BayesIngenuo

Cristopher Barrios, Carlos Daniel Estrada

2023-03-17

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()
```

```
library(e1071)
```

1. Use los mismos conjuntos de entrenamiento y prueba que utilizó en las dos hojas anteriores.

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

```
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 modelo de regresión usando bayes ingenuo (naive bayes), el conjunto de entrenamiento y la variable respuesta SalesPrice. Prediga con el modelo y explique los resultados a los que llega. Asegúrese que los conjuntos de entrenamiento y prueba sean los mismos de las hojas anteriores para que los modelos sean comparables.

```
#percentiles
percentil <- quantile(datos$SalePrice)


estado<-c('Estado')
datos$Estado<-estado
datos <- within(datos, Estado[SalePrice<=129975] <- 'Economica')
datos$Estado[(datos$SalePrice>129975 &datos$SalePrice<=163000)] <- 'Intermedia'
datos$Estado[datos$SalePrice>163000] <- 'Cara'


#Bayes
porcentaje<-0.7

set.seed(1234)
corte <- sample(nrow(datos),nrow(datos)*porcentaje)

#Entrenamiento
```

```
train<-datos[corte,]
#Prueba
test<-datos[-corte,]
```

3. Haga un modelo de clasificación, use la variable categórica que hizo con el precio de las casas (barata, media y cara) como variable respuesta.

```
#modelo
modelo<-naiveBayes(train$Estado~., data=train)

#Casting
test$GrLivArea<-as.numeric(test$GrLivArea)
test$YearBuilt<-as.numeric(test$YearBuilt)
test$BsmtUnfSF<-as.numeric(test$BsmtUnfSF)
test$TotalBsmtSF<-as.numeric(test$TotalBsmtSF)
test$GarageArea<-as.numeric(test$GarageArea)
test$YearRemodAdd<-as.numeric(test$YearRemodAdd)
test$SalePrice<-as.numeric(test$SalePrice)
test$LotArea<-as.numeric(test$LotArea)

#prediccion
predBayes<-predict(modelo, newdata = test[,c("GrLivArea","YearBuilt","BsmtUnfSF","TotalBsmtSF","GarageA

#Convertimos
predBayes<-as.factor(predBayes)
#confusion
cm<-caret::confusionMatrix(as.factor(predBayes),as.factor(test$Estado))
cm
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   Cara Economica Intermedia
##   Cara        204         1          4
##   Economica     2       100         11
##   Intermedia   16         5         96
##
## Overall Statistics
##
##                Accuracy : 0.9112
##                  95% CI : (0.8806, 0.9361)
##     No Information Rate : 0.5057
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8589
##
##  Mcnemar's Test P-Value : 0.0205
##
## Statistics by Class:
##
##                      Class: Cara Class: Economica Class: Intermedia
```

```
## Sensitivity              0.9189           0.9434           0.8649
## Specificity              0.9770           0.9610           0.9360
## Pos Pred Value           0.9761           0.8850           0.8205
## Neg Pred Value           0.9217           0.9816           0.9534
## Prevalence               0.5057           0.2415           0.2528
## Detection Rate           0.4647           0.2278           0.2187
## Detection Prevalence     0.4761           0.2574           0.2665
## Balanced Accuracy        0.9479           0.9522           0.9004
```

4. Utilice los modelos con el conjunto de prueba y determine la eficiencia del algoritmo para predecir y clasificar.

```
table(predBayes)
```

```
## predBayes
##       Cara  Economica Intermedia
##        209        113        117
```

```
table(test$Estado)
```

```
##
##       Cara  Economica Intermedia
##        222        106        111
```

5. Analice los resultados del modelo de regresión. ¿Qué tan bien le fue prediciendo?

6. Compare los resultados con el modelo de regresión lineal y el árbol de regresión que hizo en las hojas pasadas. ¿Cuál funcionó mejor?

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

8. Analice el modelo. ¿Cree que pueda estar sobre ajustado?

9. Haga un modelo usando validación cruzada, compare los resultados de este con los del modelo anterior. ¿Cuál funcionó mejor?

10. Compare la eficiencia del algoritmo con el resultado obtenido con el árbol de decisión (el de clasificación) y el modelo de random forest que hizo en la hoja pasada. ¿Cuál es mejor para predecir? ¿Cuál se demoró más en procesar?