TFM - Kaggle House Prices: Advanced Regression Techniques with caret

02 - 02 Ingeniería de características mediante paquete recipes

Juan Carlos Santiago Culebras

2019-09-22

Paquete recipes

Facilita las transformaciones de las variables para realizar la ingeniería de características.

Para su utilización: *Se crea un objeto récipe con la variable respuesta y los predictores, esto se realiza con el conjunto de entrenamiento.* Sobre dicho objeto se definen los pasos a realizar para transformar los datos. *Se realiza un entrenamiento, donde se aprenden los parámetros necesarios para realizar las transformaciones.* Se aplican las transformaciones a los conjuntos deseados

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("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)
```

Cargamos datos

Partimos del dataset generado en la etapa anterior fichero F01_dsDataAll (datos limpios)

Donde se ha realizado: *Verificación de datos y campos* Tratamiento de valores perdidos *Tratamiento de outliers

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

Preparamos datos para realizar la ingenieria con recipe

Interesa mantener la variable objetivo SalePrice con el mismo formato que en los otros conjuntos. Además 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 utilizara 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, ]
```

Ingeniería de características con recipe

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 Estandarización y escalado, sobre las variables numéricas *Binarización de variables nominales (dummy)

*Se repite la eliminación de variables con varianza cero, ya que muchas de las variables dummy tendrán este problema.

```
# 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 entrena el objeto recipe y se muestra el reusltado 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_X150, ... [trained]
```

Se aplican las transformaciones a todo el conjunto de entrenamiento y de test

```
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])

save(dsDataAll, file = './F02_Datos/F02_03_dsDataAll_Recipe.RData')
# load('./F02_Datos/F02_03_dsDataAll_Recipe.RData')
```