TFM - Kaggle House Prices: Advanced Regression Techniques with caret

05 Selección del modelo definitivo y presentación de resultados

Juan Carlos Santiago Culebras

2019-09-22

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("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

```
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")
  }
```

```
}
```

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

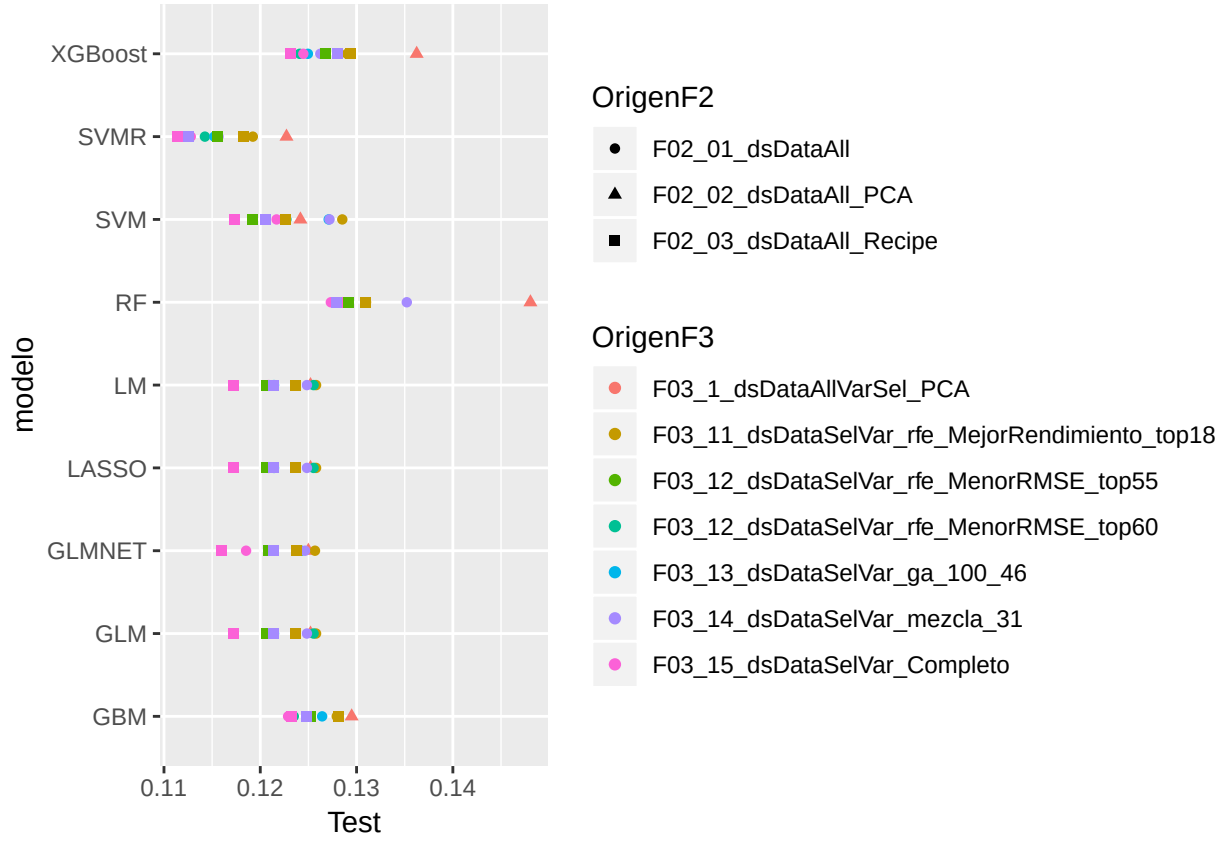
En total se han estudiado 11 modelos y se han realizado 100 entrenamientos distintos.

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  0.122
## 3 LASSO   0.123
## 4 GLM     0.123
## 5 LM      0.123
## 6 SVM     0.123
## 7 GBM     0.126
## 8 XGBoost 0.127
## 9 RF      0.131
```

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

```
## # A tibble: 3 x 2
##   OrigenF2                media
##   <fct>                   <dbl>
## 1 F02_03_dsDataAll_Recipe 0.122
## 2 F02_01_dsDataAll        0.124
## 3 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_15_dsDataSelVar_Completo                    0.120
## 2 F03_12_dsDataSelVar_rfe_MenorRMSE_top55         0.122
## 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/'
load(paste(dir,'modelo_svmRadial.RData',sep=''))

load('./F03_SelPredictores/F02_03_dsDataAll_Recipe/F03_15_dsDataSelVar_Completo.RData')
load('./F01_Datos/F01_dsDataAll.RData')

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   87
```

```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))

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

```
## [1] 0.1114359
```

```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] 22574.91
```

```
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

```
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
```

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(0.0004, 0.002, by=0.0002))
                               ,C = (5:15))

## sigma = 5e-04 and C = 50.

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
##  911.31    6.15  935.30
```

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


# Presento estudio
fnEstudioModelo(modelo_svmRadial)
```

Modelo final

Support Vector Machine object of class "ksvm"

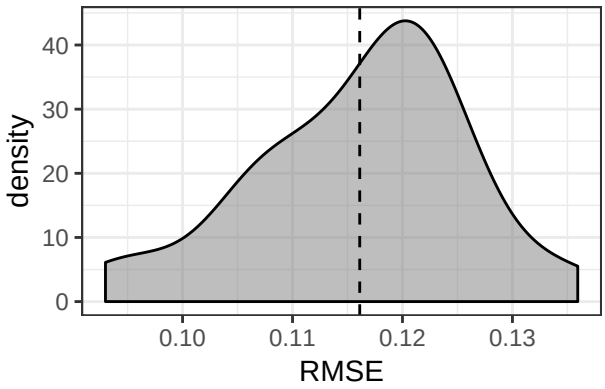
SV type: eps−svr (regression)
parameter : epsilon = 0.1 cost C = 12

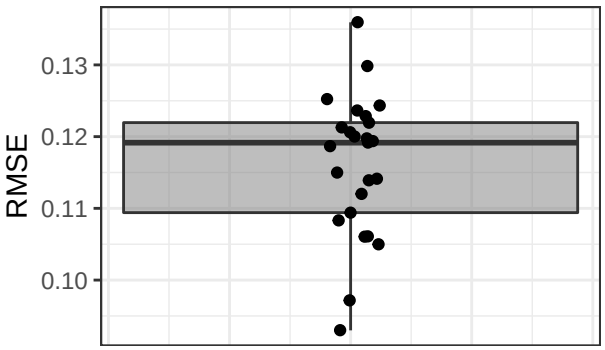
Gaussian Radial Basis kernel function.
Hyperparameter : sigma = 0.001

Number of Support Vectors : 655

Objective Function Value : −985.6482
Training error : 0.046788

RMSE obtenido en la validación



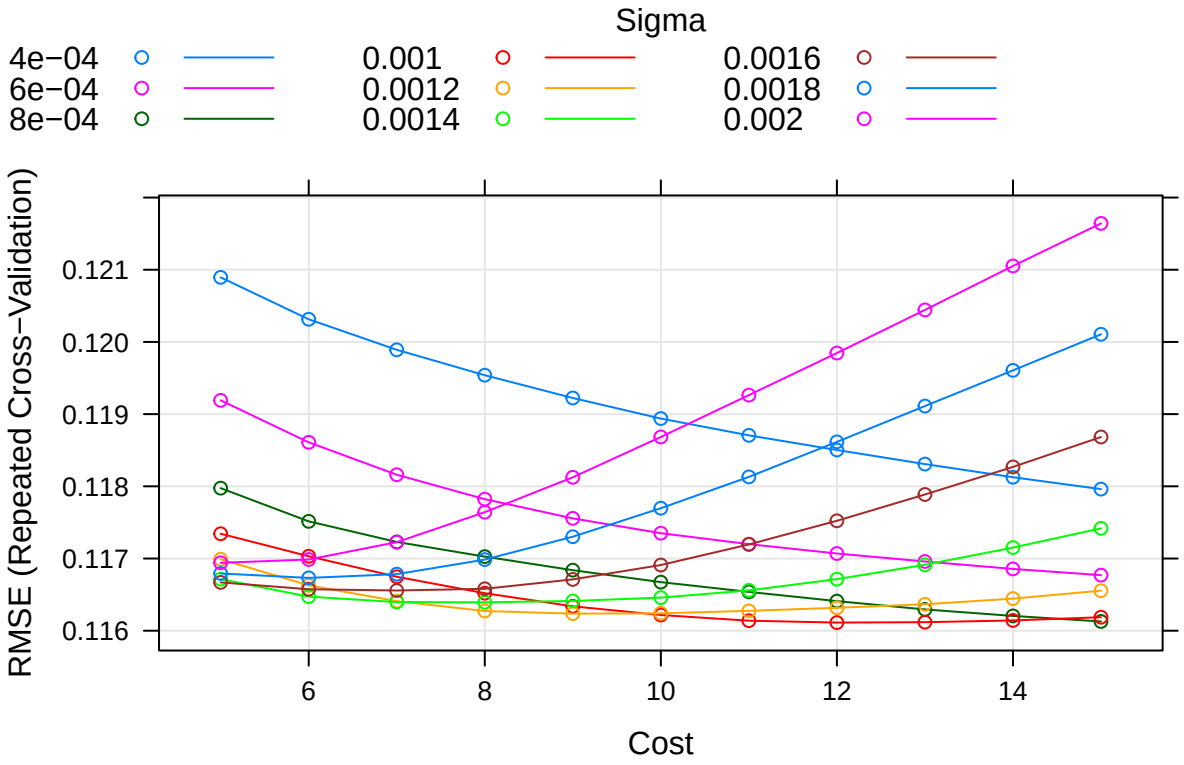


Summary resample$RMSE

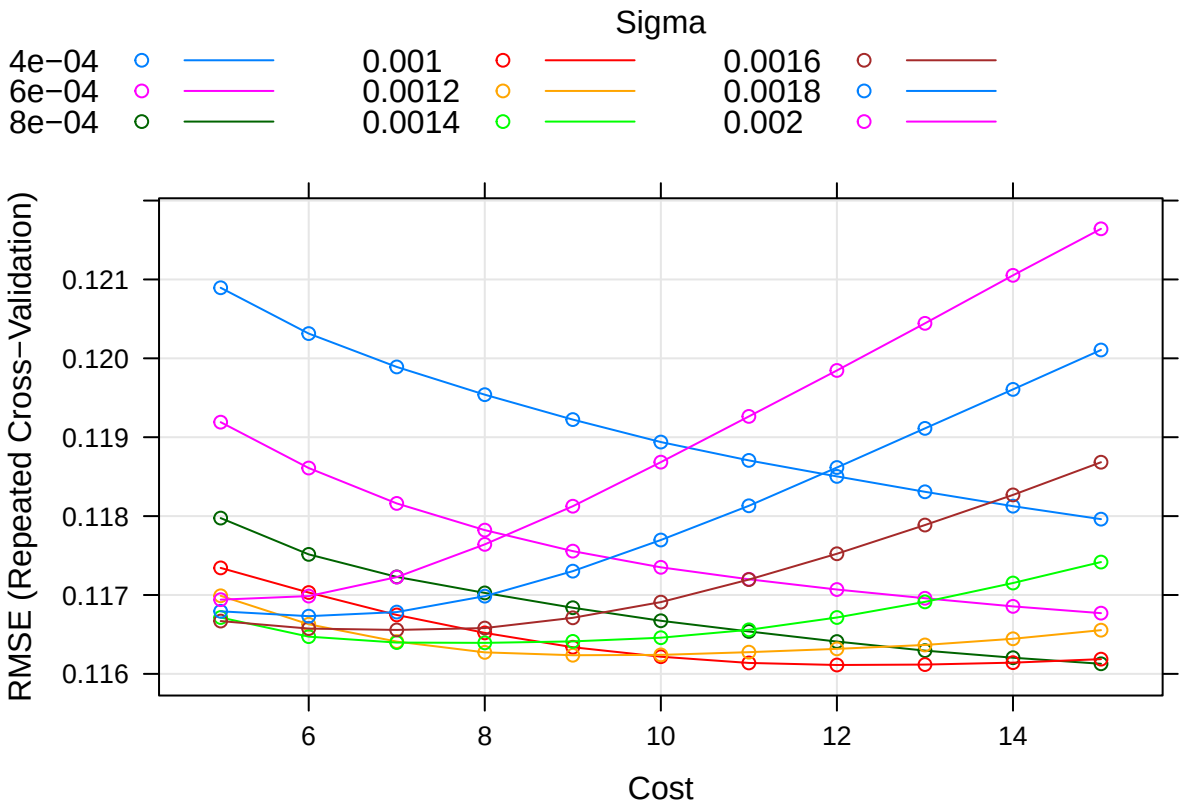
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.093	0.109	0.119	0.116	0.122	0.136

Error de test

RMSE	Rsquared	MAE
0.11123715	0.91722662	0.07995376




```
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
```