens       Spec       Accuracy   Kappa
##   0.015  0.15  0.9150621  0.7188193  0.9474513  0.8334473  0.6666854
##   0.015  0.17  0.9152014  0.7196180  0.9471108  0.8336749  0.6671420
##   0.015  0.19  0.9152422  0.7201884  0.9462595  0.8335326  0.6668585
##   0.015  0.21  0.9152750  0.7215005  0.9453513  0.8337316  0.6672588
##   0.015  0.23  0.9153172  0.7223563  0.9447839  0.8338739  0.6675447
##   0.015  0.25  0.9152450  0.7235543  0.9442165  0.8341868  0.6681725
##   0.015  0.27  0.9153895  0.7244672  0.9432516  0.8341584  0.6681172
##   0.015  0.29  0.9152357  0.7251519  0.9426842  0.8342153  0.6682323
##   0.015  0.31  0.9153434  0.7259506  0.9417763  0.8341584  0.6681200
##   0.015  0.33  0.9151953  0.7266353  0.9412090  0.8342154  0.6682352
##   0.015  0.35  0.9151679  0.7268065  0.9406982  0.8340446  0.6678941
##   0.017  0.15  0.9150420  0.7187052  0.9467703  0.8330490  0.6658888
##   0.017  0.17  0.9150969  0.7196180  0.9465432  0.8333904  0.6665731
##   0.017  0.19  0.9151886  0.7205309  0.9456352  0.8333904  0.6665748
##   0.017  0.21  0.9152555  0.7215577  0.9447839  0.8334757  0.6667471
##   0.017  0.23  0.9152159  0.7229842  0.9441597  0.8338740  0.6675460
##   0.017  0.25  0.9152063  0.7241251  0.9435354  0.8341300  0.6680599
##   0.017  0.27  0.9153617  0.7256655  0.9423440  0.8343008  0.6684042
##   0.017  0.29  0.9153697  0.7262931  0.9414927  0.8341870  0.6681777
##   0.017  0.31  0.9152481  0.7270348  0.9410954  0.8343577  0.6685203
##   0.017  0.33  0.9151796  0.7282331  0.9397333  0.8342722  0.6683517
##   0.017  0.35  0.9152020  0.7287467  0.9388256  0.8340732  0.6679547
##   0.019  0.15  0.9149581  0.7189331  0.9463163  0.8329352  0.6656616
##   0.019  0.17  0.9151742  0.7197891  0.9459190  0.8331628  0.6661184
##   0.019  0.19  0.9149596  0.7209304  0.9454081  0.8334757  0.6667461
##   0.019  0.21  0.9149821  0.7222996  0.9438757  0.8333903  0.6665777
##   0.019  0.23  0.9149996  0.7236118  0.9428544  0.8335326  0.6668647
##   0.019  0.25  0.9151480  0.7249239  0.9422869  0.8339025  0.6676064
##   0.019  0.27  0.9150894  0.7268639  0.9412089  0.8343292  0.6684633
##   0.019  0.29  0.9150704  0.7276057  0.9403009  0.8342439  0.6682940
##   0.019  0.31  0.9150366  0.7284615  0.9395630  0.8343008  0.6684093
##   0.019  0.33  0.9148906  0.7294314  0.9388253  0.8344146  0.6686385
##   0.019  0.35  0.9149864  0.7305153  0.9379739  0.8345282  0.6688676
##   0.021  0.15  0.9147000  0.7193324  0.9460326  0.8329921  0.6657761
##   0.021  0.17  0.9146729  0.7204166  0.9452947  0.8331628  0.6661194
##   0.021  0.19  0.9147655  0.7215008  0.9440461  0.8330774  0.6659507
##   0.021  0.21  0.9147250  0.7233838  0.9422869  0.8331344  0.6660680
##   0.021  0.23  0.9147270  0.7244677  0.9416058  0.8333334  0.6664680
##   0.021  0.25  0.9147551  0.7264075  0.9407546  0.8338740  0.6675522
##   0.021  0.27  0.9147631  0.7274916  0.9401304  0.8341016  0.6680093
##   0.021  0.29  0.9147560  0.7286326  0.9395629  0.8343860  0.6685800
##   0.021  0.31  0.9146954  0.7293743  0.9387684  0.8343576  0.6685244
```

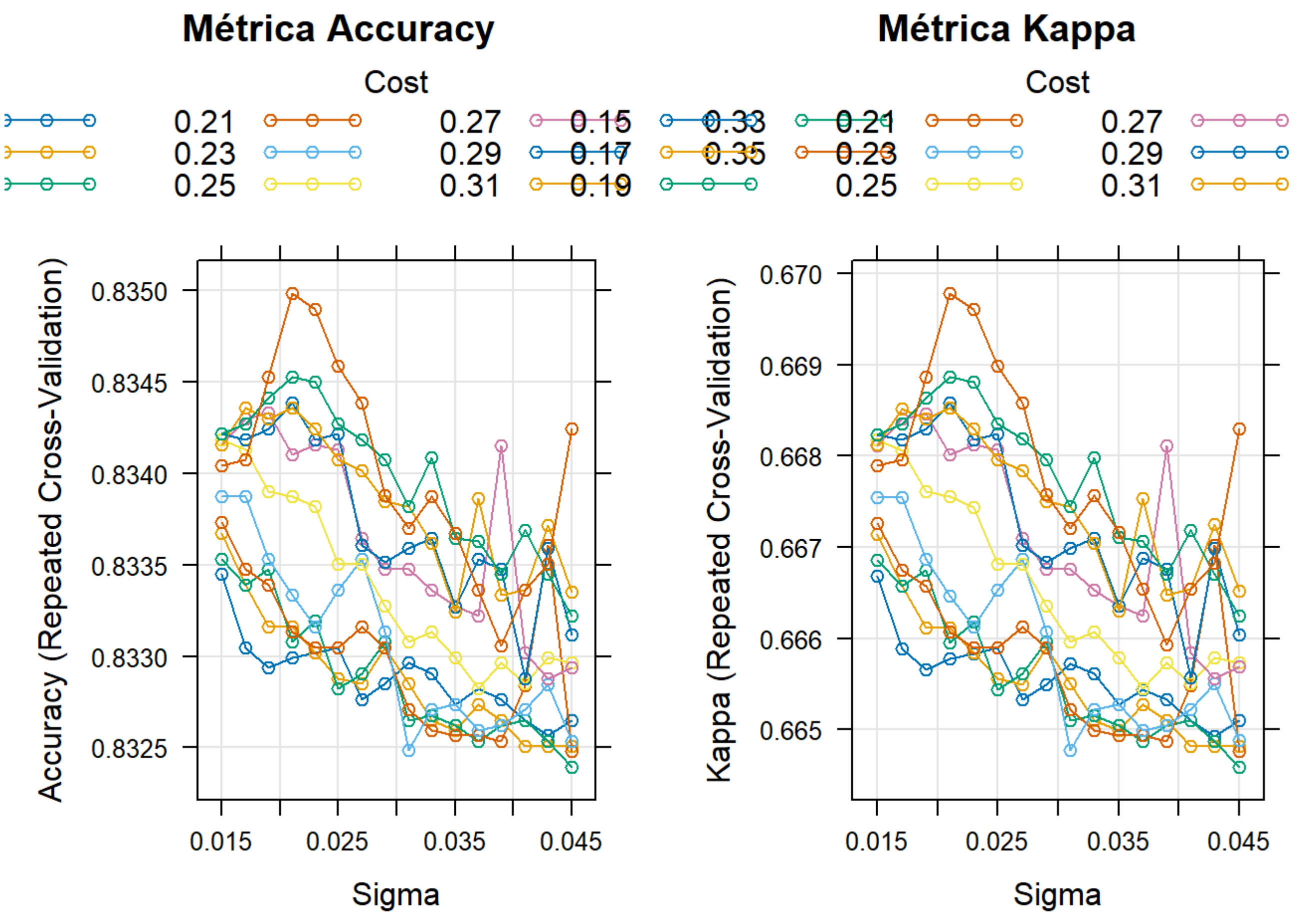
```
##    0.021  0.33  0.9146027  0.7303441  0.9381440  0.8345282  0.6688672
##    0.021  0.35  0.9146268  0.7318277  0.9375765  0.8349834  0.6697800
##    0.023  0.15  0.9145926  0.7199599  0.9454647  0.8330205  0.6658339
##    0.023  0.17  0.9145727  0.7216149  0.9438191  0.8330205  0.6658371
##    0.023  0.19  0.9146001  0.7232125  0.9425705  0.8331912  0.6661813
##    0.023  0.21  0.9145278  0.7245819  0.9409249  0.8330490  0.6658994
##    0.023  0.23  0.9145872  0.7252664  0.9404708  0.8331627  0.6661280
##    0.023  0.25  0.9146017  0.7269779  0.9400735  0.8338170  0.6674393
##    0.023  0.27  0.9144904  0.7280049  0.9397330  0.8341584  0.6681236
##    0.023  0.29  0.9143602  0.7290321  0.9387684  0.8341868  0.6681825
##    0.023  0.31  0.9143879  0.7300593  0.9378602  0.8342438  0.6682981
##    0.023  0.33  0.9143376  0.7310864  0.9373496  0.8344999  0.6688120
##    0.023  0.35  0.9142771  0.7322273  0.9370089  0.8348981  0.6696102
##    0.025  0.15  0.9143283  0.7213293  0.9441595  0.8330490  0.6658933
##    0.025  0.17  0.9143351  0.7221851  0.9429678  0.8328782  0.6655535
##    0.025  0.19  0.9142578  0.7238399  0.9412087  0.8328212  0.6654427
##    0.025  0.21  0.9142430  0.7251522  0.9403574  0.8330489  0.6659002
##    0.025  0.23  0.9141826  0.7262934  0.9398465  0.8333617  0.6665279
##    0.025  0.25  0.9141273  0.7270352  0.9393925  0.8335039  0.6668136
##    0.025  0.27  0.9140549  0.7287469  0.9389384  0.8341299  0.6680681
##    0.025  0.29  0.9139297  0.7297169  0.9381441  0.8342153  0.6682407
##    0.025  0.31  0.9138350  0.7302874  0.9372928  0.8340731  0.6679572
##    0.025  0.33  0.9139720  0.7312004  0.9367820  0.8342722  0.6683570
##    0.025  0.35  0.9139218  0.7322846  0.9363282  0.8345852  0.6689848
##    0.027  0.15  0.9140751  0.7217285  0.9431948  0.8327643  0.6653250
##    0.027  0.17  0.9140694  0.7227556  0.9423436  0.8328497  0.6654975
##    0.027  0.19  0.9139203  0.7243535  0.9408681  0.8329065  0.6656142
##    0.027  0.21  0.9139349  0.7257800  0.9399601  0.8331626  0.6661289
##    0.027  0.23  0.9138150  0.7269781  0.9395060  0.8335325  0.6668704
##    0.027  0.25  0.9137950  0.7276058  0.9388249  0.8335040  0.6668147
##    0.027  0.27  0.9135770  0.7288041  0.9379168  0.8336462  0.6671013
##    0.027  0.29  0.9133889  0.7296586  0.9369872  0.8336063  0.6670229
##    0.027  0.31  0.9134689  0.7302876  0.9371791  0.8340162  0.6678435
##    0.027  0.33  0.9133920  0.7310294  0.9367820  0.8341869  0.6681861
##    0.027  0.35  0.9134043  0.7319995  0.9362147  0.8343861  0.6685863
##    0.029  0.15  0.9137635  0.7220139  0.9430814  0.8328497  0.6654963
##    0.029  0.17  0.9136320  0.7237259  0.9417761  0.8330488  0.6658976
##    0.029  0.19  0.9136820  0.7249242  0.9406409  0.8330772  0.6659565
##    0.029  0.21  0.9134806  0.7257800  0.9397332  0.8330489  0.6659014
##    0.029  0.23  0.9134082  0.7268641  0.9388249  0.8331342  0.6660739
##    0.029  0.25  0.9133241  0.7274918  0.9384843  0.8332764  0.6663594
##    0.029  0.27  0.9131580  0.7282335  0.9381438  0.8334755  0.6667588
##    0.029  0.29  0.9127253  0.7292933  0.9371688  0.8335153  0.6668402
##    0.029  0.31  0.9128700  0.7299453  0.9371792  0.8338455  0.6675016
##    0.029  0.33  0.9127400  0.7308584  0.9367253  0.8340731  0.6679584
##    0.029  0.35  0.9124372  0.7319108  0.9352926  0.8338796  0.6675734
##    0.031  0.15  0.9134624  0.7226988  0.9426274  0.8329635  0.6657251
##    0.031  0.17  0.9132950  0.7238400  0.9412652  0.8328497  0.6654995
##    0.031  0.19  0.9131205  0.7246387  0.9400733  0.8326504  0.6651026
##    0.031  0.21  0.9129261  0.7256087  0.9392221  0.8327074  0.6652182
##    0.031  0.23  0.9127006  0.7266157  0.9377742  0.8324835  0.6647726
##    0.031  0.25  0.9126657  0.7272065  0.9383709  0.8330772  0.6659607
##    0.031  0.27  0.9124882  0.7281765  0.9382005  0.8334755  0.6667587
##    0.031  0.29  0.9124946  0.7287471  0.9378601  0.8335894  0.6669874
##    0.031  0.31  0.9123521  0.7297172  0.9373495  0.8338170  0.6674444
```

```
##    0.031  0.33  0.9122563  0.7307443  0.9363280  0.8338170  0.6674463
##    0.031  0.35  0.9119194  0.7317889  0.9350505  0.8336974  0.6672090
##    0.033  0.15  0.9130770  0.7226988  0.9425137  0.8329066  0.6656112
##    0.033  0.17  0.9129245  0.7236688  0.9410381  0.8326505  0.6651010
##    0.033  0.19  0.9125709  0.7246387  0.9401300  0.8326789  0.6651594
##    0.033  0.21  0.9124856  0.7252664  0.9393354  0.8325935  0.6649899
##    0.033  0.23  0.9121738  0.7261224  0.9387114  0.8327074  0.6652193
##    0.033  0.25  0.9121236  0.7270925  0.9385978  0.8331342  0.6660743
##    0.033  0.27  0.9119913  0.7278913  0.9382574  0.8333617  0.6665308
##    0.033  0.29  0.9119772  0.7292037  0.9375195  0.8336462  0.6671019
##    0.033  0.31  0.9118378  0.7300027  0.9366684  0.8336179  0.6670467
##    0.033  0.33  0.9115577  0.7317720  0.9358374  0.8340869  0.6679854
##    0.033  0.35  0.9116298  0.7316001  0.9355902  0.8338739  0.6675614
##    0.035  0.15  0.9125735  0.7226418  0.9422299  0.8327359  0.6652698
##    0.035  0.17  0.9123165  0.7234406  0.9411516  0.8325935  0.6649867
##    0.035  0.19  0.9121168  0.7245818  0.9400732  0.8326220  0.6650456
##    0.035  0.21  0.9120533  0.7249812  0.9395623  0.8325650  0.6649325
##    0.035  0.23  0.9118303  0.7257229  0.9391652  0.8327358  0.6652752
##    0.035  0.25  0.9117347  0.7266359  0.9387681  0.8329919  0.6657890
##    0.035  0.27  0.9116738  0.7276630  0.9383141  0.8332764  0.6663596
##    0.035  0.29  0.9113353  0.7294764  0.9365030  0.8332725  0.6663553
##    0.035  0.31  0.9111897  0.7302069  0.9357160  0.8332421  0.6662960
##    0.035  0.33  0.9113158  0.7306302  0.9361008  0.8336462  0.6671045
##    0.035  0.35  0.9110343  0.7313719  0.9354200  0.8336747  0.6671629
##    0.037  0.15  0.9121287  0.7225277  0.9425137  0.8328212  0.6654403
##    0.037  0.17  0.9119530  0.7235547  0.9413218  0.8327358  0.6652713
##    0.037  0.19  0.9118631  0.7240682  0.9404137  0.8325366  0.6648740
##    0.037  0.21  0.9117335  0.7245817  0.9399596  0.8325650  0.6649317
##    0.037  0.23  0.9114052  0.7253805  0.9392219  0.8325935  0.6649901
##    0.037  0.25  0.9111960  0.7265218  0.9385412  0.8328211  0.6654474
##    0.037  0.27  0.9111976  0.7277772  0.9380871  0.8332195  0.6662462
##    0.037  0.29  0.9109763  0.7289756  0.9375197  0.8335325  0.6668742
##    0.037  0.31  0.9115431  0.7296875  0.9374731  0.8338641  0.6675386
##    0.037  0.33  0.9107003  0.7301905  0.9364990  0.8336266  0.6670645
##    0.037  0.35  0.9103903  0.7312008  0.9349661  0.8333618  0.6665369
##    0.039  0.15  0.9117709  0.7225277  0.9424001  0.8327643  0.6653265
##    0.039  0.17  0.9115940  0.7232123  0.9414920  0.8326504  0.6650999
##    0.039  0.19  0.9114668  0.7238399  0.9408109  0.8326219  0.6650441
##    0.039  0.21  0.9112582  0.7242963  0.9401866  0.8325365  0.6648743
##    0.039  0.23  0.9109550  0.7255518  0.9391083  0.8326219  0.6650473
##    0.039  0.25  0.9109340  0.7265218  0.9388246  0.8329634  0.6657317
##    0.039  0.27  0.9109742  0.7285026  0.9392277  0.8341531  0.6681140
##    0.039  0.29  0.9103875  0.7289185  0.9374630  0.8334756  0.6667603
##    0.039  0.31  0.9100828  0.7295463  0.9365549  0.8333333  0.6664770
##    0.039  0.33  0.9099903  0.7304590  0.9358739  0.8334471  0.6667062
##    0.039  0.35  0.9098965  0.7305107  0.9350471  0.8330552  0.6659248
##    0.041  0.15  0.9114530  0.7222994  0.9424001  0.8326505  0.6650984
##    0.041  0.17  0.9112743  0.7229269  0.9414919  0.8325081  0.6648149
##    0.041  0.19  0.9110124  0.7237258  0.9409811  0.8326503  0.6651008
##    0.041  0.21  0.9104592  0.7259870  0.9391025  0.8328399  0.6654813
##    0.041  0.23  0.9106324  0.7255519  0.9392786  0.8327073  0.6652180
##    0.041  0.25  0.9103971  0.7266931  0.9384275  0.8328495  0.6655045
##    0.041  0.27  0.9102119  0.7273778  0.9380870  0.8330203  0.6658472
##    0.041  0.29  0.9095437  0.7287461  0.9364425  0.8328780  0.6655652
##    0.041  0.31  0.9095069  0.7280761  0.9380776  0.8333637  0.6665350
```

```
##   0.041  0.33  0.9093600  0.7301294  0.9366807  0.8336872  0.6671857
##   0.041  0.35  0.9082453  0.7323043  0.9338615  0.8333622  0.6665372
##   0.043  0.15  0.9111430  0.7221853  0.9423433  0.8325651  0.6649275
##   0.043  0.17  0.9108918  0.7228129  0.9416054  0.8325081  0.6648148
##   0.043  0.19  0.9107171  0.7233834  0.9410946  0.8325365  0.6648725
##   0.043  0.21  0.9107938  0.7250943  0.9415275  0.8336067  0.6670153
##   0.043  0.23  0.9100812  0.7255203  0.9395901  0.8328477  0.6654987
##   0.043  0.25  0.9098036  0.7267502  0.9386544  0.8329918  0.6657891
##   0.043  0.27  0.9091661  0.7277114  0.9374715  0.8328779  0.6655632
##   0.043  0.29  0.9090383  0.7289403  0.9376737  0.8335884  0.6669882
##   0.043  0.31  0.9092092  0.7303568  0.9365091  0.8337179  0.6672451
##   0.043  0.33  0.9084865  0.7300026  0.9363280  0.8334470  0.6667053
##   0.043  0.35  0.9082067  0.7308584  0.9355903  0.8335040  0.6668206
##   0.045  0.15  0.9108113  0.7221282  0.9425703  0.8326505  0.6650981
##   0.045  0.17  0.9106283  0.7226988  0.9417189  0.8325081  0.6648146
##   0.045  0.19  0.9103267  0.7234407  0.9407540  0.8323943  0.6645883
##   0.045  0.21  0.9099953  0.7242966  0.9400730  0.8324797  0.6647606
##   0.045  0.23  0.9096535  0.7252097  0.9392786  0.8325365  0.6648761
##   0.045  0.25  0.9091507  0.7265221  0.9388247  0.8329634  0.6657318
##   0.045  0.27  0.9085831  0.7277114  0.9375925  0.8329386  0.6656845
##   0.045  0.29  0.9079450  0.7284482  0.9372145  0.8331165  0.6660415
##   0.045  0.31  0.9082453  0.7286687  0.9374677  0.8333534  0.6665156
##   0.045  0.33  0.9076567  0.7296031  0.9362712  0.8332194  0.6662495
##   0.045  0.35  0.9080600  0.7308762  0.9370487  0.8342440  0.6683003
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.015 and C = 0.27.
```

```
grafico_metricas(svm_train)
```






```
resultados(svm_train, "SVM")
```

RESULTADOS DEL MODELO SVM

sigma	C	ROC	Sens	Spec	Accuracy	Kappa	ROCSD	SensSD
0.015	0.15	0.9150621	0.7188193	0.9474513	0.8334473	0.6666854	0.0045509	0.0067786
0.015	0.17	0.9152014	0.7196180	0.9471108	0.8336749	0.6671420	0.0046336	0.0065293
0.015	0.19	0.9152422	0.7201884	0.9462595	0.8335326	0.6668585	0.0045601	0.0062362
0.015	0.21	0.9152750	0.7215005	0.9453513	0.8337316	0.6672588	0.0045696	0.0061166
0.015	0.23	0.9153172	0.7223563	0.9447839	0.8338739	0.6675447	0.0045873	0.0061534
0.015	0.25	0.9152450	0.7235543	0.9442165	0.8341868	0.6681725	0.0046141	0.0060831
0.015	0.27	0.9153895	0.7244672	0.9432516	0.8341584	0.6681172	0.0047338	0.0058503
0.015	0.29	0.9152357	0.7251519	0.9426842	0.8342153	0.6682323	0.0047219	0.0058850
0.015	0.31	0.9153434	0.7259506	0.9417763	0.8341584	0.6681200	0.0046832	0.0064365
0.015	0.33	0.9151953	0.7266353	0.9412090	0.8342154	0.6682352	0.0047311	0.0066959
0.015	0.35	0.9151679	0.7268065	0.9406982	0.8340446	0.6678941	0.0047605	0.0066416
0.017	0.15	0.9150420	0.7187052	0.9467703	0.8330490	0.6658888	0.0048062	0.0067360

```
mejor_modelo(svm_train)
```

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

	sigma	C
7	0.015	0.27

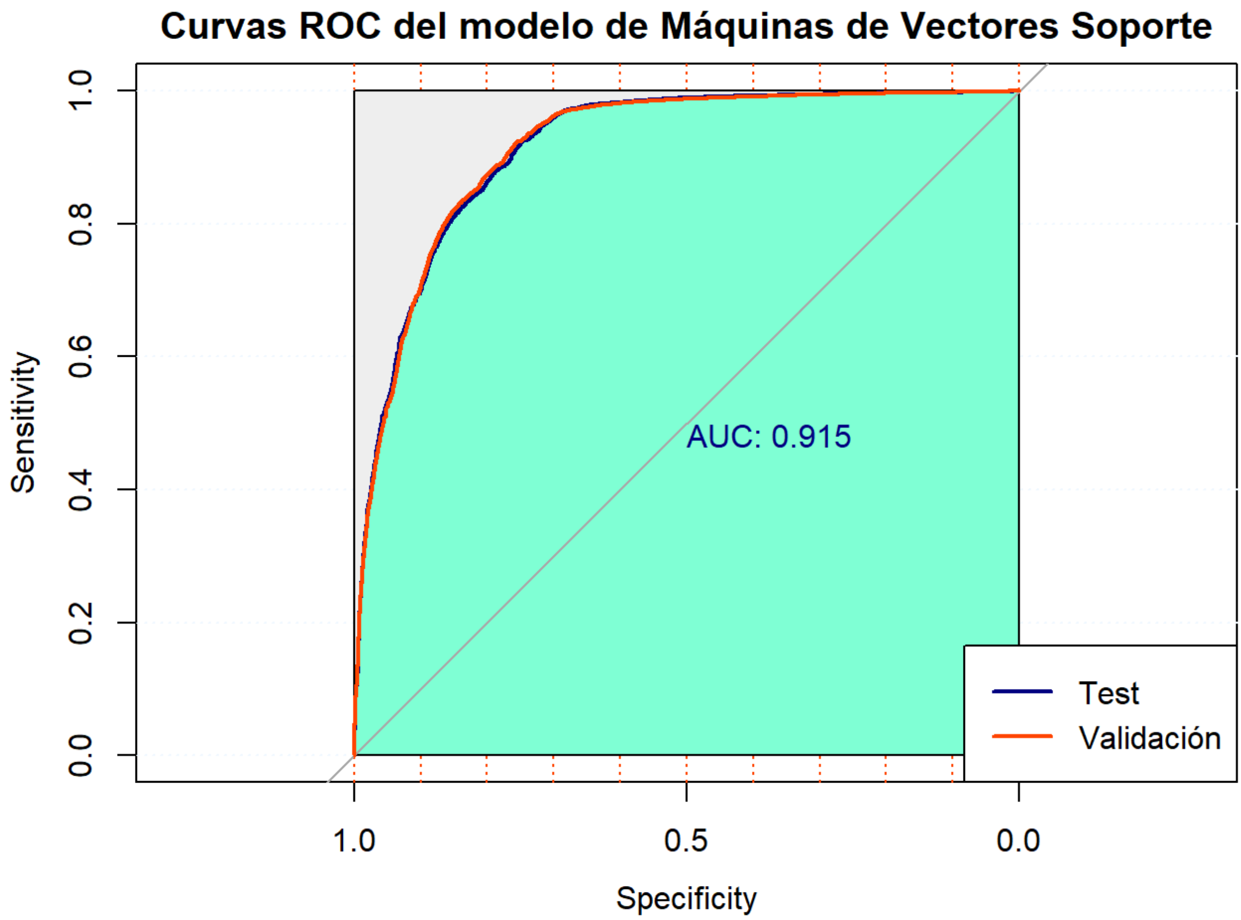
```
curvas_ROC(svm_train, "de Máquinas de Vectores Soporte", train_general, test_general)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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




```
## [1] "ROC del modelo con el fichero de test: 0.914663170977597"
```

```
validation(svm_train, "SVM", train_general, test_general)
```

```
## [1] "Modelo SVM - Tabla de confusión para los datos de entrenamiento"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  32071  2601
##        Yes 11864 41334
##
##                Accuracy : 0.8354
##                  95% CI : (0.8329, 0.8378)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6708
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7300
##             Specificity : 0.9408
##          Pos Pred Value : 0.9250
##          Neg Pred Value : 0.7770
##              Prevalence : 0.5000
##          Detection Rate : 0.3650
##    Detection Prevalence : 0.3946
##       Balanced Accuracy : 0.8354
##
##        'Positive' Class : No
##
## [1] "Modelo SVM - Tabla de confusión para los datos de validación"
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    No   Yes
##        No   7916   630
##        Yes  3067 10353
## 
##                Accuracy : 0.8317
##                  95% CI : (0.8267, 0.8366)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.6634
## 
##  Mcnemar's Test P-Value : < 2.2e-16
## 
##             Sensitivity : 0.7208
##             Specificity : 0.9426
##          Pos Pred Value : 0.9263
##          Neg Pred Value : 0.7715
##              Prevalence : 0.5000
##          Detection Rate : 0.3604
##    Detection Prevalence : 0.3891
##       Balanced Accuracy : 0.8317
## 
##        'Positive' Class : No
## 
```

```
resumen_svm <- resumen(svm_train, train_general, test_general)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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

```
resumen_svm %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ",
              "Máquinas de Vectores Soporte " = 7))
```

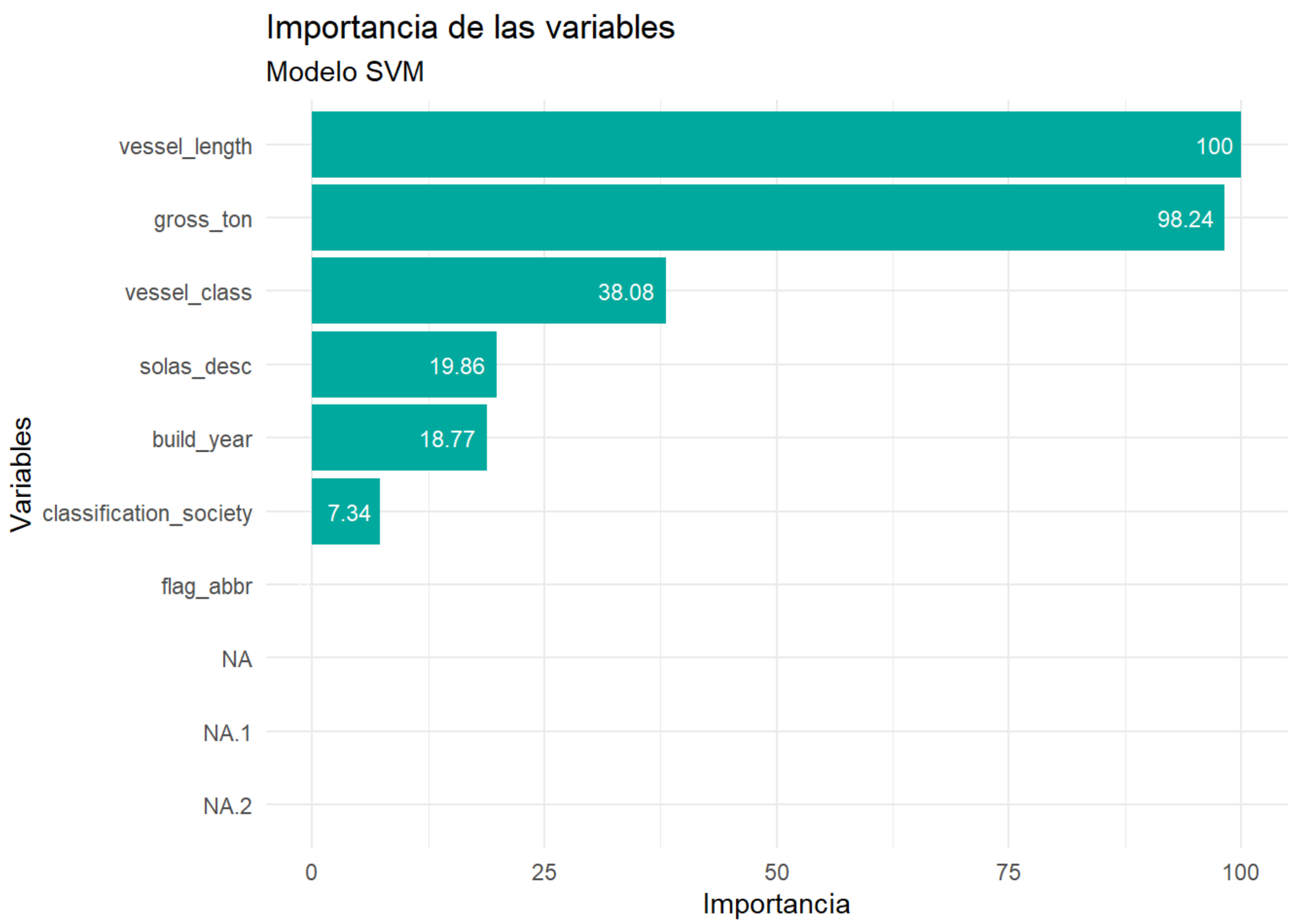
	Máquinas de Vectores Soporte						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**

	Máquinas de Vectores Soporte						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.915	0.835	0.925	0.777	0.671	0.730	0.941
Datos Validación	0.915	0.832	0.926	0.771	0.663	0.721	0.943

```
importancia_var(svm_train, "SVM")
```

```
## Warning: Removed 3 rows containing missing values (`position_stack()`).
## Removed 3 rows containing missing values (`position_stack()`).
```

```
## Warning: Removed 3 rows containing missing values (`geom_text()`).
```



2.3.3. Perceptrón multicapa

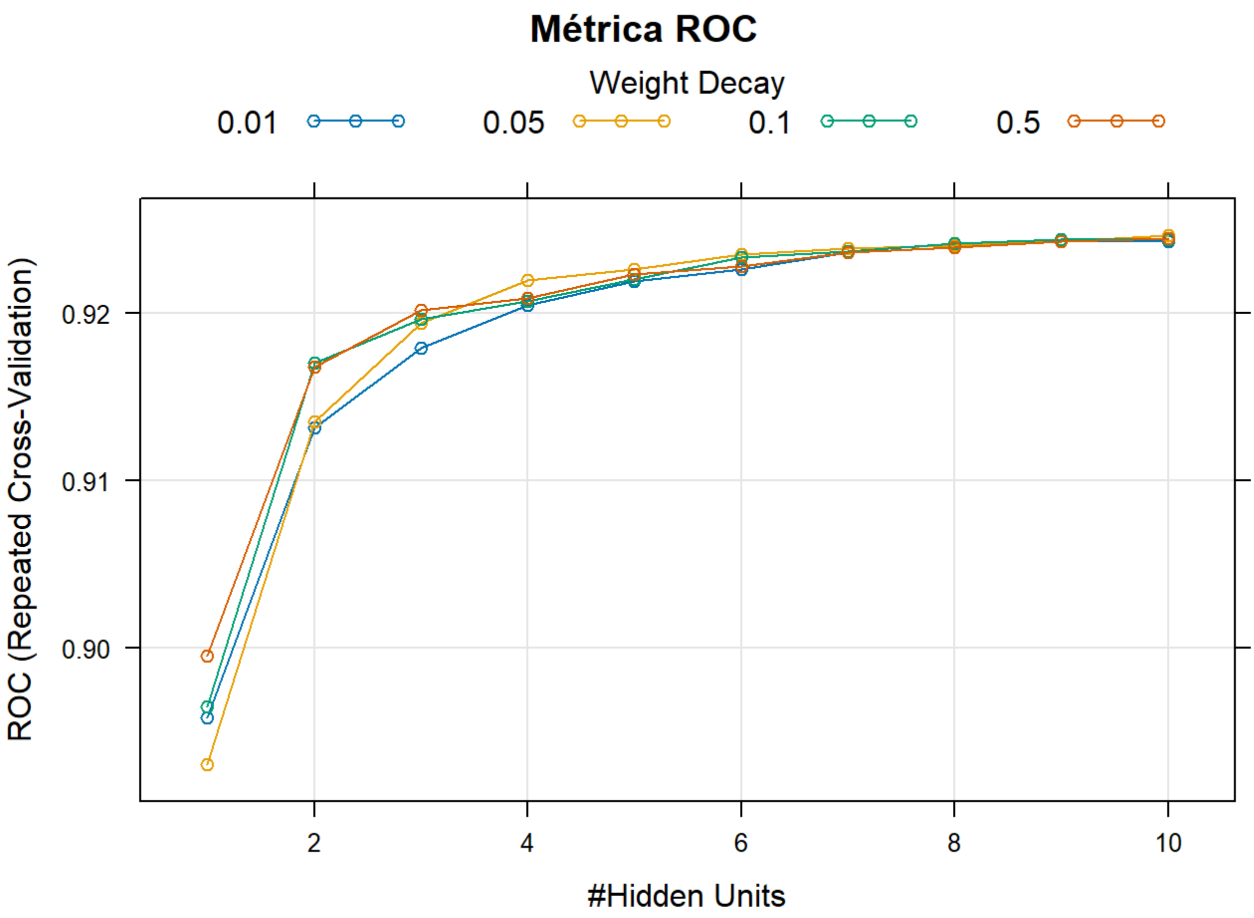
```
if (train_switch  == 1) {
set.seed(7)

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

stopCluster(clusterCPU)

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

toc()

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

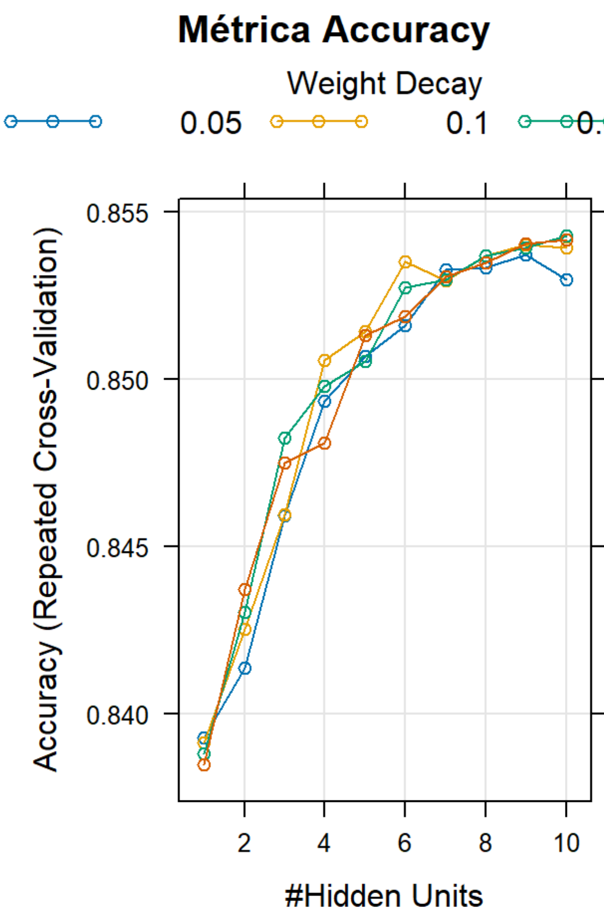
```
nnet_train
```

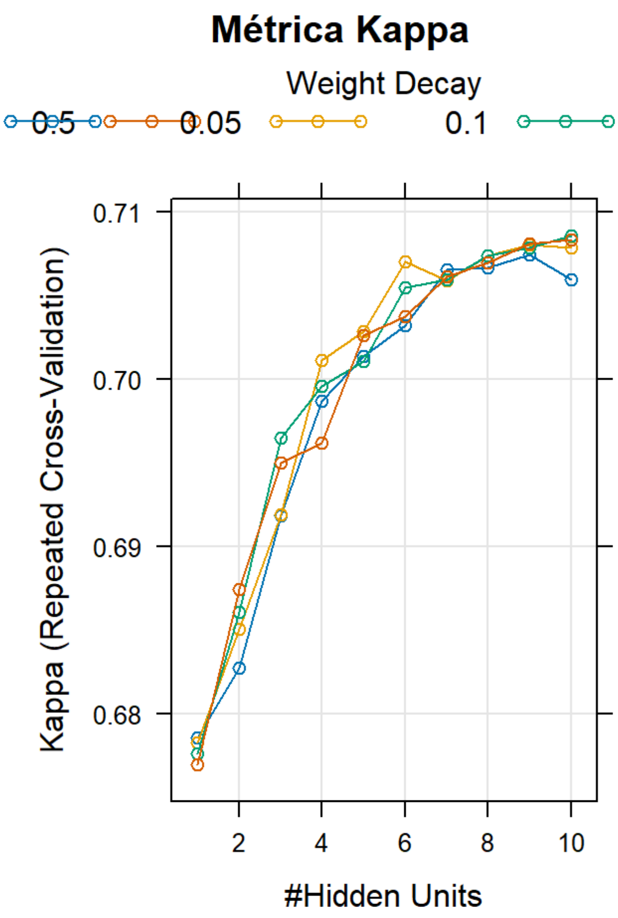
```
## Neural Network
##
## 87870 samples
##     7 predictor
##     2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 76887, 76886, 76886, 76887, 76886, 76886, ...
## Resampling results across tuning parameters:
##
##   size  decay  ROC        Sens       Spec       Accuracy   Kappa
##    1    0.01   0.8958398  0.7527596  0.9258449  0.8393024  0.6786048
##    1    0.05   0.8930302  0.7505748  0.9277340  0.8391545  0.6783089
##    1    0.10   0.8965027  0.7484013  0.9292479  0.8388244  0.6776490
##    1    0.50   0.8995052  0.7510641  0.9259247  0.8384944  0.6769888
##    2    0.01   0.9131135  0.7653803  0.9173554  0.8413679  0.6827358
##    2    0.05   0.9134713  0.7726534  0.9124047  0.8425287  0.6850577
##    2    0.10   0.9170281  0.7800386  0.9060545  0.8430465  0.6860931
##    2    0.50   0.9167416  0.7722773  0.9152156  0.8437465  0.6874930
##    3    0.01   0.9179163  0.7812225  0.9106406  0.8459316  0.6918632
##    3    0.05   0.9193759  0.7812790  0.9106410  0.8459599  0.6919199
##    3    0.10   0.9196180  0.7863777  0.9101059  0.8482418  0.6964836
##    3    0.50   0.9201803  0.7858771  0.9091497  0.8475135  0.6950269
##    4    0.01   0.9204471  0.7940026  0.9047226  0.8493627  0.6987254
##    4    0.05   0.9219529  0.7945032  0.9066463  0.8505747  0.7011495
##    4    0.10   0.9207194  0.7921931  0.9074087  0.8498009  0.6996018
##    4    0.50   0.9208451  0.7855697  0.9106636  0.8481166  0.6962332
##    5    0.01   0.9218560  0.7932402  0.9081256  0.8506829  0.7013657
##    5    0.05   0.9226209  0.7947196  0.9081483  0.8514340  0.7028680
##    5    0.10   0.9220268  0.7935473  0.9075339  0.8505406  0.7010812
##    5    0.50   0.9222884  0.7926939  0.9099579  0.8513259  0.7026518
##    6    0.01   0.9225956  0.7947197  0.9085239  0.8516218  0.7032436
##    6    0.05   0.9235087  0.7993627  0.9077160  0.8535393  0.7070787
##    6    0.10   0.9233187  0.7987141  0.9067942  0.8527541  0.7055082
##    6    0.50   0.9227545  0.7931491  0.9106180  0.8518835  0.7037670
##    7    0.01   0.9235994  0.8006260  0.9059293  0.8532776  0.7065552
##    7    0.05   0.9238260  0.7992488  0.9066576  0.8529533  0.7059065
##    7    0.10   0.9236759  0.7999431  0.9060317  0.8529874  0.7059748
##    7    0.50   0.9235951  0.7967794  0.9093548  0.8530671  0.7061342
##    8    0.01   0.9239073  0.8010584  0.9056333  0.8533459  0.7066918
##    8    0.05   0.9240031  0.8006941  0.9067144  0.8537043  0.7074086
##    8    0.10   0.9241283  0.8006829  0.9067372  0.8537100  0.7074201
##    8    0.50   0.9239321  0.7987938  0.9081940  0.8534938  0.7069877
##    9    0.01   0.9242622  0.8025606  0.9048822  0.8537214  0.7074429
##    9    0.05   0.9242653  0.8005236  0.9075111  0.8540173  0.7080347
##    9    0.10   0.9244035  0.8000455  0.9078525  0.8539490  0.7078981
##    9    0.50   0.9242413  0.7985320  0.9095824  0.8540572  0.7081144
##   10    0.01   0.9242781  0.8016161  0.9043701  0.8529931  0.7059862
##   10    0.05   0.9246347  0.8009674  0.9069080  0.8539377  0.7078754
##   10    0.10   0.9244034  0.8023445  0.9062706  0.8543075  0.7086151
##   10    0.50   0.9244681  0.8005008  0.9078525  0.8541766  0.7083533
##
## ROC was used to select the optimal model using the largest value.
```

```
## The final values used for the model were size = 10 and decay = 0.05.
```

```
grafico_metricas(nnet_train)
```








```
resultados(nnet_train, "Perceptrón multicapa")
```

RESULTADOS DEL MODELO Perceptrón multicapa

size	decay	ROC	Sens	Spec	Accuracy	Kappa	ROCSD	SensSD
1	0.01	0.8958398	0.7527596	0.9258449	0.8393024	0.6786048	0.0109440	0.0119610
1	0.05	0.8930302	0.7505748	0.9277340	0.8391545	0.6783089	0.0096471	0.0129538
1	0.10	0.8965027	0.7484013	0.9292479	0.8388244	0.6776490	0.0116083	0.0162460
1	0.50	0.8995052	0.7510641	0.9259247	0.8384944	0.6769888	0.0103985	0.0104261
2	0.01	0.9131135	0.7653803	0.9173554	0.8413679	0.6827358	0.0066410	0.0166484
2	0.05	0.9134713	0.7726534	0.9124047	0.8425287	0.6850577	0.0061574	0.0156100
2	0.10	0.9170281	0.7800386	0.9060545	0.8430465	0.6860931	0.0042734	0.0137424
2	0.50	0.9167416	0.7722773	0.9152156	0.8437465	0.6874930	0.0040788	0.0135554
3	0.01	0.9179163	0.7812225	0.9106406	0.8459316	0.6918632	0.0031599	0.0118045
3	0.05	0.9193759	0.7812790	0.9106410	0.8459599	0.6919199	0.0037073	0.0164935
3	0.10	0.9196180	0.7863777	0.9101059	0.8482418	0.6964836	0.0032563	0.0102994

size	decay	ROC	Sens	Spec	Accuracy	Kappa	ROCSD	SensSD
3	0.50	0.9201803	0.7858771	0.9091497	0.8475135	0.6950269	0.0037572	0.0134395

```
mejor_modelo(nnet_train)
```

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

	size	decay
38	10	0.05

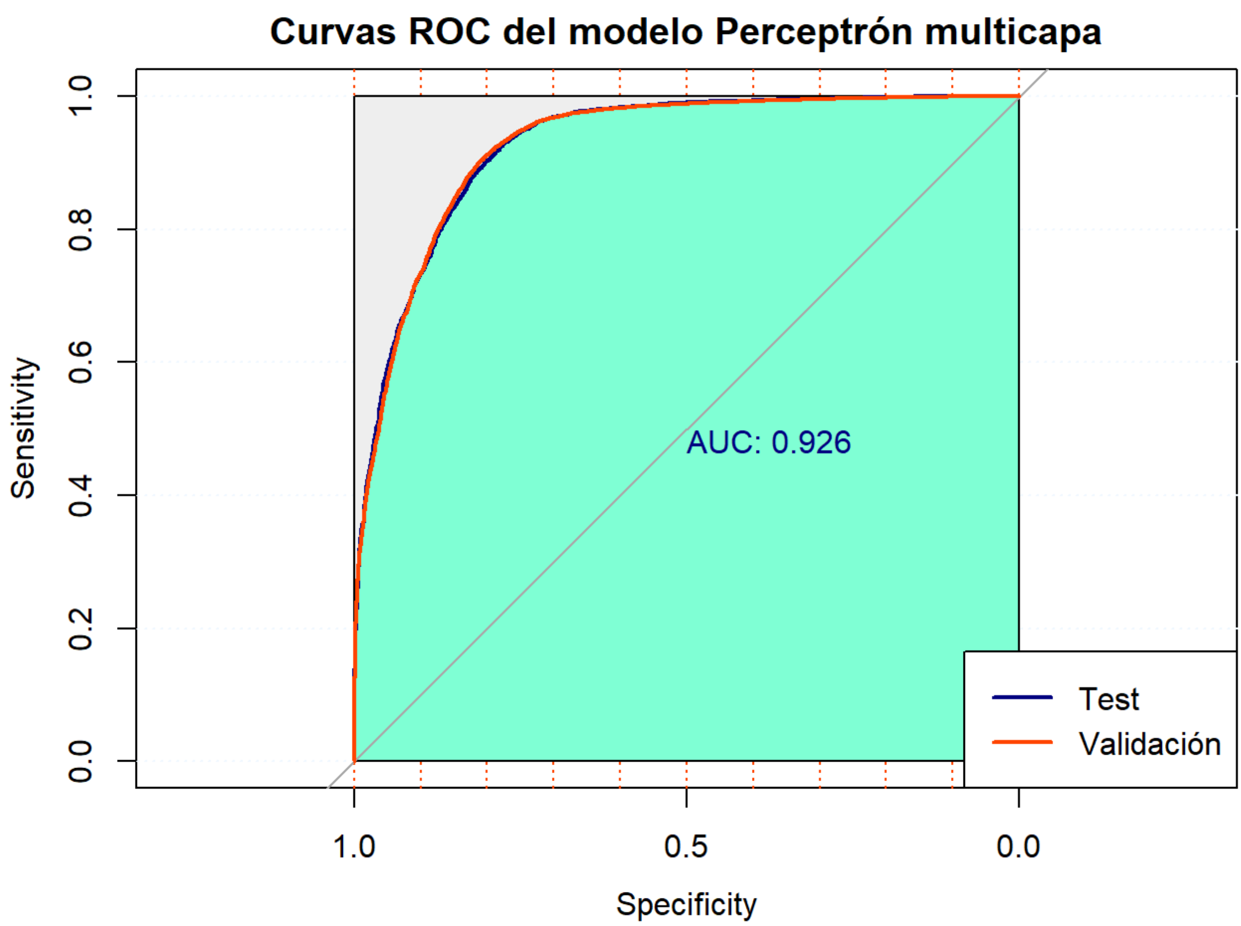
```
curvas_ROC(nnet_train, "Perceptrón multicapa", train_general, test_general)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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



```
## [1] "ROC del modelo con el fichero de test: 0.926434046230171"
```

```
validation(nnet_train, "Perceptrón multicapa", train_general, test_general)
```

```
## [1] "Modelo Perceptrón multicapa - Tabla de confusión para los datos de entrenamiento"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  35250  3968
##        Yes  8685 39967
##
##                Accuracy : 0.856
##                  95% CI : (0.8537, 0.8583)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.712
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8023
##             Specificity : 0.9097
##          Pos Pred Value : 0.8988
##          Neg Pred Value : 0.8215
##              Prevalence : 0.5000
##          Detection Rate : 0.4012
##    Detection Prevalence : 0.4463
##       Balanced Accuracy : 0.8560
##
##        'Positive' Class : No
##
## [1] "Modelo Perceptrón multicapa - Tabla de confusión para los datos de validación"
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   No  Yes
##        No  8715  997
##        Yes 2268 9986
##                                           
##                Accuracy : 0.8514          
##                  95% CI : (0.8466, 0.856) 
##     No Information Rate : 0.5             
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.7027          
##                                           
##  Mcnemar's Test P-Value : < 2.2e-16       
##                                           
##             Sensitivity : 0.7935          
##             Specificity : 0.9092          
##          Pos Pred Value : 0.8973          
##          Neg Pred Value : 0.8149          
##              Prevalence : 0.5000          
##          Detection Rate : 0.3967          
##    Detection Prevalence : 0.4421          
##       Balanced Accuracy : 0.8514          
##                                           
##        'Positive' Class : No              
## 
```

```
resumen_nnet <- resumen(nnet_train, train_general, test_general)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

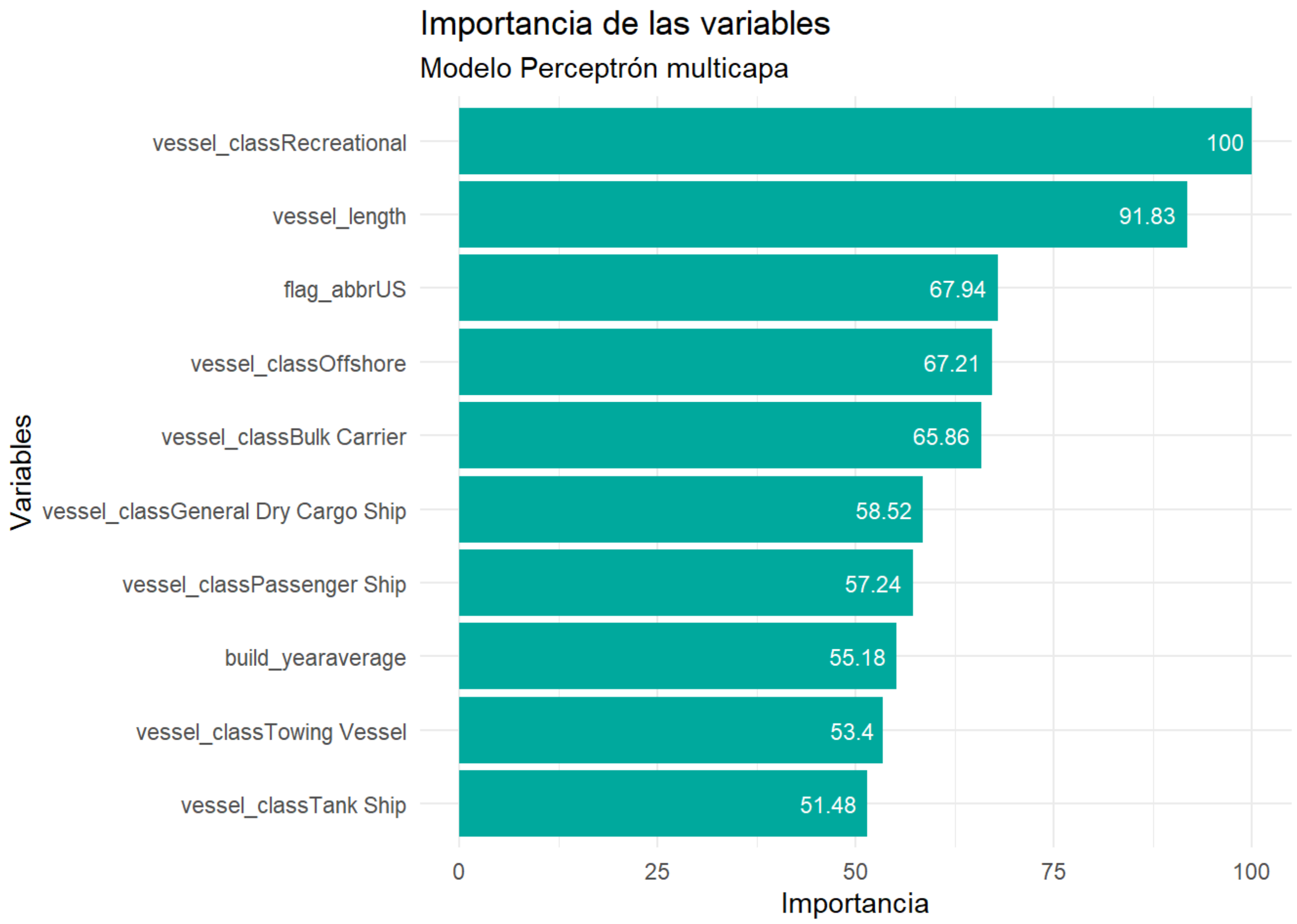
```
## Setting direction: controls < cases
```

```
resumen_nnet %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ",
              "Red Neuronal. Perceptrón Multicapa " = 7))
```

	Red Neuronal. Perceptrón Multicapa						
	AUC	Accuracy	Aciertos Clase SI	Aciertos Clase NO	Kappa	Sensitivity	Specificity

Red Neuronal. Perceptrón Multicapa							
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
Datos Entrenamiento	0.926	0.856	0.899	0.821	0.712	0.802	0.910
Datos Validación	0.926	0.851	0.897	0.815	0.703	0.793	0.909

```
importancia_var(nnet_train, "Perceptrón multicapa")
```



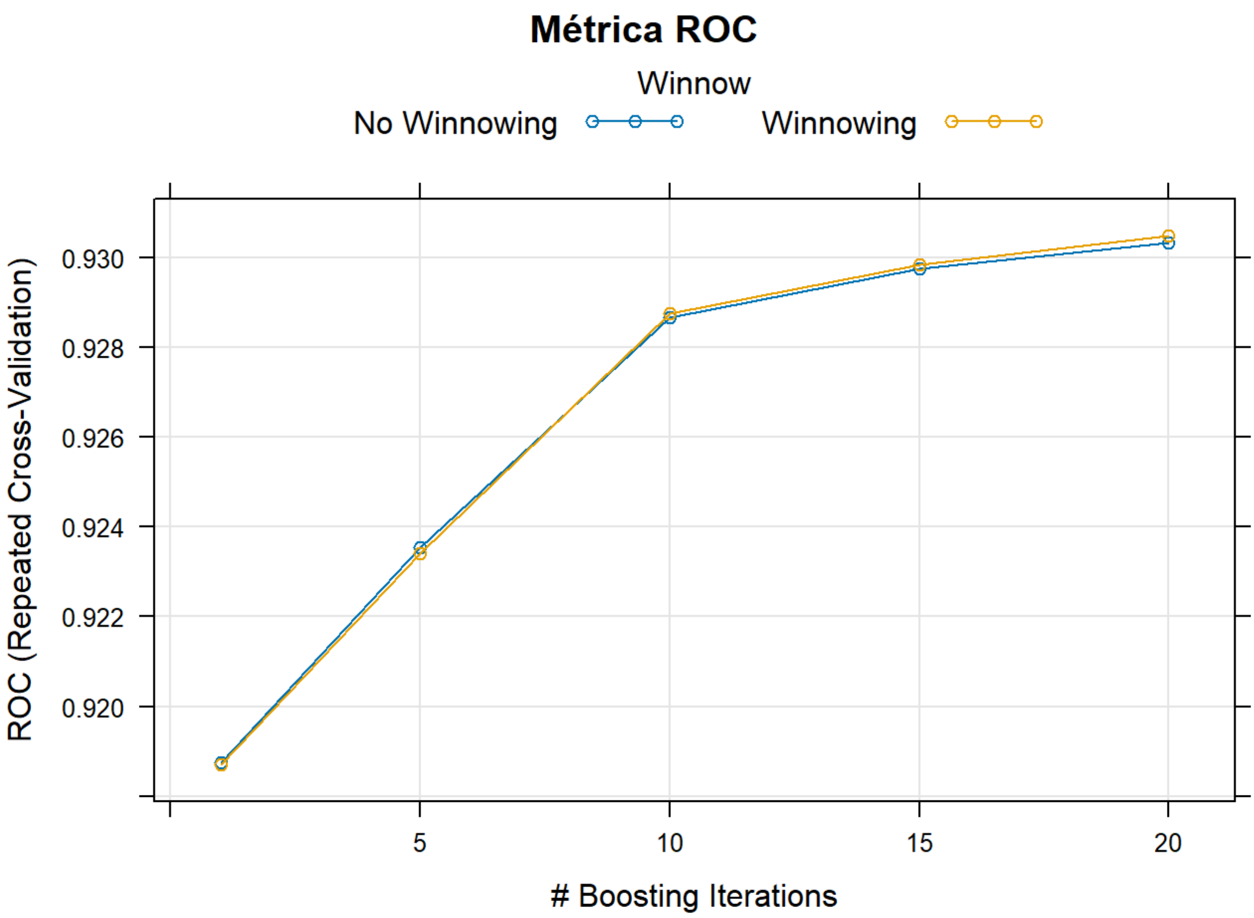
2.3.4. Árbol C5.0

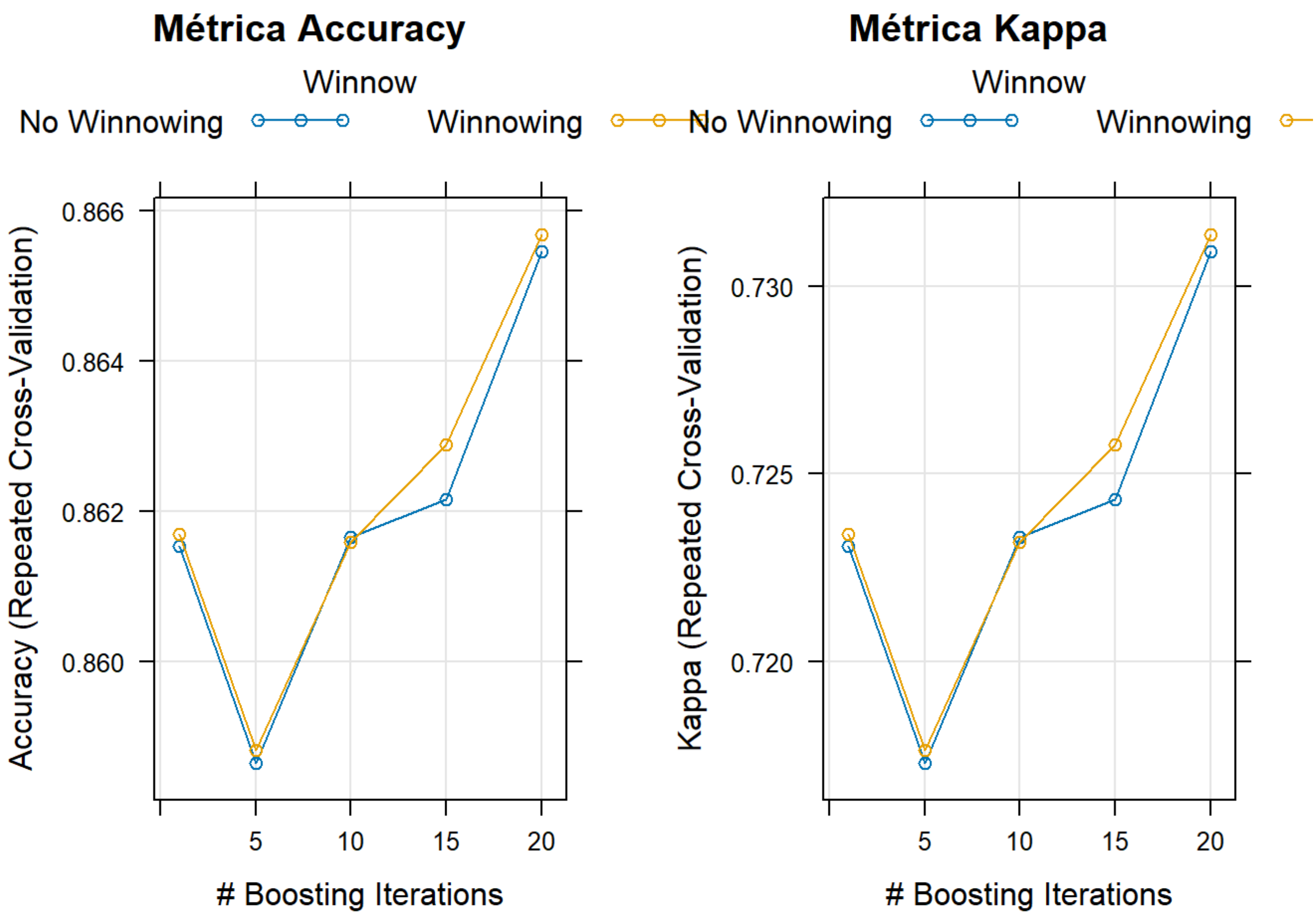
```r
# Entrenamiento
 if (train_switch  == 1) {
set.seed(7)

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

  stopCluster(clusterCPU)
  clusterCPU <- NULL

  saveRDS(C5_train, "Models/C5_train.RDS")

toc()

}else{
  C5_train <- readRDS("Models/C5_train.RDS")
}
```

```r
C5_train
```

```
## C5.0
##
## 87870 samples
##     7 predictor
##     2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 76887, 76886, 76886, 76887, 76886, 76886, ...
## Resampling results across tuning parameters:
##
##   winnow  trials  ROC        Sens       Spec       Accuracy   Kappa
##   FALSE    1      0.9187377  0.8134176  0.9096619  0.8615398  0.7230795
##   FALSE    5      0.9235430  0.8181177  0.8991694  0.8586435  0.7172871
##   FALSE   10      0.9286628  0.8170480  0.9062706  0.8616593  0.7233186
##   FALSE   15      0.9297690  0.8158302  0.9084786  0.8621543  0.7243087
##   FALSE   20      0.9303298  0.8194264  0.9114943  0.8654603  0.7309207
##    TRUE    1      0.9187033  0.8133834  0.9100147  0.8616991  0.7233982
##    TRUE    5      0.9234188  0.8168544  0.9007852  0.8588199  0.7176397
##    TRUE   10      0.9287647  0.8187436  0.9044270  0.8615853  0.7231706
##    TRUE   15      0.9298416  0.8152612  0.9105042  0.8628827  0.7257654
##    TRUE   20      0.9304996  0.8180380  0.9133379  0.8656879  0.7313759
##
## Tuning parameter 'model' was held constant at a value of tree
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were trials = 20, model = tree and winnow
##  = TRUE.
```

```
grafico_metricas(C5_train)
```






```
resultados(C5_train, "Árbol C5")
```

RESULTADOS DEL MODELO Árbol C5

model	winnow	trials	ROC	Sens	Spec	Accuracy	Kappa	ROCSD	
tree	FALSE	1	0.9187377	0.8134176	0.9096619	0.8615398	0.7230795	0.0034533	0
tree	TRUE	1	0.9187033	0.8133834	0.9100147	0.8616991	0.7233982	0.0034385	0
tree	FALSE	5	0.9235430	0.8181177	0.8991694	0.8586435	0.7172871	0.0034317	0
tree	TRUE	5	0.9234188	0.8168544	0.9007852	0.8588199	0.7176397	0.0033801	0
tree	FALSE	10	0.9286628	0.8170480	0.9062706	0.8616593	0.7233186	0.0025735	0
tree	TRUE	10	0.9287647	0.8187436	0.9044270	0.8615853	0.7231706	0.0025654	0
tree	FALSE	15	0.9297690	0.8158302	0.9084786	0.8621543	0.7243087	0.0024477	0
tree	TRUE	15	0.9298416	0.8152612	0.9105042	0.8628827	0.7257654	0.0024303	0
tree	FALSE	20	0.9303298	0.8194264	0.9114943	0.8654603	0.7309207	0.0025655	0
tree	TRUE	20	0.9304996	0.8180380	0.9133379	0.8656879	0.7313759	0.0025366	0

```
mejor_modelo(C5_train)
```

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

	trials	model	winnow
10	20	tree	TRUE

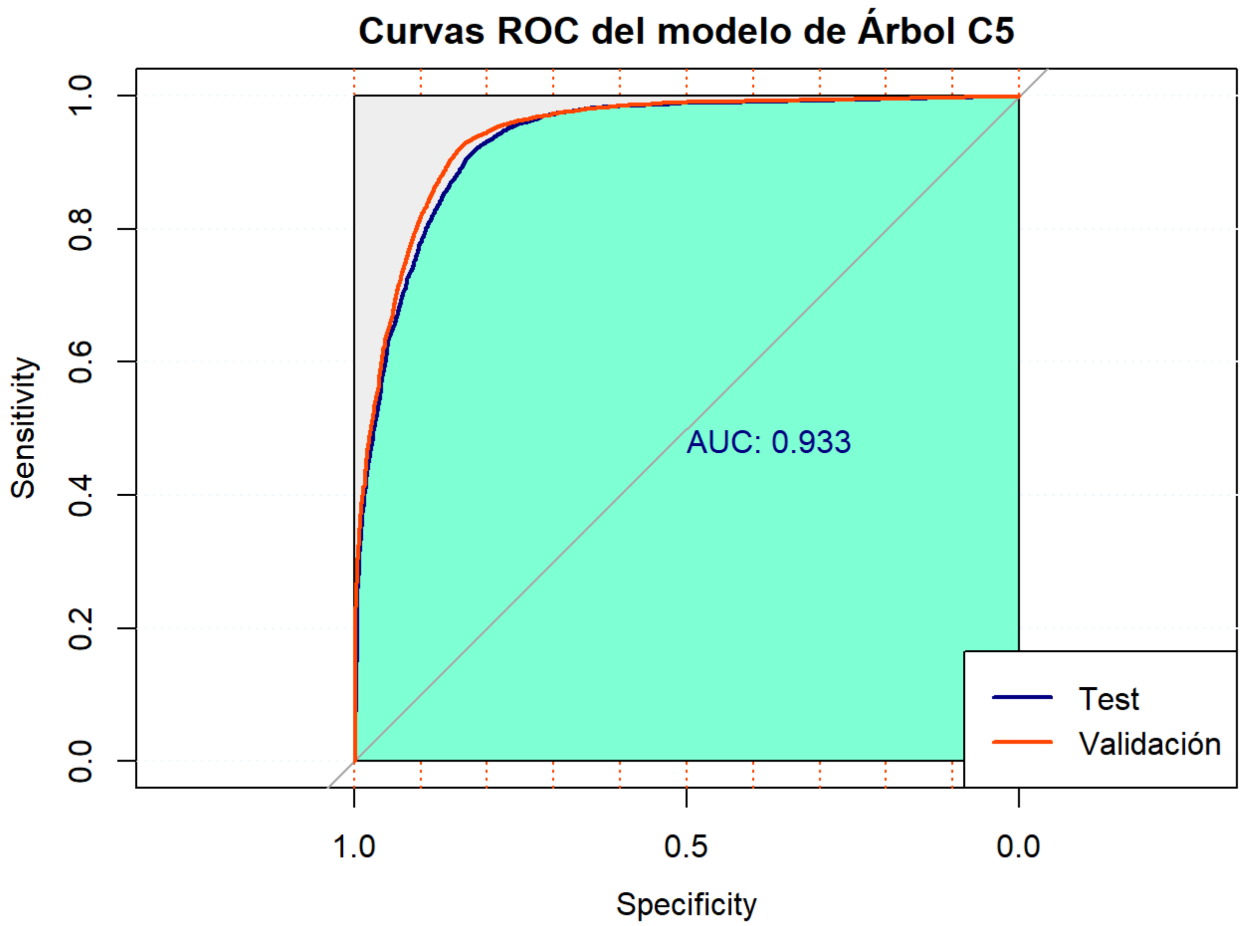
```
curvas_ROC(C5_train, "de Árbol C5", train_general, test_general)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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




```
## [1] "ROC del modelo con el fichero de test: 0.933488830946296"
```

```
validation(C5_train, "de Árbol C5", train_general, test_general)
```

```
## [1] "Modelo de Árbol C5 - Tabla de confusión para los datos de entrenamiento"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  36948  3437
##        Yes  6987 40498
##
##                Accuracy : 0.8814
##                  95% CI : (0.8792, 0.8835)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7627
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8410
##             Specificity : 0.9218
##          Pos Pred Value : 0.9149
##          Neg Pred Value : 0.8529
##              Prevalence : 0.5000
##          Detection Rate : 0.4205
##    Detection Prevalence : 0.4596
##       Balanced Accuracy : 0.8814
##
##        'Positive' Class : No
##
## [1] "Modelo de Árbol C5 - Tabla de confusión para los datos de validación"
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    No   Yes
##        No   9061   971
##        Yes  1922 10012
## 
##                Accuracy : 0.8683
##                  95% CI : (0.8638, 0.8727)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.7366
## 
##  Mcnemar's Test P-Value : < 2.2e-16
## 
##             Sensitivity : 0.8250
##             Specificity : 0.9116
##          Pos Pred Value : 0.9032
##          Neg Pred Value : 0.8389
##              Prevalence : 0.5000
##          Detection Rate : 0.4125
##    Detection Prevalence : 0.4567
##       Balanced Accuracy : 0.8683
## 
##        'Positive' Class : No
## 
```

```
resumen_C5 <- resumen(C5_train, train_general, test_general)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

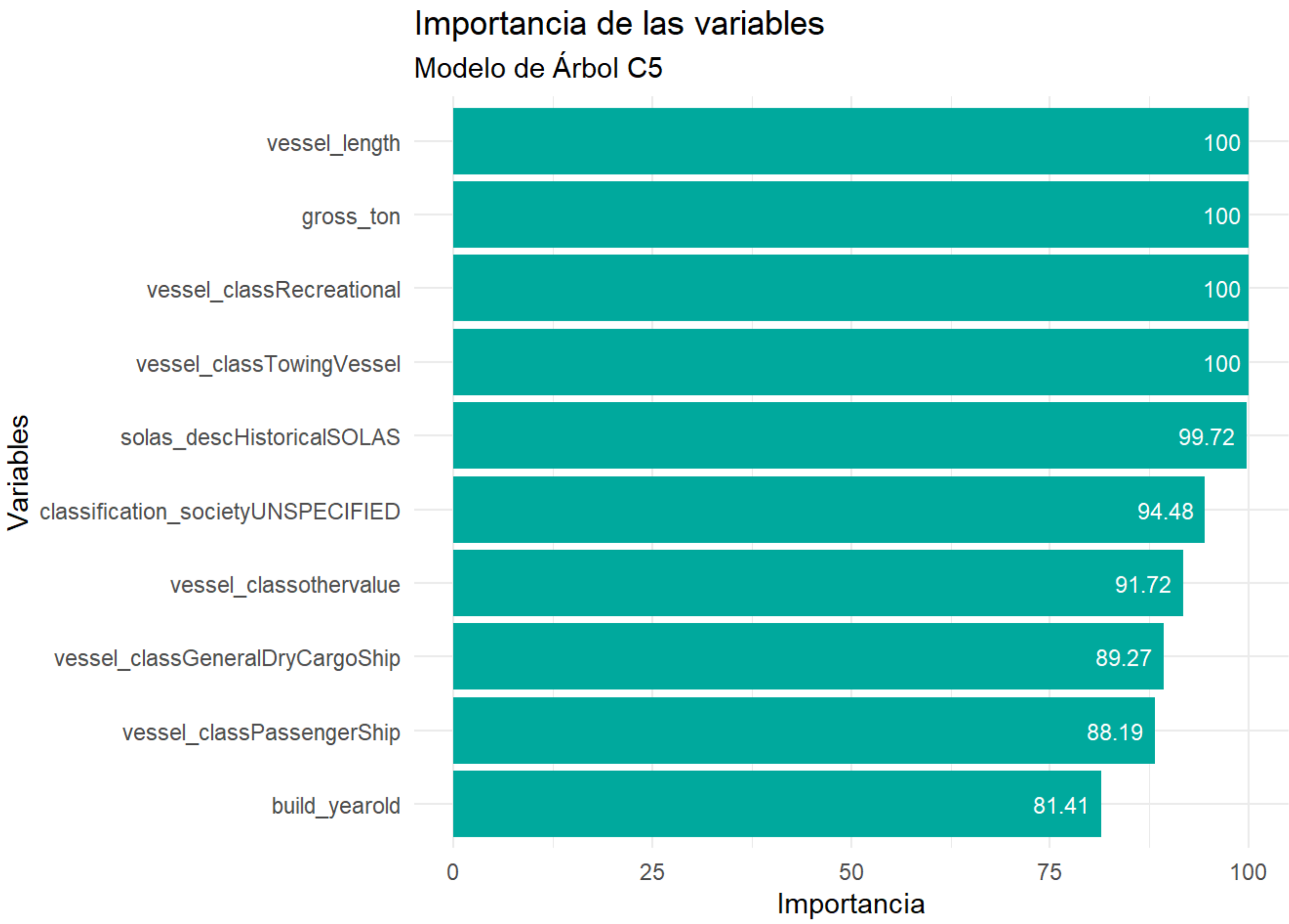
```
## Setting direction: controls < cases
```

```
resumen_C5 %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ",
            "Árbol C5 " = 7))
```

	Árbol C5						
	AUC	Accuracy	Aciertos Clase SI	Aciertos Clase NO	Kappa	Sensitivity	Specificity

	Árbol C5						
	AUC	Accuracy	Aciertos Clase SI	Aciertos Clase NO	Kappa	Sensitivity	Specificity
Datos Entrenamiento	0.941	0.881	0.915	0.853	0.763	0.841	0.922
Datos Validación	0.933	0.868	0.903	0.839	0.737	0.825	0.912

```
importancia_var(C5_train, "de Árbol C5")
```



2.3.5. Regresión logística

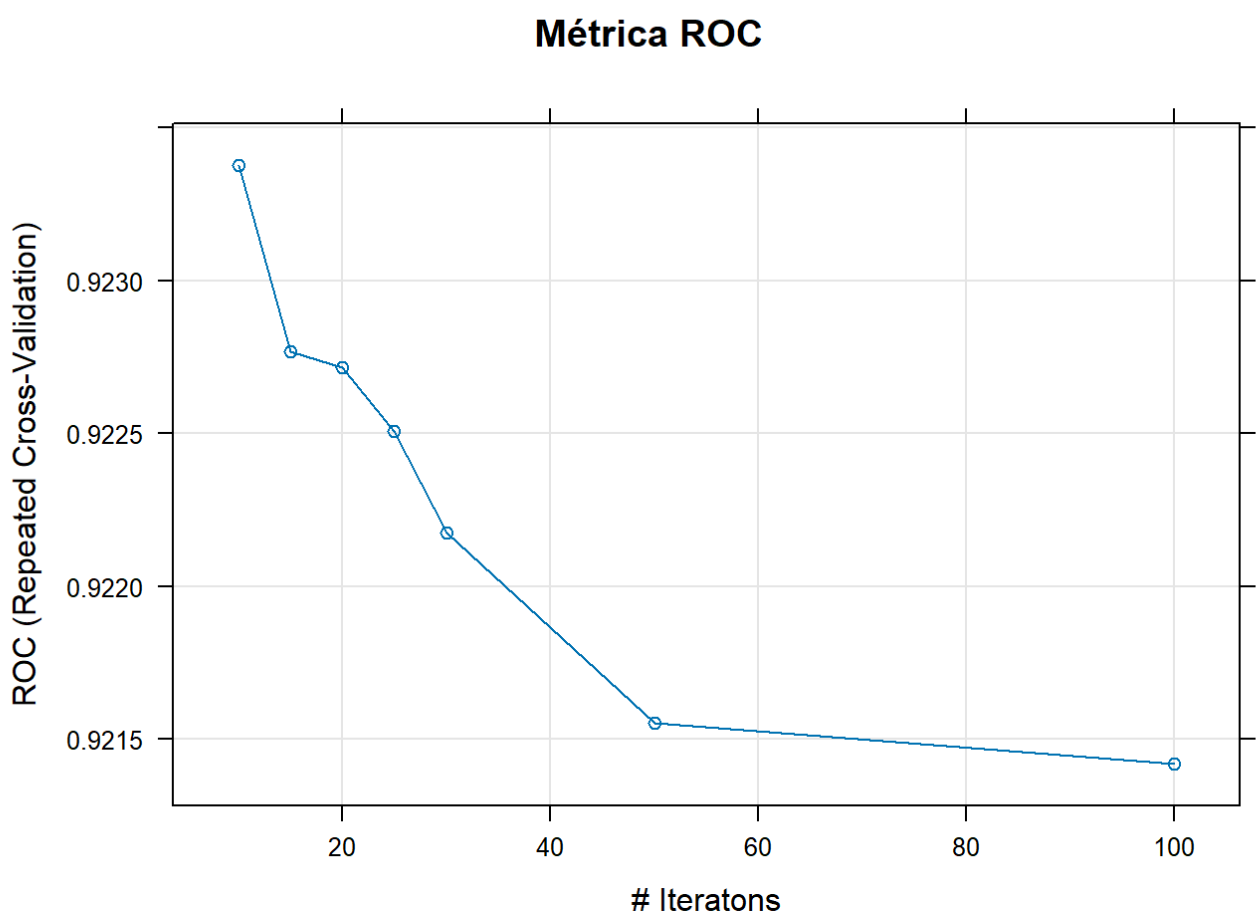
```r
# Nota:
# El entrenamiento de este modelo (rl_train) no se exporta correctamente.
# Al cargarlo de una sesión anterior, produce fallos en varios apartados.
# Es conveniente ejecutar este entrenamiento SIEMPRE en cada sesión o renderizado.

# Entrenamiento
if (is.numeric(train_switch)) {
set.seed(7)

tic()

grid_lmt <- expand.grid(iter = c(10, 15, 20, 25, 30, 50, 100))

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

rl_train <- train(y~.,
                  data = train_general,
                  method = 'LMT',
                  metric = metrica,
                  tuneGrid = grid_lmt,
                  trControl = control)

stopCluster(clusterCPU)
clusterCPU <- NULL

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

toc()

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

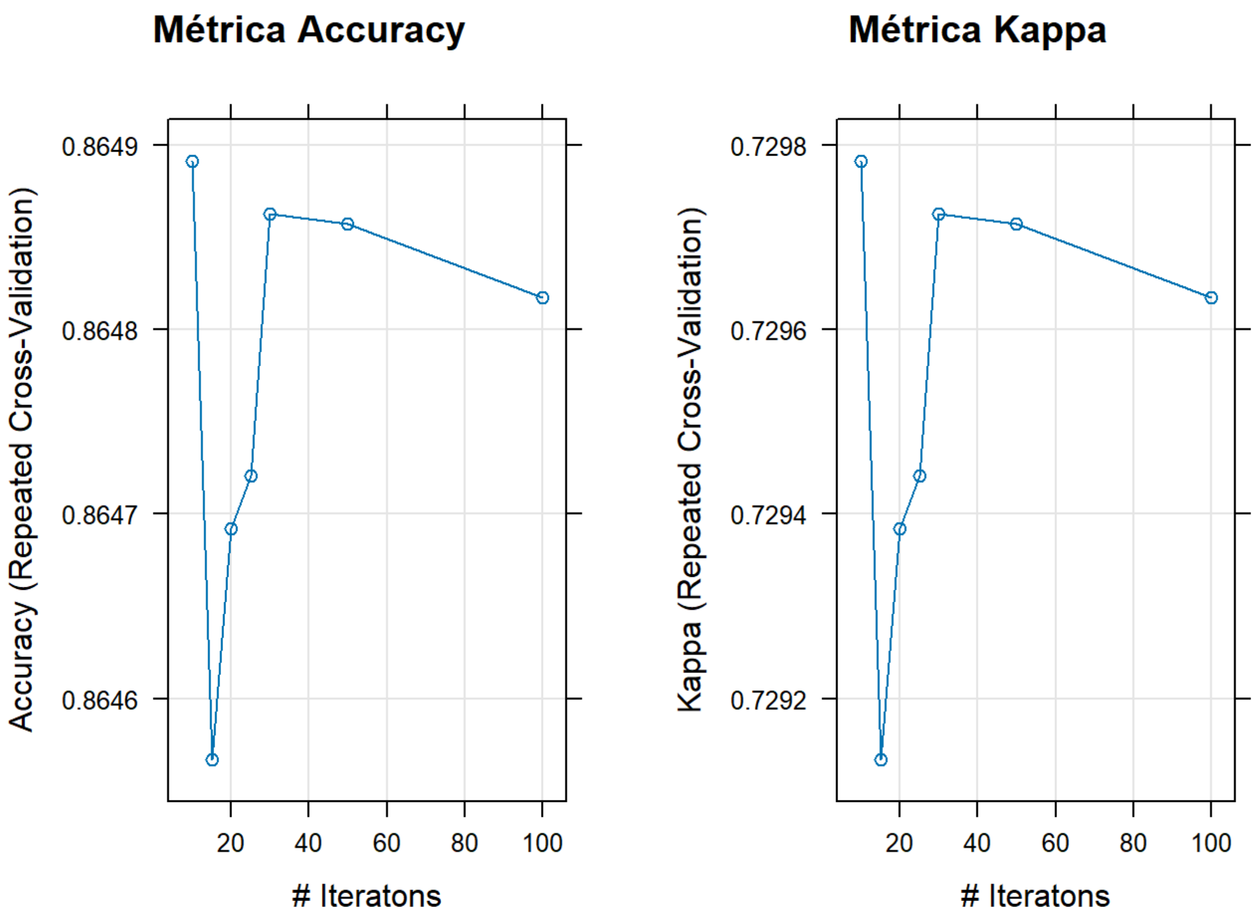
```
## Aggregating results
## Selecting tuning parameters
## Fitting iter = 10 on full training set
## 1288.44 sec elapsed
```

```r
rl_train
```

```
## Logistic Model Trees
##
## 87870 samples
##     7 predictor
##     2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (8 fold, repeated 2 times)
## Summary of sample sizes: 76887, 76886, 76886, 76887, 76886, 76886, ...
## Resampling results across tuning parameters:
##
##   iter  ROC        Sens       Spec       Accuracy   Kappa
##    10   0.9233779  0.8205531  0.9092295  0.8648913  0.7297826
##    15   0.9227672  0.8209515  0.9081825  0.8645669  0.7291339
##    20   0.9227168  0.8208035  0.9085808  0.8646921  0.7293843
##    25   0.9225091  0.8210083  0.9084328  0.8647206  0.7294412
##    30   0.9221742  0.8211222  0.9086035  0.8648628  0.7297257
##    50   0.9215541  0.8211222  0.9085922  0.8648571  0.7297143
##   100   0.9214197  0.8212474  0.9083873  0.8648173  0.7296346
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was iter = 10.
```

```
grafico_metricas(rl_train)
```






```
resultados(rl_train , "Regresión Logística")
```

RESULTADOS DEL MODELO Regresión Logística

iter	ROC	Sens	Spec	Accuracy	Kappa	ROCSD	SensSD	SpecSD
10	0.9233779	0.8205531	0.9092295	0.8648913	0.7297826	0.0032017	0.0067664	0.005131
15	0.9227672	0.8209515	0.9081825	0.8645669	0.7291339	0.0038093	0.0060470	0.005129
20	0.9227168	0.8208035	0.9085808	0.8646921	0.7293843	0.0034372	0.0066772	0.004414
25	0.9225091	0.8210083	0.9084328	0.8647206	0.7294412	0.0039273	0.0060173	0.004549
30	0.9221742	0.8211222	0.9086035	0.8648628	0.7297257	0.0040987	0.0059821	0.004868
50	0.9215541	0.8211222	0.9085922	0.8648571	0.7297143	0.0043242	0.0059259	0.005071
100	0.9214197	0.8212474	0.9083873	0.8648173	0.7296346	0.0044219	0.0058860	0.004863

```
mejor_modelo(rl_train)
```

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

iter

iter
10

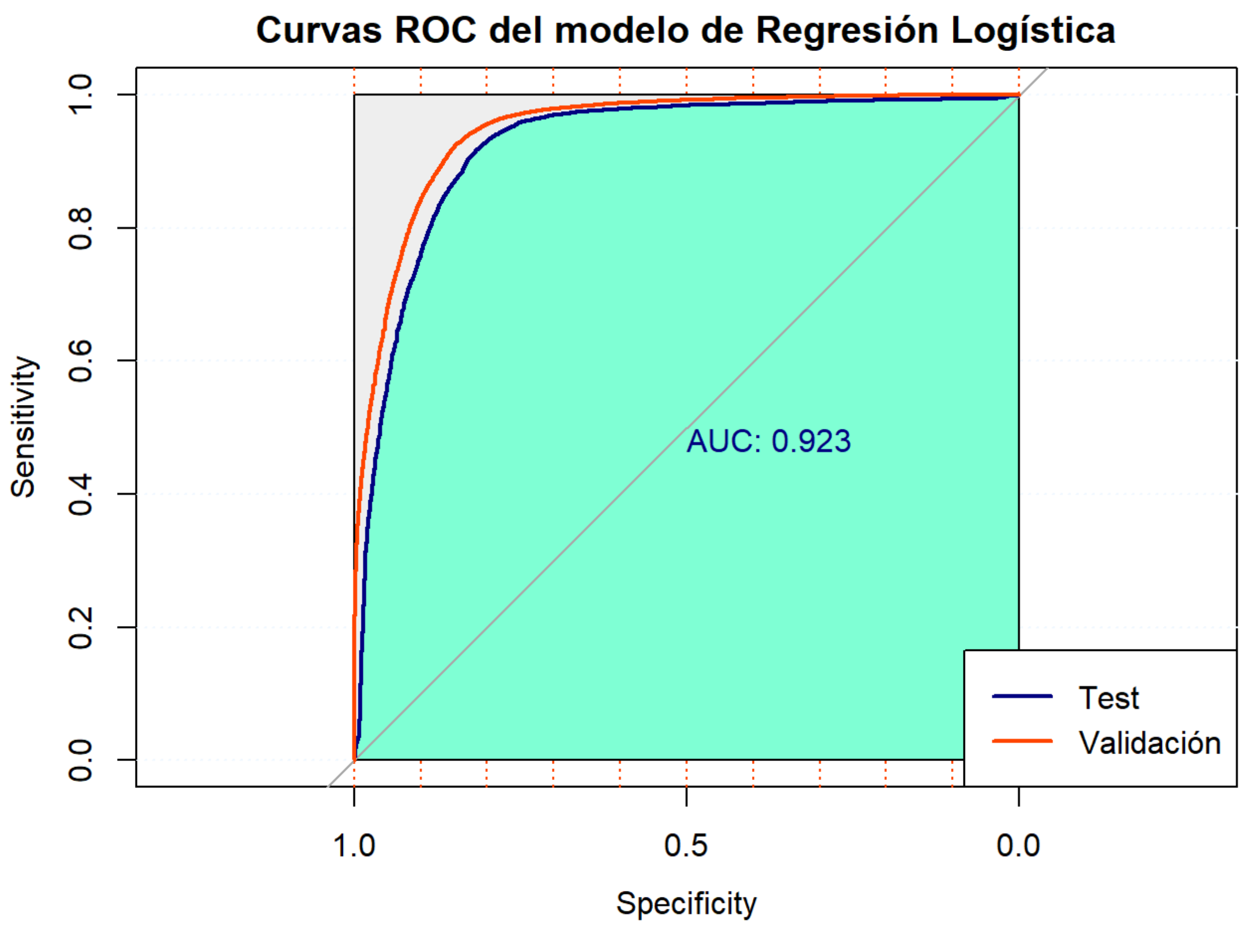
```
curvas_ROC(rl_train, "de Regresión Logística", train_general, test_general)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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




```
## [1] "ROC del modelo con el fichero de test: 0.922996109911"
```

```
# Nota: Se producirá un Error in .jcall("RWekaInterfaces" si se carga un rl_train previo
```

```
validation(rl_train, "de Regresión Logística", train_general, test_general)
```

```
## [1] "Modelo de Regresión Logística - Tabla de confusión para los datos de entrenamient
o"
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    No   Yes
##        No  37184  3247
##        Yes  6751 40688
##
##                Accuracy : 0.8862
##                  95% CI : (0.8841, 0.8883)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7724
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.8463
##             Specificity : 0.9261
##          Pos Pred Value : 0.9197
##          Neg Pred Value : 0.8577
##              Prevalence : 0.5000
##          Detection Rate : 0.4232
##    Detection Prevalence : 0.4601
##       Balanced Accuracy : 0.8862
##
##        'Positive' Class : No
##
## [1] "Modelo de Regresión Logística - Tabla de confusión para los datos de validación"
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    No   Yes
##        No   9026   976
##        Yes  1957 10007
##                                           
##                Accuracy : 0.8665          
##                  95% CI : (0.8619, 0.8709)
##     No Information Rate : 0.5             
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.733           
##                                           
##  Mcnemar's Test P-Value : < 2.2e-16       
##                                           
##             Sensitivity : 0.8218          
##             Specificity : 0.9111          
##          Pos Pred Value : 0.9024          
##          Neg Pred Value : 0.8364          
##              Prevalence : 0.5000          
##          Detection Rate : 0.4109          
##    Detection Prevalence : 0.4553          
##       Balanced Accuracy : 0.8665          
##                                           
##        'Positive' Class : No              
## 
```

```
# Nota: Se producirá un Error in .jcall("RWekaInterfaces" si se carga un rl_train previo
```

```
resumen_rl <- resumen(rl_train, train_general, test_general)
```

```
## Setting levels: control = No, case = Yes
```

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

```
## Setting levels: control = No, case = Yes
```

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

```
# Nota: Se producirá un Error in .jcall("RWekaInterfaces" si se carga un rl_train previo
```

```
resumen_rl %>% kable(escape = F) %>%
  kable_styling("hover", full_width = F) %>%
  add_header_above(c(" ", "Naïve Bayes Classifier" = 7))
```

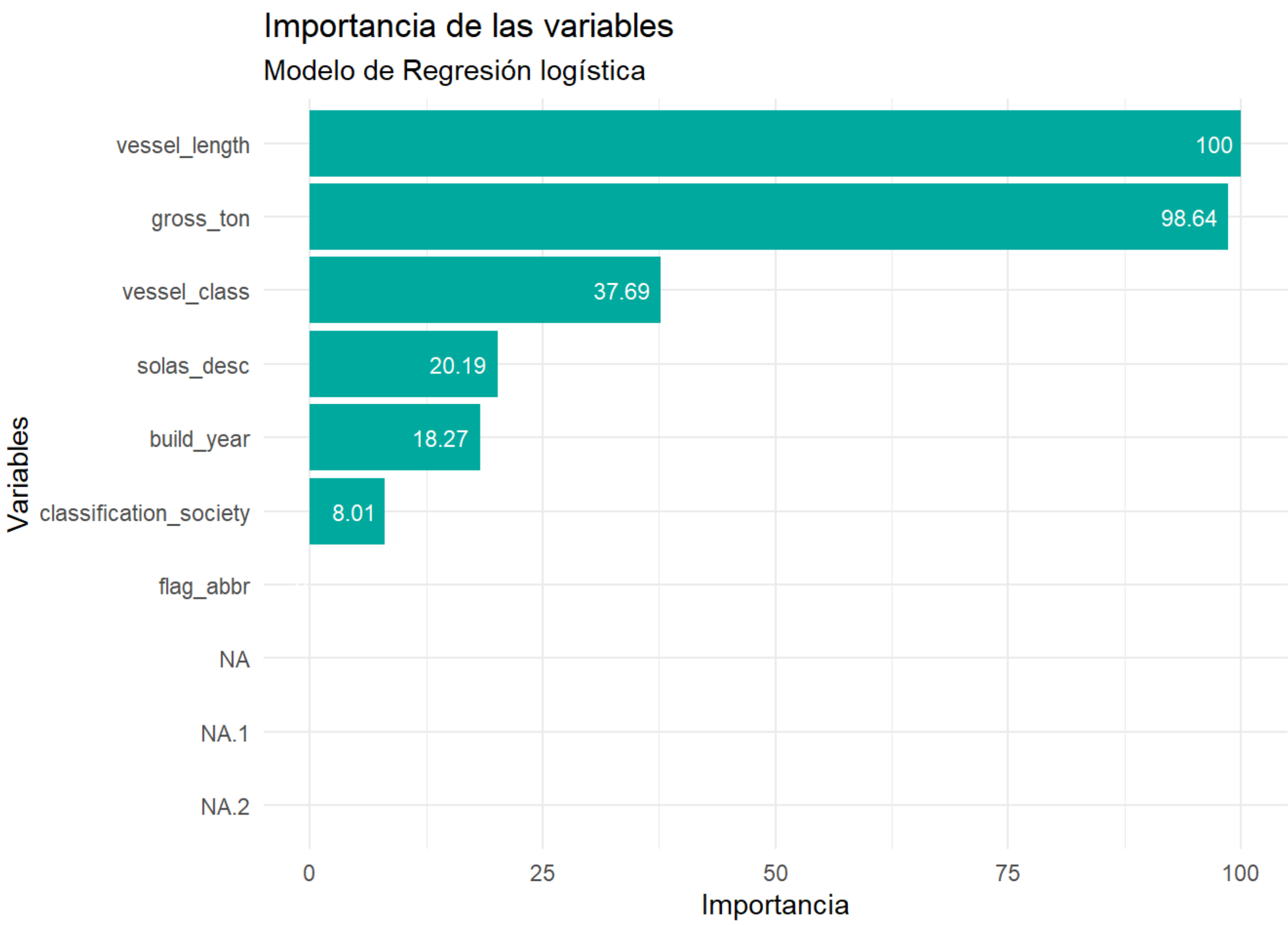
Naïve Bayes Classifier

	AUC	Accuracy	Aciertos Clase SI	Aciertos Clase NO	Kappa	Sensitivity	Specificity
Datos Entrenamiento	0.948	0.886	0.920	0.858	0.772	0.846	0.926
Datos Validación	0.923	0.866	0.902	0.836	0.733	0.822	0.911

```
importancia_var(rl_train, "de Regresión logística ")
```

```
## Warning: Removed 3 rows containing missing values (`position_stack()`).
## Removed 3 rows containing missing values (`position_stack()`).
```

```
## Warning: Removed 3 rows containing missing values (`geom_text()`).
```



3. Comparación de los modelos

3.1. Importancia de las variables

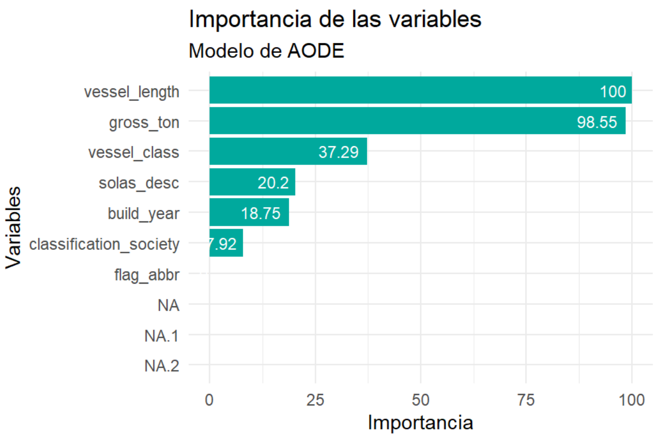
```
ggarrange(importancia_var(aode_train, "de AODE"),
          importancia_var(XGBFinal_train, "de Extreme Gradient Boosting"),
          importancia_var(rf_train, "de Random Forest"),
          importancia_var(nnet_train, "de Red Neuronal"),
          importancia_var(C5_train, "de C5"),
          importancia_var(rl_train, "de Regresión logística"),
          ncol=2,nrow=3)
```

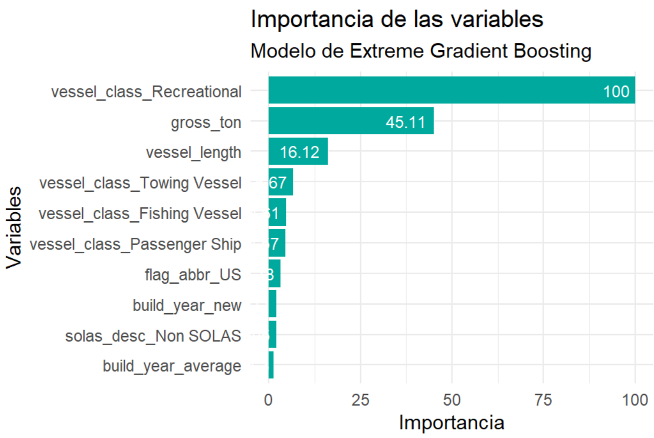
```
## Warning: Removed 3 rows containing missing values (`position_stack()`).
## Removed 3 rows containing missing values (`position_stack()`).
```

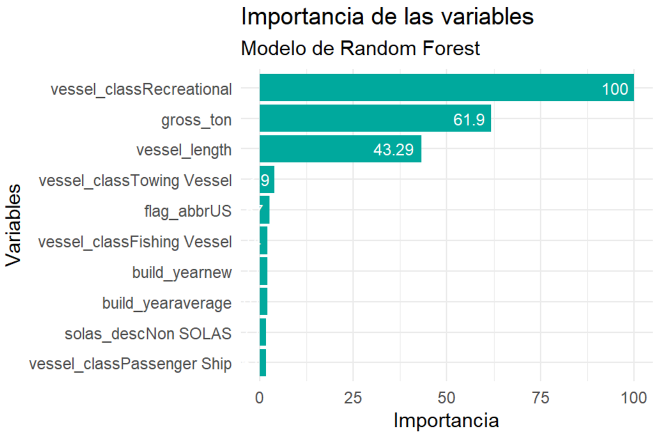
```
## Warning: Removed 3 rows containing missing values (`geom_text()`).
```

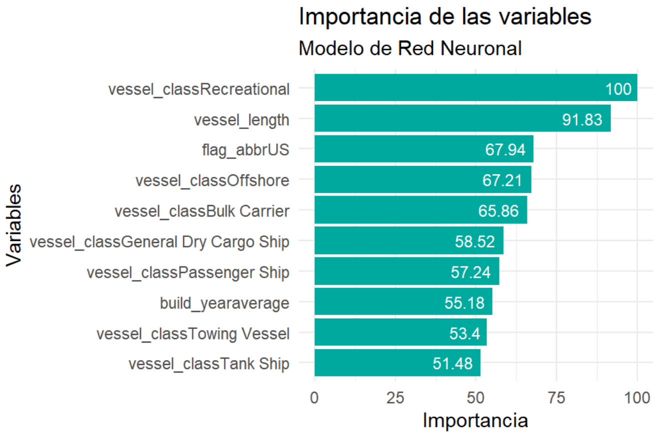
```
## Warning: Removed 3 rows containing missing values (`position_stack()`).
## Removed 3 rows containing missing values (`position_stack()`).
```

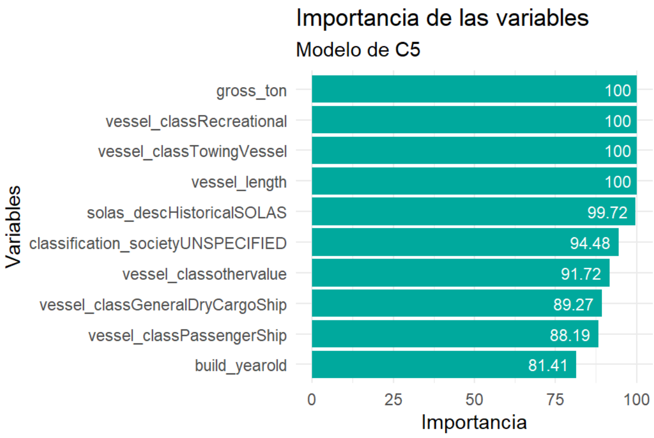
```
## Warning: Removed 3 rows containing missing values (`geom_text()`).
```

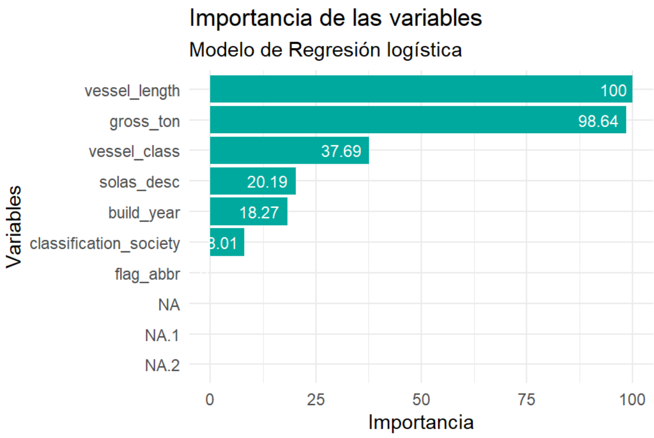












3.2. Desempeño de los modelos

```
# Los dos cuadros con los algoritmos utilizados los construimos uniendo la salida de la fu
nción resumen
Nombresmodelos <- c("NB", "TAN", "TANSE", "TANHC", "AODE", "GBM", "XGB", "RF", "SVM", "ML
P","C5","RL")

# Para los datos de entrenamiento
DatosEntrenamiento <- rbind(resumen_nb[1,], resumen_tan[1,], resumen_tanse[1,], resumen_ta
nhc[1,], resumen_AODE[1,], resumen_GBM[1,], resumen_XGBFinal[1,], resumen_rf[1,], resumen_
svm[1,], resumen_nnet[1,], resumen_C5[1,], resumen_rl[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" = 7))
```

	Comparación con la Muestra de Entrenamiento						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
RF	0.960	0.946	0.966	0.927	0.892	0.924	0.968
RL	0.948	0.886	0.920	0.858	0.772	0.846	0.926
XGB	0.947	0.881	0.920	0.849	0.762	0.835	0.927
C5	0.941	0.881	0.915	0.853	0.763	0.841	0.922
GBM	0.933	0.865	0.904	0.832	0.729	0.816	0.913
MLP	0.926	0.856	0.899	0.821	0.712	0.802	0.910
TAN	0.924	0.855	0.879	0.833	0.709	0.822	0.887
AODE	0.924	0.849	0.871	0.830	0.698	0.819	0.879
SVM	0.915	0.835	0.925	0.777	0.671	0.730	0.941
NB	0.894	0.825	0.847	0.805	0.649	0.793	0.856

	Comparación con la Muestra de Entrenamiento						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
TANSE	0.894	0.825	0.846	0.806	0.649	0.794	0.856
TANHC	0.894	0.825	0.846	0.806	0.649	0.794	0.856

```
# Para los datos de validación
DatosValidación <- rbind(resumen_nb[2,], resumen_tan[2,], resumen_tanse[2,], resumen_tanhc
[2,], resumen_AODE[2,], resumen_GBM[2,], resumen_XGBFinal[2,], resumen_rf[2,], resumen_svm
[2,], resumen_nnet[2,], resumen_C5[2,], resumen_rl[2,])

rownames(DatosValidación) <- Nombresmodelos
DatosValidación <-as.data.frame(DatosValidación)

DatosValidación %>% 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 Validacion" = 7))
```

	Comparación con la Muestra de Validacion						
	AUC	**Accuracy**	**Aciertos Clase SI**	**Aciertos Clase NO**	**Kappa**	**Sensitivity**	**Specificity**
XGB	0.949	0.883	0.923	0.850	0.766	0.835	0.931
GBM	0.935	0.865	0.907	0.830	0.729	0.813	0.917
RF	0.933	0.884	0.921	0.852	0.768	0.839	0.928
C5	0.933	0.868	0.903	0.839	0.737	0.825	0.912
TAN	0.926	0.855	0.880	0.833	0.710	0.822	0.888
AODE	0.926	0.853	0.878	0.831	0.706	0.820	0.886
MLP	0.926	0.851	0.897	0.815	0.703	0.793	0.909
RL	0.923	0.866	0.902	0.836	0.733	0.822	0.911

Comparación con la Muestra de Validacion							
	AUC	Accuracy	Aciertos Clase SI	Aciertos Clase NO	Kappa	Sensitivity	Specificity
SVM	0.915	0.832	0.926	0.771	0.663	0.721	0.943
NB	0.896	0.828	0.852	0.808	0.657	0.795	0.862
TANSE	0.896	0.829	0.852	0.808	0.657	0.796	0.861
TANHC	0.896	0.829	0.852	0.808	0.657	0.796	0.861

3.3. Contraste de hipótesis

```
modelos <- list(RL= rl_train,NB = nb_train, TAN = tan_train, TANSE = tanse_train, TANHC =
tanhc_train, AODE= aode_train, GBM = GBM_train, XGBTree = XGBFinal_train, RF = rf_train,
MLP = nnet_train, SVM = svm_train, C5 = C5_train)

comp_modelos <- resamples(modelos)
comp_modelos
```

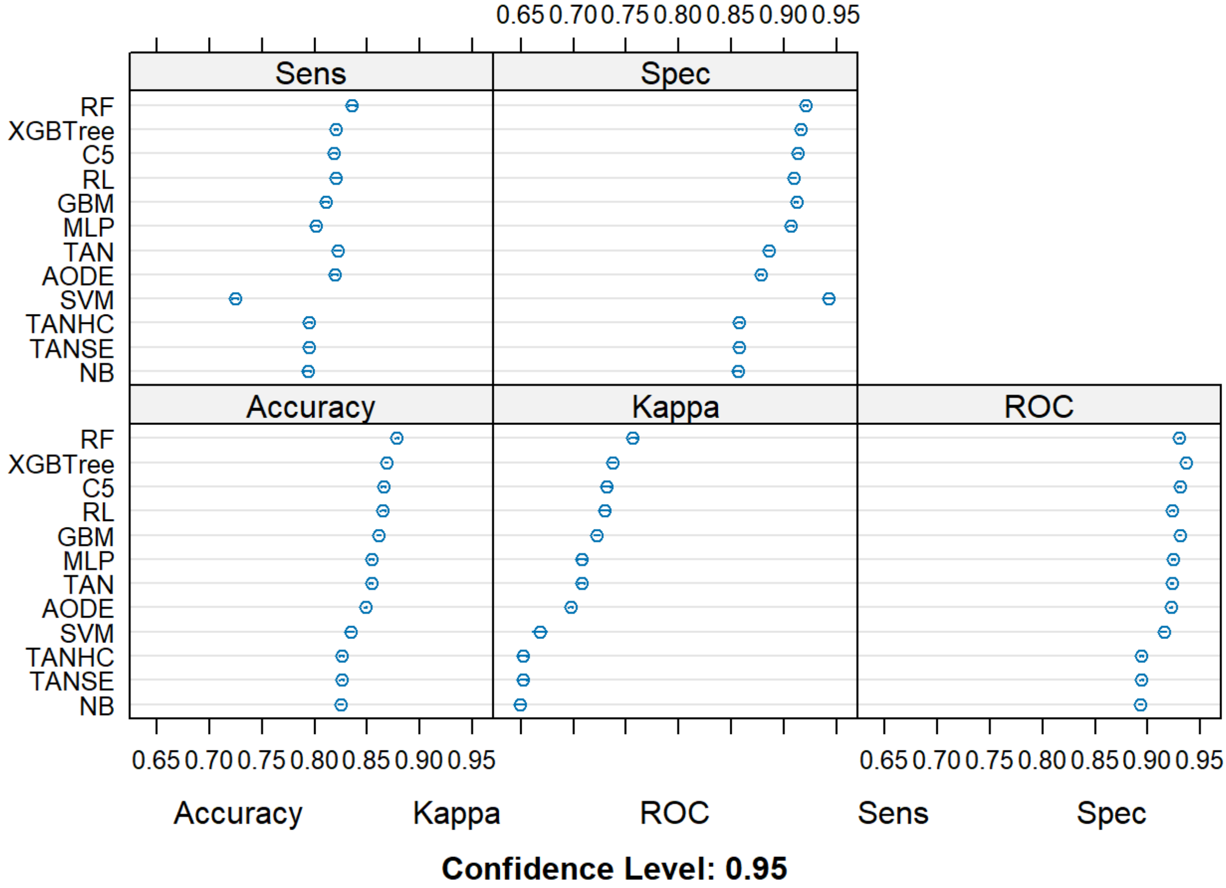
```
##
## Call:
## resamples.default(x = modelos)
##
## Models: RL, NB, TAN, TANSE, TANHC, AODE, GBM, XGBTree, RF, MLP, SVM, C5
## Number of resamples: 16
## Performance metrics: Accuracy, Kappa, ROC, Sens, Spec
## Time estimates for: everything, final model fit
```

```
summary(comp_modelos)
```

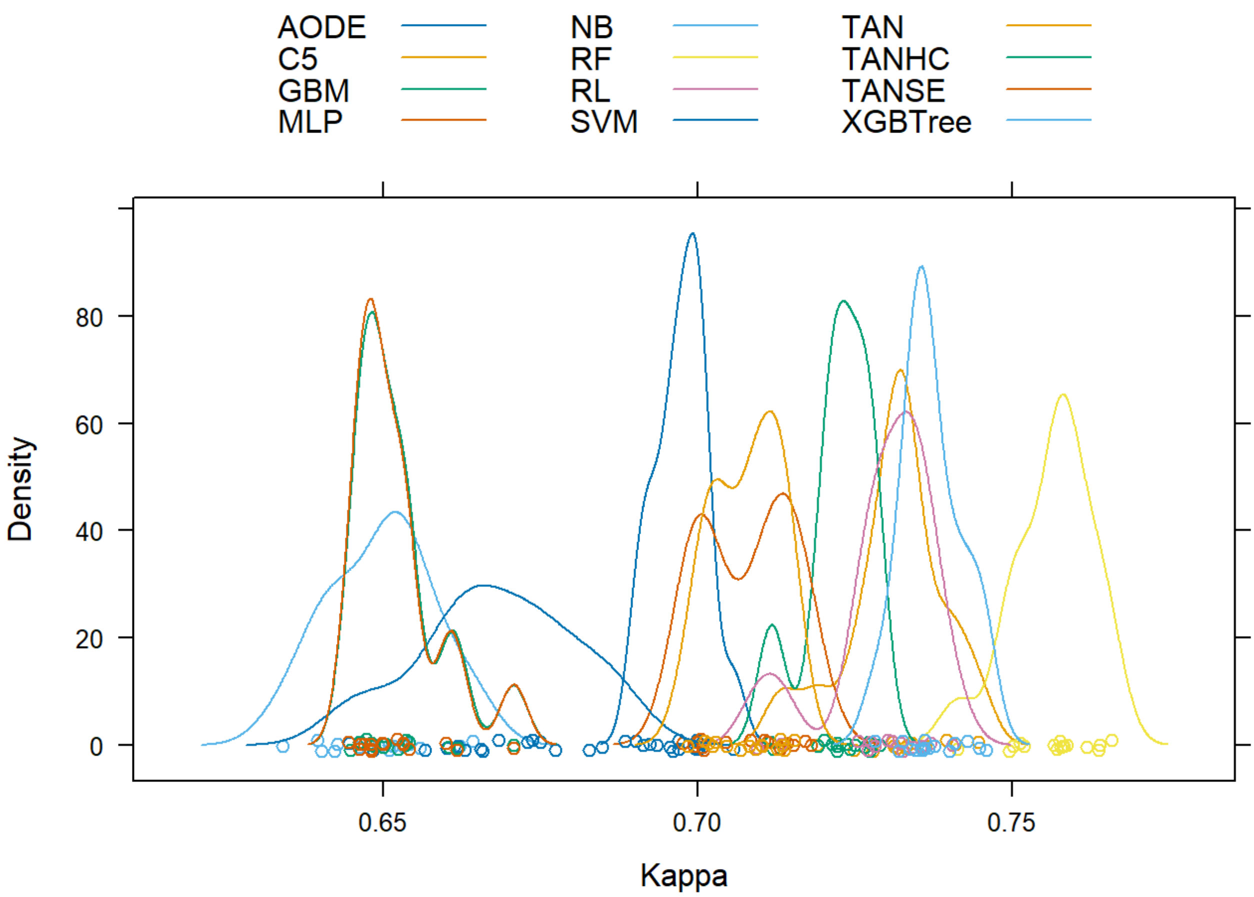
```
##
## Call:
## summary.resamples(object = comp_modelos)
##
## Models: RL, NB, TAN, TANSE, TANHC, AODE, GBM, XGBTree, RF, MLP, SVM, C5
## Number of resamples: 16
##
## Accuracy
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## RL      0.8547756 0.8637336 0.8658048 0.8648913 0.8677394 0.8702658    0
## NB      0.8170066 0.8213083 0.8254655 0.8248208 0.8273170 0.8321194    0
## TAN     0.8492215 0.8513713 0.8543791 0.8538978 0.8564048 0.8577021    0
## TANSE   0.8222870 0.8238989 0.8249647 0.8260555 0.8267589 0.8353969    0
## TANHC   0.8223780 0.8239900 0.8251013 0.8261522 0.8268955 0.8353969    0
## AODE    0.8451384 0.8475247 0.8486890 0.8486912 0.8500057 0.8528769    0
## GBM     0.8557902 0.8605472 0.8614740 0.8613121 0.8634150 0.8645302    0
## XGBTree 0.8641537 0.8672584 0.8680808 0.8685842 0.8701748 0.8729971    0
## RF      0.8707093 0.8759104 0.8788693 0.8782804 0.8797770 0.8829206    0
## MLP     0.8488711 0.8503733 0.8547822 0.8539377 0.8569510 0.8590677    0
## SVM     0.8224044 0.8308288 0.8336748 0.8341584 0.8378842 0.8443332    0
## C5      0.8566096 0.8639840 0.8661235 0.8656879 0.8673495 0.8723598    0
##
## Kappa
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## RL      0.7095534 0.7274672 0.7316096 0.7297826 0.7354789 0.7405317    0
## NB      0.6340131 0.6426165 0.6509299 0.6496417 0.6546340 0.6642389    0
## TAN     0.6984448 0.7027432 0.7087582 0.7077956 0.7128095 0.7154042    0
## TANSE   0.6445739 0.6477984 0.6499305 0.6521111 0.6535168 0.6707939    0
## TANHC   0.6447560 0.6479805 0.6502036 0.6523045 0.6537899 0.6707939    0
## AODE    0.6902768 0.6950499 0.6973780 0.6973825 0.7000111 0.7057538    0
## GBM     0.7115805 0.7210943 0.7229493 0.7226243 0.7268299 0.7290605    0
## XGBTree 0.7283051 0.7345174 0.7361617 0.7371685 0.7403496 0.7459942    0
## RF      0.7414206 0.7518208 0.7577385 0.7565608 0.7595535 0.7658412    0
## MLP     0.6977422 0.7007465 0.7095630 0.7078754 0.7139020 0.7181355    0
## SVM     0.6446080 0.6614575 0.6671607 0.6681172 0.6755831 0.6884445    0
## C5      0.7132192 0.7279680 0.7322469 0.7313759 0.7346984 0.7447196    0
##
## ROC
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## RL      0.9153282 0.9212803 0.9245943 0.9233779 0.9255961 0.9269482    0
## NB      0.8885531 0.8922260 0.8937159 0.8934897 0.8946761 0.8978968    0
## TAN     0.9204292 0.9219766 0.9231702 0.9233983 0.9250089 0.9258626    0
## TANSE   0.8905105 0.8929672 0.8936389 0.8942328 0.8963616 0.8984764    0
## TANHC   0.8905627 0.8930342 0.8936596 0.8942812 0.8964572 0.8983329    0
## AODE    0.9192975 0.9220464 0.9233638 0.9228257 0.9241768 0.9250107    0
## GBM     0.9279521 0.9295587 0.9309064 0.9307190 0.9321061 0.9326922    0
## XGBTree 0.9335461 0.9355622 0.9366212 0.9363417 0.9370810 0.9393523    0
## RF      0.9256101 0.9283364 0.9301070 0.9301246 0.9322742 0.9348083    0
## MLP     0.9188074 0.9227747 0.9249608 0.9246347 0.9267656 0.9299460    0
## SVM     0.9061412 0.9124032 0.9159291 0.9153895 0.9185022 0.9227440    0
## C5      0.9248845 0.9290935 0.9313947 0.9304996 0.9324458 0.9339609    0
##
## Sens
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
```

```
## RL      0.8060816 0.8165513 0.8208303 0.8205531 0.8245845 0.8310269    0
## NB      0.7836854 0.7894210 0.7917881 0.7932856 0.7972870 0.8033503    0
## TAN     0.8128186 0.8177349 0.8203587 0.8218278 0.8268390 0.8301165    0
## TANSE   0.7833212 0.7904224 0.7946103 0.7946171 0.7992079 0.8040430    0
## TANHC   0.7835033 0.7907866 0.7947924 0.7948219 0.7993445 0.8044072    0
## AODE    0.8102695 0.8153223 0.8175528 0.8189371 0.8236526 0.8297214    0
## GBM     0.8002549 0.8063738 0.8119082 0.8104018 0.8133649 0.8197050    0
## XGBTree 0.8153678 0.8180991 0.8211945 0.8205986 0.8221959 0.8266570    0
## RF      0.8268390 0.8307993 0.8359432 0.8357005 0.8399418 0.8457757    0
## MLP     0.7898762 0.7953842 0.8019667 0.8009674 0.8052534 0.8113620    0
## SVM     0.7116788 0.7219816 0.7246575 0.7244672 0.7271689 0.7363139    0
## C5      0.8079024 0.8151857 0.8182648 0.8180380 0.8212400 0.8288420    0
## 
## Spec 
##              Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## RL      0.8972865 0.9056810 0.9092316 0.9092295 0.9132374 0.9176985    0
## NB      0.8410415 0.8527403 0.8576111 0.8563561 0.8609796 0.8623452    0
## TAN     0.8789148 0.8816916 0.8861981 0.8859679 0.8877003 0.8949381    0
## TANSE   0.8492353 0.8523307 0.8576892 0.8574941 0.8607065 0.8716315    0
## TANHC   0.8492353 0.8523762 0.8576892 0.8574827 0.8607065 0.8714494    0
## AODE    0.8705390 0.8760757 0.8781865 0.8784455 0.8807811 0.8852877    0
## GBM     0.9053168 0.9104151 0.9118638 0.9122226 0.9147396 0.9173343    0
## XGBTree 0.9111435 0.9143715 0.9170612 0.9165698 0.9182447 0.9246176    0
## RF      0.9142389 0.9199290 0.9216061 0.9208603 0.9228423 0.9242535    0
## MLP     0.8956664 0.9038602 0.9074108 0.9069080 0.9101753 0.9178806    0
## SVM     0.9246140 0.9371597 0.9437127 0.9432516 0.9502725 0.9600726    0
## C5      0.8978514 0.9111394 0.9130554 0.9133379 0.9159687 0.9213401    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
##         RL        NB         TAN        TANSE      TANHC      AODE
## RL                 4.007e-02  1.099e-02  3.884e-02  3.874e-02  1.620e-02
## NB      2.624e-14            -2.908e-02 -1.235e-03 -1.331e-03 -2.387e-02
## TAN     2.838e-06 8.045e-11             2.784e-02  2.775e-02  5.207e-03
## TANSE   9.738e-13 1.0000000  1.439e-12            -9.673e-05 -2.264e-02
## TANHC   1.068e-12 1.0000000  1.552e-12  0.0240107            -2.254e-02
## AODE    3.212e-08 1.881e-10  1.279e-06  5.895e-13  5.985e-13
## GBM     0.9001710 1.371e-12  1.394e-07  7.447e-16  7.617e-16  2.920e-11
## XGBTree 0.4689159 2.138e-13  3.286e-08  8.161e-16  8.342e-16  4.638e-11
## RF      5.569e-09 6.741e-16  5.360e-12  < 2.2e-16  < 2.2e-16  4.924e-14
## MLP     7.636e-08 1.967e-14  1.0000000  3.896e-10  4.186e-10  0.0107044
## SVM     2.706e-08 0.0046213  1.109e-06  0.0149517  0.0159165  1.603e-05
## C5      1.0000000 8.747e-16  7.338e-06  5.930e-13  6.330e-13  1.734e-08
##         GBM        XGBTree    RF         MLP        SVM        C5
## RL       3.579e-03 -3.693e-03 -1.339e-02  1.095e-02  3.073e-02 -7.967e-04
## NB      -3.649e-02 -4.376e-02 -5.346e-02 -2.912e-02 -9.338e-03 -4.087e-02
## TAN     -7.414e-03 -1.469e-02 -2.438e-02 -3.989e-05  1.974e-02 -1.179e-02
## TANSE   -3.526e-02 -4.253e-02 -5.222e-02 -2.788e-02 -8.103e-03 -3.963e-02
## TANHC   -3.516e-02 -4.243e-02 -5.213e-02 -2.779e-02 -8.006e-03 -3.954e-02
## AODE    -1.262e-02 -1.989e-02 -2.959e-02 -5.246e-03  1.453e-02 -1.700e-02
## GBM                -7.272e-03 -1.697e-02  7.374e-03  2.715e-02 -4.376e-03
## XGBTree 0.0001475             -9.696e-03  1.465e-02  3.443e-02  2.896e-03
## RF      5.602e-09  4.200e-06              2.434e-02  4.412e-02  1.259e-02
## MLP     0.0014726  4.308e-07  1.634e-11             1.978e-02 -1.175e-02
## SVM     4.909e-09  3.597e-10  6.227e-12  3.212e-06            -3.153e-02
## C5      0.4803708  1.0000000  1.497e-08  1.981e-09  9.944e-09
##
## Kappa
##         RL        NB         TAN        TANSE      TANHC      AODE
## RL                 8.014e-02  2.199e-02  7.767e-02  7.748e-02  3.240e-02
## NB      2.624e-14            -5.815e-02 -2.469e-03 -2.663e-03 -4.774e-02
## TAN     2.838e-06 8.045e-11             5.568e-02  5.549e-02  1.041e-02
## TANSE   9.737e-13 1.0000000  1.439e-12            -1.935e-04 -4.527e-02
## TANHC   1.068e-12 1.0000000  1.552e-12  0.0240092            -4.508e-02
## AODE    3.211e-08 1.881e-10  1.279e-06  5.894e-13  5.985e-13
## GBM     0.9001825 1.370e-12  1.394e-07  7.447e-16  7.616e-16  2.919e-11
## XGBTree 0.4688296 2.138e-13  3.284e-08  8.157e-16  8.338e-16  4.636e-11
## RF      5.569e-09 6.742e-16  5.360e-12  < 2.2e-16  < 2.2e-16  4.923e-14
## MLP     7.638e-08 1.967e-14  1.0000000  3.895e-10  4.185e-10  0.0107023
## SVM     2.560e-08 0.0051265  1.028e-06  0.0167235  0.0178188  1.457e-05
## C5      1.0000000 8.746e-16  7.338e-06  5.930e-13  6.329e-13  1.734e-08
##         GBM        XGBTree    RF         MLP        SVM        C5
## RL       7.158e-03 -7.386e-03 -2.678e-02  2.191e-02  6.167e-02 -1.593e-03
## NB      -7.298e-02 -8.753e-02 -1.069e-01 -5.823e-02 -1.848e-02 -8.173e-02
## TAN     -1.483e-02 -2.937e-02 -4.877e-02 -7.979e-05  3.968e-02 -2.358e-02
```

```
## TANSE   -7.051e-02 -8.506e-02 -1.044e-01 -5.576e-02 -1.601e-02 -7.926e-02
## TANHC   -7.032e-02 -8.486e-02 -1.043e-01 -5.557e-02 -1.581e-02 -7.907e-02
## AODE    -2.524e-02 -3.979e-02 -5.918e-02 -1.049e-02  2.927e-02 -3.399e-02
## GBM                -1.454e-02 -3.394e-02  1.475e-02  5.451e-02 -8.752e-03
## XGBTree 0.0001474             -1.939e-02  2.929e-02  6.905e-02  5.793e-03
## RF      5.602e-09  4.199e-06              4.869e-02  8.844e-02  2.518e-02
## MLP     0.0014724  4.306e-07  1.634e-11              3.976e-02 -2.350e-02
## SVM     4.621e-09  3.417e-10  5.951e-12  2.982e-06             -6.326e-02
## C5      0.4803525  1.0000000  1.498e-08  1.981e-09  9.433e-09
##
## ROC
##         RL         NB          TAN         TANSE       TANHC       AODE
## RL                 2.989e-02 -2.039e-05  2.915e-02  2.910e-02  5.523e-04
## NB      < 2.2e-16            -2.991e-02 -7.431e-04 -7.915e-04 -2.934e-02
## TAN     1.0000000 6.704e-16              2.917e-02  2.912e-02  5.727e-04
## TANSE   2.061e-12 1.0000000  < 2.2e-16             -4.843e-05 -2.859e-02
## TANHC   2.036e-12 1.0000000  < 2.2e-16  0.2700157             -2.854e-02
## AODE    1.0000000 6.688e-16  1.0000000  < 2.2e-16  < 2.2e-16
## GBM     2.400e-05 < 2.2e-16  8.004e-13  < 2.2e-16  < 2.2e-16  2.022e-14
## XGBTree 9.521e-09 < 2.2e-16  2.823e-10  < 2.2e-16  < 2.2e-16  2.387e-11
## RF      6.771e-08 < 2.2e-16  4.701e-07  4.438e-15  4.244e-15  2.131e-07
## MLP     1.0000000 < 2.2e-16  1.0000000  5.143e-13  5.047e-13  1.0000000
## SVM     0.0166590 4.214e-09  0.0014300  1.191e-09  1.271e-09  0.0022931
## C5      1.309e-08 < 2.2e-16  1.019e-06  9.773e-15  9.689e-15  4.421e-07
##         GBM        XGBTree    RF         MLP        SVM        C5
## RL      -7.341e-03 -1.296e-02 -6.747e-03 -1.257e-03  7.988e-03 -7.122e-03
## NB      -3.723e-02 -4.285e-02 -3.663e-02 -3.114e-02 -2.190e-02 -3.701e-02
## TAN     -7.321e-03 -1.294e-02 -6.726e-03 -1.236e-03  8.009e-03 -7.101e-03
## TANSE   -3.649e-02 -4.211e-02 -3.589e-02 -3.040e-02 -2.116e-02 -3.627e-02
## TANHC   -3.644e-02 -4.206e-02 -3.584e-02 -3.035e-02 -2.111e-02 -3.622e-02
## AODE    -7.893e-03 -1.352e-02 -7.299e-03 -1.809e-03  7.436e-03 -7.674e-03
## GBM                -5.623e-03  5.944e-04  6.084e-03  1.533e-02  2.194e-04
## XGBTree 2.600e-06              6.217e-03  1.171e-02  2.095e-02  5.842e-03
## RF      1.0000000  3.593e-05              5.490e-03  1.474e-02 -3.750e-04
## MLP     0.0001394  7.299e-08  6.810e-07              9.245e-03 -5.865e-03
## SVM     2.362e-07  4.378e-10  3.986e-06  0.0035280             -1.511e-02
## C5      1.0000000  6.606e-05  1.0000000  1.946e-10  3.077e-06
##
## Sens
##         RL         NB          TAN         TANSE       TANHC       AODE
## RL                 2.727e-02 -1.275e-03  2.594e-02  2.573e-02  1.616e-03
## NB      8.143e-11            -2.854e-02 -1.332e-03 -1.536e-03 -2.565e-02
## TAN     1.0000000 1.920e-08              2.721e-02  2.701e-02  2.891e-03
## TANSE   5.160e-07 1.0000000  8.263e-14             -2.048e-04 -2.432e-02
## TANHC   5.859e-07 1.0000000  1.261e-13  0.0187885             -2.412e-02
## AODE    1.0000000 1.240e-08  0.0153458  2.554e-15  2.715e-15
## GBM     0.0240890 2.459e-05  3.405e-07  8.628e-11  7.439e-11  5.042e-07
## XGBTree 1.0000000 1.167e-09  1.0000000  1.351e-08  1.453e-08  1.0000000
## RF      1.417e-07 8.835e-14  0.0006302  2.302e-10  2.410e-10  1.708e-05
## MLP     1.967e-10 0.0001435  1.660e-06  0.7306717  0.8990708  8.546e-06
## SVM     5.105e-16 1.566e-13  1.442e-15  3.763e-13  3.212e-13  1.238e-15
## C5      1.0000000 8.526e-12  1.0000000  8.066e-07  8.984e-07  1.0000000
##         GBM        XGBTree    RF         MLP        SVM        C5
## RL       1.015e-02 -4.544e-05 -1.515e-02  1.959e-02  9.609e-02  2.515e-03
## NB      -1.712e-02 -2.731e-02 -4.241e-02 -7.682e-03  6.882e-02 -2.475e-02
```

```
## TAN       1.143e-02  1.229e-03 -1.387e-02  2.086e-02  9.736e-02  3.790e-03
## TANSE    -1.578e-02 -2.598e-02 -4.108e-02 -6.350e-03  7.015e-02 -2.342e-02
## TANHC    -1.558e-02 -2.578e-02 -4.088e-02 -6.145e-03  7.035e-02 -2.322e-02
## AODE      8.535e-03 -1.661e-03 -1.676e-02  1.797e-02  9.447e-02  8.991e-04
## GBM                 -1.020e-02 -2.530e-02  9.434e-03  8.593e-02 -7.636e-03
## XGBTree 0.0013666              -1.510e-02  1.963e-02  9.613e-02  2.561e-03
## RF      2.561e-07  4.072e-06              3.473e-02  1.112e-01  1.766e-02
## MLP     0.0502042  7.349e-07  8.438e-14             7.650e-02 -1.707e-02
## SVM     3.599e-15  < 2.2e-16  < 2.2e-16  2.065e-14            -9.357e-02
## C5      0.1883289  1.0000000  2.240e-10  1.335e-09  < 2.2e-16
##
## Spec
##         RL        NB         TAN        TANSE      TANHC      AODE
## RL                 5.287e-02  2.326e-02  5.174e-02  5.175e-02  3.078e-02
## NB      1.104e-12            -2.961e-02 -1.138e-03 -1.127e-03 -2.209e-02
## TAN     1.737e-08 1.143e-07             2.847e-02  2.849e-02  7.522e-03
## TANSE   2.153e-11 1.0000000  9.742e-11             1.138e-05 -2.095e-02
## TANHC   1.981e-11 1.0000000  9.227e-11  1.0000000            -2.096e-02
## AODE    5.029e-10 3.801e-06  3.294e-07  1.412e-09  1.333e-09
## GBM     1.0000000 1.082e-12  2.590e-13  1.211e-14  1.112e-14  5.615e-15
## XGBTree 0.0278491 1.253e-12  1.046e-11  4.888e-16  4.488e-16  9.033e-14
## RF      7.357e-07 2.732e-14  2.826e-12  1.536e-14  1.426e-14  1.652e-13
## MLP     1.0000000 4.679e-14  1.900e-06  2.077e-10  1.917e-10  1.262e-08
## SVM     4.855e-06 4.069e-13  1.209e-09  1.034e-12  1.022e-12  1.040e-10
## C5      0.6313074 6.063e-14  5.638e-09  5.814e-13  5.257e-13  1.530e-10
##         GBM       XGBTree    RF         MLP        SVM        C5
## RL      -2.993e-03 -7.340e-03 -1.163e-02  2.321e-03 -3.402e-02 -4.108e-03
## NB      -5.587e-02 -6.021e-02 -6.450e-02 -5.055e-02 -8.690e-02 -5.698e-02
## TAN     -2.625e-02 -3.060e-02 -3.489e-02 -2.094e-02 -5.728e-02 -2.737e-02
## TANSE   -5.473e-02 -5.908e-02 -6.337e-02 -4.941e-02 -8.576e-02 -5.584e-02
## TANHC   -5.474e-02 -5.909e-02 -6.338e-02 -4.943e-02 -8.577e-02 -5.586e-02
## AODE    -3.378e-02 -3.812e-02 -4.241e-02 -2.846e-02 -6.481e-02 -3.489e-02
## GBM                -4.347e-03 -8.638e-03  5.315e-03 -3.103e-02 -1.115e-03
## XGBTree 0.0621488             -4.290e-03  9.662e-03 -2.668e-02  3.232e-03
## RF      4.567e-06  0.1135270             1.395e-02 -2.239e-02  7.522e-03
## MLP     0.5440653  0.0111472  1.042e-05            -3.634e-02 -6.430e-03
## SVM     2.571e-06  6.596e-06  0.0001073  7.603e-07             2.991e-02
## C5      1.0000000  1.0000000  0.0028001  0.0342006  1.817e-05
```

4. Interpretabilidad

Para simplificar la comparación de modelos, vamos a comparar los modelos más representativos del análisis:

· AODE

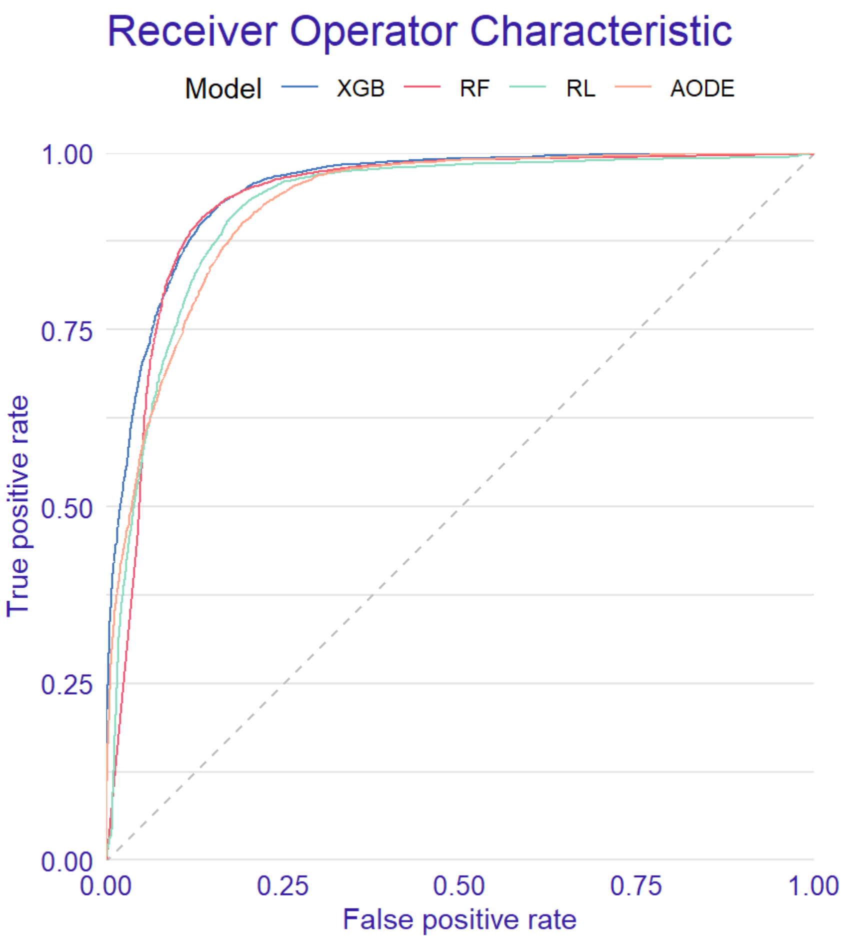
· Extreme Gradient Boosting (XGB)

· Random Forest (RF)

· Regresión logística

```
# Creamos el explicador de AODE
explainer_AODE <- DALEX::explain(
  model = aode_train,
  data = test_factor,
  y= test_factor$y=="Yes",
  label    = "AODE",
  type = "classification",
  verbose = FALSE)

# Creamos el explicador de XGB
explainer_XGB <- DALEX::explain(
  model = XGBFinal_train,
  data = test_num,
  y= test_num$y=="Yes",
  label    = "XGB",
  type = "classification",
  verbose = FALSE)

# Creamos el explicador de Random Forest
explainer_rf <- DALEX::explain(
  model = rf_train,
  data = test_general,
  y= test_general$y=="Yes",
  label    = "RF",
  type = "classification",
  verbose = FALSE)

# Creamos el explicador de RL
explainer_rl <- DALEX::explain(
  model = rl_train,
  data = test_general,
  y= test_general$y=="Yes",
  label    = "RL",
  type = "classification",
  verbose = FALSE)
```

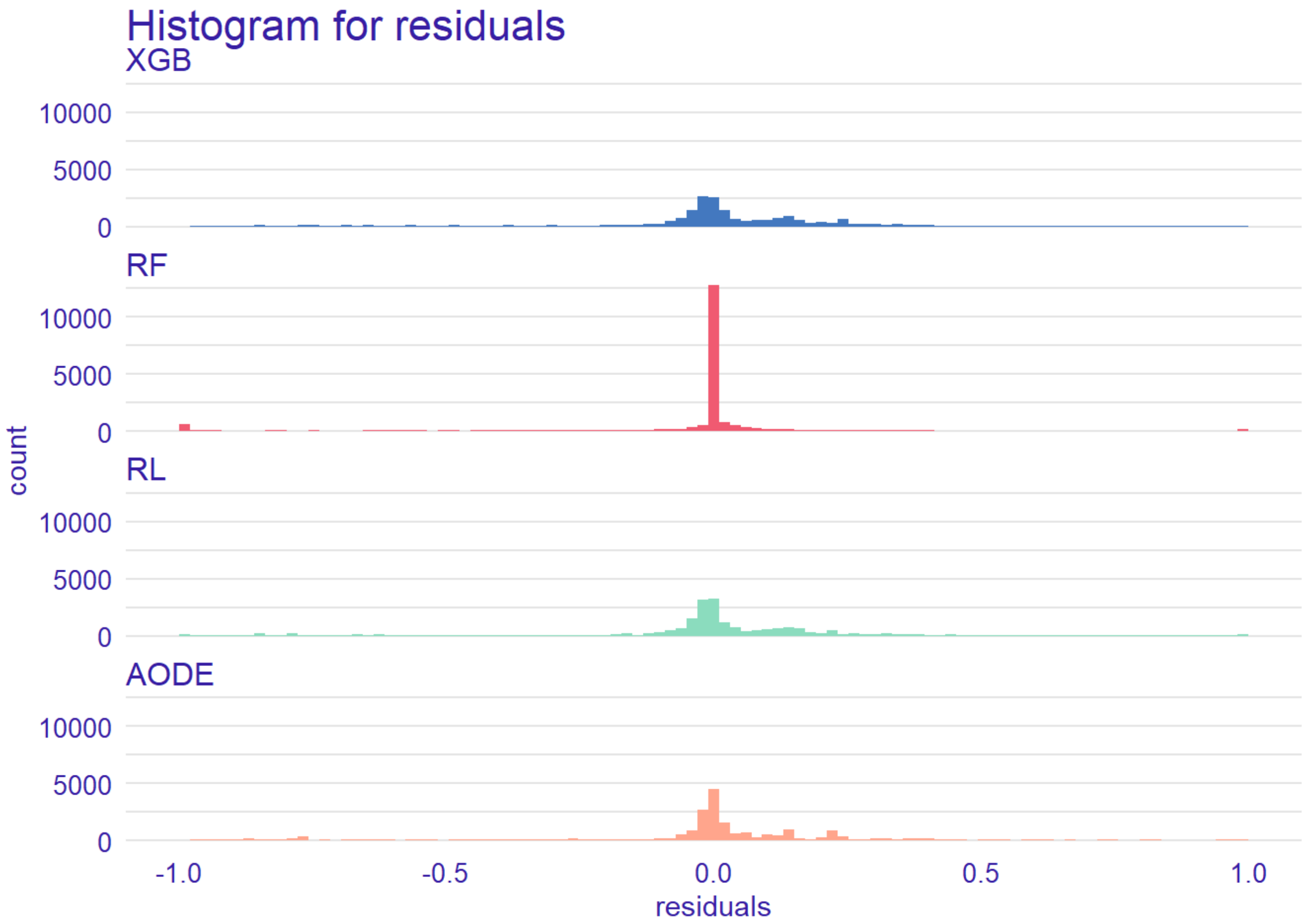
```
# Comparativa de curvas ROC
plot(model_performance(explainer_AODE),
     model_performance(explainer_XGB),
     model_performance(explainer_rf),
     model_performance(explainer_rl),
     geom = "roc"
     )
```



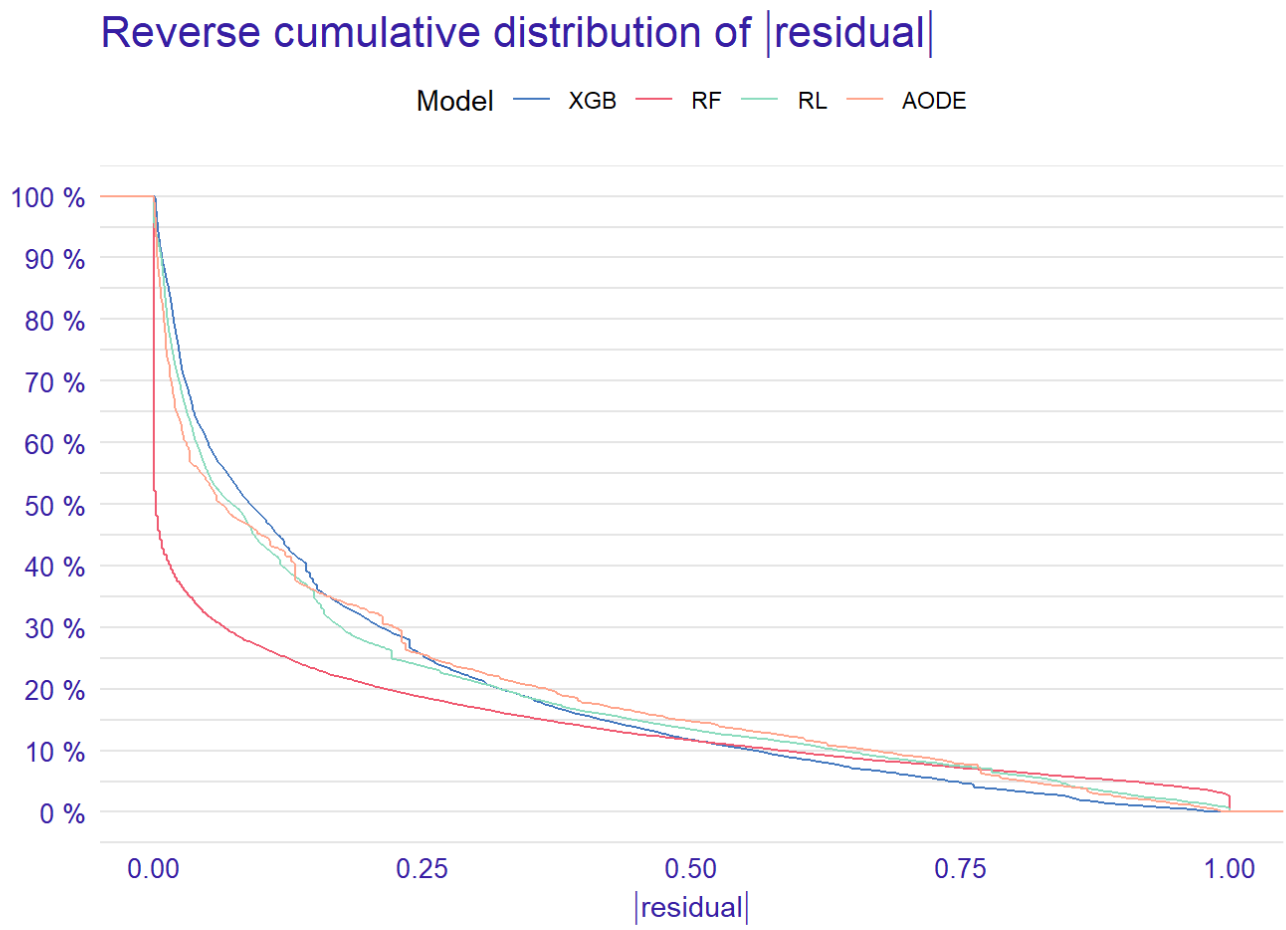

```
# Nota: Se producirá un error con el explainer_rl si se carga un rl_train previo (en vez e
jecutarlo en esta sesión)
```

4.1. Comparativa de residuos

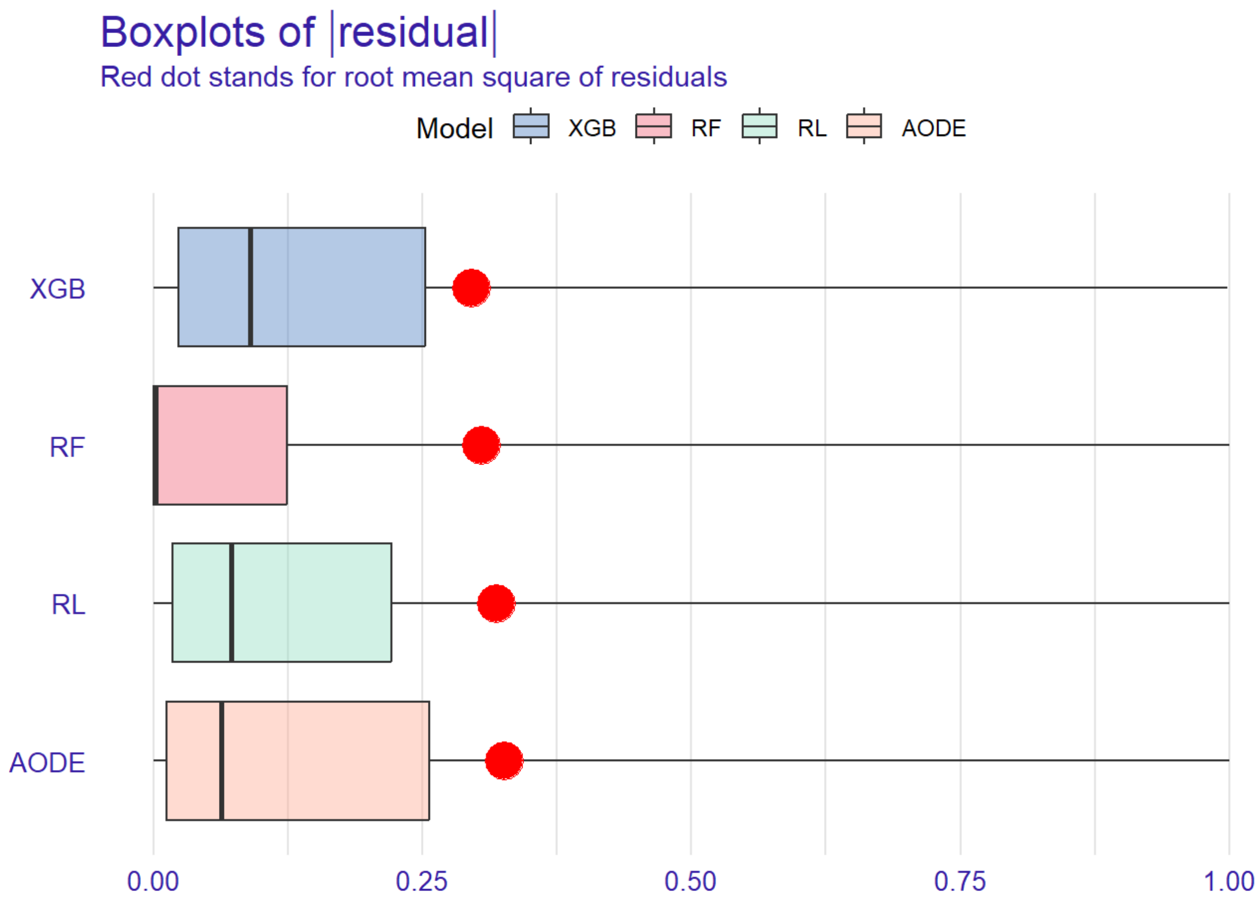
```
# Comparativa de histogramas de residuos
plot(model_performance(explainer_AODE),
     model_performance(explainer_XGB),
     model_performance(explainer_rf),
     model_performance(explainer_rl),
     geom = "histogram"
     )
```




```r
# Distribución acumulativa de los residuos
plot(model_performance(explainer_AODE),
     model_performance(explainer_XGB),
     model_performance(explainer_rf),
     model_performance(explainer_rl)
     )
```

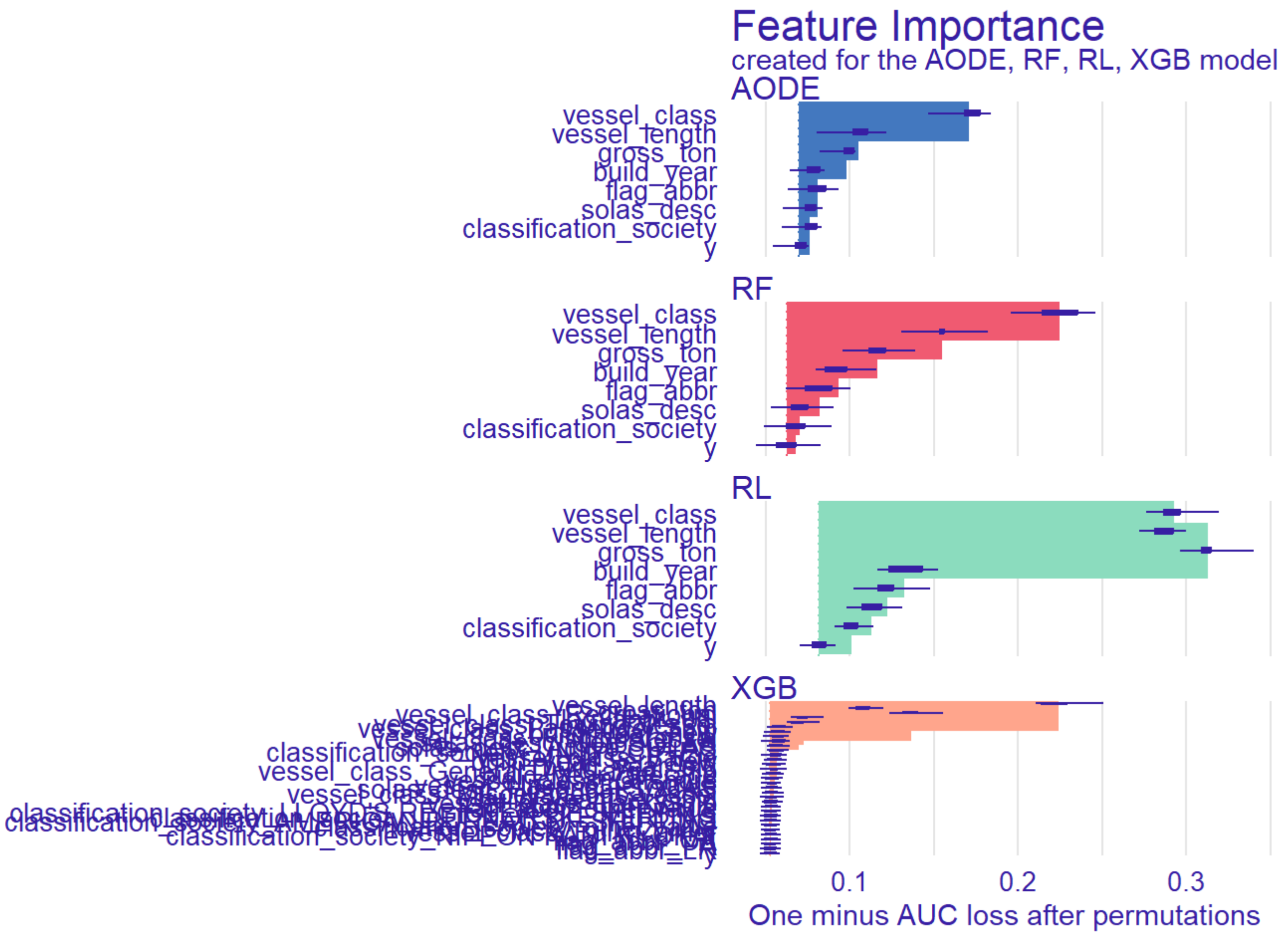



```
# En formato de boxplots
plot(model_performance(explainer_AODE),
     model_performance(explainer_XGB),
     model_performance(explainer_rf),
     model_performance(explainer_rl),
     geom = "boxplot"
     )
```

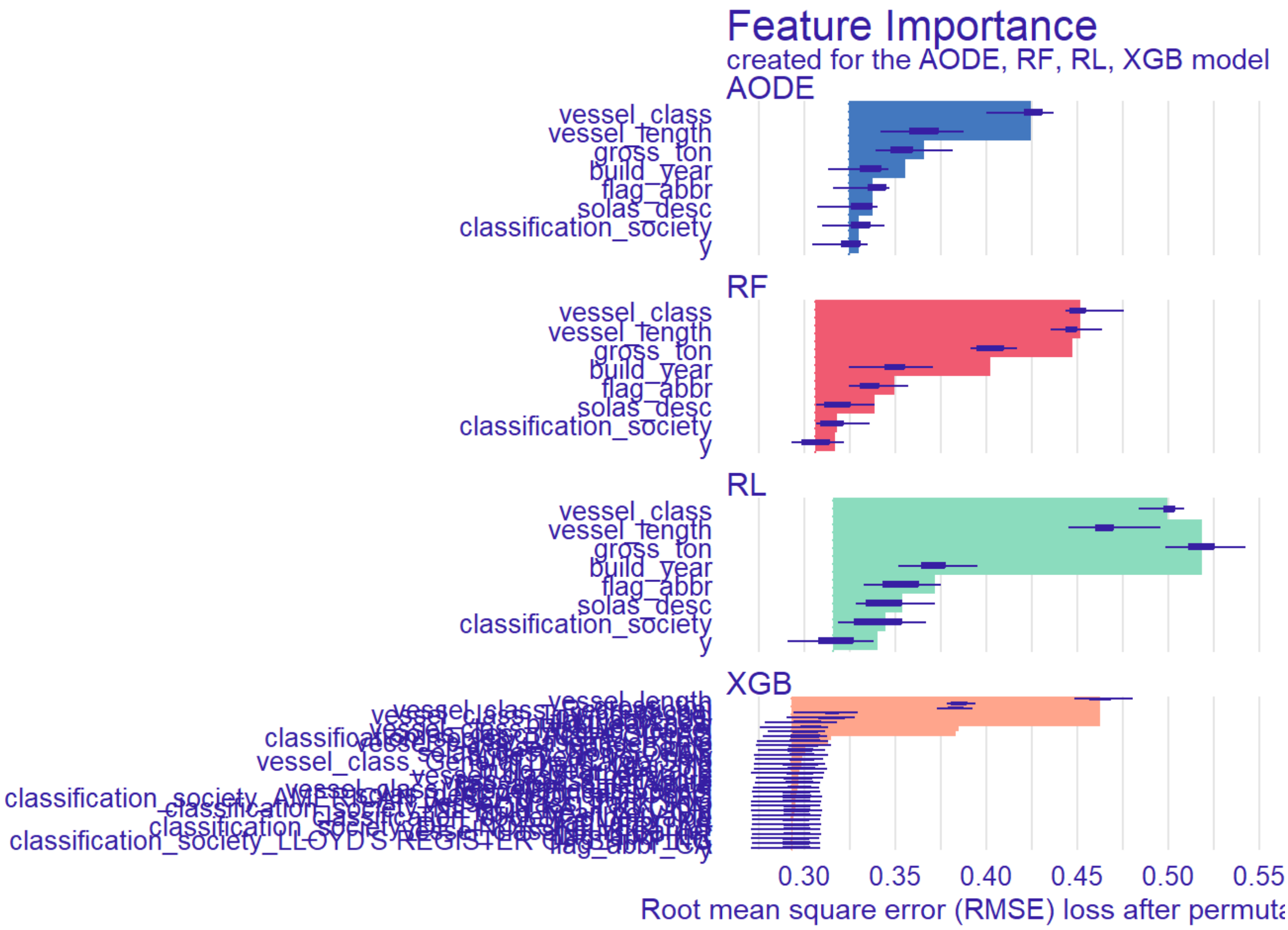


4.2. Importancia de las variables

```
# Generamos gráficos de importancia en base al Root Mean Square Error (RMSE)
plot(model_parts(explainer_AODE, type = "raw"),
     model_parts(explainer_XGB, type = "raw"),
     model_parts(explainer_rf, type = "raw"),
     model_parts(explainer_rl, type = "raw")
     )
```



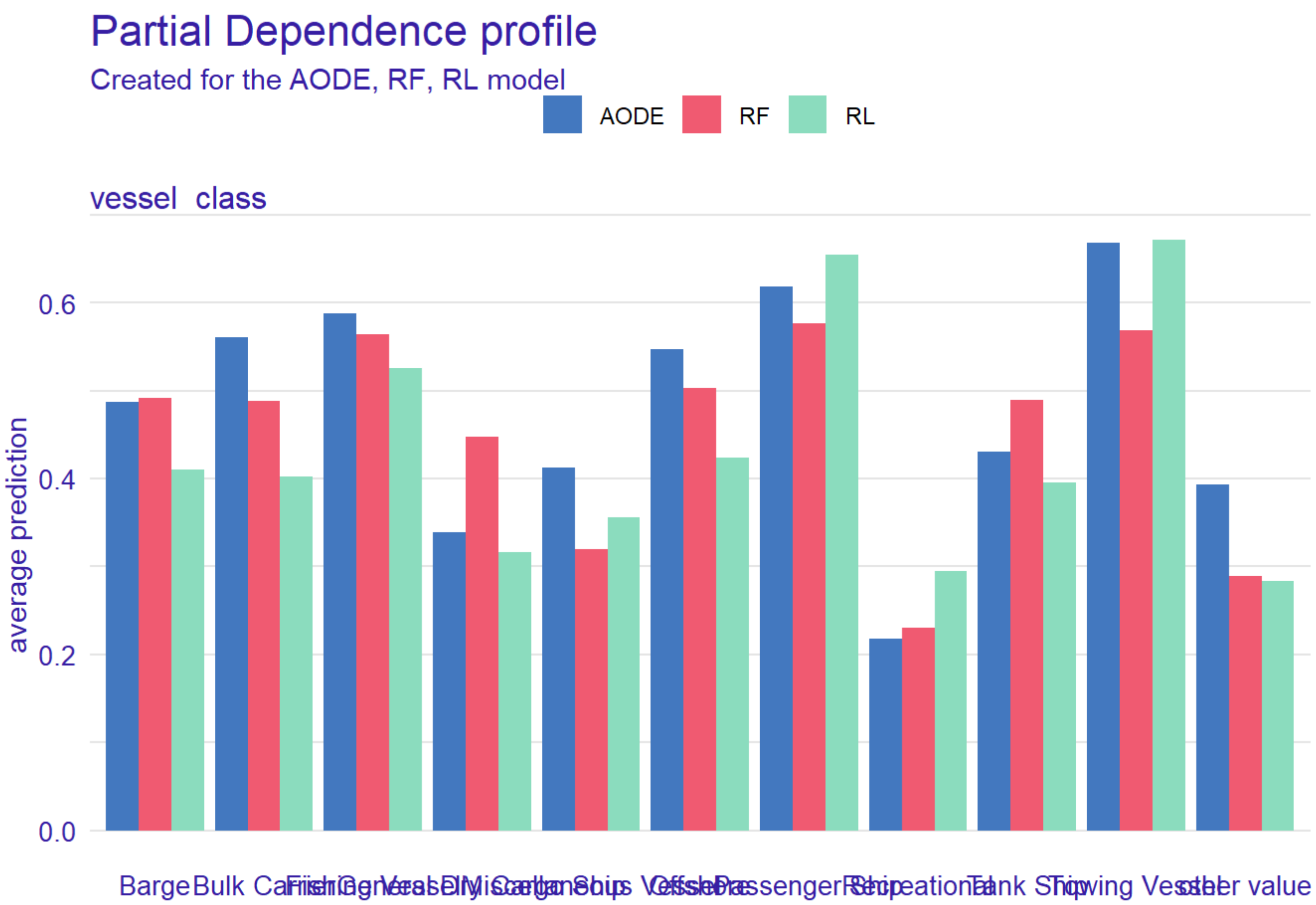

```
# Generamos gráficos de importancia en base al Root Mean Square Error (RMSE)
plot(model_parts(explainer_AODE, loss_function = loss_root_mean_square),
     model_parts(explainer_XGB, loss_function = loss_root_mean_square),
     model_parts(explainer_rf, loss_function = loss_root_mean_square),
     model_parts(explainer_rl, loss_function = loss_root_mean_square)
     )
```



4.3. Partial Dependence Plot

```
# Gráfico de dependencia parcial de los modelos AODE, Random Forset y Regresión Logística
plot(model_profile(explainer_AODE, variables = "vessel_class", type="partial"),
     model_profile(explainer_rf, variables = "vessel_class", type="partial"),
     model_profile(explainer_rl, variables = "vessel_class", type="partial")
     )
```

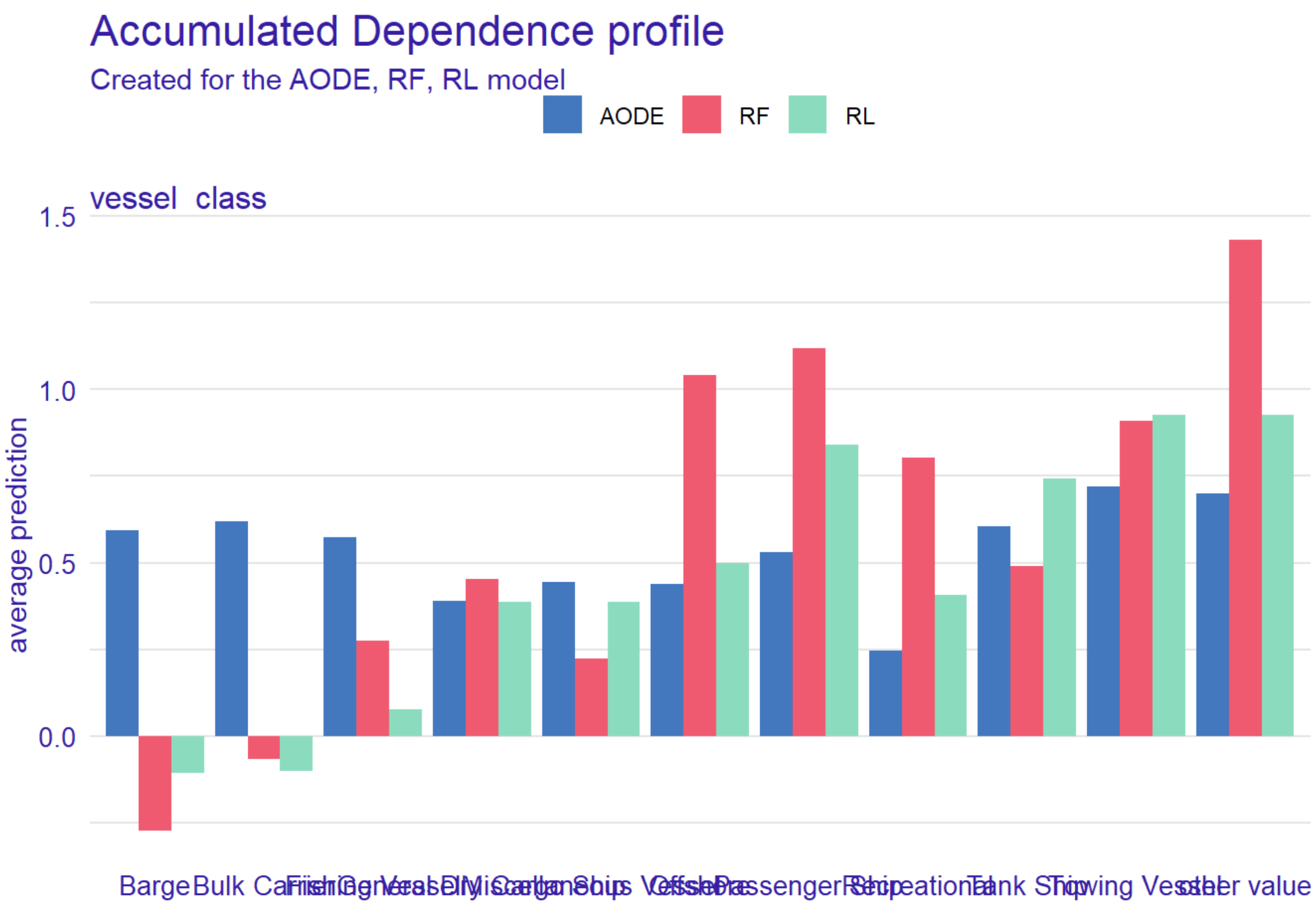
```
## 'variable_type' changed to 'categorical' due to lack of numerical variables.
## 'variable_type' changed to 'categorical' due to lack of numerical variables.
## 'variable_type' changed to 'categorical' due to lack of numerical variables.
```



4.4. Accumulated Local Effects plot

```
# Gráfico de dependencia acumulada de los modelos AODE, Random Forset y Regresión logística
plot(model_profile(explainer_AODE, variables = "vessel_class", type="accumulated"),
     model_profile(explainer_rf, variables = "vessel_class", type="accumulated"),
     model_profile(explainer_rl, variables = "vessel_class", type="accumulated")
     )
```

```
## 'variable_type' changed to 'categorical' due to lack of numerical variables.
## 'variable_type' changed to 'categorical' due to lack of numerical variables.
## 'variable_type' changed to 'categorical' due to lack of numerical variables.
```

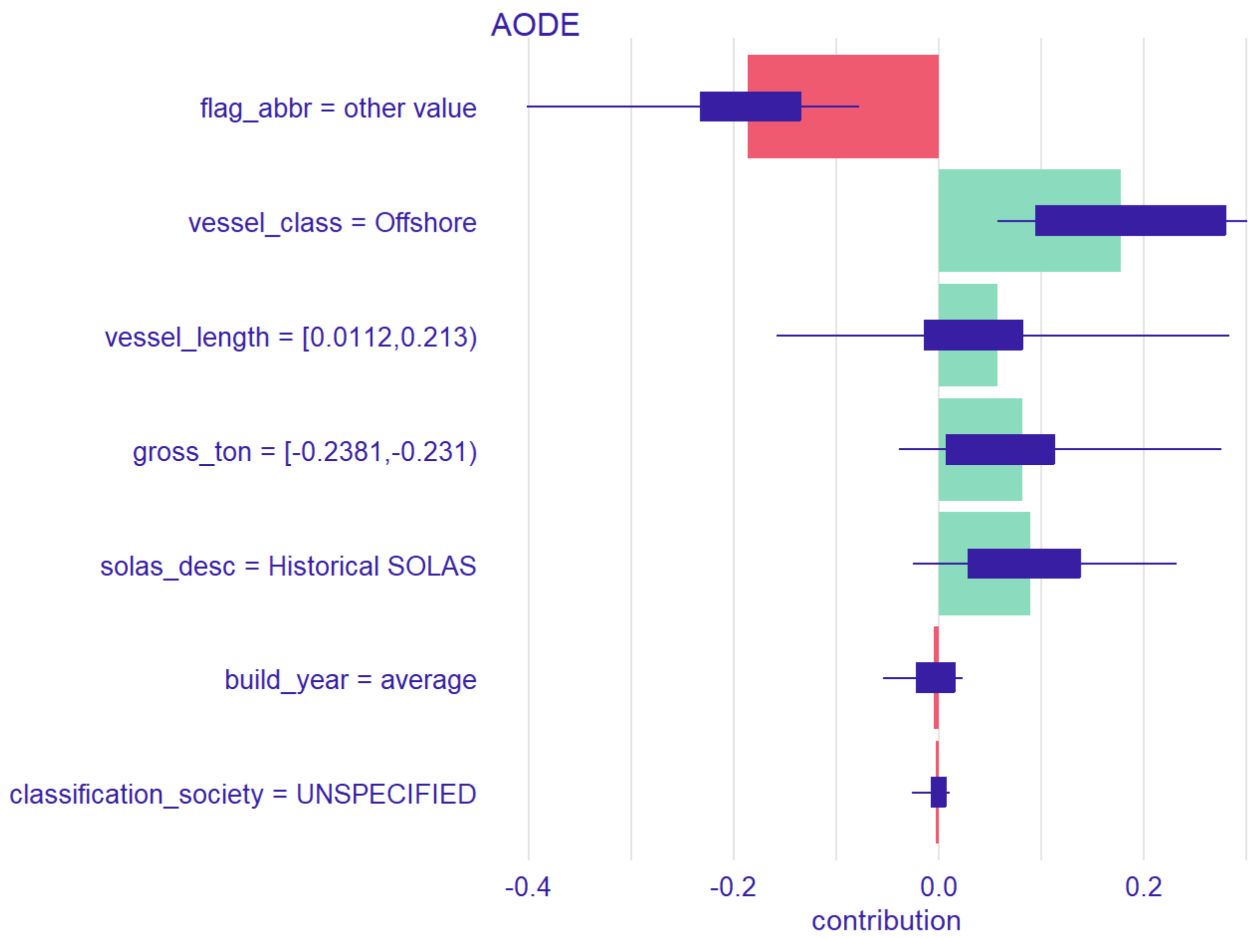


4.5. Explicabilidad local - Criterio SHAP

Para variables factoriales, podemos analizar la importancia de cada variable en predicciones específicas usando el paquete iBreakDown. Seleccionando un registro al azar, calculamos la contribución de cada variable, destacando su impacto positivo (verde) o negativo (rojo) en la predicción. Este enfoque permite una comprensión detallada de la explicatividad a nivel local.

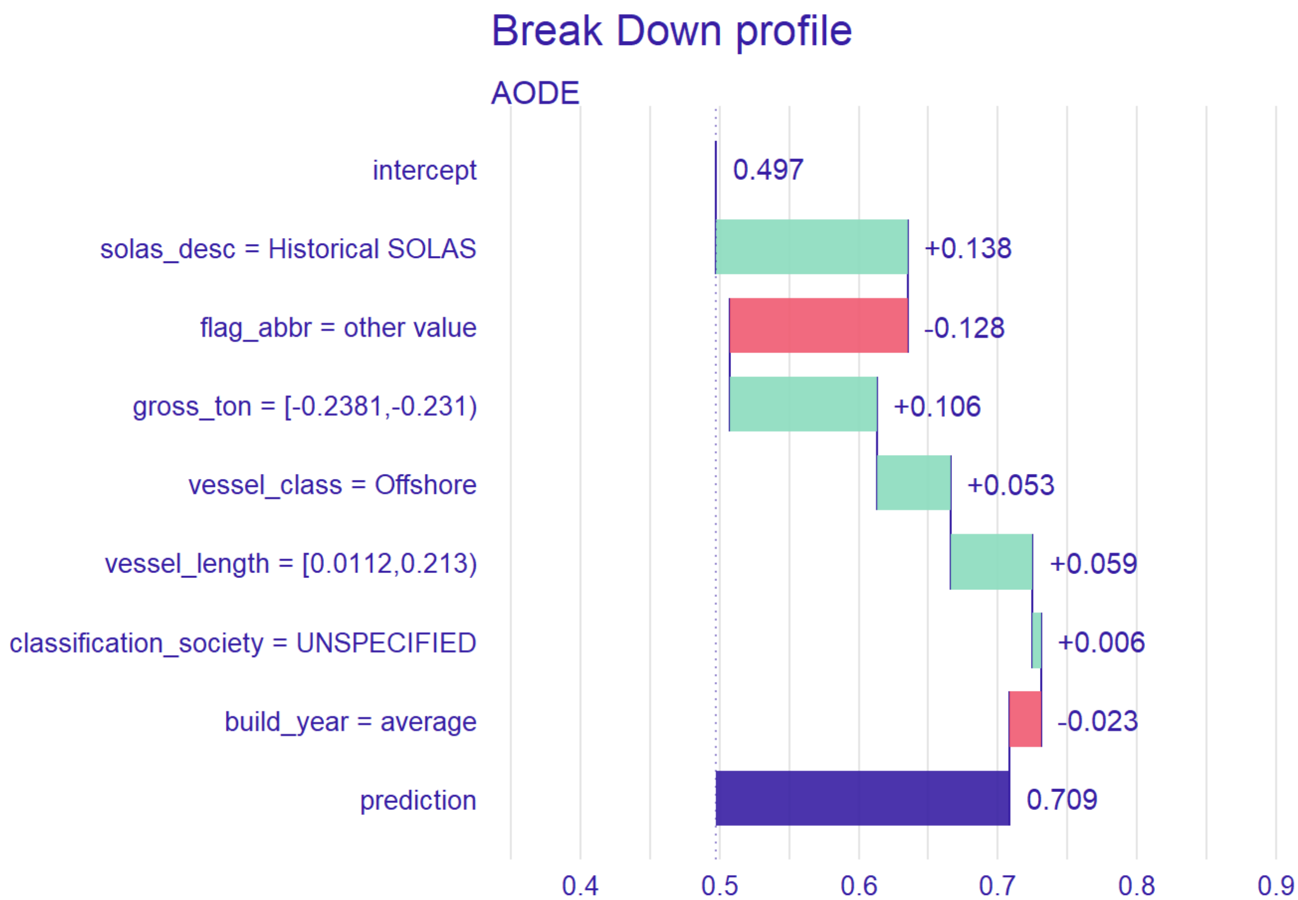
```
set.seed(7)
vessel_sample <- train_factor %>%
  select(-y) %>%
  slice_sample(n=1)

shap_AODE <- shap(explainer_AODE, vessel_sample)

bd_AODE <- break_down(explainer_AODE, vessel_sample, keep_distributions = TRUE)
```

```
plot(shap_AODE)
```



AODE
flag_abbr = other value
vessel_class = Offshore
vessel_length = [0.0112,0.213)
gross_ton = [-0.2381,-0.231)
solas_desc = Historical SOLAS
build_year = average
classification_society = UNSPECIFIED
-0.4
-0.2
0.0
0.2
contribution


```
plot(bd_AODE)
```



Break Down profile
AODE
intercept
solas_desc = Historical SOLAS
flag_abbr = other value
gross_ton = [-0.2381,-0.231)
vessel_class = Offshore
vessel_length = [0.0112,0.213)
classification_society = UNSPECIFIED
build_year = average
prediction
0.497
+0.138
-0.128
+0.106
+0.053
+0.059
+0.006
-0.023
0.709
0.4
0.5
0.6
0.7
0.8
0.9

4.6. Explicatibilidad interactiva

```
modelStudio(explainer_rf)
```

```
## `new_observation` argument is NULL. `new_observation_n` observations needed to calculat
e local explanations are taken from the data.
```

```
## Warning in value[[3L]](cond):
## Error occurred in ingredients::partial_dependence (numerical) function: variable 'gross
_ton' was fitted with type "nmatrix.1" but type "numeric" was supplied
```

```
## Warning in value[[3L]](cond):
## Error occurred in ingredients::partial_dependence (categorical) function: variable 'gro
ss_ton' was fitted with type "nmatrix.1" but type "numeric" was supplied
```

```
## Warning in value[[3L]](cond):
## Error occurred in ingredients::accumulated_dependence (numerical) function: variable 'g
ross_ton' was fitted with type "nmatrix.1" but type "numeric" was supplied
```

```
## Warning in value[[3L]](cond):
## Error occurred in ingredients::accumulated_dependence (categorical) function: variable
'gross_ton' was fitted with type "nmatrix.1" but type "numeric" was supplied
```

```
## Warning in value[[3L]](cond):
## Error occurred in iBreakDown::local_attributions (1)       function: replacement has 1
row, data has 21966
```

```
## Warning in value[[3L]](cond):
## Error occurred in iBreakDown::shap (1)                      function: replacement has 1
row, data has 900
```

```
## Warning in value[[3L]](cond):
## Error occurred in ingredients::ceteris_paribus (1)         function: variable 'gross_to
n' was fitted with type "nmatrix.1" but type "numeric" was supplied
```

```
## Warning in value[[3L]](cond):
## Error occurred in iBreakDown::local_attributions (2)       function: replacement has 1
row, data has 21966
```

```
## Warning in value[[3L]](cond):
## Error occurred in iBreakDown::shap (2)                      function: replacement has 1
row, data has 900
```

```
## Warning in value[[3L]](cond):
## Error occurred in ingredients::ceteris_paribus (2)         function: variable 'gross_to
n' was fitted with type "nmatrix.1" but type "numeric" was supplied
```

```
## Warning in value[[3L]](cond):
## Error occurred in iBreakDown::local_attributions (3)       function: replacement has 1
row, data has 21966
```

```
## Warning in value[[3L]](cond):
## Error occurred in iBreakDown::shap (3)                      function: replacement has 1
row, data has 900
```

```
## Warning in value[[3L]](cond):
## Error occurred in ingredients::ceteris_paribus (3)         function: variable 'gross_to
n' was fitted with type "nmatrix.1" but type "numeric" was supplied
```