TFM - Kaggle House Prices: Advanced Regression Techniques with caret

05 Selección del modelo definitivo y presentación de resultados

Juan Carlos Santiago Culebras

2019-10-01

Primeros pasos

Librerías

Realizamos la carga de las librerías necesarias

```r
if(!is.element("dplyr", installed.packages()[, 1]))
      install.packages("dplyr", repos = 'http://cran.us.r-project.org')
library(dplyr)

if(!is.element("tidyr", installed.packages()[, 1]))
      install.packages("tidyr", repos = 'http://cran.us.r-project.org')
library(tidyr)

if(!is.element("ggplot2", installed.packages()[, 1]))
      install.packages("ggplot2", repos = 'http://cran.us.r-project.org')
library(ggplot2)

if(!is.element("grid", installed.packages()[, 1]))
      install.packages("grid", repos = 'http://cran.us.r-project.org')
library(grid)

if(!is.element("gridExtra", installed.packages()[, 1]))
      install.packages("gridExtra", repos = 'http://cran.us.r-project.org')
library(gridExtra)

if(!is.element("readr", installed.packages()[, 1]))
      install.packages("readr", repos = 'http://cran.us.r-project.org')
library(readr)

if(!is.element("caret", installed.packages()[, 1]))
      install.packages("caret", repos = 'http://cran.us.r-project.org')
library(caret)

if(!is.element("ggpubr", installed.packages()[, 1]))
      install.packages("ggpubr", repos = 'http://cran.us.r-project.org')
library(ggpubr)
```

Funciones

```r
fnEstudioModelo <- function ( modelo , estudioParam = TRUE){

  # modelo
  # modelo$finalModel


  p1 <- ggplot(data = modelo$resample, aes(x = RMSE)) +
        geom_density(alpha = 0.5, fill = "gray50") +
        geom_vline(xintercept = mean(modelo$resample$RMSE),
                   linetype = "dashed") +
        theme_bw()

  p2 <- ggplot(data = modelo$resample, aes(x = 1, y = RMSE)) +
        geom_boxplot(outlier.shape = NA, alpha = 0.5, fill = "gray50") +
        geom_jitter(width = 0.05) +
        labs(x = "") +
        theme_bw() +
        theme(axis.text.x = element_blank(), axis.ticks.x = element_blank())

  #Estudio de hiperparámtros
  if (estudioParam){
    p3 <- plot(modelo)
  }

  # Error de test
  predicciones <- predict(modelo
                          , newdata = dsTrain.CV
                          , type = "raw")

  # RMSE(predicciones, dsTrain.CV$SalePrice)
  # MAE(predicciones, dsTrain.CV$SalePrice)
  # R2(predicciones, dsTrain.CV$SalePrice, form = "traditional")


  t1 <- capture.output(summary(modelo$resample$RMSE, digits=3))
  t1 <- paste("Summary resample$RMSE", " ", paste(t1, collapse="\n"), sep = "\n")
  t1 <- text_grob(t1, size = 10)

  t2 <- capture.output(postResample(pred = predicciones, obs = dsTrain.CV$SalePrice))
  t2 <- paste("Error de test", " ", paste(t2, collapse="\n"), sep = "\n")
  t2 <- text_grob(t2, size = 10)

  t3 <- capture.output(modelo$finalModel)
  t3 <- text_grob(paste(t3, collapse="\n"), size = 9)

  grid.arrange(t3, top="Modelo final")
  grid.arrange(p1, p2, t1, t2, nrow = 2, top="RMSE obtenido en la validación")

  if (estudioParam){
    grid.arrange(p3, nrow = 1, top="Evolución del RMSE del modelo en función de hiperparámetros")
  }
```

```
}


# helper function for the plots
tuneplot <- function(x, probs = .90) {
  ggplot(x) +
    coord_cartesian(ylim = c(quantile(x$results$RMSE, probs = probs), min(x$results$RMSE))) +
    theme_bw()
}
```

Cargamos datos

En este caso partimos de las métricas guardadas en etapas anteriores.

Excluyo el modelo KNN que claramente está muy alejado del resultado de los demás modelos

```
# Conjunto seleccionado en paso anterior
load('./F04_Modelos/F04_200_metricas.RData')

metricas <- metricasGuardadas %>%
    filter(modelo!='KNN') %>%
    mutate_if(is.character, as.factor)

head(arrange(metricas,Test),10)
```

```
##    modelo      Test   Training                OrigenF2
## 1    SVMR 0.1114359 0.08632997 F02_03_dsDataAll_Recipe
## 2    SVMR 0.1125683 0.10668322 F02_03_dsDataAll_Recipe
## 3    SVMR 0.1127808 0.09661025        F02_01_dsDataAll
## 4    SVMR 0.1142498 0.10157601        F02_01_dsDataAll
## 5    SVMR 0.1152299 0.09849094        F02_01_dsDataAll
## 6    SVMR 0.1155539 0.09393204 F02_03_dsDataAll_Recipe
## 7    SVMR 0.1156660 0.10630596        F02_01_dsDataAll
## 8  GLMNET 0.1159645 0.11149980 F02_03_dsDataAll_Recipe
## 9     GLM 0.1172617 0.10981097 F02_03_dsDataAll_Recipe
## 10     LM 0.1172617 0.10981097 F02_03_dsDataAll_Recipe
##                                 OrigenF3        fch
## 1             F03_15_dsDataSelVar_Completo 2019-09-20
## 2            F03_14_dsDataSelVar_mezcla_31 2019-09-20
## 3             F03_15_dsDataSelVar_Completo 2019-09-20
## 4  F03_12_dsDataSelVar_rfe_MenorRMSE_top60 2019-09-19
## 5            F03_13_dsDataSelVar_ga_100_46 2019-09-20
## 6  F03_12_dsDataSelVar_rfe_MenorRMSE_top55 2019-09-20
## 7            F03_14_dsDataSelVar_mezcla_31 2019-09-20
## 8             F03_15_dsDataSelVar_Completo 2019-09-20
## 9             F03_15_dsDataSelVar_Completo 2019-09-20
## 10            F03_15_dsDataSelVar_Completo 2019-09-20
```

Selección del mejor modelo

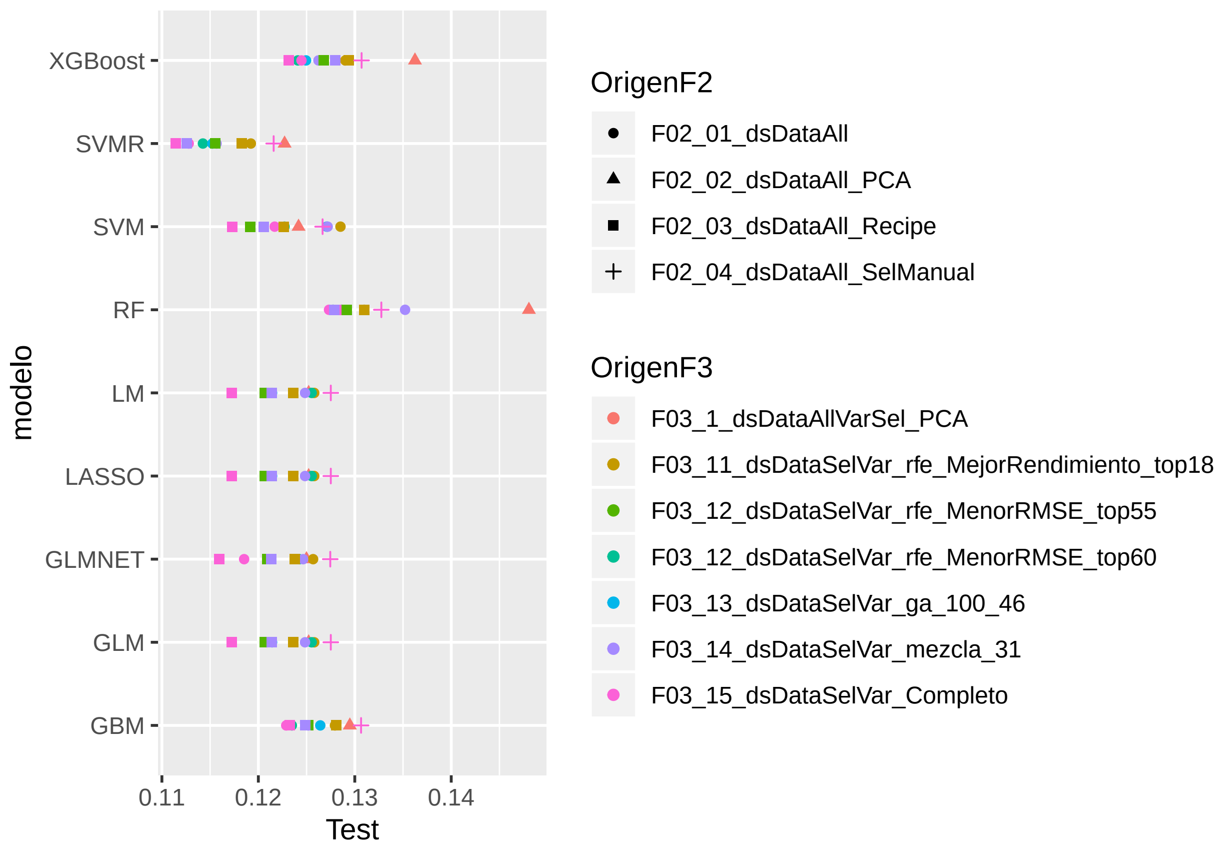
En total se han estudiado 11 modelos y se han realizado 100 entrenamientos distintos.

Seguidamente presento una gráfica con el error de Test obtenido en las distintas ejecuciones realizadas,

Estudio comparativo

Comparativa gráfica

```
ggplot(metricas,aes(x=Test, y=modelo,shape=OrigenF2,color=OrigenF3 )) +
  geom_point()
```



También presento las medias de RMSE por tipo de ingeniería de características usada, selección de predictores y tipo de modelo:

```
metricas %>%
  group_by(modelo) %>%
  summarise(media = mean(Test)) %>%
  arrange(media)
```

```
## # A tibble: 9 x 2
##   modelo  media
##   <fct>   <dbl>
## 1 SVMR    0.116
## 2 GLMNET  0.123
## 3 LASSO   0.123
## 4 GLM     0.123
```

```
## 5 LM      0.123
## 6 SVM     0.123
## 7 GBM     0.126
## 8 XGBoost 0.128
## 9 RF      0.132
```

```
metricas %>%
  group_by(OrigenF2) %>%
  summarise(media = mean(Test)) %>%
  arrange(media)
```

```
## # A tibble: 4 x 2
##   OrigenF2                   media
##   <fct>                      <dbl>
## 1 F02_03_dsDataAll_Recipe    0.122
## 2 F02_01_dsDataAll           0.124
## 3 F02_04_dsDataAll_SelManual 0.128
## 4 F02_02_dsDataAll_PCA       0.129
```

```
metricas %>%
  group_by(OrigenF3) %>%
  summarise(media = mean(Test)) %>%
  arrange(media)
```

```
## # A tibble: 7 x 2
##   OrigenF3                                     media
##   <fct>                                        <dbl>
## 1 F03_12_dsDataSelVar_rfe_MenorRMSE_top55        0.122
## 2 F03_15_dsDataSelVar_Completo                   0.123
## 3 F03_13_dsDataSelVar_ga_100_46                  0.123
## 4 F03_12_dsDataSelVar_rfe_MenorRMSE_top60        0.124
## 5 F03_14_dsDataSelVar_mezcla_31                  0.124
## 6 F03_11_dsDataSelVar_rfe_MejorRendimiento_top18 0.126
## 7 F03_1_dsDataAllVarSel_PCA                      0.129
```

Seleccionamos el modelo con mejor Error de Test

Modelo Support Vector Machines with Radial Basis Function Kernel como mejor modelo y como conjunto de predictores todos los generados con Recipe

Cargamos los datos del modelo seleccionado, de los predictores y el conjunto original

```
#dir <- './F04_Modelos/F02_03_dsDataAll_Recipe/F03_15_dsDataSelVar_Completo/'
#dir <- './F04_Modelos/F02_03_dsDataAll_Recipe/F03_14_dsDataSelVar_mezcla_31/'
dir <- './F04_Modelos/F02_03_dsDataAll_Recipe/F03_12_dsDataSelVar_rfe_MenorRMSE_top55/'

load(paste(dir,'modelo_svmRadial.RData',sep=''))

#load('./F03_SelPredictores/F02_03_dsDataAll_Recipe/F03_15_dsDataSelVar_Completo.RData')
#load('./F03_SelPredictores/F02_03_dsDataAll_Recipe/F03_14_dsDataSelVar_mezcla_31.RData')
load('./F03_SelPredictores/F02_03_dsDataAll_Recipe/F03_12_dsDataSelVar_rfe_MenorRMSE_top55.RData')

load('./F01_Datos/F01_dsDataAll.RData')
```

```r
modeloSel <- modelo_svmRadial

# Guardo resultado del calculo
save(modeloSel, file = './F04_Modelos/F04_100_modeloSel.RData')
```

Separamos los datos

Optenemos 4 dataset

dsTrain - Que a su vez se divide en dsTrain.training dsTrain.CV

dsTest

```r
dsTrain <- dsDataAllVarSel %>%
  filter(indTrain == 1) %>%
  select(SalePrice, everything()) %>%
  select(-c(Id,indTrain))

dim(dsTrain)
```

```
## [1] 1458   56
```

```r
set.seed(123)
iTrain  <- createDataPartition(y=dsTrain$SalePrice, p=0.7, list=F)

dsTrain.training <- dsTrain[iTrain, ]
dsTrain.CV       <- dsTrain[-iTrain, ]

dsTest <- dsDataAllVarSel %>%
  filter(indTrain == 0) %>%
  select(SalePrice, everything())
```

Presentación de resultados

Una vez seleccionado el modelo, lo ejecutamos contra el conjunto de test, para verificar el resultado, también he verificado la conversión a dólares.

```r
prediccionPrecioVentaLog <- predict(modeloSel, newdata = dsTrain.CV, type = "raw")

dsTrainOriginal.CV <- dsDataAll %>%
  filter(indTrain == 1) %>%
  select(Id,SalePrice)

dsTrainOriginal.CV <- dsTrainOriginal.CV[-iTrain, ]

dsTrainOriginal.CV <- dsTrainOriginal.CV %>%
    mutate(SalePrice.log = log(SalePrice))

dsTrainOriginal.CV <- cbind(dsTrainOriginal.CV, prediccionPrecioVentaLog)

dsTrainOriginal.CV <- mutate(dsTrainOriginal.CV, p = exp(prediccionPrecioVentaLog))
```

```r
RMSE(dsTrainOriginal.CV$prediccionPrecioVentaLog, dsTrainOriginal.CV$SalePrice.log)
```

```
## [1] 0.1155539
```

```r
# Compruebo el RMSE sobre el precio real (aunque en la competición se utilizará sobre los logaritmos)
RMSE(dsTrainOriginal.CV$p, dsTrainOriginal.CV$SalePrice)
```

```
## [1] 23492.12
```

```r
dsSubmission <- select(dsTrainOriginal.CV, Id, p)
```

Entrega

Generamos las predicciones para el conjunto de Test original Aplicamos la función exp a la predicción para cálcular la predicción el dolares Generamos el fichero con las predicciones

```r
prediccionPrecioVentaLog <- predict(modeloSel, newdata = dsTest, type = "raw")

p <- exp(prediccionPrecioVentaLog)

dsSubmission <- cbind(select(dsTest, Id), SalePrice=p)

dsSubmission %>%
  write_csv(path = './output/submission.csv')

# Entrega 1
# Model Support Vector Machines with Radial Basis Function Kernel set of predictors generated with Reci
# Score 0.12401 Posición 1454 / 4548


# dsSubmission %>%
#   write_csv(path = './output/submission 03.csv')

# Entrega 3
# Model Support Vector Machines with Radial Basis Function Kernel set of predictors generated with Reci
# Score 0.12508


# dsSubmission %>%
#   write_csv(path = './output/submission 04.csv')

# Entrega 4
# Model Support Vector Machines with Radial Basis Function Kernel set of predictors generated with Reci
# Score 0.12485
```

Mejora

Support Vector Machines with Radial Basis Function Kernel

```
particiones  <- 5
repeticiones <- 5

# Entrenamiento con conjunto de hiperparametros
fitControl <- trainControl(method = "repeatedcv",
                              number = particiones,
                              repeats = repeticiones,
                              returnResamp = "final",
                              verboseIter = FALSE)
```

```
# hiperparametros <- expand.grid(sigma = c(0.0001, 0.0005, 0.001)
#                                 ,C = seq(10, 100, by=10))

# hiperparametros <- expand.grid(sigma = c(0.001)
#                                 ,C = (1:20))

hiperparametros <- expand.grid(sigma = c(seq(0.0006, 0.002, by=0.0002))
                               ,C = (10:25))

# hiperparametros <- data.frame(sigma=0.001, C=14)

t <- proc.time() # Inicia el cronómetro
modelo_svmRadial <- train(SalePrice ~ .
                          , data = dsTrain.training
                          , method = "svmRadial"
                          , tuneGrid = hiperparametros
                          , metric = "RMSE"
                          , trControl = fitControl)
proc.time()-t    # Detiene el cronómetro
```

```
##    user  system elapsed
##  785.77    2.13  789.31
```

```
# Guardo resultado del calculo
# save(modelo_svmRadial, file ='./F04_Modelos/modelo_svmRadial.RData')


# Presento estudio
# fnEstudioModelo(modelo_svmRadial)
fnEstudioModelo(modelo_svmRadial,estudioParam=FALSE)
```

Modelo final

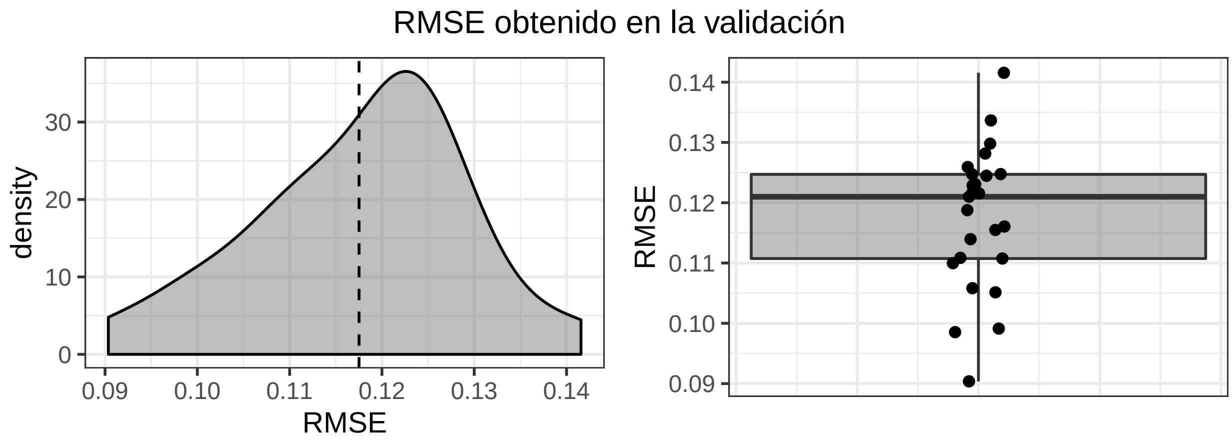
Support Vector Machine object of class "ksvm"

SV type: eps−svr (regression)
parameter : epsilon = 0.1 cost C = 24

Gaussian Radial Basis kernel function.
Hyperparameter : sigma = 0.0012

Number of Support Vectors : 655

Objective Function Value : −2121.422
Training error : 0.055008



Summary resample$RMSE

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0904	0.1110	0.1210	0.1180	0.1250	0.1420

Error de test

RMSE	Rsquared	MAE
0.11494909	0.91160004	0.08148172

```
# plot(modelo_svmRadial)

# RMSE 0.116 sigma = 0.001 and C = 10
# RMSE 0.116 sigma = 0.001 and C = 13
# RMSE 0.114 sigma = 0.001 and C = 14
# RMSE 0.114 sigma = 0.001 and C = 15 ****
# RMSE 0.114 sigma = 0.0004 and C = 15
# RMSE 0.119 sigma = 0.001 and C = 19
```

```
modeloSel = modelo_svmRadial

prediccionPrecioVentaLog <- predict(modeloSel, newdata = dsTest, type = "raw")

p <- exp(prediccionPrecioVentaLog)

dsSubmission <- cbind(select(dsTest, Id), SalePrice=p)

dsSubmission %>%
  write_csv(path = './output/submission 02.csv')

# Entrega 1
# Model Support Vector Machines with Radial Basis Function Kernel set of predictors generated with Reci
# Score 0.12406 Posición 1454 / 4548
```

XGBoost

https://www.kaggle.com/benumeh/advanced-prediction-of-house-prices-top-10

```
  # nrounds = seq(from = 200, to = nrounds, by = 50),
  # max_depth = c(2, 3, 4, 5, 6),
  # eta = c(0.025, 0.05, 0.1, 0.3),
  # gamma = 0,
  # colsample_bytree = 1,
  # min_child_weight = 1,
  # subsample = 1

  # nrounds = seq(from = 50, to = nrounds, by = 50),
  # eta = modelo_XGBoost$bestTune$eta,
  # max_depth = ifelse(modelo_XGBoost$bestTune$max_depth == 2,
  #   c(2:4), # si es valor mínimo
  #   modelo_XGBoost$bestTune$max_depth - 1:modelo_XGBoost$bestTune$max_depth + 1),
  # gamma = 0,
  # colsample_bytree = 1,
  # min_child_weight = c(1, 2, 3),
  # subsample = 1

  # nrounds = seq(from = 250, to = nrounds, by = 50),
  # eta = modelo_XGBoost$bestTune$eta,
  # max_depth = modelo_XGBoost$bestTune$max_depth,
  # gamma = 0,
  # colsample_bytree = c(0.1, 0.15, 0.2, 0.25, 0.4),
  # min_child_weight = modelo_XGBoost$bestTune$min_child_weight,
  # subsample = 1

  # nrounds = seq(from = 250, to = nrounds, by = 50),
  # eta = modelo_XGBoost$bestTune$eta,
  # max_depth = modelo_XGBoost$bestTune$max_depth,
  # gamma = 0,
  # colsample_bytree = c(0.1, 0.14, 0.16, 0.18, 0.20, 0.22, 0.24, 0.26, 0.4),
  # min_child_weight = modelo_XGBoost$bestTune$min_child_weight,
  # subsample = c(0.3, 0.4, 0.5, 0.6)

  # nrounds = seq(from = 250, to = nrounds, by = 50),
  # eta = modelo_XGBoost$bestTune$eta,
  # max_depth = modelo_XGBoost$bestTune$max_depth,
  # gamma = c(0, 0.05, 0.1, 0.5, 0.7, 0.9, 1.0),
  # colsample_bytree = modelo_XGBoost$bestTune$colsample_bytree,
  # min_child_weight = modelo_XGBoost$bestTune$min_child_weight,
  # subsample = modelo_XGBoost$bestTune$subsample

# Maximum Depth
# hiperparametros <- expand.grid(
#   nrounds = seq(from = 750, to = 2500, by = 50),
#   eta = c(0.01, 0.015, 0.025, 0.05, 0.1),
#   max_depth = modelo_XGBoost$bestTune$max_depth,
#   gamma = modelo_XGBoost$bestTune$gamma,
#   colsample_bytree = modelo_XGBoost$bestTune$colsample_bytree,
#   min_child_weight = modelo_XGBoost$bestTune$min_child_weight,
```

```r
#   subsample = modelo_XGBoost$bestTune$subsample
# )

# Maximum Depth
hiperparametros <- expand.grid(
  nrounds = seq(from = 750, to = 2000, by = 50),
  eta = 0.05, #c(0.01, 0.05, 0.1),
  max_depth = 2,
  gamma = 0,
  colsample_bytree =c(0.16, 0.18, 0.19, 0.20, 0.21, 0.22, 0.24),
  min_child_weight = 3,
  subsample =  0.3
)

t <- proc.time() # Inicia el cronómetro
modelo_XGBoost <- train(SalePrice ~ .
                        , data = dsTrain.training
                        , method = "xgbTree"
                        , tuneGrid = hiperparametros
                        , metric = "RMSE"
                        , trControl = fitControl)
proc.time()-t    # Detiene el cronómetro
```

```
##    user  system elapsed
##  861.75  576.51  683.22
```

```r
# Guardo resultado del calculo
save(modelo_XGBoost, file ='./F04_Modelos/modelo_XGBoost.RData')

#modelo_XGBoost

# Presento estudio
fnEstudioModelo(modelo_XGBoost, estudioParam = FALSE)
```

Modelo final

xgb.Booster

raw: 322.1 Kb

call:

xgboost::xgb.train(params = list(eta = param$eta, max_depth = param$max_depth,
gamma = param$gamma, colsample_bytree = param$colsample_bytree,
min_child_weight = param$min_child_weight, subsample = param$subsample),
data = x, nrounds = param$nrounds, objective = "reg:linear")

params (as set within xgb.train):

x_depth = "2", gamma = "0", colsample_bytree = "0.24", min_child_weight = "3", subsample = "0.3", objective = "reg:li

xgb.attributes:

niter

callbacks:

cb.print.evaluation(period = print_every_n)

# of features: 55

niter: 850

nfeatures : 55

d BedroomAbvGr BsmtUnfSF GarageType_Attchd MSZoning_RL WoodDeckSF MSSubClass_X60 BsmtFullBath LotS

problemType : Regression

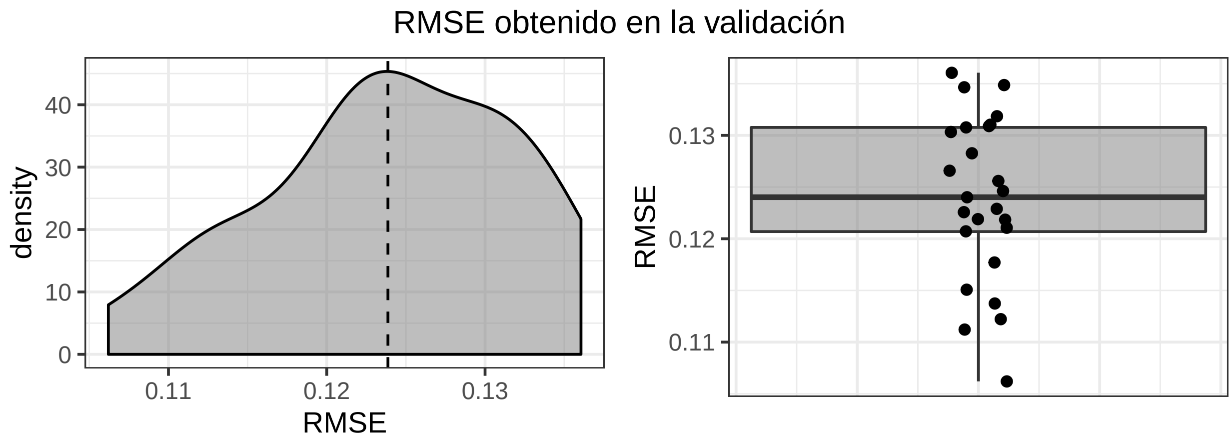
tuneValue :

	nrounds	max_depth	eta	gamma	colsample_bytree	min_child_weight
159	850	2	0.05	0	0.24	3

	subsample
159	0.3

obsLevels : NA

param :



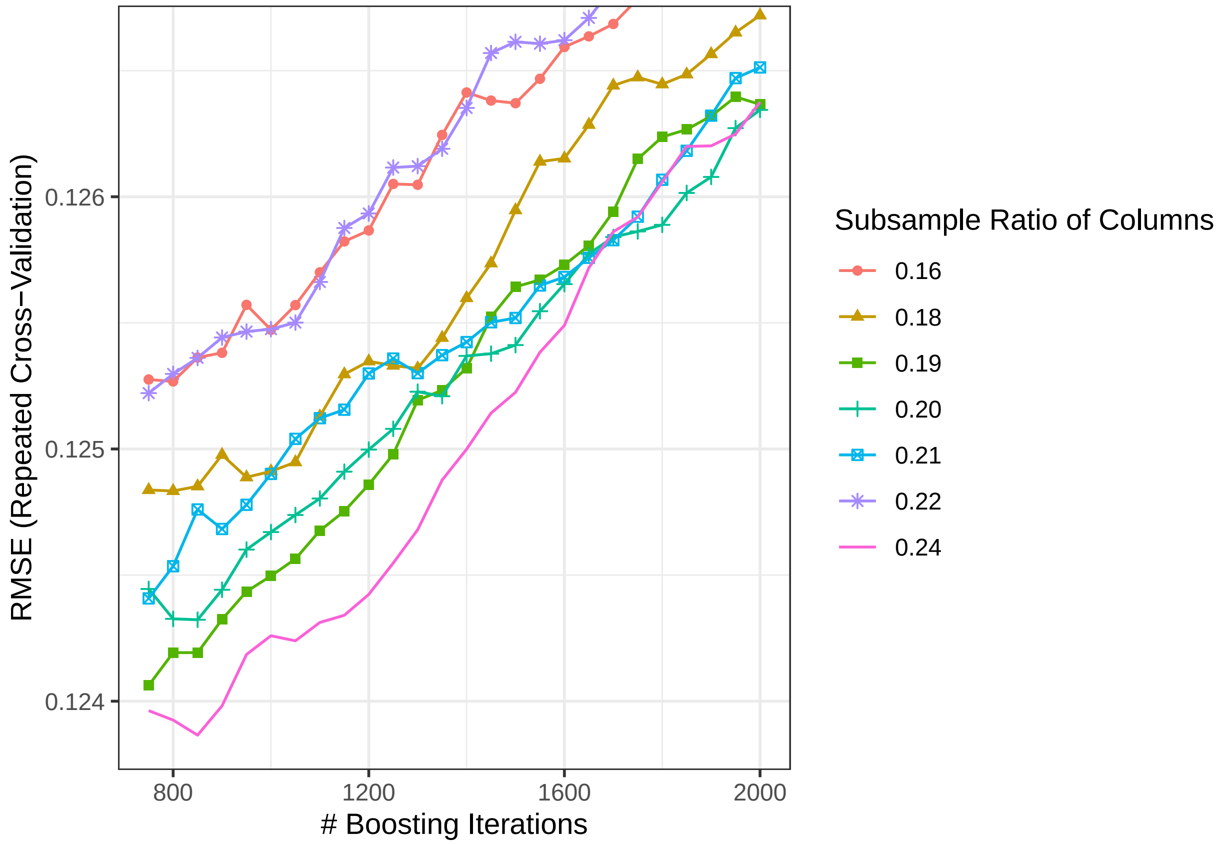
Summary resample$RMSE

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.106	0.121	0.124	0.124	0.131	0.136

Error de test

RMSE	Rsquared	MAE
0.12771898	0.89547379	0.09097713

```
tuneplot(modelo_XGBoost)
```




```r
modelo_XGBoost$bestTune
```

```
##     nrounds max_depth  eta gamma colsample_bytree min_child_weight
## 159     850         2 0.05     0             0.24                3
##     subsample
## 159       0.3
```

```r
# RMSE 0,123
# nrounds = 850, max_depth = 2, eta = 0.05, gamma = 0, colsample_bytree = 1, min_child_weight = 1 and s
# RMSE 0,125
# nrounds = 800, max_depth = 2, eta = 0.05, gamma = 0, colsample_bytree = 1, min_child_weight = 3 and s
# RMSE 0,125
# nrounds = 1000, max_depth = 2, eta = 0.05, gamma = 0, colsample_bytree = 0.1, min_child_weight = 3 an
# RMSE 0,124
# nrounds = 1150, max_depth = 2, eta = 0.05, gamma = 0, colsample_bytree = 0.2, min_child_weight = 3 an

# RMSE 0,119
# nrounds = 1100, max_depth = 2, eta = 0.05, gamma = 0, colsample_bytree = 0.15, min_child_weight = 3 a

# RMSE 0,117  Test 0.1245
# nrounds = 1150, max_depth = 2, eta = 0.05, gamma = 0, colsample_bytree = 0.15, min_child_weight = 3 a

# RMSE 0,121  Test 0.1225
# nrounds = 850, max_depth = 2, eta = 0.05, gamma = 0, colsample_bytree = 0.26, min_child_weight = 3 an
```

```
# RMSE 0,121  Test 0.1221
# nrounds = 1250, max_depth = 2, eta = 0.05, gamma = 0, colsample_bytree = 0.26, min_child_weight = 3 a

# RMSE 0,119  Test 0.126
# nrounds = 1200, max_depth = 2, eta =  0.05, gamma = 0, colsample_bytree = 0.2, min_child_weight = 3 a
```