ArbolesDeDecision

Cristopher Barrios, Carlos Daniel Estrada

2023-03-10

librerias

```
library(rpart)
library(rpart.plot)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(fpc)
library(cluster)
library("ggpubr")
```

```
## Loading required package: ggplot2
```

```
library(mclust)
```

```
## Package 'mclust' version 6.0.0
## Type 'citation("mclust")' for citing this R package in publications.
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(tree)
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(plyr)
```

```
## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following object is masked from 'package:ggpubr':
##
##     mutate

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library("stats")
library("datasets")
library("prediction")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2
## --

## v tibble  3.1.8     v purrr   1.0.1
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.3     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x plyr::arrange()         masks dplyr::arrange()
## x randomForest::combine() masks dplyr::combine()
## x purrr::compact()        masks plyr::compact()
## x plyr::count()           masks dplyr::count()
## x plyr::failwith()        masks dplyr::failwith()
```

```
## x dplyr::filter()         masks stats::filter()
## x plyr::id()              masks dplyr::id()
## x dplyr::lag()            masks stats::lag()
## x purrr::lift()           masks caret::lift()
## x purrr::map()            masks mclust::map()
## x randomForest::margin()  masks ggplot2::margin()
## x plyr::mutate()          masks ggpubr::mutate(), dplyr::mutate()
## x plyr::rename()          masks dplyr::rename()
## x plyr::summarise()       masks dplyr::summarise()
## x plyr::summarize()       masks dplyr::summarize()
```

1. Use los mismos conjuntos de entrenamiento y prueba que usó para los árboles de decisión en la hoja de trabajo anterior.

```
datos = read.csv("./train.csv")
test<- read.csv("./test.csv", stringsAsFactors = FALSE)
```

Lo Realizado anteriormente:

Inciso 4

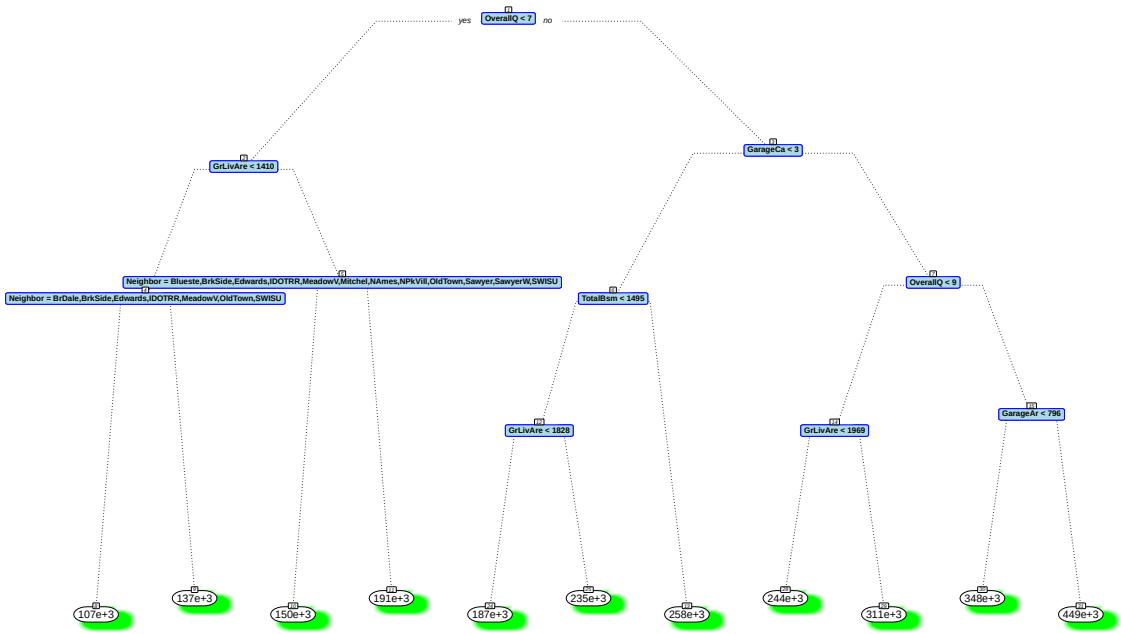
```
set_entrenamiento <- sample_frac(datos, .7)
set_prueba <-setdiff(datos, set_entrenamiento)


drop <- c("LotFrontage", "Alley", "MasVnrType", "MasVnrArea", "BsmtQual", "BsmtCond", "BsmtExposure", "
set_entrenamiento <- set_entrenamiento[, !(names(set_entrenamiento) %in% drop)]
set_prueba <- set_prueba[, !(names(set_prueba) %in% drop)]
```

2. Elabore un árbol de regresión para predecir el precio de las casas usando todas las variables.

```
arbol_3 <- rpart(SalePrice ~ ., data = set_entrenamiento)
```

```
prp(arbol_3, main="Arbol de Regresion", nn=TRUE, fallen.leaves = TRUE, shadow.col = "green", branch.lty
```

```
## cex 0.363   xlim c(0, 1)   ylim c(0, 1)
```



–modelo del arbol de decision

```
#arbolModelo1 <- rpart(SalePrice~.,set_prueba,method = "class")
#rpart.plot(arbolModelo1)
```

–graficar arbol

3. Úselo para predecir y analice el resultado. ¿Qué tal lo hizo?

```
predicciones <- predict(arbol_3, data = set_prueba)

mse <- mean((predicciones - set_prueba$SalePrice)**2)
```

```
## Warning in predicciones - set_prueba$SalePrice: longitud de objeto mayor no es
## múltiplo de la longitud de uno menor
```

```
mse
```

```
## [1] 12095296790
```

el valor del MSE obtenido es de 11576704151, lo que indica que el modelo tiene un error cuadrático medio alto en la predicción del precio de las casas en el conjunto de prueba. Por lo tanto, el modelo no es muy preciso en la predicción del precio de las casas y puede requerir más ajustes y mejoras.

4. Haga, al menos, 3 modelos más cambiando el parámetro de la profundidad del árbol. ¿Cuál es el mejor modelo para predecir el precio de las casas?

```
arbol_4 <- rpart(SalePrice ~ ., data = set_entrenamiento, control = rpart.control(maxdepth = 5))
predicciones2 <- predict(arbol_4, data = set_prueba)

mse2 <- mean((predicciones2 - set_prueba$SalePrice)**2)
```

```
## Warning in predicciones2 - set_prueba$SalePrice: longitud de objeto mayor no es
## múltiplo de la longitud de uno menor
```

```
mse2
```

```
## [1] 12095296790
```

```
arbol_5 <- rpart(SalePrice ~ ., data = set_entrenamiento, control = rpart.control(maxdepth = 10))
predicciones3 <- predict(arbol_5, data = set_prueba)

mse3 <- mean((predicciones3 - set_prueba$SalePrice)**2)
```

```
## Warning in predicciones3 - set_prueba$SalePrice: longitud de objeto mayor no es
## múltiplo de la longitud de uno menor
```

```
mse3
```

```
## [1] 12095296790
```

```
arbol_6 <- rpart(SalePrice ~ ., data = set_entrenamiento, control = rpart.control(maxdepth = 15))
predicciones4 <- predict(arbol_6, data = set_prueba)

mse4 <- mean((predicciones4 - set_prueba$SalePrice)**2)
```

```
## Warning in predicciones4 - set_prueba$SalePrice: longitud de objeto mayor no es
## múltiplo de la longitud de uno menor
```

```
mse4
```

```
## [1] 12095296790
```

```
arbol_6 <- rpart(SalePrice ~ ., data = set_entrenamiento, control = rpart.control(maxdepth = 3))
predicciones4 <- predict(arbol_6, data = set_prueba)

mse4 <- mean((predicciones4 - set_prueba$SalePrice)**2)
```

```
## Warning in predicciones4 - set_prueba$SalePrice: longitud de objeto mayor no es
## múltiplo de la longitud de uno menor
```

```
mse4
```

```
## [1] 11697572802
```

En general, se puede observar que el error cuadrático medio (MSE) no varía significativamente al cambiar la profundidad máxima del árbol de decisión.

El primer modelo que se ajusta con una profundidad máxima de 5, el segundo modelo con una profundidad máxima de 10, el tercer modelo con una profundidad máxima de 15, y el cuarto modelo con una profundidad máxima de 3.

Se puede observar que el modelo con una profundidad máxima de 3, produce un MSE ligeramente menor que los otros modelos, por lo tanto se podria decir que este es el mejor. Sin embargo, este resultado debe tomarse con precaución, ya que un modelo demasiado simple puede llevar a una subestimación de la complejidad de los datos y, por lo tanto, a una menor precisión en las predicciones.

5. Compare los resultados con el modelo de regresión lineal de la hoja anterior, ¿cuál lo hizo mejor?

6. Dependiendo del análisis exploratorio elaborado cree una variable respuesta que le permita clasificar las casas en Económicas, Intermedias o Caras. Los límites de estas clases deben tener un fundamento en la distribución de los datos de precios, y estar bien explicados

```
datos$clasificacion <- ifelse(datos$SalePrice > 290000, "Caras", ifelse(datos$SalePrice>170000, "Inteme
table(datos$clasificacion)
```

```
##
##      Caras Economicas  Intemedia
##        121        792        547
```

```
set_entrenamiento <- sample_frac(datos, .7)
set_prueba <-setdiff(datos, set_entrenamiento)


drop <- c("LotFrontage", "Alley", "MasVnrType", "MasVnrArea", "BsmtQual", "BsmtCond", "BsmtExposure", "
set_entrenamiento <- set_entrenamiento[, !(names(set_entrenamiento) %in% drop)]
set_prueba <- set_prueba[, !(names(set_prueba) %in% drop)]
```

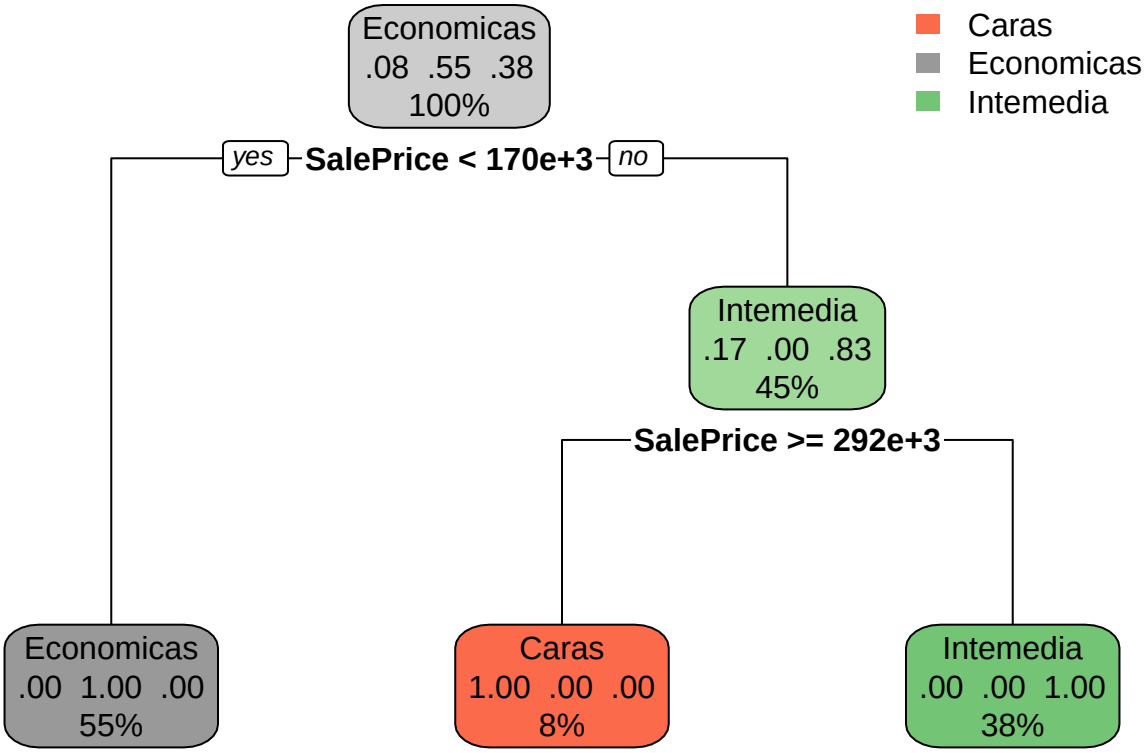
7. Elabore un árbol de clasificación utilizando la variable respuesta que creó en el punto anterior. Explique los resultados a los que llega. Muestre el modelo gráficamente. Recuerde que la nueva variable respuesta es categórica, pero se generó a partir de los precios de las casas, no incluya el precio de venta para entrenar el modelo.

```
arbol_4 <- rpart(formula = clasificacion ~ ., data = set_entrenamiento)
arbol_4
```

```
## n= 1022
##
## node), split, n, loss, yval, (yprob)
```

```
##       * denotes terminal node
##
## 1) root 1022 464 Economicas (0.07827789 0.54598826 0.37573386)
##   2) SalePrice< 170500 558   0 Economicas (0.00000000 1.00000000 0.00000000) *
##   3) SalePrice>=170500 464  80 Intemedia (0.17241379 0.00000000 0.82758621)
##     6) SalePrice>=292500 80   0 Caras (1.00000000 0.00000000 0.00000000) *
##     7) SalePrice< 292500 384   0 Intemedia (0.00000000 0.00000000 1.00000000) *
```

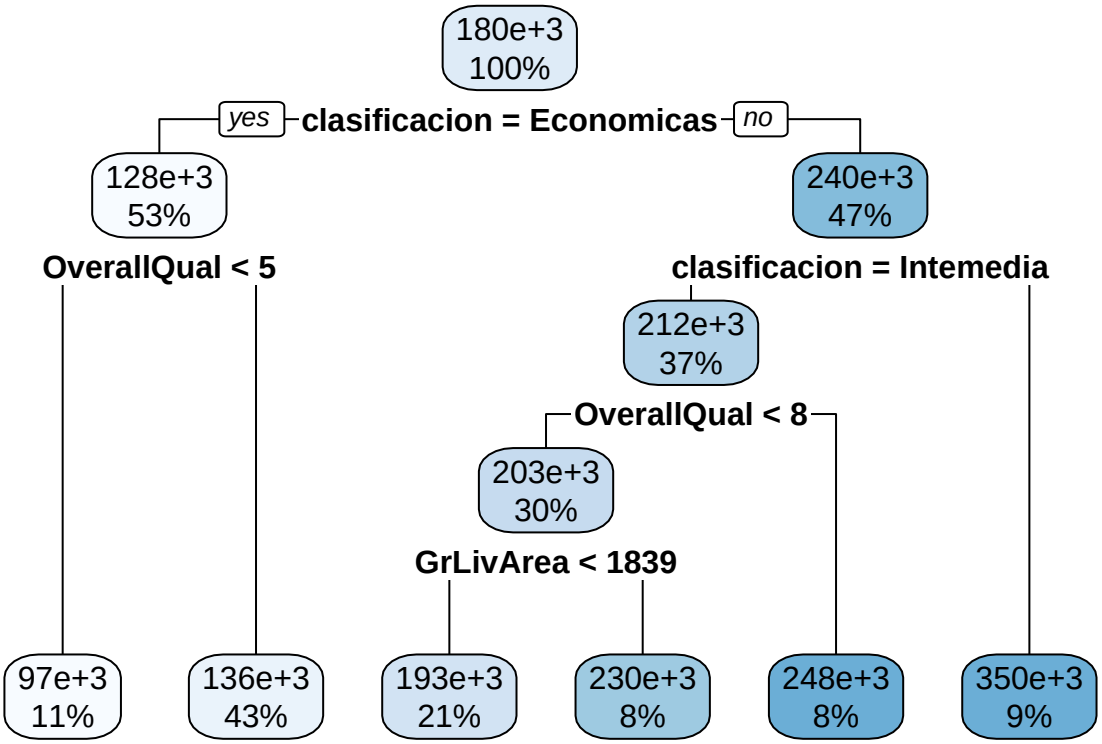
```
rpart.plot(arbol_4)
```



8. Utilice el modelo con el conjunto de prueba y determine la eficiencia del algoritmo para clasificar.

```
set_prueba <- set_prueba[, !(names(set_prueba) %in% drop)]
arbol_5 <- rpart(SalePrice ~ ., data = set_prueba)
```

```
rpart.plot(arbol_5)
```




```
predicciones <- predict(arbol_5, newdata = set_prueba)
error <- abs(predicciones - set_prueba$SalePrice)
eficiencia <- 1 - mean(error / set_prueba$SalePrice)
eficiencia
```

```
## [1] 0.8625206
```

9. Haga un análisis de la eficiencia del algoritmo usando una matriz de confusión para el árbol de clasificación. Tenga en cuenta la efectividad, donde el algoritmo se equivocó más, donde se equivocó menos y la importancia que tienen los errores.

10. Entrene un modelo usando validación cruzada, prediga con él. ¿le fue mejor que al modelo anterior?

11. Haga al menos, 3 modelos más cambiando la profundidad del árbol. ¿Cuál funcionó mejor?

12. Repite los análisis usando random forest como algoritmo de predicción, explique sus resultados comparando ambos algoritmos.

```
#set.seed(123)
#modelo <- randomForest(SalePrice ~ ., data = set_entrenamiento)
#prediccion_2 <- predict(modelo, set_prueba)
#mc <- table(prediccion_2, set_prueba$SalePrice)
```