TFM - UNED Kaggle House Prices: Advanced Regression Techniques with caret

Support Vector Machines with Radial Basis Function Kernel

Juan Carlos Santiago Culebras

2019-10-01

Proceso de evaluación del "Master de BigData – UNED".

Este documento es un resumen del TFM realizado para el "Máster Big Data y Business Analytics UNED 2018/2019" sobre la competición de Kaggle "House Prices: Advanced Regression Techniques"

Gran parte de este proyecto se ha realizado sobre la guía de RPubs "Machine Learning con R y caret" de Joaquín Amat Rodrigo, al cual doy las gracias por esta y todas sus contribuciones en RPubs.

Primeros pasos

Librerías

Realizamos la carga de las librerías necesarias

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

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

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

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

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

Cargamos datos

```
dsTrain <- read.csv("./input/train.csv")
dsTest <- read.csv("./input/test.csv")
```

Conjunto Unificado

Juntamos los datos de entrenamiento con los de test para realizar el estudio y las transformaciones pertinentes sobre todos los datos.

- Añadimos SalePrice al conjunto de Test con valor NA
- Marcamos datos de entrenamiento y test

```
dsTest <- dsTest %>%
  mutate(SalePrice = as.integer(NA), indTrain = 0)

dsDataAll <- dsTrain %>%
  mutate(indTrain = 1) %>%
  union(dsTest) %>%
  select(SalePrice, indTrain, everything())

dsDataAll$indTrain <- as.factor(dsDataAll$indTrain)

# Elimino los conjuntos originales
rm(dsTrain)
rm(dsTest)
```

Fase 01

Antes de implementar esta fase se ha realizado un estudio completo del dataset, que no incluyo por la extensión que implica. En ella se han realizado las siguientes acciones: * Análisis descriptivo * Verificación de contenido de campos nominales * Búsqueda de valores faltantes * Verificaciones de los tipos de datos * Búsqueda y eliminación de valores atípicos * Estudio de correlaciones

Seguidamente se muestra la limpieza que se ha generado.

Limpieza y preparación de los datos

Convertimos variables factor a texto, para posteriormente corregir valores

```
dsDataAll <- dsDataAll %>%
  mutate_if(is.factor, as.character)

# Dejamos el indicador de entrenamiento como factor
dsDataAll$indTrain <- as.factor(dsDataAll$indTrain)
```

Verificación y corrección de contenido de datos campos Nominales y Ordinales

Corrección de errores

```
# Normalizar valores para los campos Exterior1st / Exterior2nd -> cambio valor en excel a
#  Wd Sdng: Wood Siding
#  Wd Shng: Wood Shingles
dsDataAll <- dsDataAll %>% mutate(Exterior1st = ifelse(Exterior1st=="WdShing","WdShng",Exterior1st))
dsDataAll <- dsDataAll %>% mutate(Exterior1st = ifelse(Exterior1st=="Wd Sdng","WdSdng",Exterior1st))
dsDataAll <- dsDataAll %>% mutate(Exterior1st = ifelse(Exterior1st=="Wd Shng","WdShng",Exterior1st))

# Exterior2nd CmentBd el valor real es CemntBd (al igual que Exterior1st) filter(dsDataAll,Exterior1st=
dsDataAll <- dsDataAll %>% mutate(Exterior2nd = ifelse(Exterior2nd=="CmentBd","CemntBd",Exterior2nd))
dsDataAll <- dsDataAll %>% mutate(Exterior2nd = ifelse(Exterior2nd=="Wd Sdng","WdSdng",Exterior2nd))
dsDataAll <- dsDataAll %>% mutate(Exterior2nd = ifelse(Exterior2nd=="Wd Shng","WdShng",Exterior2nd))
dsDataAll <- dsDataAll %>% mutate(Exterior2nd = ifelse(Exterior2nd=="Brk Cmn","BrkComm",Exterior2nd))

dsDataAll <- dsDataAll %>% mutate(MSZoning = ifelse(MSZoning=="C (all)","C",MSZoning))
dsDataAll <- dsDataAll %>% mutate(RoofMatl = ifelse(RoofMatl=="Tar&Grv","Tar",RoofMatl))
```

Valores faltantes - Missing Data

La gran mayoría de algoritmos no aceptan observaciones incompletas, por lo que, cuando el set de datos contiene valores ausentes, se puede:

- Eliminar aquellas observaciones que estén incompletas.
- Eliminar aquellas variables que contengan valores ausentes.
- Tratar de estimar los valores ausentes empleando el resto de información disponible (imputación).

Identifico valores faltantes

```
missingData <- dsDataAll %>%
  summarise_all(funs(sum(is.na(.)))) %>%
  gather("column") %>%
  rename(NumNAs = value) %>%
  mutate(PrcNAs = NumNAs/nrow(dsDataAll)) %>%
  filter(NumNAs!=0) %>%
  arrange(desc(PrcNAs))

head(missingData) # presento solo los primeros
```

```
##         column NumNAs    PrcNAs
## 1       PoolQC   2909 0.9965742
## 2 MiscFeature   2814 0.9640288
## 3        Alley   2721 0.9321686
## 4        Fence   2348 0.8043851
## 5    SalePrice   1459 0.4998287
## 6 FireplaceQu   1420 0.4864680
```

Las variables con un porcentaje de valores asuntes muy alto (>80%) las excluimos del modelo, ya que pueden dar errores al realizar subconjuntos de datos pare entrenar y validar los modelos.

```
# PoolQC - Calidad de la piscina
# MiscFeature -  características varias no cubiertas en otras categorías
# Alley - tipo de acceso al callejón
# Fence - calidad de la cerca

eliminar <- filter(missingData, PrcNAs > 0.80) %>% select(column)

dsDataAll <- dsDataAll %>%
    select(-c(eliminar$column))

rm(eliminar)
```

Del resto de valores pendientes realizo estudio y modifico valores faltantes.

```
dsDataAll <- select(dsDataAll, -Utilities)

#Ordinales asigno texto None
dsDataAll <- mutate(dsDataAll, FireplaceQu = ifelse(is.na(FireplaceQu),"None",FireplaceQu))
dsDataAll <- mutate(dsDataAll, GarageCond = ifelse(is.na(GarageCond),"None",GarageCond))
dsDataAll <- mutate(dsDataAll, GarageQual = ifelse(is.na(GarageQual),"None",GarageQual))
dsDataAll <- mutate(dsDataAll, GarageFinish = ifelse(is.na(GarageFinish),"None",GarageFinish))
dsDataAll <- mutate(dsDataAll, GarageType = ifelse(is.na(GarageType),"None",GarageType))
dsDataAll <- mutate(dsDataAll, BsmtFinType2 = ifelse(is.na(BsmtFinType2),"None",BsmtFinType2))
dsDataAll <- mutate(dsDataAll, BsmtQual = ifelse(is.na(BsmtQual),"None",BsmtQual))
dsDataAll <- mutate(dsDataAll, BsmtCond = ifelse(is.na(BsmtCond),"None",BsmtCond))
dsDataAll <- mutate(dsDataAll, BsmtExposure = ifelse(is.na(BsmtExposure),"None",BsmtExposure))
dsDataAll <- mutate(dsDataAll, BsmtFinType1 = ifelse(is.na(BsmtFinType1),"None",BsmtFinType1))
dsDataAll <- mutate(dsDataAll, MasVnrType = ifelse(is.na(MasVnrType),"None",MasVnrType))

#Discretas y continuas 0
dsDataAll <- mutate(dsDataAll, GarageYrBlt = ifelse(is.na(GarageYrBlt),0,GarageYrBlt))
dsDataAll <- mutate(dsDataAll, GarageCars = ifelse(is.na(GarageCars),0,GarageCars))
dsDataAll <- mutate(dsDataAll, GarageArea = ifelse(is.na(GarageArea),0,GarageArea))
dsDataAll <- mutate(dsDataAll, TotalBsmtSF = ifelse(is.na(TotalBsmtSF),0,TotalBsmtSF))
dsDataAll <- mutate(dsDataAll, BsmtFinSF1 = ifelse(is.na(BsmtFinSF1),0,BsmtFinSF1))
dsDataAll <- mutate(dsDataAll, BsmtFinSF2 = ifelse(is.na(BsmtFinSF2),0,BsmtFinSF2))
dsDataAll <- mutate(dsDataAll, BsmtUnfSF = ifelse(is.na(BsmtUnfSF),0,BsmtUnfSF))
dsDataAll <- mutate(dsDataAll, BsmtFullBath = ifelse(is.na(BsmtFullBath),0,BsmtFullBath))
dsDataAll <- mutate(dsDataAll, BsmtHalfBath = ifelse(is.na(BsmtHalfBath),0,BsmtHalfBath))
dsDataAll <- mutate(dsDataAll, MasVnrArea = ifelse(is.na(MasVnrArea),0,MasVnrArea))

dsDataAll <- mutate(dsDataAll, LotFrontage = ifelse(is.na(LotFrontage),mean(dsDataAll$LotFrontage,na.rm
```

```
# Nominales asigno valor medio
dsDataAll <- mutate(dsDataAll, MSZoning = ifelse(is.na(MSZoning),"RL",MSZoning))
dsDataAll <- mutate(dsDataAll, Functional = ifelse(is.na(Functional),"Typ",Functional))
dsDataAll <- mutate(dsDataAll, Exterior1st = ifelse(is.na(Exterior1st),"VinylSd",Exterior1st))
dsDataAll <- mutate(dsDataAll, Exterior2nd = ifelse(is.na(Exterior2nd),"VinylSd",Exterior2nd))
dsDataAll <- mutate(dsDataAll, Electrical = ifelse(is.na(Electrical),"SBrkr",Electrical))
dsDataAll <- mutate(dsDataAll, KitchenQual = ifelse(is.na(KitchenQual),"TA",KitchenQual))
dsDataAll <- mutate(dsDataAll, SaleType = ifelse(is.na(SaleType),"WD",SaleType))

missingData <- dsDataAll %>%
  summarise_all(funs(sum(is.na(.)))) %>%
  gather("column") %>%
  rename(NumNAs = value) %>%
  mutate(PrcNAs = NumNAs/nrow(dsDataAll)) %>%
  filter(NumNAs!=0) %>%
  arrange(desc(PrcNAs))

rm(missingData)
```

Conversión de variables ordinales a numéricas

```
dsDataAll$ExterQual <- factor(dsDataAll$ExterQual, levels = rev(c("Ex","Gd","TA","Fa","Po")))
dsDataAll$ExterQual <- as.numeric(c(dsDataAll$ExterQual))

dsDataAll$ExterCond <- factor(dsDataAll$ExterCond, levels = rev(c("Ex","Gd","TA","Fa","Po")))
dsDataAll$ExterCond <- as.numeric(c(dsDataAll$ExterCond))

dsDataAll$LotShape <- factor(dsDataAll$LotShape, levels = rev(c("Reg","IR1","IR2","IR3")))
dsDataAll$LotShape <- as.numeric(c(dsDataAll$LotShape))

dsDataAll$LandSlope <- factor(dsDataAll$LandSlope, levels = rev(c("Gtl","Mod","Sev")))
dsDataAll$LandSlope <- as.numeric(c(dsDataAll$LandSlope))

dsDataAll$BsmtQual <- factor(dsDataAll$BsmtQual, levels = rev(c("Ex","Gd","TA","Fa","Po","None")))
dsDataAll$BsmtQual <- as.numeric(c(dsDataAll$BsmtQual))-1

dsDataAll$BsmtCond <- factor(dsDataAll$BsmtCond, levels = rev(c("Ex","Gd","TA","Fa","Po","None")))
dsDataAll$BsmtCond <- as.numeric(c(dsDataAll$BsmtCond))-1

dsDataAll$BsmtExposure <- factor(dsDataAll$BsmtExposure, levels = rev(c("Gd","Av","Mn","No","None")))
dsDataAll$BsmtExposure <- as.numeric(c(dsDataAll$BsmtExposure))-1

dsDataAll$BsmtFinType1 <- factor(dsDataAll$BsmtFinType1, levels = rev(c("GLQ","ALQ","BLQ","Rec","LwQ","U
dsDataAll$BsmtFinType1 <- as.numeric(c(dsDataAll$BsmtFinType1))-1

dsDataAll$BsmtFinType2 <- factor(dsDataAll$BsmtFinType2, levels = rev(c("GLQ","ALQ","BLQ","Rec","LwQ","U
dsDataAll$BsmtFinType2 <- as.numeric(c(dsDataAll$BsmtFinType2))-1

dsDataAll$HeatingQC <- factor(dsDataAll$HeatingQC, levels = rev(c("Ex","Gd","TA","Fa","Po")))
dsDataAll$HeatingQC <- as.numeric(c(dsDataAll$HeatingQC))
```

```
dsDataAll$Electrical <- factor(dsDataAll$Electrical, levels = rev(c("SBrkr","FuseA","FuseF","FuseP","Mi
dsDataAll$Electrical <- as.numeric(c(dsDataAll$Electrical))

dsDataAll$KitchenQual <- factor(dsDataAll$KitchenQual, levels = rev(c("Ex","Gd","TA","Fa","Po")))
dsDataAll$KitchenQual <- as.numeric(c(dsDataAll$KitchenQual))

dsDataAll$Functional <- factor(dsDataAll$Functional, levels = rev(c("Typ","Min1","Min2","Mod","Maj1","Ma
dsDataAll$Functional <- as.numeric(c(dsDataAll$Functional))

dsDataAll$FireplaceQu <- factor(dsDataAll$FireplaceQu, levels = rev(c("Ex","Gd","TA","Fa","Po","None")))
dsDataAll$FireplaceQu <- as.numeric(c(dsDataAll$FireplaceQu))-1

dsDataAll$GarageFinish <- factor(dsDataAll$GarageFinish, levels = rev(c("Fin","RFn","Unf","None")))
dsDataAll$GarageFinish <- as.numeric(c(dsDataAll$GarageFinish))-1

dsDataAll$GarageQual <- factor(dsDataAll$GarageQual, levels = rev(c("Ex","Gd","TA","Fa","Po","None")))
dsDataAll$GarageQual <- as.numeric(c(dsDataAll$GarageQual))-1

dsDataAll$GarageCond <- factor(dsDataAll$GarageCond, levels = rev(c("Ex","Gd","TA","Fa","Po","None")))
dsDataAll$GarageCond <- as.numeric(c(dsDataAll$GarageCond))-1

dsDataAll$PavedDrive <- factor(dsDataAll$PavedDrive, levels = rev(c("Y","P","N")))
dsDataAll$PavedDrive <- as.numeric(c(dsDataAll$PavedDrive))
```

Variables Nominales paso a factor

Todas las variables que quedan con carácter son nominales

```
dsDataAll <- dsDataAll %>%
    mutate_if(is.character, as.factor)

dsDataAll$MSSubClass <- as.factor(dsDataAll$MSSubClass)
```

Variables con solo dos valores se pasan a numéricas

$ Street : Factor w/ 2 levels “Grvl”,“Pave” $ CentralAir : Factor w/ 2 levels “N”,“Y”

```
#  Grvl:  12
#  Pave:2907

dsDataAll$StreetPave[dsDataAll$Street != "Pave"] <- "0"
dsDataAll$StreetPave[dsDataAll$Street == "Pave"] <- "1"
dsDataAll$StreetPave <- as.numeric(dsDataAll$StreetPave)
dsDataAll <- select(dsDataAll, -Street)

# CentralAir
# Y:2723
# N: 196

dsDataAll$CentralAir <- as.character(dsDataAll$CentralAir)
dsDataAll$CentralAir[dsDataAll$CentralAir != "Y"] <- "0"
dsDataAll$CentralAir[dsDataAll$CentralAir == "Y"] <- "1"
dsDataAll$CentralAir <- as.numeric(dsDataAll$CentralAir)
```

Valores atípicos - outliers

GrLivArea superficie habitable por encima del nivel del suelo (pies cuadrados) Existen 2 valores atípicos son muy altos para el precio que tienen en el conjunto de entrenamiento, estas filas se eliminarán al ser esta una variable principal para el proceso de predicción

```
# Se eliminan las filas
eliminar <- dsDataAll %>%
    filter(indTrain==1&GrLivArea>4500) %>%
    select(Id, GrLivArea, SalePrice, indTrain)

dsDataAll <- dsDataAll %>%
  anti_join(eliminar,by="Id")

rm(eliminar)
```

LotArea tamaño del lote en pies cuadrados Existen 4 valores claramente fuera de rango, creo variable nueva actualizándolos con los valores con la mediana según el tipo de construcción

```
# Calculo mediana por tipo de construcción
lotAreaMedian <- select(dsDataAll,BldgType,LotArea) %>%
    group_by(BldgType) %>%
    summarise(medianLotArea = median(LotArea))

f <- function(x){
  a <- as.numeric(lotAreaMedian[lotAreaMedian$BldgType==x,2])
  return(a)
}

# Seleccion Outliers
outlier_values <- as.data.frame(boxplot.stats(dsDataAll$LotArea)$out)
names(outlier_values) = "LotArea"
outlier_values$LotArea <- as.numeric(outlier_values$LotArea)
outlier_values <- outlier_values %>%
  arrange(desc(LotArea)) %>%
  top_n(4)

# Modificación directa
dsDataAll <- dsDataAll %>%
  rowwise() %>%
    mutate(LotArea = ifelse(LotArea>=115149,f(BldgType),LotArea))

rm(outlier_values)
rm(lotAreaMedian)
rm(f)
```

LowQualFinSF pies cuadrados terminados de baja calidad (todos los pisos) Parece que existe un par de valores extraños, creo variable nueva y actualizo a la mediana de todos los valores no cero

```
a <- dsDataAll %>%
    select(LowQualFinSF) %>%
    filter(LowQualFinSF!=0)
```

```
# Parece que existe un par de valores extraños
# Actualizo a la mediana de todos los valores no cero
medianLowQualFinSF <- median(a$LowQualFinSF)

#select(data,Id,LowQualFinSF) %>% filter(LowQualFinSF>600)

# Modificación directa
dsDataAll <- dsDataAll %>%
  rowwise() %>%
    mutate(LowQualFinSF = ifelse(LowQualFinSF>600,medianLowQualFinSF,LowQualFinSF))

rm(medianLowQualFinSF)
rm(a)
```

MasVnrArea área de revestimiento de mampostería en pies cuadrados Identifico un valor extraño

```
a <- dsDataAll %>%
    select(MasVnrArea) %>%
    filter(MasVnrArea!=0)

# Parece que existe un valor extraño
# Actualizo a la mediana de todos los valores no cero
medianMasVnrArea <- median(a$MasVnrArea)

# Modificación directa
dsDataAll <- dsDataAll %>%
  rowwise() %>%
    mutate(MasVnrArea = ifelse(MasVnrArea>1500,medianMasVnrArea,MasVnrArea))


rm(medianMasVnrArea)
rm(a)
```

WoodDeckSF área de cubierta de madera en pies cuadrados Parece que existe un valor extraño, sin embargo, existe la posibilidad de que sea una casa completamente de madera, pero como el valor está en el conjunto de test no se puede usar para entrenar y el modelo resultante no podrá calcular precios para casas solo de madera, por lo que actualizo el valor a la mediana según la superficie

```
a <- dsDataAll %>%
    filter(WoodDeckSF!=0 & GrLivArea > 1300 & GrLivArea < 1400) %>%
    select(WoodDeckSF)

medianWoodDeckSF <- median(a$WoodDeckSF)

# Modificación directa
dsDataAll <- dsDataAll %>%
  rowwise() %>%
    mutate(WoodDeckSF = ifelse(WoodDeckSF>1500,medianWoodDeckSF,WoodDeckSF))

rm(medianWoodDeckSF)
rm(a)
```

OpenPorchSF área de porche abierto en pies cuadrados Identifico un par de valores extraños Uno en el

conjunto de entrenamiento, con un porche muy grande y un precio bajo y Otro en el conjunto de test, con una superficie muy grande

```
a <- dsDataAll %>%
    filter(OpenPorchSF!=0 & GrLivArea > 700 & GrLivArea < 750) %>%
    select(OpenPorchSF)

medianOpenPorchSF <- median(a$OpenPorchSF)

# Modificación directa
dsDataAll <- dsDataAll %>%
  rowwise() %>%
    mutate(OpenPorchSF = ifelse(OpenPorchSF>500&GrLivArea<1000,medianOpenPorchSF,OpenPorchSF))


a <- dsDataAll %>%
    filter(OpenPorchSF!=0 & GrLivArea > 2550 & GrLivArea < 2650) %>%
    select(OpenPorchSF)

medianOpenPorchSF <- median(a$OpenPorchSF)

# Modificación directa
dsDataAll <- dsDataAll %>%
  rowwise() %>%
    mutate(OpenPorchSF = ifelse(OpenPorchSF>600,medianOpenPorchSF,OpenPorchSF))

rm(medianOpenPorchSF)
rm(a)
```

EnclosedPorch área de porche cerrado en pies cuadrados Parece que existen un valor extraño en el conjunto de test, con una superficie muy grande

```
## Asigno mediana segun el area
a <- dsDataAll %>%
    filter(EnclosedPorch!=0 & GrLivArea > 1800 & GrLivArea < 1850) %>%
    select(EnclosedPorch)

medianEnclosedPorch <- median(a$EnclosedPorch)

# Modificación directa
dsDataAll <- dsDataAll %>%
  rowwise() %>%
    mutate(EnclosedPorch = ifelse(OpenPorchSF>600,medianEnclosedPorch,EnclosedPorch))

rm(medianEnclosedPorch)
rm(a)
```

YearRemodAdd Año de remodelación Parece que a las casas que se construyeron antes de 1950 se les puso una fecha de remodelación 1950 Modifico la fecha de remodelación para casas anteriores a 1950 asignándoles la fecha de construcción

```
dsDataAll <- dsDataAll %>%
  mutate(YearRemodAdd = ifelse(YearBuilt<1950 & YearRemodAdd==1950,YearBuilt,YearRemodAdd))
```

GarageYrBlt año en que se construyó el garaje Los datos de esta variable parecen incorrectos (Elimino)

```
dsDataAll <- select(dsDataAll, -GarageYrBlt)
```

Fase 02 Ingeniería de características con recipe

En esta fase se intentará modificar el conjunto de características para aumentar su eficacia predictiva, para lo cual se utilizará el paquete "recipes".

Preparamos datos para realizar la ingeniería con recipe

La variable objetivo SalePrice no se modificará con caret, se normalizará directamente con la función log ya que la competición se basa en la medida RMSE del logaritmo de SalePrice.

```
dsDataAllRecipe <- dsDataAll %>%
  mutate(SalePrice = log(SalePrice))
```

Separamos los datos

Optenemos 3 dataset (dsTest no se utilizará en esta fase)

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

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

dim(dsTrain)
```

```
## [1] 1458   74
```

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

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

Objeto inicial

Se crea un objeto recipe() con la variable respuesta y los predictores

```
objRecipe <- recipe(formula = SalePrice ~ ., data =  dsTrain.training)
```

Generamos los pasos

*Eliminación variables con varianza próxima a cero: Si una variable tiene casi todas las observaciones con un mismo valor, su varianza será próxima a cero. Estas variables pueden añadir más ruido que información, también dan problemas cuando se seleccionan los conjuntos de entrenamiento ya que si en la variable solo queda un valor puede producir que el entrenamiento sea erróneo.

*Estandarización y escalado, sobre las variables numéricas: En el análisis inicial se detectó que la mayoría de las variables continuas están sesgadas.

*Binarización de variables nominales (dummy): Todas las variables nominales se convertirán a numéricas binarias, para ello cada variable generara nuevas variables una por cada valor existente en el conjunto de datos, indicando como valor 0 o 1, ausencia o presencia del valor.

*Se repite la eliminación de variables con varianza cero, ya que muchas de las variables dummy tendrán este problema. Una varianza cero puede generar problemas con algunos modelos.

```
# Eliminación variables con varianza próxima a cero
objRecipe <- objRecipe %>% step_nzv(all_predictors())

# Estandarización y escalado, sobre las variables numéricas
objRecipe <- objRecipe %>% step_center(all_numeric(), -SalePrice)
objRecipe <- objRecipe %>% step_scale(all_numeric(), -SalePrice)

#Binarización de variables nominales
objRecipe <- objRecipe %>% step_dummy(all_nominal(), -all_outcomes())

# Eliminación variables con varianza próxima a cero para DUMMY
objRecipe <- objRecipe %>% step_nzv(all_predictors())
```

Comprobamos el objeto Recipe

```
objRecipe
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         73
##
## Operations:
##
## Sparse, unbalanced variable filter on all_predictors
## Centering for all_numeric, -, SalePrice
## Scaling for all_numeric, -, SalePrice
## Dummy variables from all_nominal, -, all_outcomes()
## Sparse, unbalanced variable filter on all_predictors
```

Entrenamiento

Se realiza un entrenamiento del objeto recipe, donde se aprenden los parámetros necesarios para realizar las transformaciones y se muestra el resultado del entrenamiento

```
trained_recipe <- prep(objRecipe, training = dsTrain.training)
trained_recipe
```

```
## Data Recipe
##
```

```
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         73
##
## Training data contained 1023 data points and no missing data.
##
## Operations:
##
## Sparse, unbalanced variable filter removed LandContour, LandSlope, ... [trained]
## Centering for LotFrontage, LotArea, ... [trained]
## Scaling for LotFrontage, LotArea, ... [trained]
## Dummy variables from MSSubClass, MSZoning, LotConfig, ... [trained]
## Sparse, unbalanced variable filter removed MSSubClass_X30, ... [trained]
```

Se aplican las transformaciones a los conjuntos deseados, en nuestro caso se realizará sobre todos los datos conjunto de entrenamiento y de test, ya que será la entrada de la siguiente fase.

```
dsDataAllRecipe.prep <- bake(trained_recipe, new_data = dsDataAllRecipe)

#Guardamos en el dataset dsDataAll los resultados incluidos los campos Id e indTrain
dsDataAll <- cbind(dsDataAllRecipe[,1:3], dsDataAllRecipe.prep[,-1])
```

Fase 03 Selección de predictores con caret

El objetivo es utilizar una de las muchas formas de reducir el volumen de características que ofrece caret, de tal forma que únicamente los predictores que están relacionados con la variable respuestas se incluyan en el modelo.

Los métodos de selección de predictores se pueden clasificar como:

- Métodos wrapper: Evalúan múltiples modelos utilizando procedimientos que agregan y/o eliminan predictores para encontrar la combinación óptima que maximice el rendimiento del modelo, son algoritmos de búsqueda donde los predictores son las entradas y el modelo a optimizar es la salida.
  - Eliminación de características recursivas
  - Algoritmos genéticos
  - Simulated annealing
- Métodos de filtro: Analizan la relación que tiene cada predictor con la variable respuesta, evaluando la relevancia de los predictores fuera de los modelos y seleccionando los que pasan algún criterio.

Además, cada uno de estos métodos puede utilizar distintos algoritmos (regresión lineal, naive bayes, random forest) y métodos de entrenamiento (Validación cruzada o bootstrapping).

El método seleccionado para realizar la selección de predictores es Eliminación Recursiva con Randon Forest y el método de evaluación, la validación cruzada con 5 particiones y 5 repeticiones,

Separamos de nuevo los datos

Separamos los datos modificados con recipe

Obtenemos 4 dataset

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

dsTest

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

dim(dsTrain)
```

```
## [1] 1458   87
```

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

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

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

Eliminación Recursiva con Random Forest y validación cruzada

EJECUCIÓN

```
#Defino conjuntos de número de predictores a probar.
subsets <- c(5, 10, 15, 16, 17, 18, 19, 20, 25, 30, 40, 50, 60, 70)

control <- rfeControl(functions = rfFuncs
                      ,method = "repeatedcv" # Validación cruzada
                      ,repeats = 5
                      ,verbose = FALSE)

t <- proc.time() # Inicia el cronómetro
rf_rfe <- rfe(SalePrice ~ .
              , data = dsTrain.training
              , sizes = subsets
              , metric = "RMSE"
              , rfeControl = control)
proc.time()-t    # Detiene el cronómetro
```

```
##    user  system elapsed
## 3457.78   11.89 3477.17
```

```
#load('./F03_SelPredictores/F02_03_dsDataAll_Recipe/F03_2_rfe_rf.RData')

#    user  system elapsed
# 3044.24   16.42 3072.58

rf_rfe
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold, repeated 5 times)
##
## Resampling performance over subset size:
##
##  Variables   RMSE Rsquared     MAE  RMSESD RsquaredSD   MAESD Selected
##          5 0.1749   0.8179 0.12761 0.02175    0.04152 0.01254
##         10 0.1481   0.8692 0.10442 0.02096    0.03343 0.01091
##         15 0.1378   0.8885 0.09520 0.02033    0.02837 0.01039
##         16 0.1371   0.8899 0.09467 0.02019    0.02784 0.01036
##         17 0.1368   0.8904 0.09450 0.02006    0.02753 0.01035
##         18 0.1368   0.8900 0.09457 0.02059    0.02879 0.01092
##         19 0.1363   0.8913 0.09425 0.02036    0.02782 0.01051
##         20 0.1364   0.8914 0.09427 0.02057    0.02771 0.01065
##         25 0.1367   0.8912 0.09432 0.02017    0.02752 0.01049
##         30 0.1359   0.8928 0.09365 0.02046    0.02704 0.01040
##         40 0.1359   0.8931 0.09342 0.02017    0.02669 0.01024
##         50 0.1349   0.8950 0.09271 0.01973    0.02588 0.01020
##         60 0.1350   0.8948 0.09267 0.01987    0.02620 0.01033
##         70 0.1348   0.8954 0.09252 0.01990    0.02604 0.01044        *
##         86 0.1350   0.8955 0.09241 0.02022    0.02579 0.01056
##
## The top 5 variables (out of 70):
##    GrLivArea, OverallQual, TotalBsmtSF, X1stFlrSF, BsmtFinSF1
```

Estudio de resultados

Presentamos gráficamente la evolución de RMSE según el número de predictores seleccionados.

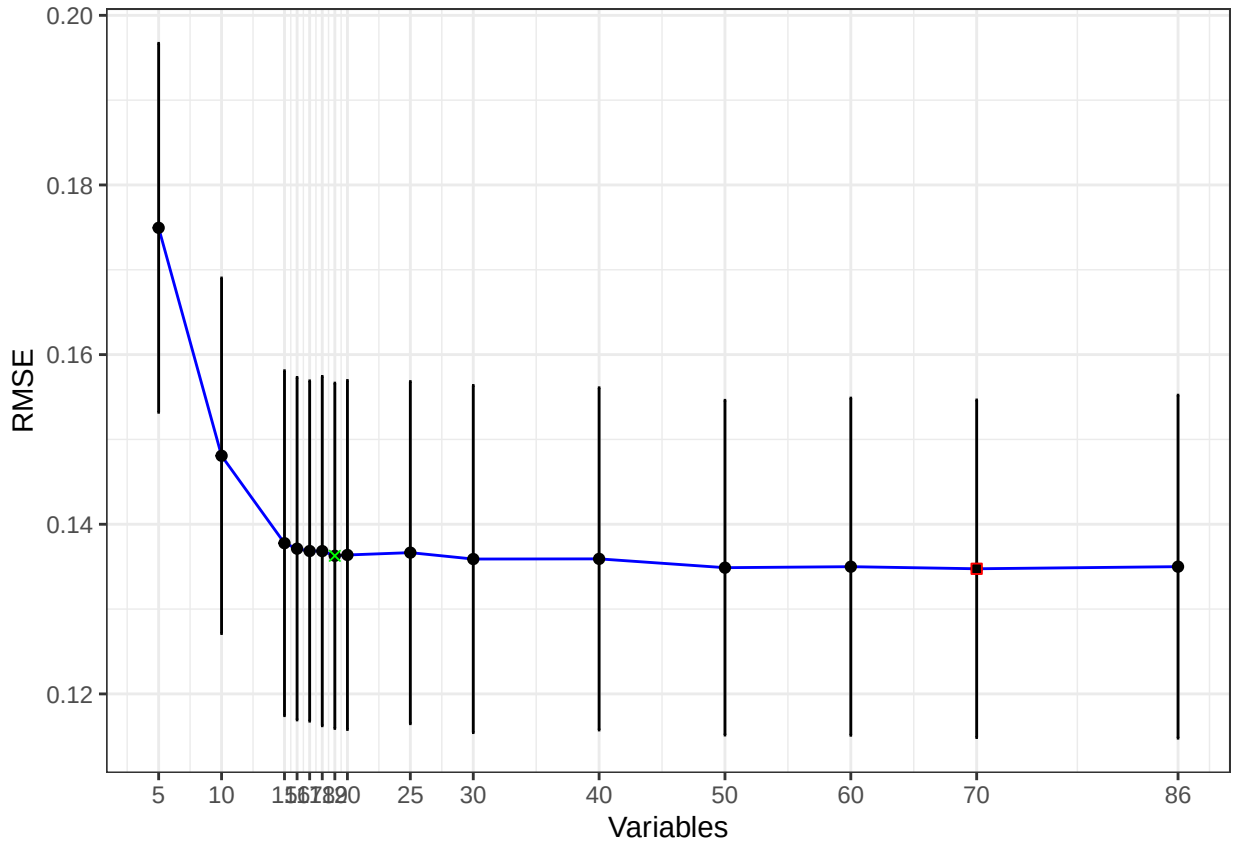
Se marcan los números de predictores con mejor RMSE Absoluto y los que mejor Rendimiento ofrecen según la función pickSizeTolerance.

```
dsResults <- rf_rfe$results

# Métricas promedio de cada tamaño
dsResults %>%
  group_by(Variables) %>%
  summarise(media_RMSE = mean(RMSE), media_Rsquared = mean(Rsquared)) %>%
  arrange(media_RMSE)
```

```
## # A tibble: 15 x 3
##    Variables media_RMSE media_Rsquared
##        <dbl>      <dbl>          <dbl>
##  1        70      0.135          0.895
```

```
##  2        50      0.135          0.895
##  3        60      0.135          0.895
##  4        86      0.135          0.895
##  5        30      0.136          0.893
##  6        40      0.136          0.893
##  7        19      0.136          0.891
##  8        20      0.136          0.891
##  9        25      0.137          0.891
## 10        18      0.137          0.890
## 11        17      0.137          0.890
## 12        16      0.137          0.890
## 13        15      0.138          0.889
## 14        10      0.148          0.869
## 15         5      0.175          0.818
```

```
mejorAbsoluto <- pickSizeBest(select(dsResults,RMSE,Variables)
                              , metric = "RMSE"
                              , maximize = FALSE)
mejorRendimiento <- pickSizeTolerance(select(dsResults,RMSE,Variables)
                                    , metric = "RMSE"
                                    , maximize = FALSE)

## Percent Loss in performance (positive)
# ToDo: example$PctLoss <- (example$RMSE - min(example$RMSE))/min(example$RMSE)*100

# Gráfica de disminución de RMSE
ggplot(data = dsResults, aes(x = Variables, y = RMSE)) +
  geom_line(color = "blue") +
  scale_x_continuous(breaks = unique(dsResults$Variables)) +
  geom_point() +
  geom_errorbar(aes(ymin = RMSE - RMSESD, ymax = RMSE + RMSESD),
                width = 0.2) +

  geom_point(data = filter(dsResults, Variables==mejorAbsoluto)
             , shape=0, cex= 1.5, color = "red") +

  geom_point(data = filter(dsResults, Variables==mejorRendimiento)
             , shape = 4, cex= 1.5, color = "green") +

  theme_bw()
```



Selecciono las variables indicadas como mejor rendimiento, utilizando el principio de parsimonia que nos indica que el modelo más simple es posiblemente el mejor.

```
dsVarSel001 <- as.data.frame(rf_rfe$optVariables) %>%
  rename(Campo = 1) %>%
  rownames_to_column("Orden") %>%
  mutate(Orden = as.numeric(Orden), Campo = as.character(Campo))

# Selecciono los 25 primeros selectores.
dsVarSel <- dsVarSel001 %>% top_n(0-mejorAbsoluto, Orden)

# Guardo un data set con los valores seleccionados
dsDataAllVarSel <- dsDataAll %>%
    select(SalePrice, indTrain, Id, c(dsVarSel$Campo))
```

Eliminamos objetos que no se seguirán usando

```
rm(list= ls()[!(ls() == 'dsDataAllVarSel')])
```

Fase 04 Selección de modelos predictivos con caret

En esta fase aplicaremos distintos algoritmos de machine learning para generar modelos de regresión, que sean capaces de predecir la variable objetivo (SalePrice).

Creamos una función de apoyo para presentar los resultados obtenidos al entrenar un modelo.

Presenta:

- Modelo final: Resumen del modelo seleccionado en el entrenamiento y cuáles son los mejores parámetros.
- Estudio del RMSE obtenido en la validación
  - Para ello se presentan dos gráficas:
    - Gráfica de densidad de RMSE de los distintos cálculos realizados en el entrenamiento.
    - Boxplot de RMSE de los distintos cálculos
  - El resumen del RMSE obtenido para el entrenamiento: Summary de resample$RMSE
  - Y por último el error de test: para ello la función llama a predict sobre el modelo con el conjunto dsTrain.CV

```
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())

  #trellis.par.set(caretTheme())
  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")
  }

}

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

Separamos los datos

Obtenemos 4 dataset

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

dsTest

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

dim(dsTrain)
```

```
## [1] 1458   71
```

```
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())
```

Definimos entrenamiento

En esta sección se entrenarán distintos modelos para evaluar cual puede ser el mejor.

Sobre cada modelo se realizará:

- Entrenamiento

- Ajuste de hiperparámetros
- Evaluación mediante validación cruzada

```
particiones  <- 5
repeticiones <- 5

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

Modelos

He seleccionado 5 de los modelos que he probado en el TFM.

Support Vector Machines with Radial Basis Function Kernel

Aunque Las máquinas de vectores soporte fueron pensadas para resolver problemas de clasificación también pueden adaptarse para resolver problemas de regresión, estos modelos dan bastante buenos resultados cuando la variable objetivo no es separables linealmente dentro del espacio vectorial de los predictores y evitan en gran medida el problema del sobreajuste a los ejemplos de entrenamiento, por ello es una buena elección para este problema.

Las máquinas de soporte utilizan una función denominada Kernel para la búsqueda del hiperplano de separación, para ello mapean los datos en espacios de dimensiones superiores con la esperanza de que en este espacio de dimensiones superiores los datos puedan separarse más fácilmente o estar mejor estructurados.

Radial Basis permite seleccionar círculos (o hiperesferas)

```
# hiperparametros <- expand.grid(sigma = c(0.0005, 0.001, 0.005)
#                                ,C = c(1 , 20, 50, 100, 150, 200))
# RMSE 0.128 TEST 0.11829

# hiperparametros <- expand.grid(sigma = c(seq(0.0006, 0.002, by=0.0002))
#                                ,C = (10:25))
# RMSE 0.126 TEST 0.1184 -- sigma = 0.0018 and C = 24.

hiperparametros <- expand.grid(sigma = c(seq(0.0014, 0.0024, by=0.0002))
                               ,C = (15:40))
# RMSE 0.125 TEST 0.1184 -- sigma = 0.0014 and C = 34.

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
## 1209.32   20.75 1232.11
```

```
modelo_svmRadial$bestTune
```

```
##    sigma  C
## 1 0.0014 15
```

```
# Presento estudio
fnEstudioModelo(modelo_svmRadial)
```

Modelo final

Support Vector Machine object of class "ksvm"

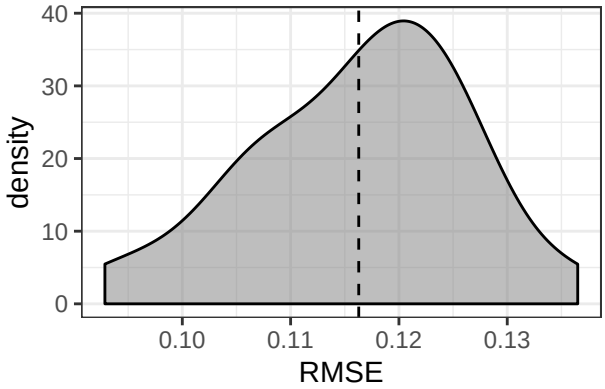
SV type: eps−svr (regression)
parameter : epsilon = 0.1 cost C = 15

Gaussian Radial Basis kernel function.
Hyperparameter : sigma = 0.0014

Number of Support Vectors : 653

Objective Function Value : −1139.697
Training error : 0.045146

RMSE obtenido en la validación

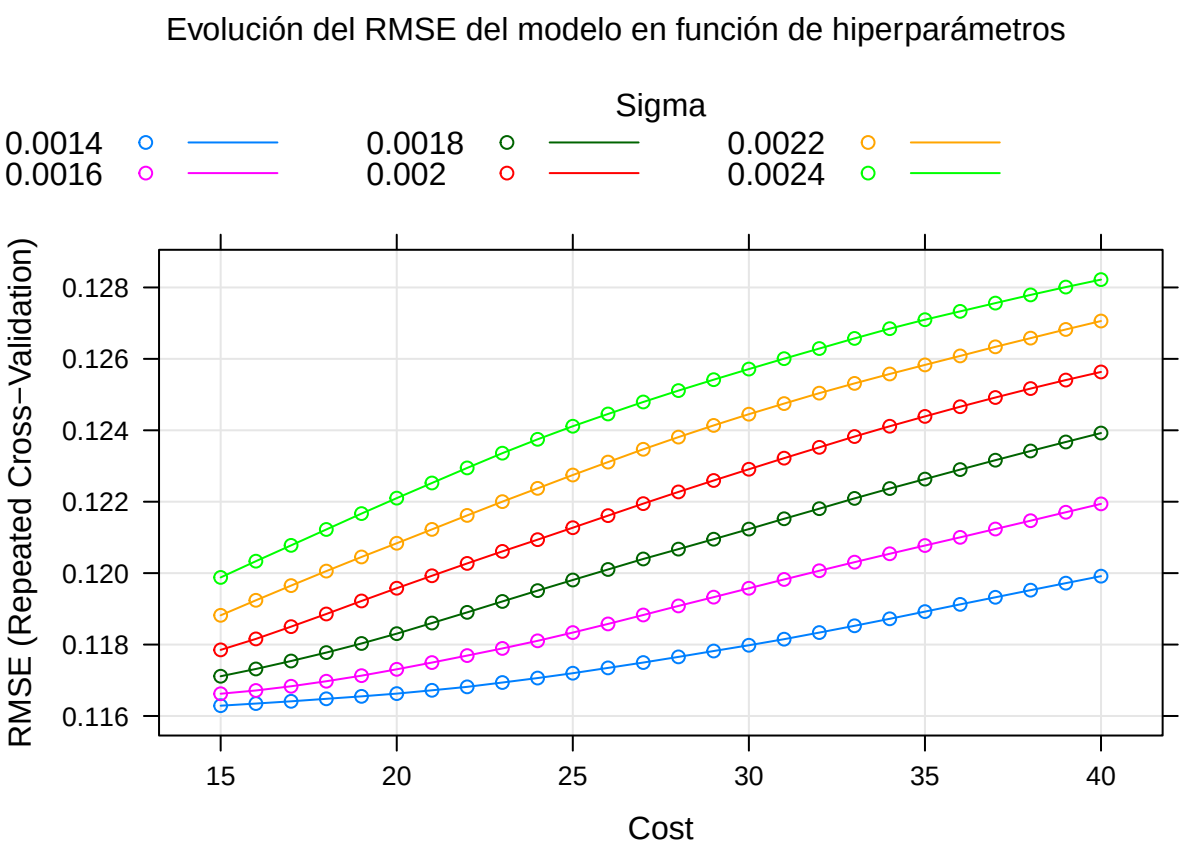


Summary resample$RMSE

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0929	0.1080	0.1180	0.1160	0.1240	0.1370

Error de test

RMSE	Rsquared	MAE
0.11368779	0.91344543	0.08118396



Elasticnet

Es una combinación de LASSO y Ridge regression, donde predictores altamente correlacionados presentan coeficientes estimados similares.

```
hiperparametros <- expand.grid(alpha=seq(0,2,by=.5),lambda=seq(0,0.1,by=.02))
# RMSE 0.130 TEST 0.1237 -- alpha = 1 and lambda = 0

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

```
##    user  system elapsed
##    4.25    0.04    4.33
```

```
modelo_glmnet$bestTune
```

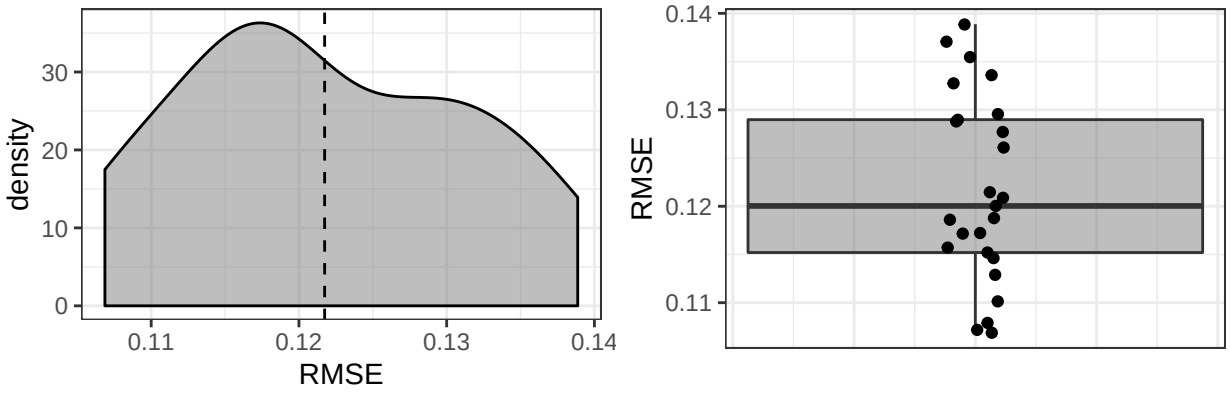
```
##   alpha lambda
## 2     0   0.02
```

```
# Presento estudio
fnEstudioModelo(modelo_glmnet)
```

Modelo final

```
[36,] 70 4.086e-01 12.86000
[37,] 70 4.399e-01 11.72000
[38,] 70 4.635e-01 10.68000
[39,] 70 4.871e-01  9.72700
[40,] 70 5.107e-01  8.86300
[41,] 70 5.342e-01  8.07500
[42,] 70 5.573e-01  7.35800
[43,] 70 5.801e-01  6.70400
[44,] 70 6.023e-01  6.10900
[45,] 70 6.239e-01  5.56600
[46,] 70 6.448e-01  5.07200
[47,] 70 6.648e-01  4.62100
[48,] 70 6.840e-01  4.21000
[49,] 70 7.022e-01  3.83600
[50,] 70 7.194e-01  3.49600
[51,] 70 7.356e-01  3.18500
[52,] 70 7.508e-01  2.90200
[53,] 70 7.649e-01  2.64400
[54,] 70 7.781e-01  2.40900
[55,] 70 7.903e-01  2.19500
[56,] 70 8.016e-01  2.00000
[57,] 70 8.121e-01  1.82300
[58,] 70 8.216e-01  1.66100
[59,] 70 8.304e-01  1.51300
[60,] 70 8.384e-01  1.37900
```

RMSE obtenido en la validación

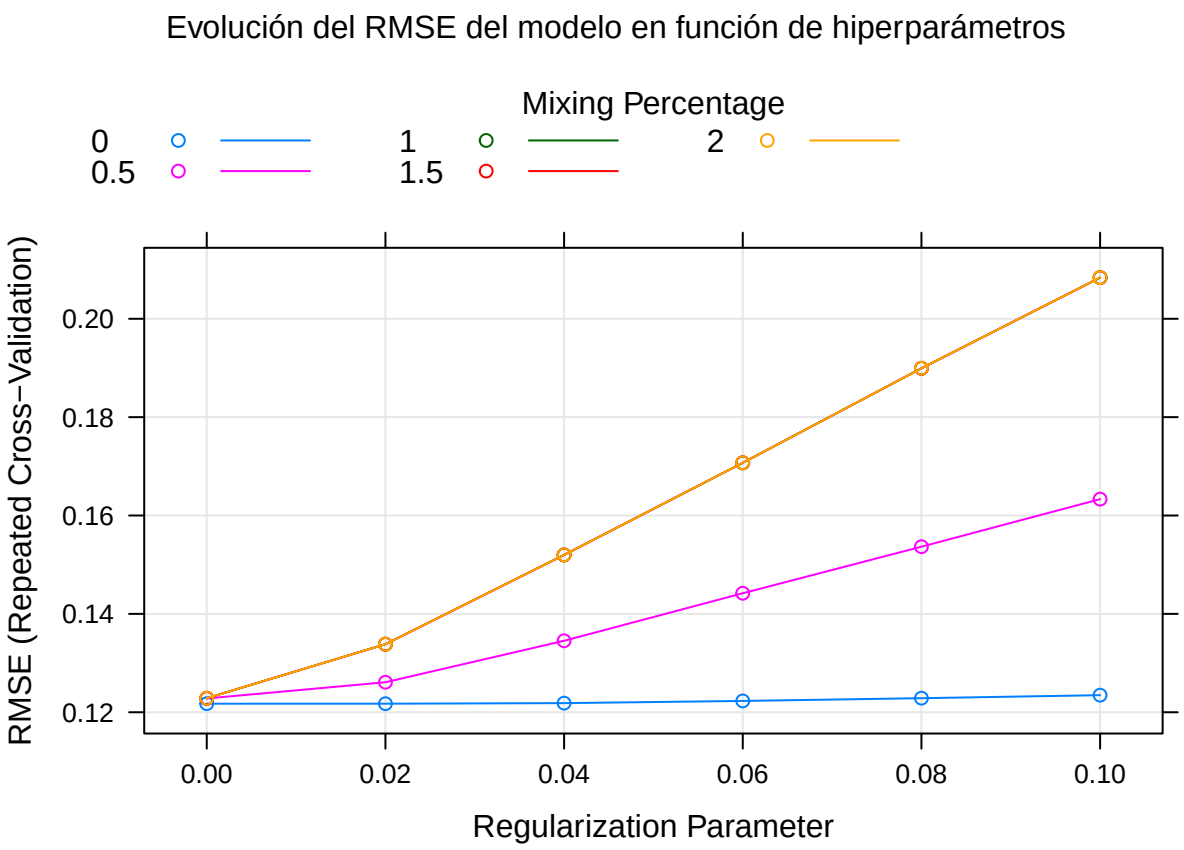


Summary resample$RMSE

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.107	0.115	0.120	0.122	0.129	0.139

Error de test

RMSE	Rsquared	MAE
0.11850060	0.90704931	0.08855263



LASSO

0perador de mínima contracción y selección absoluta. (least absolute shrinkage and selection operator) se utiliza para modelos de sistemas no lineales.

Realiza selección de variables y regularización para mejorar la exactitud e interpretabilidad del modelo. Establece algunos coeficientes a cero lo que permite eliminar variables.

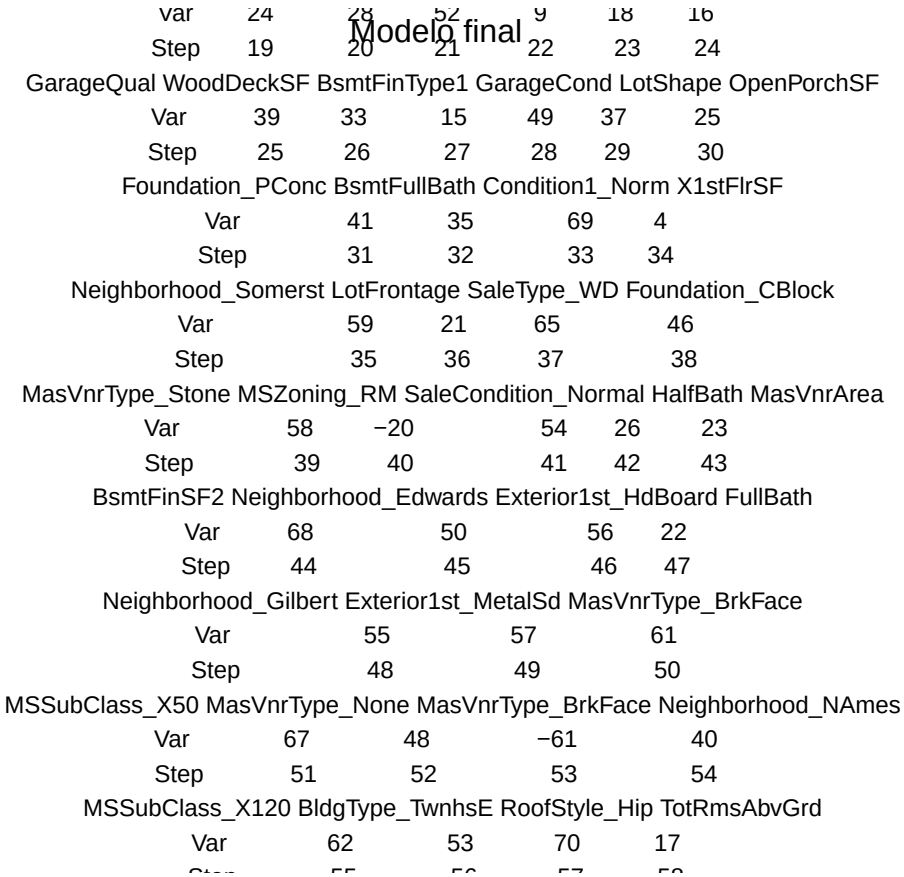
```
hiperparametros <- expand.grid(fraction=c(0.001,0.01,0.1,1))

t <- proc.time() # Inicia el cronómetro
modelo_lasso <- train(SalePrice ~ .
                      , data = dsTrain.training
                      , method = "lasso"
                      , tuneGrid = hiperparametros
                      #, tuneLength = 10
                      , metric = "RMSE"
                      , trControl = fitControl)
proc.time()-t    # Detiene el cronómetro
```

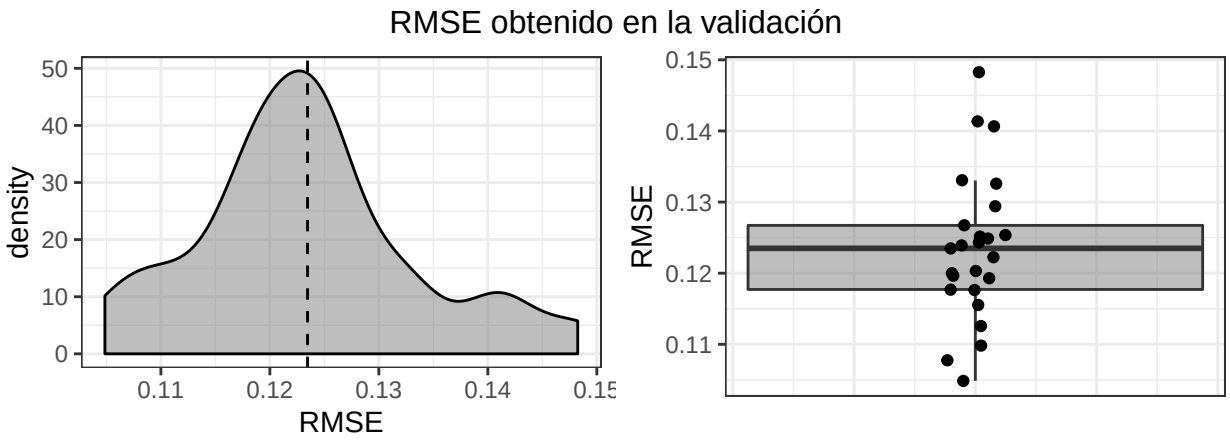
```
##    user  system elapsed
##    3.68    0.05    3.80
```

```
modelo_lasso$bestTune
```

```
##   fraction
## 4        1
```

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





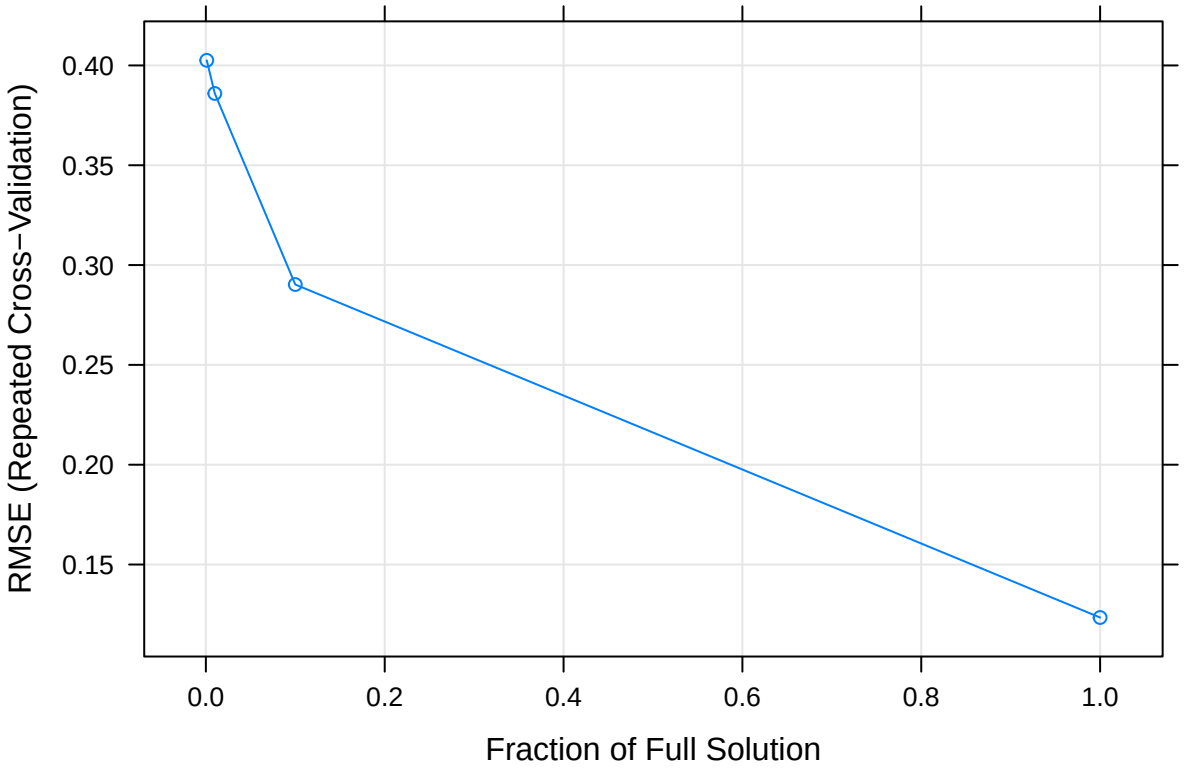
Summary resample$RMSE

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.105	0.118	0.123	0.123	0.127	0.148

Error de test

RMSE	Rsquared	MAE
0.12050943	0.90405044	0.08968673

```
plot(modelo_lasso)
```



Generalized Linear Model

Es una generalización flexible de la regresión lineal, permite que el modelo lineal se relacione con la variable de respuesta a través de una función de enlace. Este modelo parece más apto para nuestro problema ya que permite que la variable respuesta tenga una distribución arbitraria, nuestra variable es solo positiva y varia gran escala, es una distribución sesgada.

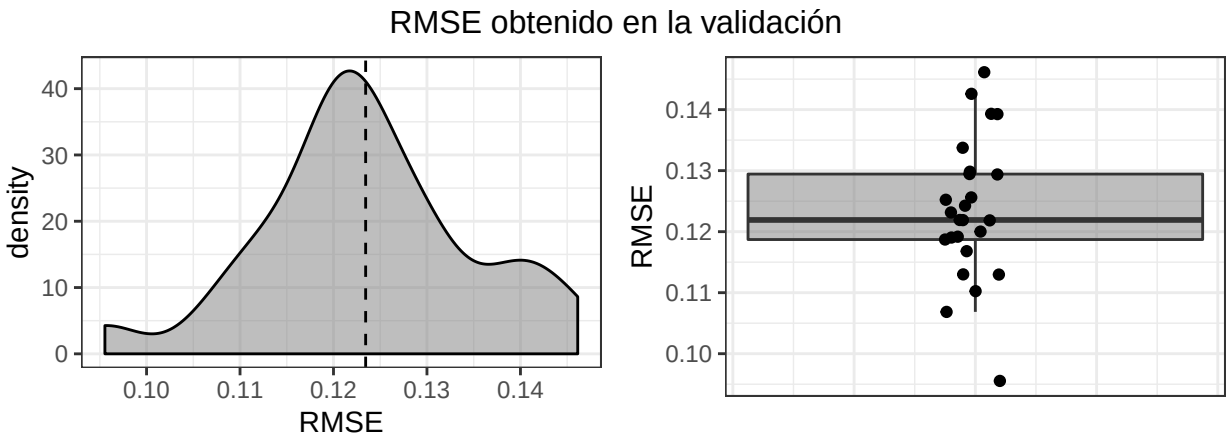
```
hiperparametros <- data.frame(parameter = "none")

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

```
##    user  system elapsed
##    1.59    0.00    1.60
```

```
modelo_glm$bestTune
```

```
##   parameter
## 1      none
```

```r
# Presento estudio
fnEstudioModelo(modelo_glm, estudioParam = FALSE)
```



Summary resample$RMSE

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0955	0.1190	0.1220	0.1230	0.1290	0.1460

Error de test

RMSE	Rsquared	MAE
0.12050943	0.90405044	0.08968673

XGBoost

XGBoost ha sido uno de los modelos más utilizados, esto es así porque se adapta fácilmente ya que es muy flexible, se puede usar tanto en regresión como en clasificación. Utiliza una combinación de modelos más simples (árboles de decisión) y potencia los resultados.

La gran desventaja de este modelo es el ajuste de su gran cantidad de parámetros.

Para realizar el ajuste hemos usado un proceso iterativo, basado en el Notebooks de Kaggle:

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

```
# ITER 1
# 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 = 850, max_depth = 2, eta = 0.025, gamma = 0, colsample_bytree =  1, min_child_weight = 1 and
# RMSE 0.134 Test 0.1288


# ITER 2
# nrounds = seq(from = 200, to = 1000, by = 50),
```

```
# max_depth = c(2, 3, 4),
# eta = 0.025,
# gamma = 0,
# colsample_bytree = 1,
# min_child_weight = c(1, 2, 3),
# subsample = 1

# RMSE 0.128 Test 0.1288
# model were nrounds = 900, max_depth = 2, eta = 0.025, gamma = 0, colsample_bytree = 1, min_child_weig

# ITER 3
# nrounds = seq(from = 500, to = 2500, by = 50),
# max_depth = 2,
# eta = 0.025,
# gamma = 0,
# colsample_bytree = 1,
# min_child_weight = c(5, 6, 7, 8, 9),
# subsample = 1

# RMSE 0.128 Test 0.1292
# nrounds = 800, max_depth = 2, eta = 0.025, gamma = 0, colsample_bytree = 1, min_child_weight = 8 and

# ITER 4
# nrounds = seq(from = 500, to = 1500, by = 50),
# max_depth = 2,
# eta = 0.025,
# gamma = 0,
# colsample_bytree = c(0.3, 0.4, 0.5, 0.6),
# min_child_weight = 8,
# subsample = c(0.4, 0.5, 0.6)


# ITER 5
# nrounds = seq(from = 500, to = 2000, by = 50),
# max_depth = 2,
# eta = 0.025,
# gamma = c(0, 0.05, 0.1, 0.2),
# colsample_bytree = 0.4,
# min_child_weight = 8,
# subsample = 0.5

# RMSE 0.127 Test 0.1277
# nrounds = 1350, max_depth = 2, eta = 0.025, gamma = 0, colsample_bytree = 0.4, min_child_weight = 8 a


# RMSE 0.125 Test 0.128
# nrounds = 1950, max_depth = 2, eta = 0.015, gamma = 0, colsample_bytree = 0.4, min_child_weight = 8 a

# Maximum Depth
hiperparametros <- expand.grid(
  nrounds = seq(from = 500, to = 2500, by = 50),
  max_depth = 2,
  eta =  0.015,
```

```
  gamma = 0,
  colsample_bytree = 0.4,
  min_child_weight = 8,
  subsample = 0.5
)

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
##  182.12   99.10  125.92
```

```
modelo_XGBoost$bestTune
```

```
##    nrounds max_depth   eta gamma colsample_bytree min_child_weight
## 39    2400         2 0.015     0              0.4                8
##    subsample
## 39       0.5
```

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

Model Final

raw: 900 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):

ax_depth = "2", gamma = "0", colsample_bytree = "0.4", min_child_weight = "8", subsample = "0.5", objective = "reg:li

xgb.attributes:

niter

callbacks:

cb.print.evaluation(period = print_every_n)

# of features: 70

niter: 2400

nfeatures : 70

tingQC GarageQual Neighborhood_NAmes Foundation_PConc HouseStyle_X1Story Exterior2nd_VinylSd CentralAir

problemType : Regression

tuneValue :

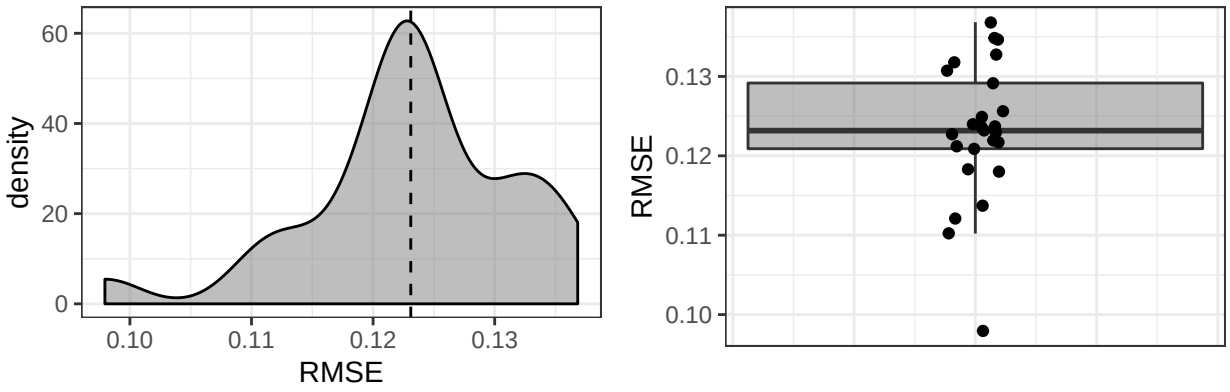
	nrounds	max_depth	eta	gamma	colsample_bytree	min_child_weight
39	2400	2	0.015	0	0.4	8

	subsample
39	0.5

obsLevels : NA

param :

RMSE obtenido en la validación



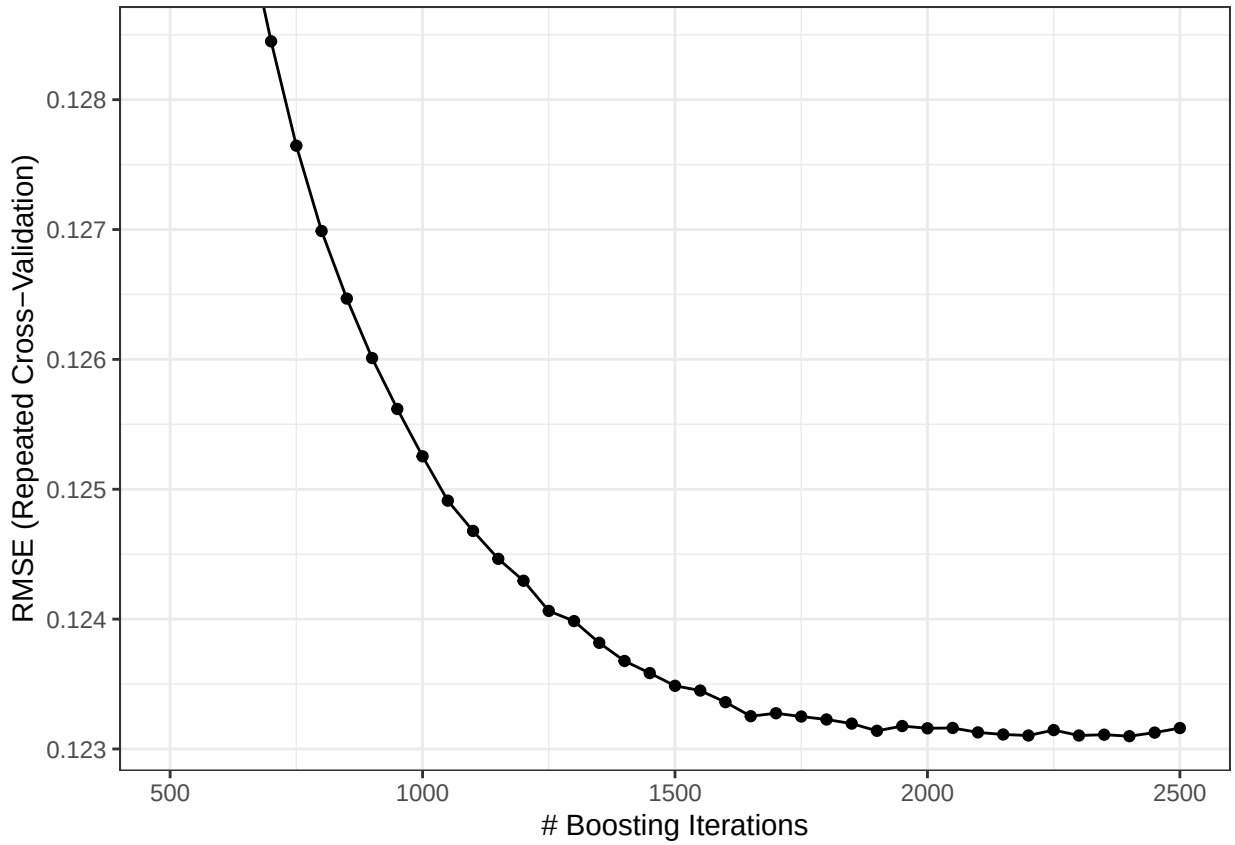
Summary resample$RMSE

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0979	0.1210	0.1230	0.1230	0.1290	0.1370

Error de test

RMSE	Rsquared	MAE
0.12142244	0.90481190	0.08676418

```
tuneplot(modelo_XGBoost)
```



Comparación de modelos

En este punto trataremos de identificar cuál de los modelos es mejor para ello tendremos en cuenta las métricas de validación calculadas en el entrenamiento y el error de test.

Utilizare la función resamples() para extraer las métricas de los modelos entrenados.

```
modelos <- list(
  SVMR = modelo_svmRadial
  ,GLMNET = modelo_glmnet
  ,GLM = modelo_glm
  ,LASSO = modelo_lasso
  ,XGBoost = modelo_XGBoost
)

resultados_resamples <- resamples(modelos)

# Separamos el nombre del modelo y las métricas en columnas distintas.
metricas_resamples <- resultados_resamples$values %>%
                        gather(key = "modelo", value = "valor", -Resample) %>%
                        separate(col = "modelo", into = c("modelo", "metrica"),
                                 sep = "~", remove = TRUE)

# Se obtienen las medias por modelo
metricas_resamples %>%
```

```
  group_by(modelo, metrica) %>%
  summarise(media = mean(valor)) %>%
  spread(key = metrica, value = media) %>%
  arrange(RMSE)
```

```
## # A tibble: 5 x 4
## # Groups:   modelo [5]
##   modelo     MAE  RMSE Rsquared
##   <chr>    <dbl> <dbl>    <dbl>
## 1 SVMR    0.0801 0.116    0.918
## 2 GLMNET  0.0861 0.122    0.910
## 3 XGBoost 0.0865 0.123    0.908
## 4 GLM     0.0880 0.123    0.907
## 5 LASSO   0.0880 0.123    0.908
```

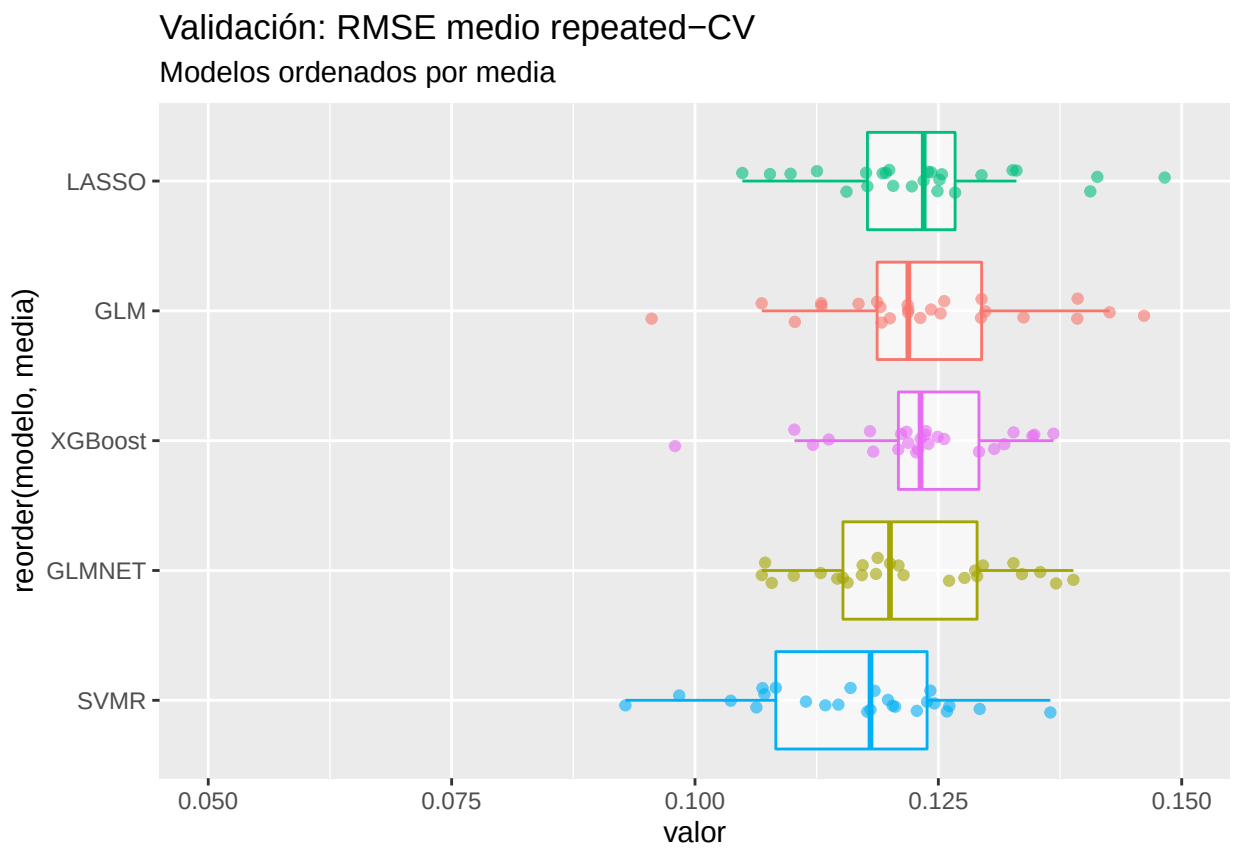
```
predicciones <- extractPrediction(
                  models = modelos,
                  testX = dsTrain.CV[, -1],
                  testY = dsTrain.CV$SalePrice
                  )

metricas_tipo <- predicciones %>%
                        group_by(object, dataType) %>%
                        summarise(RMSE = RMSE(pred, obs))

metricas <- metricas_tipo %>%
  spread(key = dataType, RMSE) %>%
  arrange(Test)

metricas
```

```
## # A tibble: 5 x 3
## # Groups:   object [5]
##   object   Test Training
##   <fct>   <dbl>    <dbl>
## 1 SVMR    0.114   0.0861
## 2 GLMNET  0.119   0.114
## 3 LASSO   0.121   0.113
## 4 GLM     0.121   0.113
## 5 XGBoost 0.121   0.0818
```

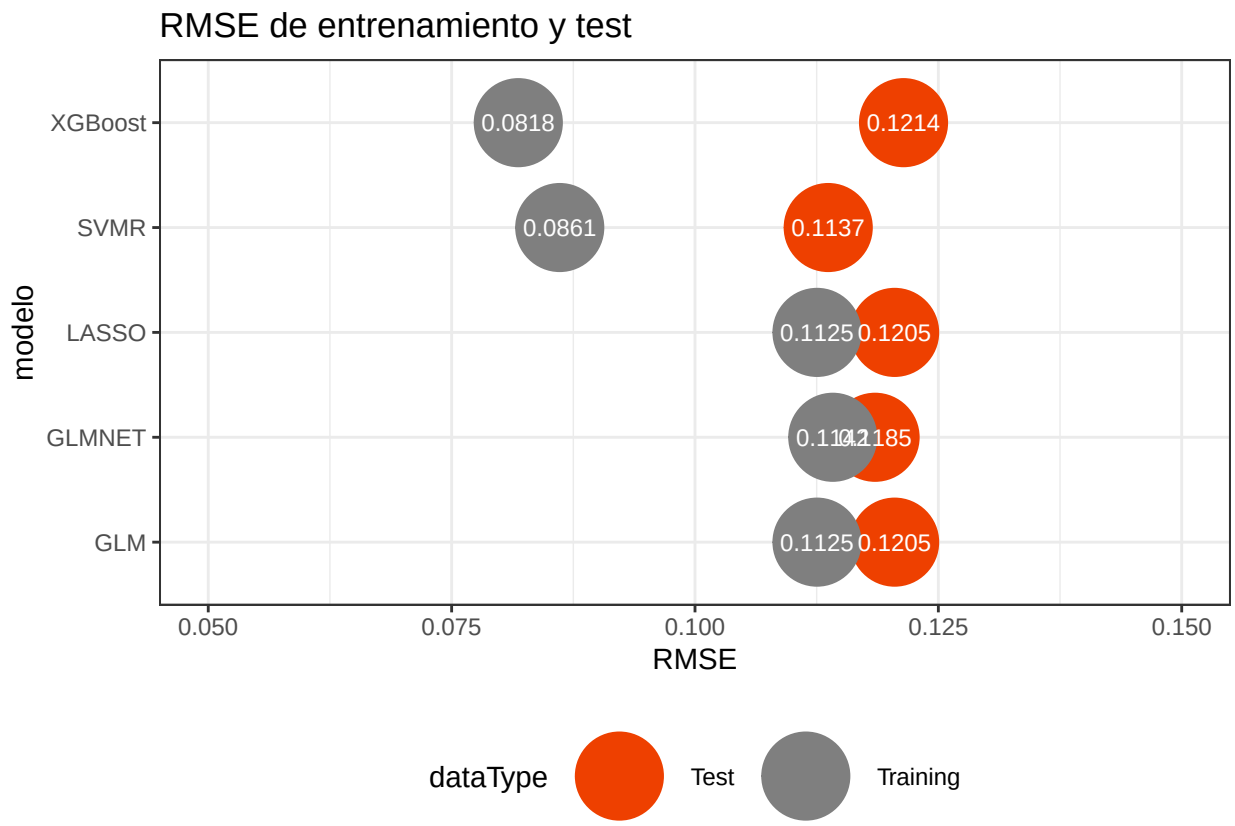
Comparativa gráfica

```
dg <-metricas_resamples %>%
  filter(metrica == "RMSE") %>%
  group_by(modelo) %>%
  mutate(media = mean(valor)) %>%
  ungroup()

ggplot(dg, aes(x = reorder(modelo, media), y = valor, color = modelo)) +
    geom_boxplot(alpha = 0.6, outlier.shape = NA) +
    geom_jitter(width = 0.1, alpha = 0.6) +
```

```
    scale_y_continuous(limits = c(0.05, 0.15)) +
    labs(title = "Validación: RMSE medio repeated-CV",
         subtitle = "Modelos ordenados por media") +
    coord_flip() +
    theme(legend.position = "none")
```




```
ggplot(data = metricas_tipo,
       aes(x = object, y = RMSE,
           color = dataType, label = round(RMSE, 4))) +
  geom_point(size = 15) +
  scale_color_manual(values = c("orangered2", "gray50")) +
  geom_text(color = "white", size = 3) +
  scale_y_continuous(limits = c(0.05, 0.15)) +
  coord_flip() +
  labs(title = "RMSE de entrenamiento y test",
       x = "modelo") +
  theme_bw() +
  theme(legend.position = "bottom")
```



Fase 5 - Selección del modelo definitivo

Seleccionamos el modelo con el RMSE más bajo

```
modeloSel <- modelo_svmRadial
```

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

```
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
```