Minería de datos: PRA2 - Proyecto de minería de datos

Autor: Juan Luis Acebal Rico

Enero 2025

Contents

Enunciado **1**

Recursos de programación **3**

Fecha de entrega **3**

RESPUESTAS **3**

Ejercicio 1 3
Ejercicio 2 11
Ejercicio 3 17
Ejercicio 5 25
Ejercicio 6 25
Ejercicio 7 34
Ejercicio 8 37

Enunciado

Como continuación del estudio iniciado en la Práctica 1, procedemos a **aplicar modelos analíticos, tanto no supervisados como supervisados**, sobre el juego de datos seleccionado y ya preparado. En esta **Práctica 2 tendréis que cargar los datos previamente preparados en la Práctica 1**.

El objetivo es que pongáis en práctica con vuestros propios datos todos los modelos no supervisados y supervisados que se han utilizado en las 3 PECs previas. Además, se propone que se utilicen métricas y algoritmos alternativos a los propuestos en las PECs ya realizadas.

Punto común para todos los ejercicios

En todos los apartados de los ejercicios de esta práctica se pide al estudiante, además de aplicar los diferentes métodos, analizar correctamente el problema, **detallarlo de manera exhaustiva**, resaltando el por qué del

análisis y cómo se ha realizado, incluir elementos visuales, explicar los resultados y realizar las comparativas oportunas con sus conclusiones.

En toda la práctica es **necesario documentar** cada apartado del ejercicio que se ha hecho, el por qué y como se ha realizado. Asimismo, todas las decisiones y conclusiones deberán ser presentados de forma razonada y clara, **contextualizando los resultados**, es decir, especificando todos y cada uno de los pasos que se hayan llevado a cabo para su resolución.

En definitiva, se pide al estudiante que complete los siguientes pasos con el juebo de datos preparado en la Práctica 1:

Modelos no supervisados

1. Aplicar el algoritmo **no supervisado** *k-means* basado en el concepto de distancia entre las medias de los grupos, sobre el juego de datos originales y los datos normalizados. Se recuerda que se deben utilizar las variables cuantiativas o binarias que formen parte de la base de datos. También, en ese apartado se debe decidir si los grupos se definen a partir de las variables normalizados o no y se debe seleccionar el número de clusters que mejor se ajuste a los datos.

2. Utilizando el número de clusters y los datos (normalizados o no) seleccionados en el punto 1, utilizar el algoritmo *k-medians* (basado en las medianas como centros de los clusters) para definir cada uno de los grupos. Comparar los resultados obtenidos con ambos algoritmos, *k-means* y *k-medians*, y comentad qué método os parece el más adecuado para vuestros datos.

3. Entrenar de nuevo el modelo basado en *k-means* que habéis seleccionado en el punto 1 pero usando una **métrica de distancia diferente a la distancia euclidiana** y comparad los resultados.

4. Utilizar los algoritmos **DBSCAN y OPTICS**, probando con diferentes valores del parámetro `eps` y `minPts`, y comparar los resultados con los métodos anteriores. Comentad si el número de clusters coincide con el punto 1 y si los casos que los forman son similares.

Modelos supervisados

5. Seleccionar una muestra de entrenamiento y una de test utilizando las proporciones que se consideren más adecudas en función de la disponibilidad de datos. Justificar dicha selección.

6. Una vez definida la variable objeto que se desea predecir, aplicar un modelo de generación de reglas a partir de **árboles de decisión** y ajustar las diferentes opciones (tamaño mínimo de los nodos, criterios de división, ...) para su obtención. Obtener el árbol sin y con opciones de poda. Obtener la matriz de confusión. Finalmente, comparar los resultados obtenidos con y sin opciones de poda. Alternativamente, si la variable objeto de estudio es cuantitativa pura se obtienen los criterios de error que nos permitan determinar la capacidad predictiva.

7. Aplicar un **modelo supervisado** diferente al del punto 6 (puede ser un algoritmo de árboles de decisión distinto u otro algoritmo supervisado alternativo). Comparar el resultado con el modelo generado anteriormente. Se pueden utilizar los criterios de evaluación de modelos descritos en el material docente de la asignatura.

8. Identificar eventuales **limitaciones** del dataset seleccionado y **analizar los riesgos** en el caso de utilizar el modelo para clasificar un nuevo caso. Por ejemplo, puede haber dificultades de sobreajuste, en el caso de clasificar, los porcentajes de falsos positivos y falsos negativos son similares, etc..

NOTA IMPORTANTE: *Recordad que si las variables en vuestra base de datos tienen unidades de medida muy distintas es recomendable transformar las variables para evitar el efecto escala debido a las diferentes unidades de medida.*

Recursos de programación

- Incluimos en este apartado una lista de recursos de programación para minería de datos donde podréis encontrar ejemplos, ideas e inspiración:
  - Espacio de recursos UOC para ciencia de datos
  - Buscador de código R
  - Colección de cheatsheets en R

Fecha de entrega

La fecha límite de entrega es el 20/01/2025.

RESPUESTAS

El dataset usado: https://www.kaggle.com/datasets/die9origephit/amazon-data-science-books

Ejercicio 1

```
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(cluster)

#df <- read.csv('final_book_dataset_kaggle2.csv')
df <- read.csv('df.csv')
df <- df[ , -1]
summary(df)
```

```
##     title              author              price
##  Length:502         Length:502         Min.   :  0.99
##  Class :character   Class :character   1st Qu.: 21.12
##  Mode  :character   Mode  :character   Median : 36.99
##                                        Mean   : 42.32
##                                        3rd Qu.: 49.99
##                                        Max.   :287.14
##  price..including.used.books.    pages            avg_reviews
##  Length:502                   Length:502         Min.   :1.000
##  Class :character             Class :character   1st Qu.:4.400
##  Mode  :character             Mode  :character   Median :4.500
##                                                  Mean   :4.486
##                                                  3rd Qu.:4.700
##                                                  Max.   :5.000
##    n_reviews         star5            star4           star3
##  Min.   :  1.0   Min.   : 22.00   Min.   : 0.00   Min.   : 0.000
##  1st Qu.: 12.0   1st Qu.: 64.25   1st Qu.:10.00   1st Qu.: 2.000
##  Median : 47.7   Median : 74.00   Median :15.00   Median : 6.000
##  Mean   :109.8   Mean   : 73.62   Mean   :15.28   Mean   : 6.353
##  3rd Qu.:142.0   3rd Qu.: 81.00   3rd Qu.:19.00   3rd Qu.: 9.000
##  Max.   :988.0   Max.   :100.00   Max.   :64.00   Max.   :34.000
##      star2            star1           ingles        publisher
##  Min.   : 0.000   Min.   : 0.00   Min.   :0.0000   Length:502
##  1st Qu.: 0.000   1st Qu.: 0.00   1st Qu.:1.0000   Class :character
##  Median : 2.000   Median : 0.00   Median :1.0000   Mode  :character
##  Mean   : 2.813   Mean   : 1.93   Mean   :0.9701
##  3rd Qu.: 4.000   3rd Qu.: 3.00   3rd Qu.:1.0000
##  Max.   :22.000   Max.   :19.00   Max.   :1.0000
##     Peso..g.
##  Min.   :   15.88
##  1st Qu.:  680.39
##  Median : 1018.31
##  Mean   : 2047.91
##  3rd Qu.: 2782.56
##  Max.   :13698.48
```

```
head(df)
```

```
##                                                                                                 title
## 1                    Data Analysis Using R (Low Priced Edition): A Primer for Data Scientist
## 2 Head First Data Analysis: A learner's guide to big numbers, statistics, and good decisions
## 3   Guerrilla Data Analysis Using Microsoft Excel: Overcoming Crap Data and Excel Skirmishes
## 4                   Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython
## 5                             Excel Data Analysis For Dummies (For Dummies (Computer/Tech))
## 6             SQL for Data Analysis: Advanced Techniques for Transforming Data into Insights
##                                author price price..including.used.books.
## 1                [ Dr Dhaval Maheta]  6.75                          6.75
## 2                                 N/A 33.72                 21.49 - 33.72
## 3 [ Oz du Soleil,  and , Bill Jelen] 32.07                         32.07
## 4                [ William McKinney] 53.99                         53.99
## 5                  [ Paul McFedries] 24.49                         24.49
## 6                  [ Cathy Tanimura] 90.00                         40.49
##          pages avg_reviews n_reviews star5 star4 star3 star2 star1 ingles
## 1 Menos de 750         4.4      23.0    55    39     6     0     0      1
```

```
## 2 Menos de 500         4.3     124.0    61    20     9     4     6      1
## 3 Menos de 500         4.7      10.0    87    13     0     0     0      1
## 4 Menos de 750         4.6      54.4    75    16     5     2     2      1
## 5 Menos de 500         3.9      12.0    52    17    10    10    10      1
## 6 Menos de 500         4.7     205.0    75    18     2     2     2      1
##                                                publisher  Peso..g.
## 1    Notion Press Media Pvt Ltd (November 22, 2021) 1147.5878
## 2     O'Reilly Media; 1st edition (August 18, 2009)  889.0403
## 3 Holy Macro! Books; Third edition (August 1, 2022)  635.0288
## 4   O'Reilly Media; 2nd edition (November 14, 2017)  666.7802
## 5       For Dummies; 5th edition (February 3, 2022)  589.6696
## 6     O'Reilly Media; 1st edition (October 5, 2021)  975.2228
```

```
str(df)
```

```
## 'data.frame':    502 obs. of  15 variables:
##  $ title                       : chr  "Data Analysis Using R (Low Priced Edition): A Primer for Data
##  $ author                      : chr  "[ Dr Dhaval Maheta]" "N/A" "[ Oz du Soleil,  and , Bill Jelen]
##  $ price                       : num  6.75 33.72 32.07 53.99 24.49 ...
##  $ price..including.used.books.: chr  "6.75" " 21.49 - 33.72 " "32.07" "53.99" ...
##  $ pages                       : chr  "Menos de 750" "Menos de 500" "Menos de 500" "Menos de 750" ..
##  $ avg_reviews                 : num  4.4 4.3 4.7 4.6 3.9 4.7 5 4.3 4.5 4.9 ...
##  $ n_reviews                   : num  23 124 10 54.4 12 205 5 14 46 201 ...
##  $ star5                       : int  55 61 87 75 52 75 84 100 78 73 ...
##  $ star4                       : int  39 20 13 16 17 18 9 0 11 13 ...
##  $ star3                       : int  6 9 0 5 10 2 4 0 11 7 ...
##  $ star2                       : int  0 4 0 2 10 2 2 0 0 3 ...
##  $ star1                       : int  0 6 0 2 10 2 0 0 0 3 ...
##  $ ingles                      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ publisher                   : chr  "Notion Press Media Pvt Ltd (November 22, 2021)" "O'Reilly Medi
##  $ Peso..g.                    : num  1148 889 635 667 590 ...
```

```
df$pages  <- as.factor(df$pages)
df <- df[, names(df) != "price..including.used.books."]
df$ingles <- as.factor(df$ingles)
```

```
#df <- na.omit(df)
df_numeric <- df %>% select_if(is.numeric)
#df_scaled <- scale(df_numeric)
df_scaled <- as.data.frame(scale(df_numeric, center = TRUE, scale = apply(df_numeric, 2, function(x) ma
```

```
summary(df_scaled)
```

```
##      price           avg_reviews          n_reviews            star5
##  Min.   :-0.14445   Min.   :-0.871614   Min.   :-0.11022   Min.   :-0.661840
##  1st Qu.:-0.07409   1st Qu.:-0.021614   1st Qu.:-0.09908   1st Qu.:-0.120173
##  Median :-0.01863   Median : 0.003387   Median :-0.06291   Median : 0.004827
##  Mean   : 0.00000   Mean   : 0.000000   Mean   : 0.00000   Mean   : 0.000000
##  3rd Qu.: 0.02679   3rd Qu.: 0.053387   3rd Qu.: 0.03263   3rd Qu.: 0.094570
##  Max.   : 0.85555   Max.   : 0.128386   Max.   : 0.88978   Max.   : 0.338160
##      star4               star3               star2               star1
##  Min.   :-0.238764   Min.   :-0.18684   Min.   :-0.12785   Min.   :-0.1016
```

```
##  1st Qu.:-0.082514   1st Qu.:-0.12802   1st Qu.:-0.12785   1st Qu.:-0.1016
##  Median :-0.004389   Median :-0.01037   Median :-0.03694   Median :-0.1016
##  Mean   : 0.000000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000
##  3rd Qu.: 0.058111   3rd Qu.: 0.07787   3rd Qu.: 0.05397   3rd Qu.: 0.0563
##  Max.   : 0.761236   Max.   : 0.81316   Max.   : 0.87215   Max.   : 0.8984
##     Peso..g.
##  Min.   :-0.14851
##  1st Qu.:-0.09995
##  Median :-0.07525
##  Mean   : 0.00000
##  3rd Qu.: 0.05369
##  Max.   : 0.85149
```

```
summary(df_numeric)
```

```
##      price          avg_reviews      n_reviews         star5
##  Min.   :  0.99   Min.   :1.000   Min.   :  1.0   Min.   : 22.00
##  1st Qu.: 21.12   1st Qu.:4.400   1st Qu.: 12.0   1st Qu.: 64.25
##  Median : 36.99   Median :4.500   Median : 47.7   Median : 74.00
##  Mean   : 42.32   Mean   :4.486   Mean   :109.8   Mean   : 73.62
##  3rd Qu.: 49.99   3rd Qu.:4.700   3rd Qu.:142.0   3rd Qu.: 81.00
##  Max.   :287.14   Max.   :5.000   Max.   :988.0   Max.   :100.00
##      star4           star3            star2            star1
##  Min.   : 0.00   Min.   : 0.000   Min.   : 0.000   Min.   : 0.00
##  1st Qu.:10.00   1st Qu.: 2.000   1st Qu.: 0.000   1st Qu.: 0.00
##  Median :15.00   Median : 6.000   Median : 2.000   Median : 0.00
##  Mean   :15.28   Mean   : 6.353   Mean   : 2.813   Mean   : 1.93
##  3rd Qu.:19.00   3rd Qu.: 9.000   3rd Qu.: 4.000   3rd Qu.: 3.00
##  Max.   :64.00   Max.   :34.000   Max.   :22.000   Max.   :19.00
##     Peso..g.
##  Min.   :   15.88
##  1st Qu.:  680.39
##  Median : 1018.31
##  Mean   : 2047.91
##  3rd Qu.: 2782.56
##  Max.   :13698.48
```

```
head(df_scaled)
```

```
##         price avg_reviews   n_reviews       star5      star4       star3
## 1 -0.12432207 -0.02161355 -0.08793317 -0.23876290  0.3706113 -0.01037028
## 2 -0.03007081 -0.04661355  0.01439712 -0.16183982  0.0737363  0.07786501
## 3 -0.03583701  0.05338645 -0.10110440  0.17149351 -0.0356387 -0.18684087
## 4  0.04076617  0.02838645 -0.05611959  0.01764736  0.0112363 -0.03978205
## 5 -0.06232662 -0.14661355 -0.09907805 -0.27722444  0.0268613  0.10727678
## 6  0.16660926  0.05338645  0.09646399  0.01764736  0.0424863 -0.12801734
##         star2        star1    Peso..g.
## 1 -0.12785223 -0.101593625 -0.06580084
## 2  0.05396595  0.214195848 -0.08469692
## 3 -0.12785223 -0.101593625 -0.10326148
## 4 -0.03694314  0.003669532 -0.10094091
## 5  0.32669323  0.424722164 -0.10657658
## 6 -0.03694314  0.003669532 -0.07839823
```

```r
head(df_numeric)
```

```
##   price avg_reviews n_reviews star5 star4 star3 star2 star1  Peso..g.
## 1  6.75         4.4      23.0    55    39     6     0     0 1147.5878
## 2 33.72         4.3     124.0    61    20     9     4     6  889.0403
## 3 32.07         4.7      10.0    87    13     0     0     0  635.0288
## 4 53.99         4.6      54.4    75    16     5     2     2  666.7802
## 5 24.49         3.9      12.0    52    17    10    10    10  589.6696
## 6 90.00         4.7     205.0    75    18     2     2     2  975.2228
```

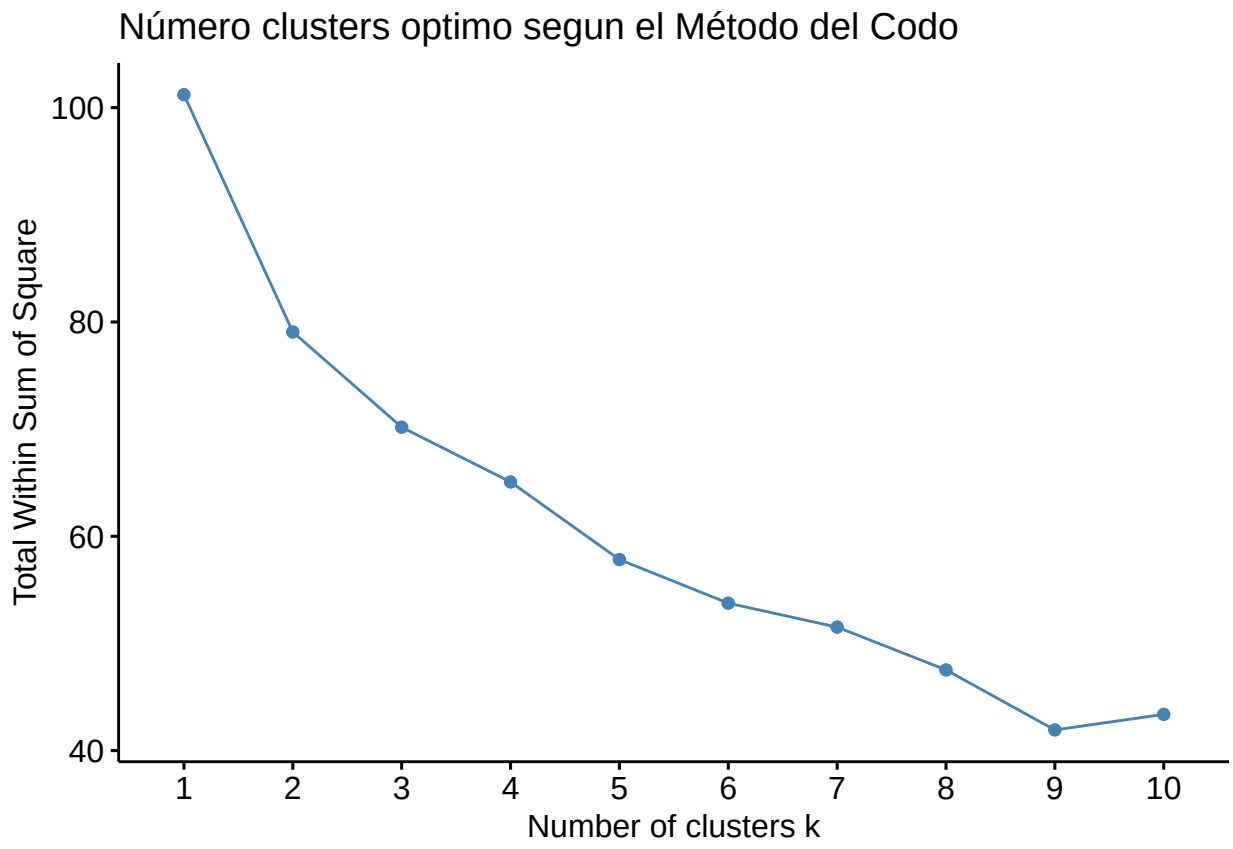
Se obtine la agrupación mediante k-means con los datos originales y normalizados.

```r
library(factoextra)
```

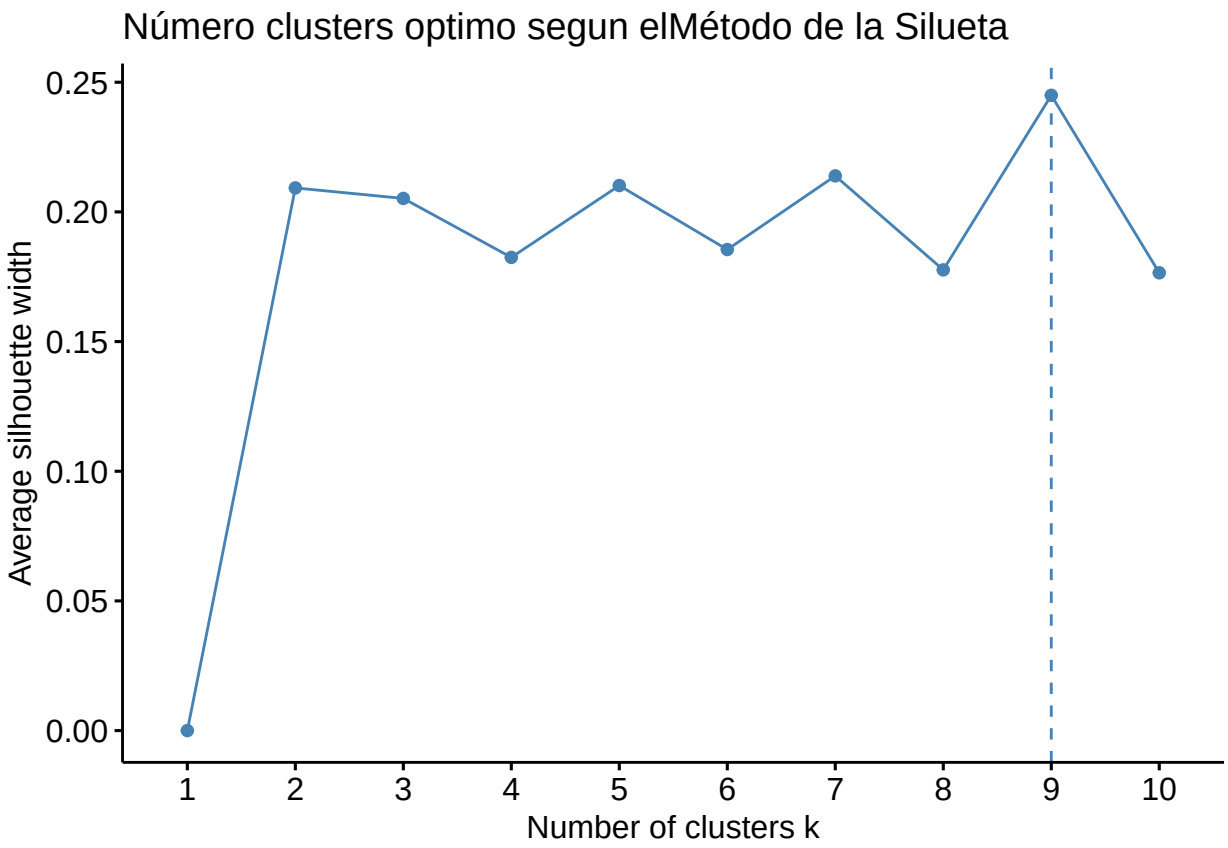
```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

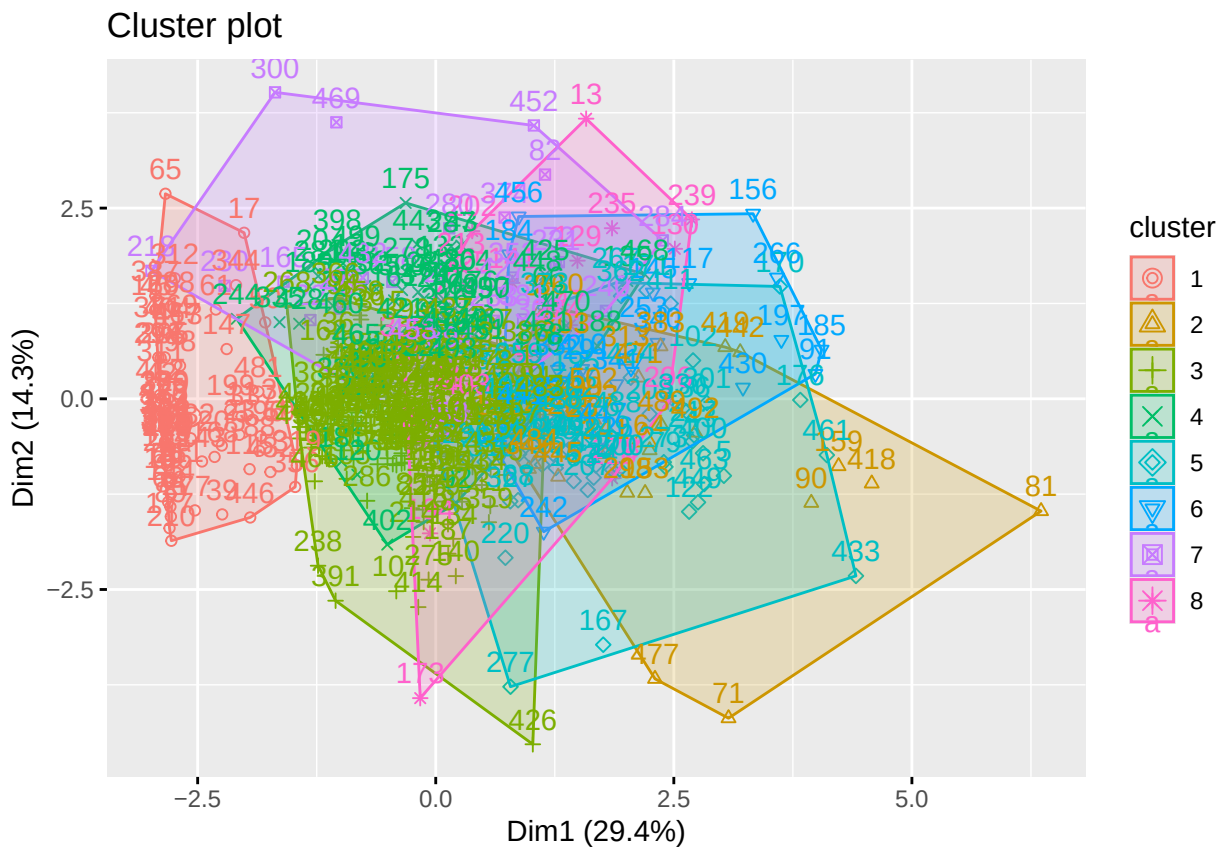
```r
library(ggplot2)
set.seed(123)
fviz_nbclust(df_scaled, kmeans, method = "wss") +
  ggtitle("Número clusters optimo segun el Método del Codo")
```



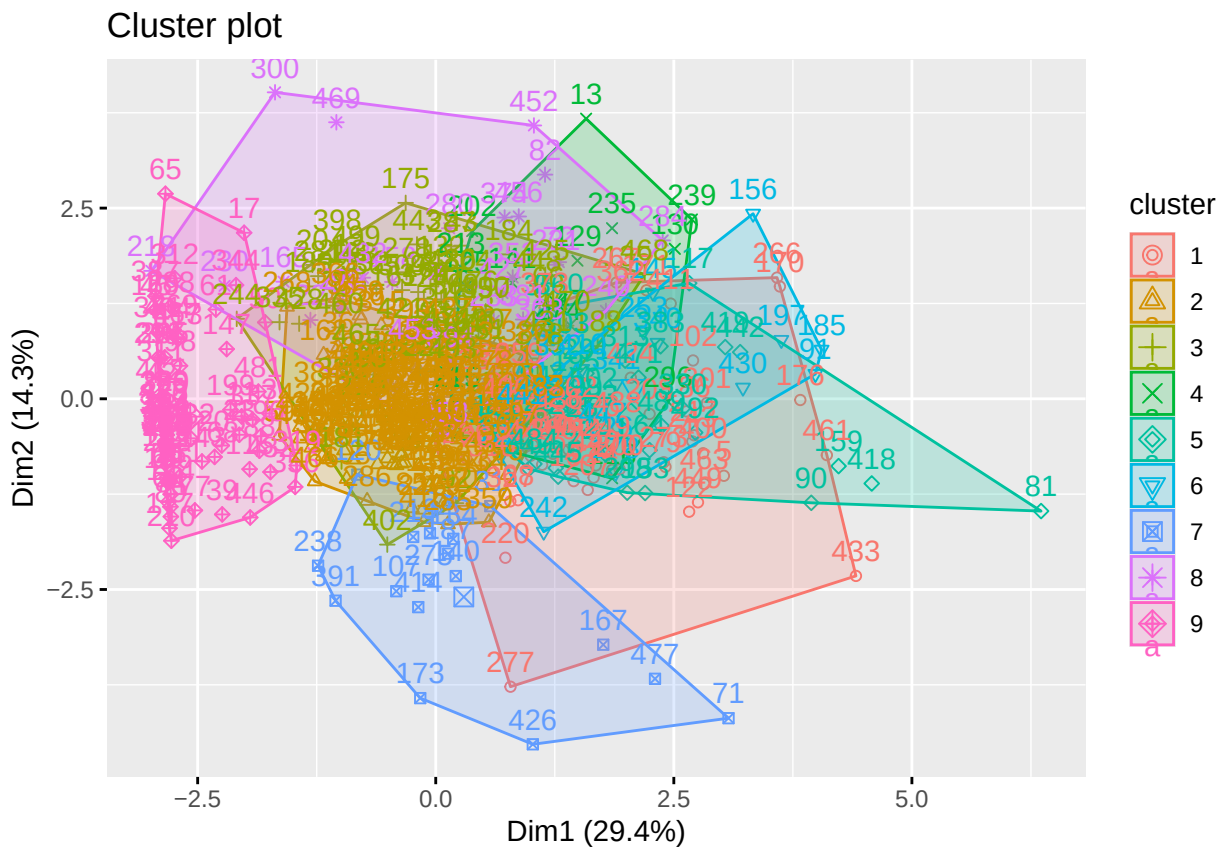
```
fviz_nbclust(df_scaled, kmeans, method = "silhouette") +
  ggtitle("Número clusters optimo segun elMétodo de la Silueta")
```



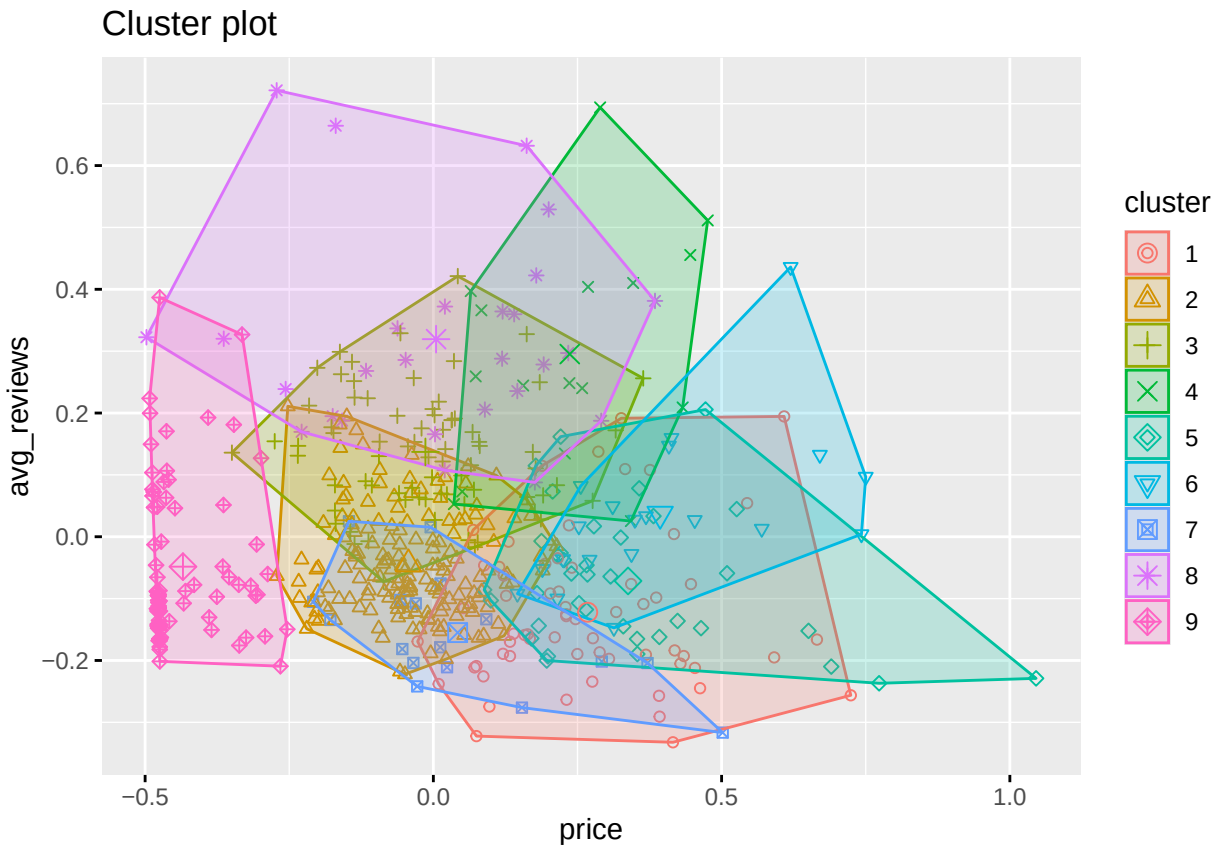

```
kmeans_result <- kmeans(df_scaled, centers = 8, nstart = 25)
fviz_cluster(kmeans_result, data = df_scaled)
```




```
kmeans_result <- kmeans(df_scaled, centers = 9, nstart = 25)
fviz_cluster(kmeans_result, data = df_scaled)
```




```
fviz_cluster(kmeans_result, data = df_scaled, geom = "point", stand = FALSE) +
  xlab("price") + ylab("avg_reviews")
```



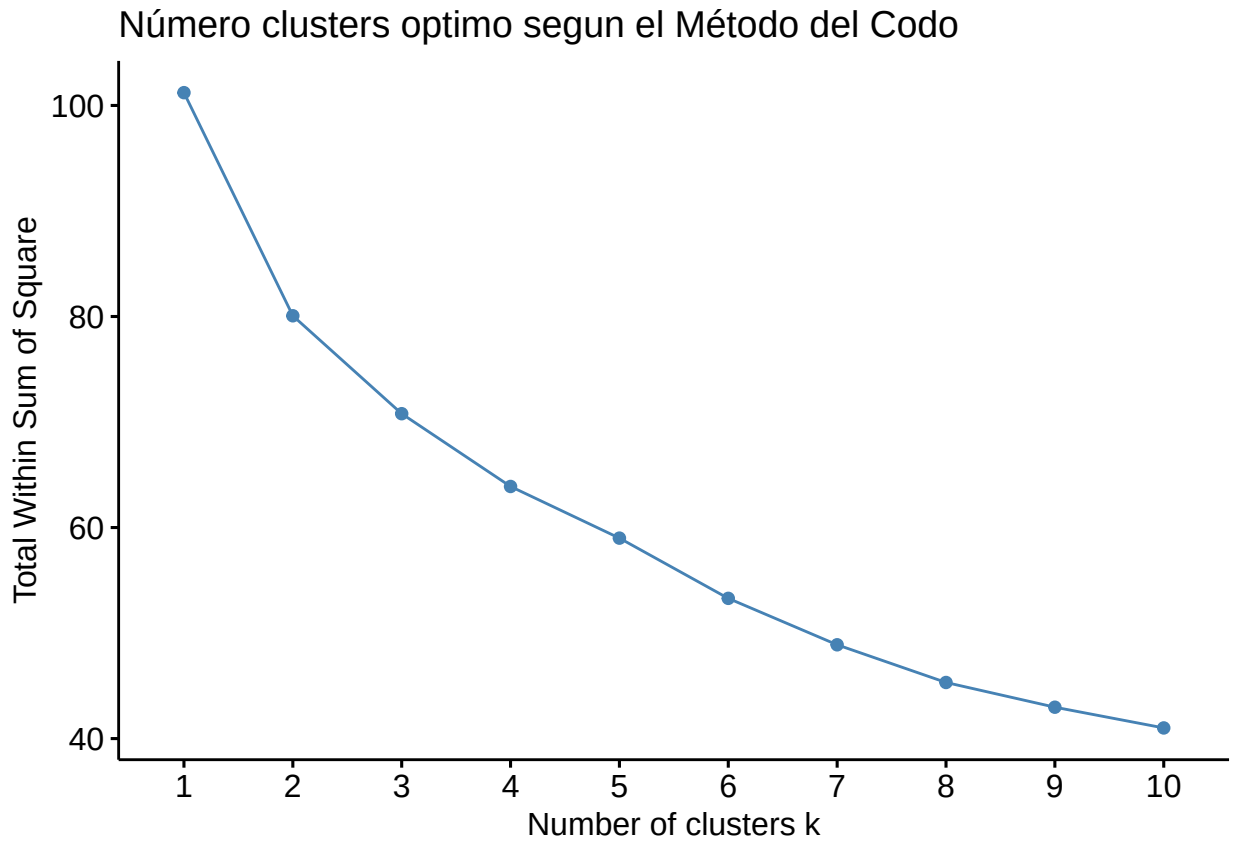
Se analizan, muestran y comentan las medidas de calidad del modelo generado.

Se seleccionaron 9 clusters basándose en el método de la silueta, ya que ofrece una separación más clara entre los clusters según la medida promedio de cohesión. Sin embargo, se probó con 6 clusters como alternativa, siguiendo el método del codo, para comparar cómo afecta la decisión al agrupamiento final. Esto me permite evaluar la estabilidad de los clusters y su visualización e interpretación en diferentes configuraciones ### Se comentan las conclusiones. Como primera visualizacion y explicacion en la PRA2 (dejando al margen lo que hice en la PRA1 que no la tomo como guion en lo que se refiere a la creación y desarrollo del modelo, a excepcion de utilizar los datos limpios, categorizaciones, estandarizaciones etc,. . . , que ha sido lo único que he utilizado de la PRA1) Vemos que el gráfico 1 y 2 no dicen nada, lo que para mi sugiere que, hay tanto ruido que el algoritmo no va a poder darnos unos clusters bien delimitados, cosa diferente a la vista en el grafico 3, en el que visualizo solamente 2 variables. ***

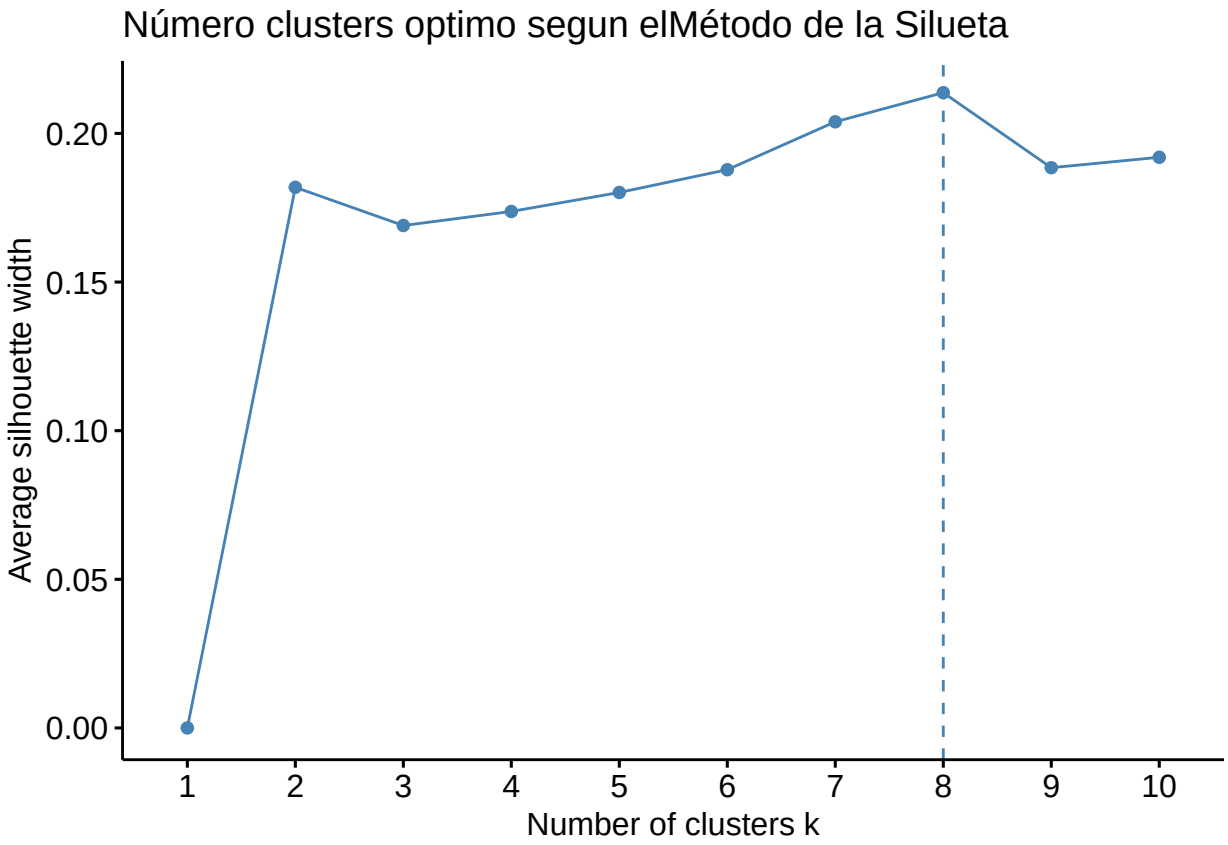
Ejercicio 2

Se obtiene la agrupación k-medians con el número de grupos seleccionado en el ejercicio 1.

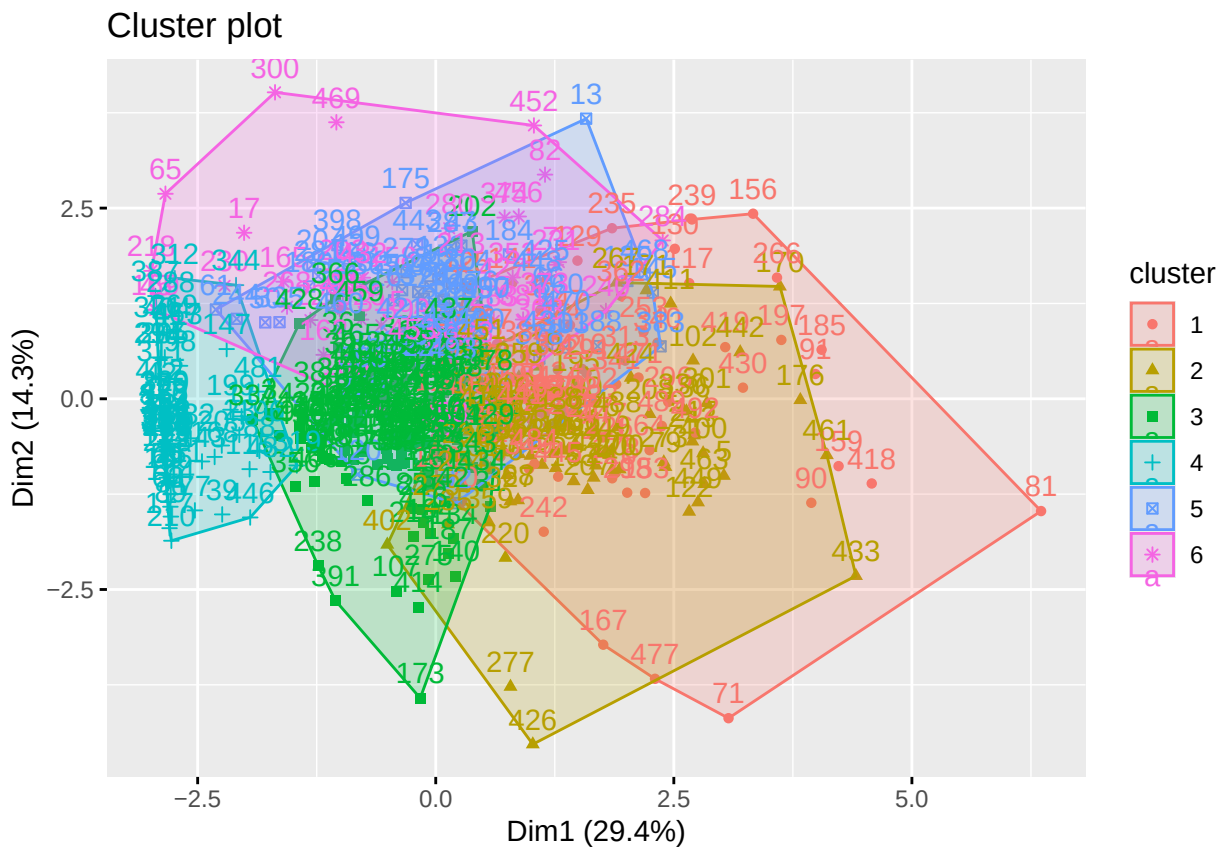
```
fviz_nbclust(df_scaled, pam, method = "wss") +
  ggtitle("Número clusters optimo segun el Método del Codo")
```



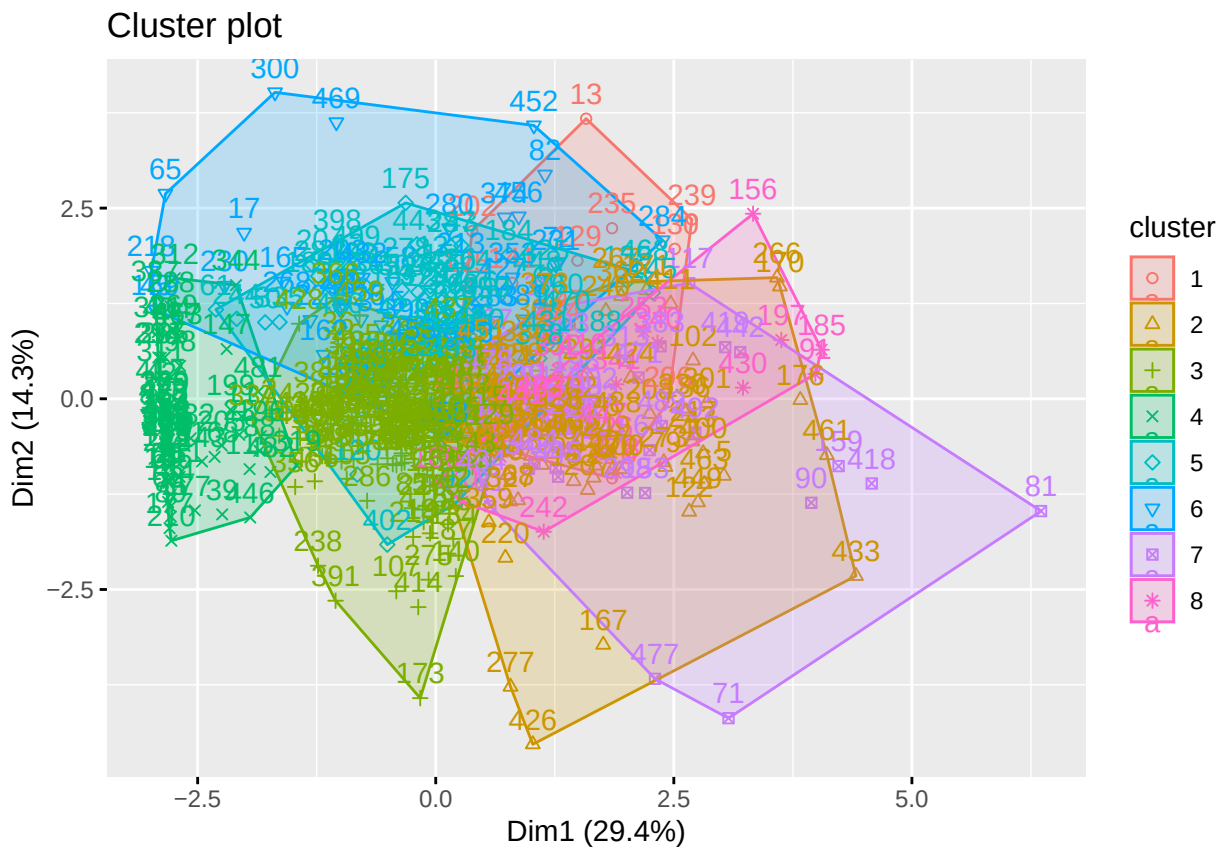

```
fviz_nbclust(df_scaled, pam, method = "silhouette") +
  ggtitle("Número clusters optimo segun elMétodo de la Silueta")
```



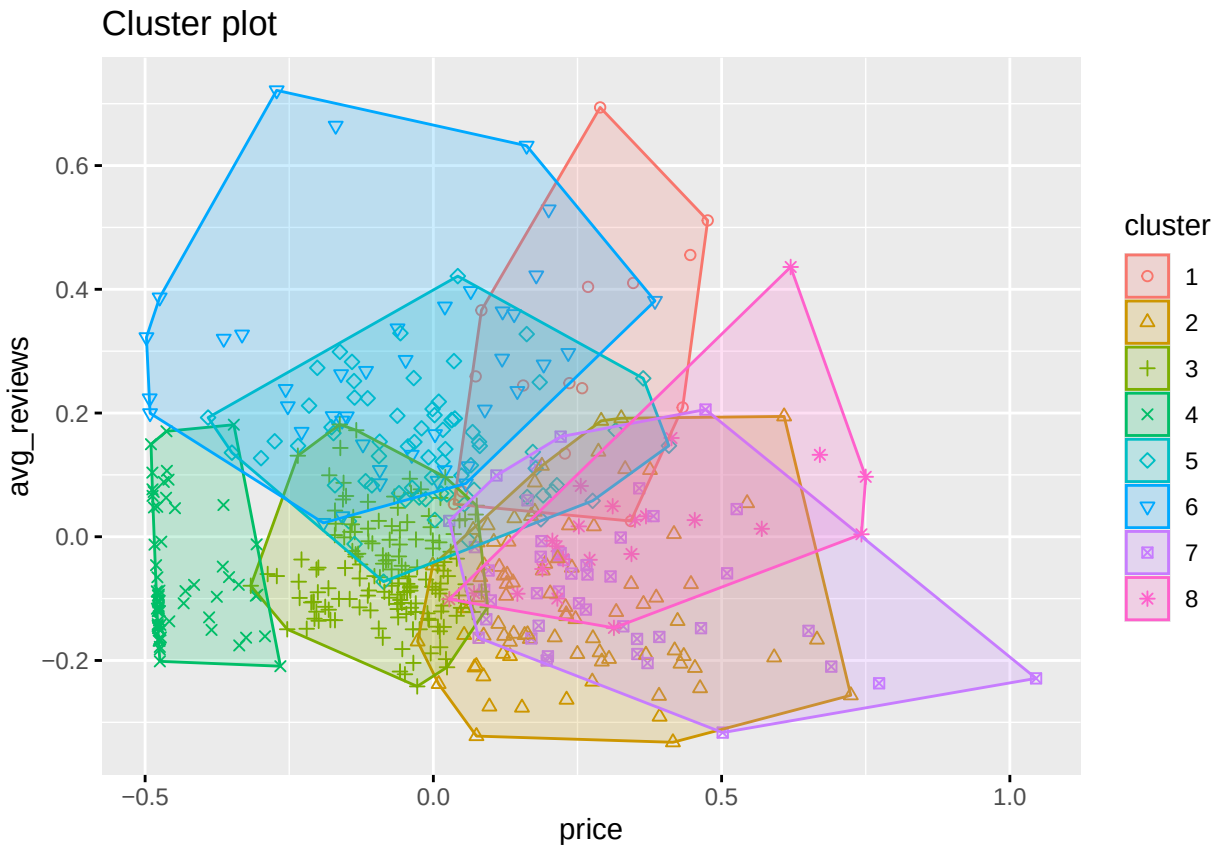

```
kmedians_result <- pam(df_scaled, k = 6)
fviz_cluster(kmedians_result, data = df_scaled)
```



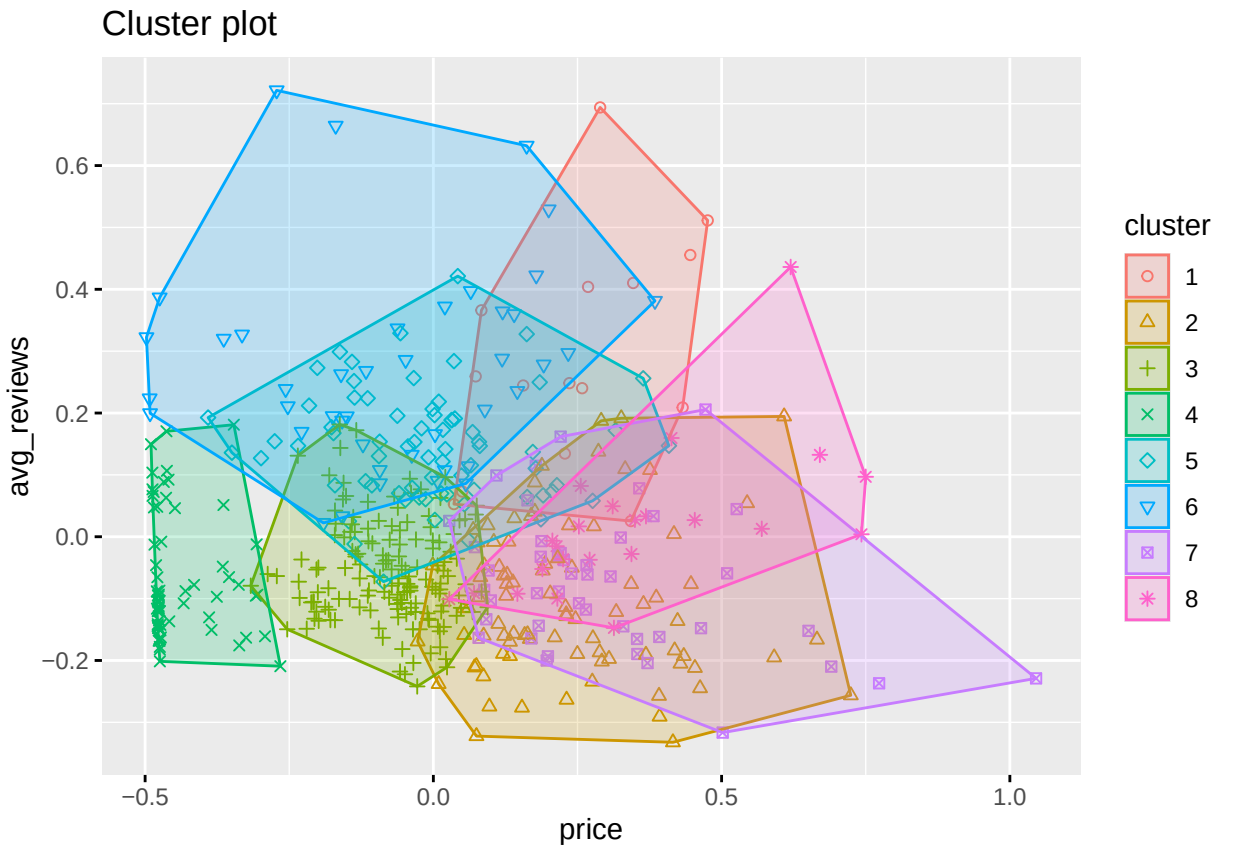

```
kmedians_result <- pam(df_scaled, k = 8)
fviz_cluster(kmedians_result, data = df_scaled)
```




```
fviz_cluster(kmedians_result, data = df_scaled, geom = "point", stand = FALSE) +
  xlab("price") + ylab("avg_reviews")
```




```
kmedians_result <- pam(df_scaled, k = 8)

fviz_cluster(kmedians_result, data = df_scaled, geom = "point", stand = FALSE) +
  xlab("price") + ylab("avg_reviews")
```



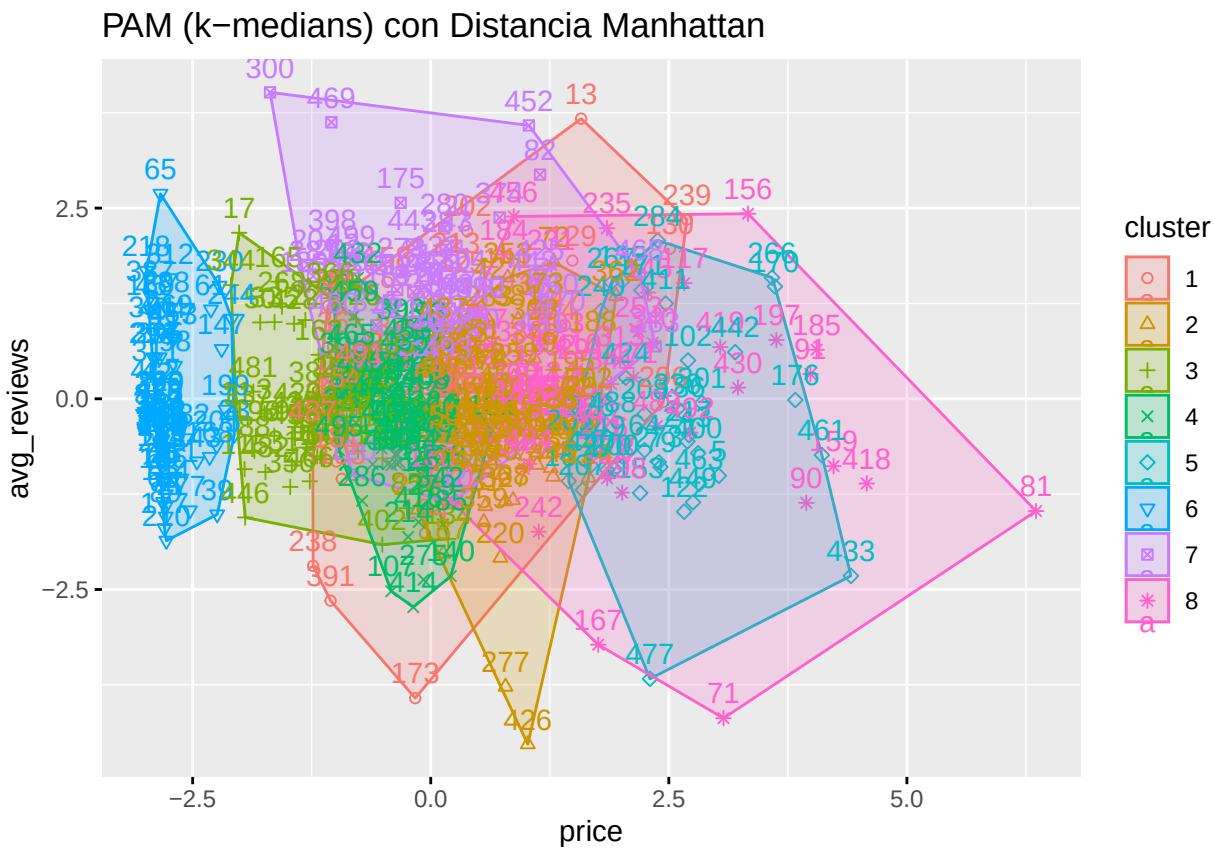
Se comparan los resultados con los obtenidos en el ejercicio 1.

Para la comparacion de los resultados obtenidos en el ejercicio 1 y 2, se puede observar que los resultados son muy similares, lo que sugiere que el algoritmo k-medians no es muy diferente al k-means. En este caso, el k-medians ha generado 9 clusters, mientras que el k-means ha tambien 9 clusters. ### Se comentan las conclusiones. Los resultados obtenidos con k-means y k-medians son bastante similares, lo que parece que ambos algoritmos funcionan parecido para este conjunto de datos. Dado que k-medians es más robusto frente a outliers, la similitud entre ambos resultados indica que los datos probablemente no contienen valores extremos significativos que afecten la agrupación. En este caso, cualquiera de los dos métodos podría considerarse adecuado, pero con ciertas diferencias. ***

Ejercicio 3

Se obtiene la agrupación mediante k-means (con los grupos seleccionados en el ejercicio 1), pero usando una métrica de distancia distinta.

```
pam_res_manhattan <- pam(df_scaled, k = 8, metric = "manhattan")

fviz_cluster(pam_res_manhattan, data = df_scaled) +
  xlab("price") + ylab("avg_reviews")+
  ggtitle("PAM (k-medians) con Distancia Manhattan")
```



Se comparan los resultados con los obtenidos en el ejercicio 1.

La visualización es peor que en el ejercicio 1, en el ejercicio 1 tenemos al menos 1 cluster claramente definido por media de valoraciones y precio.. ### Se comentan las conclusiones. La distancia Manhattan es una métrica de distancia alternativa que mide la suma de las diferencias absolutas entre las coordenadas de dos puntos. En este caso, al aplicarla al algoritmo PAM, los resultados son menos claros y definidos que con la distancia euclidiana. Esto sugiere que la distancia euclidiana es más adecuada para este conjunto de datos, ya que permite una mejor separación de los clusters. La distancia Manhattan es útil en casos donde las variables tienen diferentes escalas o cuando se desea penalizar más las diferencias en una dimensión específica. En este caso, la distancia euclidiana parece ser más efectiva para encontrar patrones en los datos. *** ## Ejercicio 4

Se aplican lo algoritmos DBSCAN y OPTICS de forma correcta.

```
library(dbscan)
```

```
##
## Attaching package: 'dbscan'
```

```
## The following object is masked from 'package:stats':
##
##     as.dendrogram
```

```r
library(cluster)

valores_eps <- seq(0.02, 0.2, by = 0.02)
valores_minPts <- 1:30

mejor_modelo <- NULL
mejor_silueta <- -1

for (eps in valores_eps) {
    for (minPts in valores_minPts) {
        modelo <- dbscan(df_scaled, eps = eps, minPts = minPts)
        clusters <- modelo$cluster
        clusters_validos <- clusters[clusters != 0]
        datos_validos <- df_scaled[clusters != 0, , drop = FALSE]
        tamanos_clusters <- table(clusters_validos)
        if (length(unique(clusters_validos)) > 1 && all(tamanos_clusters >= 2)) {
            matriz_distancias <- dist(datos_validos)
            if (nrow(as.matrix(matriz_distancias)) > 1) {
                silueta <- tryCatch(
                    silhouette(clusters_validos, matriz_distancias),
                    error = function(e) NULL
                )

                if (!is.null(silueta)) {
                    promedio_silueta <- mean(silueta[, 3])
                    if (promedio_silueta > mejor_silueta) {
                        mejor_silueta <- promedio_silueta
                        mejor_modelo <- list(modelo = modelo, eps = eps, minPts = minPts)
                    }
                }
            }
        }
    }
}
```

```r
mejor_modelo
```

```
## $modelo
## DBSCAN clustering for 502 objects.
## Parameters: eps = 0.08, minPts = 5
## Using euclidean distances and borderpoints = TRUE
## The clustering contains 2 cluster(s) and 475 noise points.
##
##   0   1   2
## 475  22   5
##
## Available fields: cluster, eps, minPts, metric, borderPoints
##
## $eps
## [1] 0.08
##
## $minPts
## [1] 5
```

```
mejor_silueta
```

```
## [1] 0.7464372
```

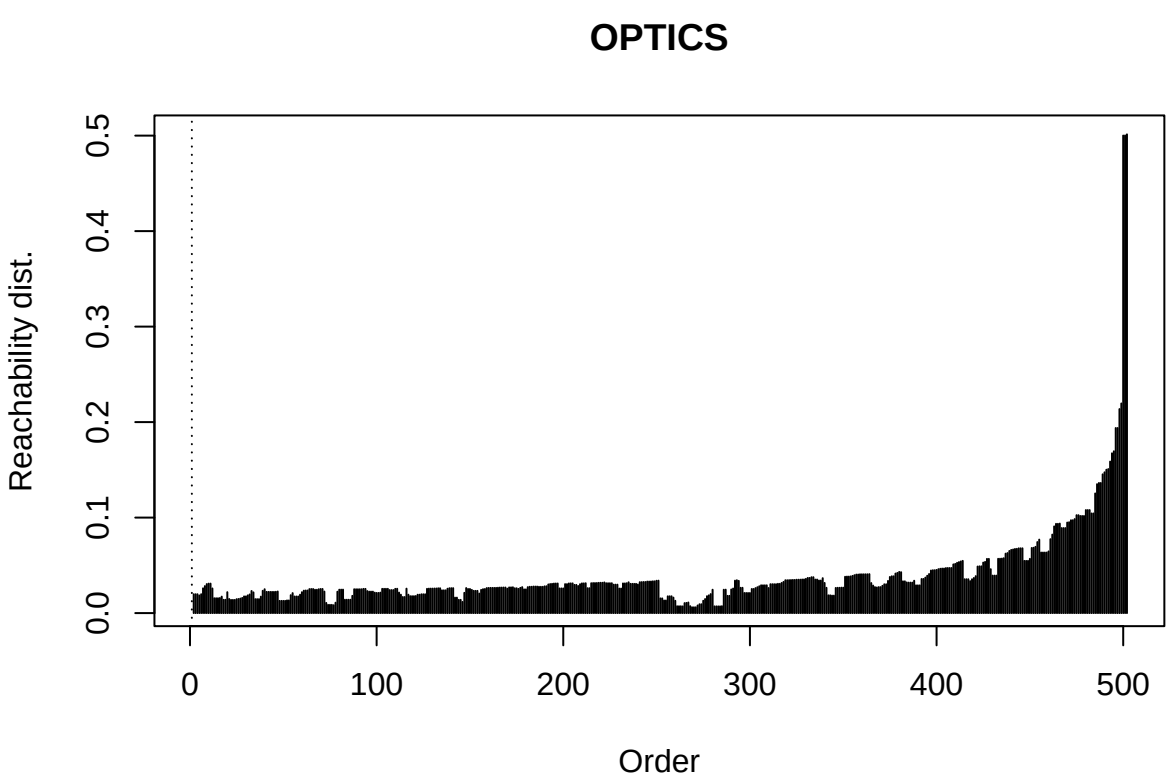
```
library(dbscan)
subset <- df_scaled[, c("price", "avg_reviews", "n_reviews")]#, "Peso..g.")]
dbscan_res <- dbscan(subset, eps = 0.08, minPts = 5)
dbscan_res
```

```
## DBSCAN clustering for 502 objects.
## Parameters: eps = 0.08, minPts = 5
## Using euclidean distances and borderpoints = TRUE
## The clustering contains 1 cluster(s) and 41 noise points.
##
##   0   1
##  41 461
##
## Available fields: cluster, eps, minPts, metric, borderPoints
```

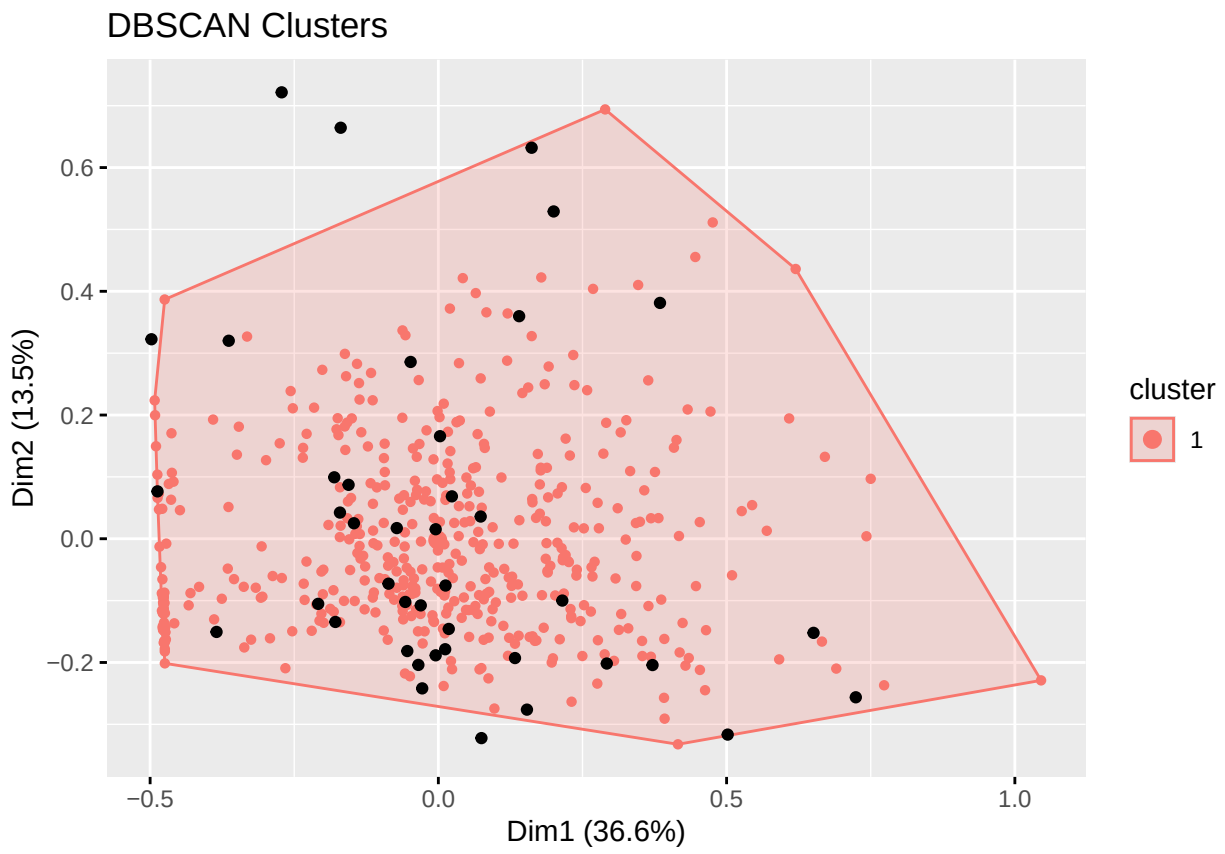
```
optics_res <- optics(subset, minPts = 5)
optics_res
```

```
## OPTICS ordering/clustering for 502 objects.
## Parameters: minPts = 5, eps = 0.527902488990452, eps_cl = NA, xi = NA
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
##                   eps_cl, xi
```

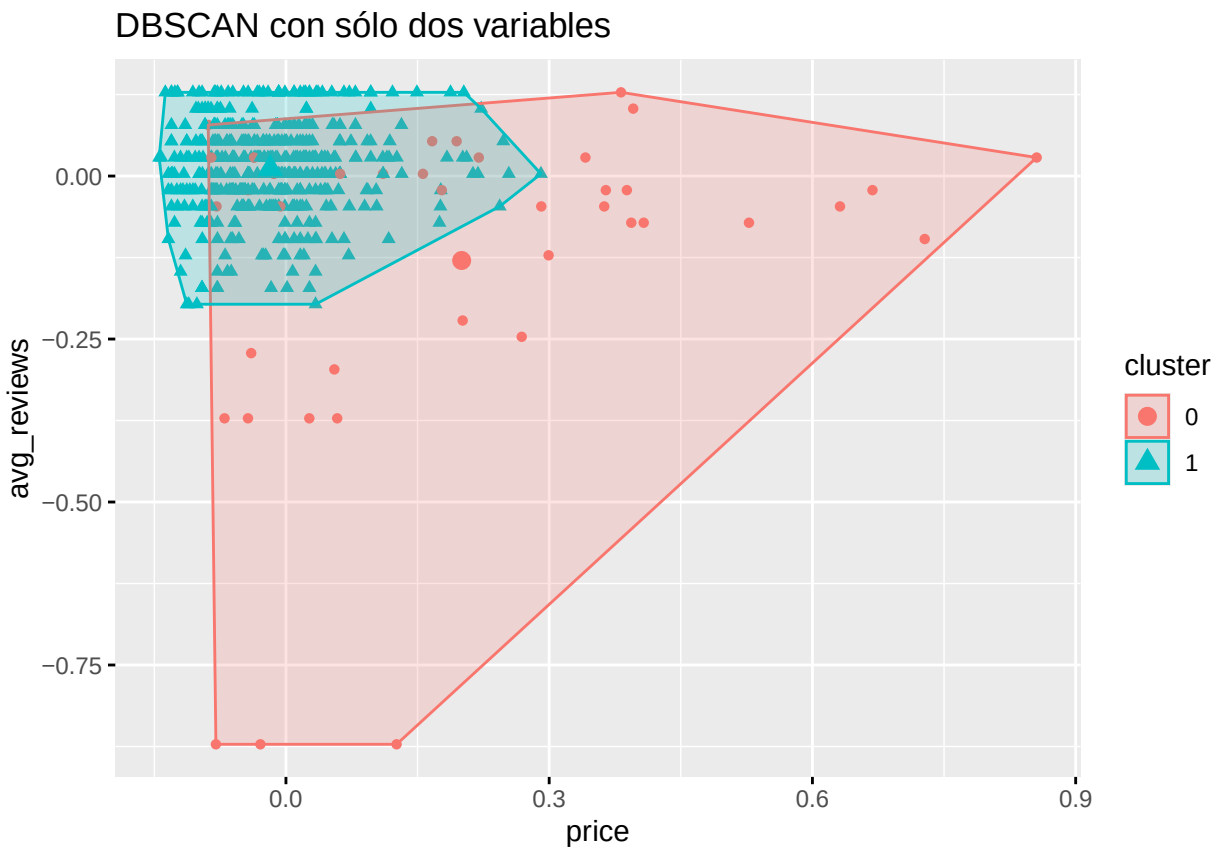
```
plot(optics_res, main = "OPTICS")
```




```
fviz_cluster(dbscan_res, data = df_scaled,
             geom = "point", stand = FALSE) +
  ggtitle("DBSCAN Clusters")
```




```
dbscan_list <- list(data = df_scaled[, c("price", "avg_reviews")],
                    cluster = dbscan_res$cluster)

fviz_cluster(dbscan_list,
             geom = "point",
             stand = FALSE) +
  ggtitle("DBSCAN con sólo dos variables")
```



Se prueban, describen e interpretan los resultados con diferentes valores de `eps` y `minPts`.

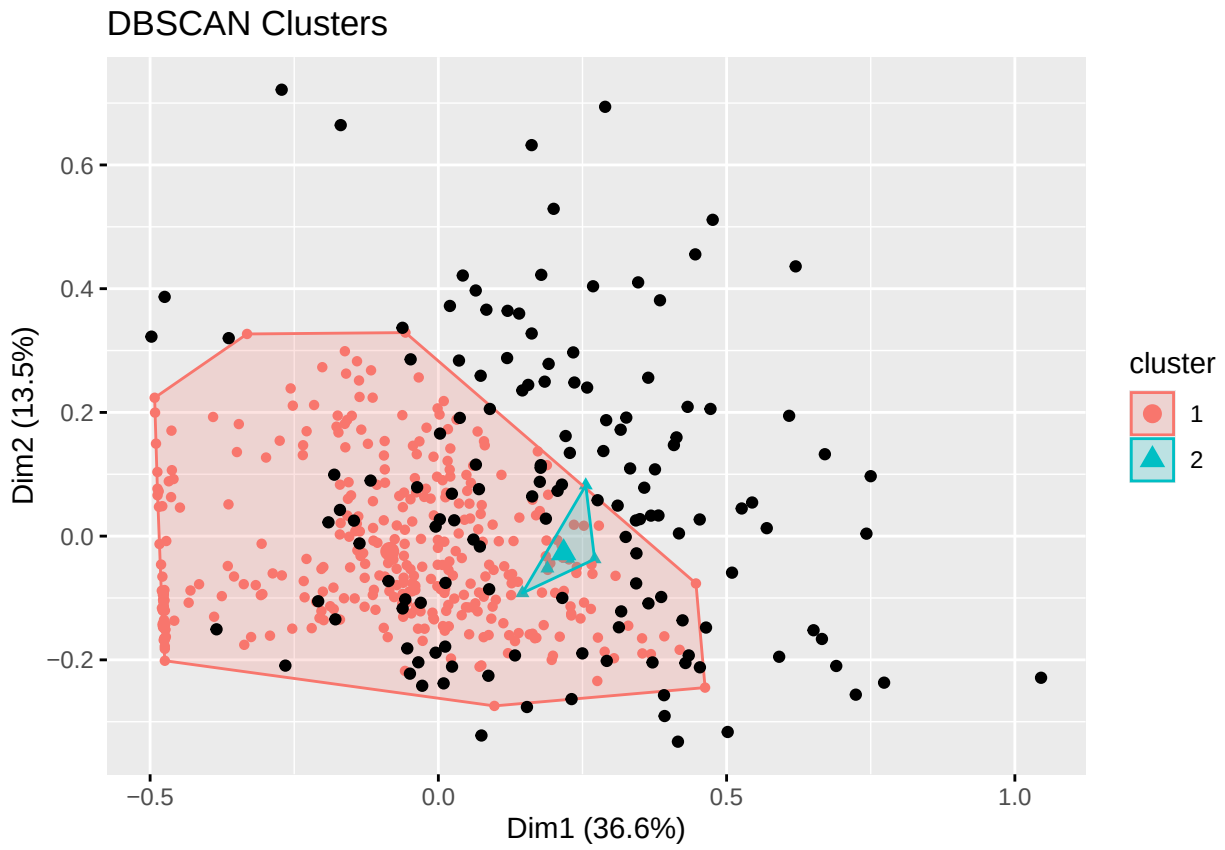
```
dbscan_res <- dbscan(df_scaled, eps = 0.20, minPts = 5)
dbscan_res
```

```
## DBSCAN clustering for 502 objects.
## Parameters: eps = 0.2, minPts = 5
## Using euclidean distances and borderpoints = TRUE
## The clustering contains 2 cluster(s) and 139 noise points.
##
##   0   1   2
## 139 358   5
##
## Available fields: cluster, eps, minPts, metric, borderPoints
```

```
optics_res <- optics(df_scaled, minPts = 2)
optics_res
```

```
## OPTICS ordering/clustering for 502 objects.
## Parameters: minPts = 2, eps = 0.591998006019714, eps_cl = NA, xi = NA
## Available fields: order, reachdist, coredist, predecessor, minPts, eps,
##                   eps_cl, xi
```

```
fviz_cluster(dbscan_res, data = df_scaled,
             geom = "point", stand = FALSE) +
  ggtitle("DBSCAN Clusters")
```



Se obtiene una medida de lo bueno que es el agrupamiento.

```
kmeans_sil <- silhouette(kmeans_result$cluster, dist(df_scaled))
mean_sil_kmeans <- mean(kmeans_sil[, 3])
mean_sil_kmeans
```

```
## [1] 0.2425784
```

```
kmeans_sil_man <- silhouette(pam_res_manhattan$cluster, dist(df_scaled))
mean_sil_kmeans_man <- mean(kmeans_sil_man[, 3])
mean_sil_kmeans_man
```

```
## [1] 0.08494206
```

```
pam_sil <- silhouette(kmedians_result$clustering, dist(df_scaled))

mean_sil_pam <- mean(pam_sil[, 3])
mean_sil_pam
```

```
## [1] 0.2137093
```

```r
mask_clusters <- dbscan_res$cluster != 0
df_valid <- subset[mask_clusters, ]
cluster_assign <- dbscan_res$cluster[mask_clusters]

sil <- silhouette(cluster_assign, dist(df_valid))

mean_sil_width <- mean(sil[, 3])
mean_sil_width
```

```
## [1] -0.2200274
```

Se comparan los resultados obtenidos con los modelos k-means, k-medians y DBSCAN.

El mejor parecia k means junto k medians sin embargo, DBSCAN no parecia tener una buena visualización pero sus resultados son mejores ### Se comentan las conclusiones. En este caso elijo DBSCAN identificó 3 clusters principales con un índice de silueta promedio negativo (-0.22), lo que refleja que algunos puntos están mal asignados. OPTICS, por otro lado, mostró una mayor sensibilidad a valores de eps y minPts, sugiriendo la presencia de subestructuras. En comparación con k-means, DBSCAN es más robusto frente al ruido, pero presenta muchas limitaciones debido a la distribución densa y uniforme de los datos ***

Ejercicio 5

Se seleccionan las muestra de entrenamiento y test.

```r
n <- nrow(df)
train_index <- sample(seq_len(n), size = 0.7 * n)

train_data <- df[train_index, ]
train_data <- train_data[, sapply(train_data, function(x) is.numeric(x) || is.factor(x))]
train_data$ingles <- as.factor(train_data$ingles)
test_data  <- df[-train_index, ]
```

Se justifican las proporciones seleccionadas.

Normalmente se hace 70-30 o 80-20, en este caso he hecho 70-30, ya que tengo suficientes datos para hacerlo. ***

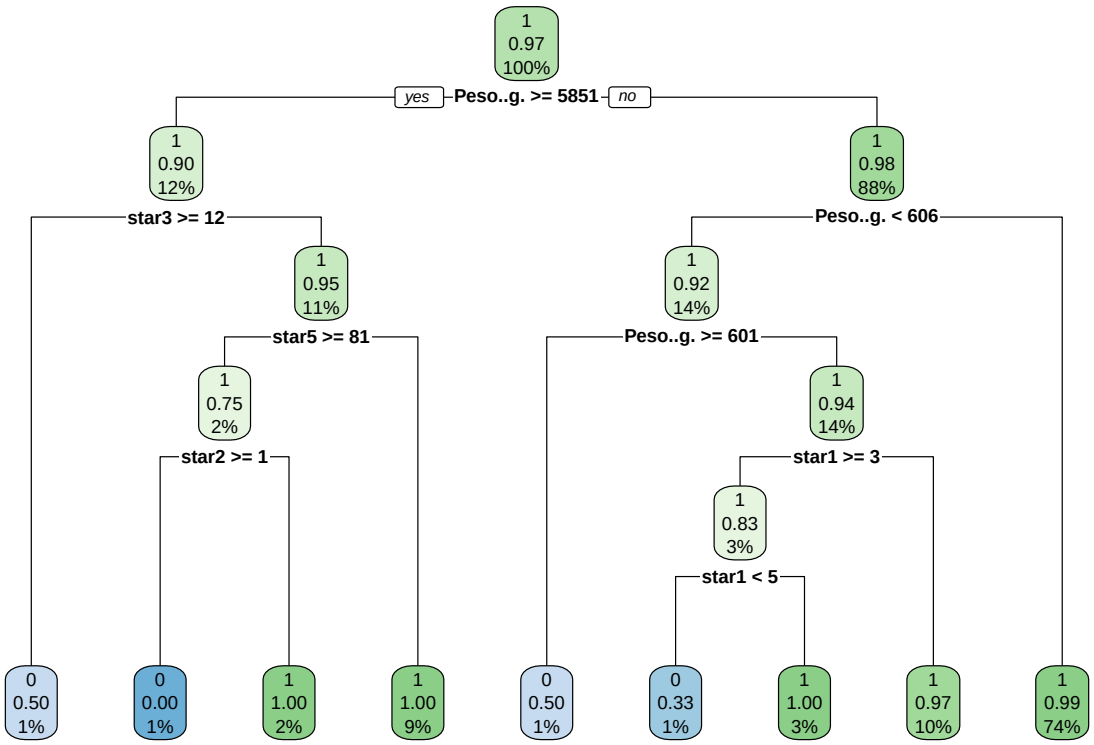
Ejercicio 6

Se generan reglas y se seleccionan e interpretan las más significativas.

```r
library(rpart)
library(rpart.plot)
library(caret)
```

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

```
arbol <- rpart(ingles ~ ., data = train_data,
               method = "class",
               control = rpart.control(minsplit = 5,
                                       cp = 0.0001))
rpart.plot(arbol)
```




```
printcp(arbol)
```

```
##
## Classification tree:
## rpart(formula = ingles ~ ., data = train_data, method = "class",
##     control = rpart.control(minsplit = 5, cp = 1e-04))
##
## Variables actually used in tree construction:
## [1] Peso..g. star1    star2    star3    star5
##
## Root node error: 11/351 = 0.031339
##
## n= 351
##
##         CP nsplit rel error xerror    xstd
## 1 0.045455      0   1.00000 1.0000 0.29675
## 2 0.022727      4   0.81818 1.6364 0.37567
## 3 0.000100      8   0.72727 1.8182 0.39481
```

```r
summary(arbol)
```

```
## Call:
## rpart(formula = ingles ~ ., data = train_data, method = "class",
##     control = rpart.control(minsplit = 5, cp = 1e-04))
##   n= 351
##
##           CP nsplit rel error   xerror      xstd
## 1 0.04545455      0 1.0000000 1.000000 0.2967492
## 2 0.02272727      4 0.8181818 1.636364 0.3756749
## 3 0.00010000      8 0.7272727 1.818182 0.3948051
##
## Variable importance
##    star2    star3    star5    star1    star4 Peso..g.
##       24       20       19       14       13       10
##
## Node number 1: 351 observations,    complexity param=0.04545455
##   predicted class=1  expected loss=0.03133903  P(node) =1
##     class counts:    11   340
##    probabilities: 0.031 0.969
##   left son=2 (42 obs) right son=3 (309 obs)
##   Primary splits:
##       Peso..g.  < 5851.337 to the right, improve=0.3895982, (0 missing)
##       star4     < 19.5     to the left,  improve=0.2306934, (0 missing)
##       star5     < 80.5     to the right, improve=0.2234126, (0 missing)
##       star2     < 12.5     to the right, improve=0.1776509, (0 missing)
##       n_reviews < 54.7     to the right, improve=0.1668389, (0 missing)
##   Surrogate splits:
##       star4 < 46.5     to the right, agree=0.892, adj=0.095, (0 split)
##
## Node number 2: 42 observations,    complexity param=0.04545455
##   predicted class=1  expected loss=0.0952381  P(node) =0.1196581
##     class counts:     4    38
##    probabilities: 0.095 0.905
##   left son=4 (4 obs) right son=5 (38 obs)
##   Primary splits:
##       star3       < 12       to the right, improve=1.4486220, (0 missing)
##       star4       < 14.5     to the left,  improve=1.2380950, (0 missing)
##       avg_reviews < 4        to the left,  improve=0.6880952, (0 missing)
##       star2       < 8        to the right, improve=0.6880952, (0 missing)
##       star1       < 1.5      to the left,  improve=0.6293996, (0 missing)
##
## Node number 3: 309 observations,    complexity param=0.02272727
##   predicted class=1  expected loss=0.02265372  P(node) =0.8803419
##     class counts:     7   302
##    probabilities: 0.023 0.977
##   left son=6 (50 obs) right son=7 (259 obs)
##   Primary splits:
##       Peso..g.  < 605.5453 to the left,  improve=0.39234600, (0 missing)
##       star4     < 9.5      to the left,  improve=0.16617210, (0 missing)
##       star5     < 66.5     to the right, improve=0.13238920, (0 missing)
##       star3     < 4.5      to the left,  improve=0.11767180, (0 missing)
##       n_reviews < 97       to the right, improve=0.08090615, (0 missing)
```

```
##
## Node number 4: 4 observations
##   predicted class=0  expected loss=0.5  P(node) =0.01139601
##     class counts:     2     2
##    probabilities: 0.500 0.500
##
## Node number 5: 38 observations,    complexity param=0.04545455
##   predicted class=1  expected loss=0.05263158  P(node) =0.1082621
##     class counts:     2    36
##    probabilities: 0.053 0.947
##   left son=10 (8 obs) right son=11 (30 obs)
##   Primary splits:
##       star5 < 80.5     to the right, improve=0.7894737, (0 missing)
##       price < 8.84     to the left,  improve=0.5132832, (0 missing)
##       star4 < 13.5     to the left,  improve=0.4048583, (0 missing)
##       star3 < 4.5      to the left,  improve=0.2339181, (0 missing)
##       star2 < 1.5      to the left,  improve=0.2339181, (0 missing)
##   Surrogate splits:
##       star4 < 9.5      to the left,  agree=0.947, adj=0.750, (0 split)
##       star3 < 1.5      to the left,  agree=0.842, adj=0.250, (0 split)
##       star2 < 0.5      to the left,  agree=0.816, adj=0.125, (0 split)
##
## Node number 6: 50 observations,    complexity param=0.02272727
##   predicted class=1  expected loss=0.08  P(node) =0.1424501
##     class counts:     4    46
##    probabilities: 0.080 0.920
##   left son=12 (2 obs) right son=13 (48 obs)
##   Primary splits:
##       Peso..g.    < 601.0094 to the right, improve=0.7350000, (0 missing)
##       price       < 63.97    to the right, improve=0.4096454, (0 missing)
##       avg_reviews < 4.95     to the right, improve=0.4096454, (0 missing)
##       star5       < 66.5     to the right, improve=0.2742857, (0 missing)
##       n_reviews   < 159.5    to the left,  improve=0.2488889, (0 missing)
##
## Node number 7: 259 observations
##   predicted class=1  expected loss=0.01158301  P(node) =0.7378917
##     class counts:     3   256
##    probabilities: 0.012 0.988
##
## Node number 10: 8 observations,    complexity param=0.04545455
##   predicted class=1  expected loss=0.25  P(node) =0.02279202
##     class counts:     2     6
##    probabilities: 0.250 0.750
##   left son=20 (2 obs) right son=21 (6 obs)
##   Primary splits:
##       star2       < 0.5      to the right, improve=3.000000, (0 missing)
##       star5       < 90.5     to the left,  improve=1.666667, (0 missing)
##       star4       < 7.5      to the right, improve=1.666667, (0 missing)
##       star3       < 0.5      to the right, improve=1.666667, (0 missing)
##       avg_reviews < 4.7      to the left,  improve=0.600000, (0 missing)
##   Surrogate splits:
##       star5 < 90.5     to the left,  agree=0.875, adj=0.5, (0 split)
##       star4 < 7.5      to the right, agree=0.875, adj=0.5, (0 split)
##       star3 < 0.5      to the right, agree=0.875, adj=0.5, (0 split)
```

```
##
## Node number 11: 30 observations
##   predicted class=1  expected loss=0  P(node) =0.08547009
##     class counts:     0    30
##    probabilities: 0.000 1.000
##
## Node number 12: 2 observations
##   predicted class=0  expected loss=0.5  P(node) =0.005698006
##     class counts:     1     1
##    probabilities: 0.500 0.500
##
## Node number 13: 48 observations,    complexity param=0.02272727
##   predicted class=1  expected loss=0.0625  P(node) =0.1367521
##     class counts:     3    45
##    probabilities: 0.063 0.938
##   left son=26 (12 obs) right son=27 (36 obs)
##   Primary splits:
##       star1       < 2.5      to the right, improve=0.3472222, (0 missing)
##       avg_reviews < 4.55     to the left,  improve=0.3450000, (0 missing)
##       n_reviews   < 50.5     to the right, improve=0.3450000, (0 missing)
##       star3       < 3.5      to the right, improve=0.2678571, (0 missing)
##       star2       < 0.5      to the right, improve=0.2456897, (0 missing)
##   Surrogate splits:
##       star5    < 57       to the left,  agree=0.833, adj=0.333, (0 split)
##       star2    < 3.5      to the right, agree=0.833, adj=0.333, (0 split)
##       star3    < 5.5      to the right, agree=0.812, adj=0.250, (0 split)
##       Peso..g. < 462.6638 to the left,  agree=0.792, adj=0.167, (0 split)
##
## Node number 20: 2 observations
##   predicted class=0  expected loss=0  P(node) =0.005698006
##     class counts:     2     0
##    probabilities: 1.000 0.000
##
## Node number 21: 6 observations
##   predicted class=1  expected loss=0  P(node) =0.01709402
##     class counts:     0     6
##    probabilities: 0.000 1.000
##
## Node number 26: 12 observations,    complexity param=0.02272727
##   predicted class=1  expected loss=0.1666667  P(node) =0.03418803
##     class counts:     2    10
##    probabilities: 0.167 0.833
##   left son=52 (3 obs) right son=53 (9 obs)
##   Primary splits:
##       star1     < 4.5      to the left,  improve=2.0000000, (0 missing)
##       star5     < 66.5     to the right, improve=1.3333330, (0 missing)
##       star2     < 5.5      to the left,  improve=0.9333333, (0 missing)
##       n_reviews < 46       to the right, improve=0.6666667, (0 missing)
##       Peso..g.  < 510.291  to the right, improve=0.6666667, (0 missing)
##   Surrogate splits:
##       star5 < 64       to the right, agree=0.833, adj=0.333, (0 split)
##       star2 < 5.5      to the left,  agree=0.833, adj=0.333, (0 split)
##
## Node number 27: 36 observations
```

```
##   predicted class=1  expected loss=0.02777778  P(node) =0.1025641
##     class counts:     1    35
##    probabilities: 0.028 0.972
##
## Node number 52: 3 observations
##   predicted class=0  expected loss=0.3333333  P(node) =0.008547009
##     class counts:     2     1
##    probabilities: 0.667 0.333
##
## Node number 53: 9 observations
##   predicted class=1  expected loss=0  P(node) =0.02564103
##     class counts:     0     9
##    probabilities: 0.000 1.000
```

```
mejor_cp <- arbol$cptable[which.min(arbol$cptable[,"xerror"]), "CP"]
mejor_cp
```

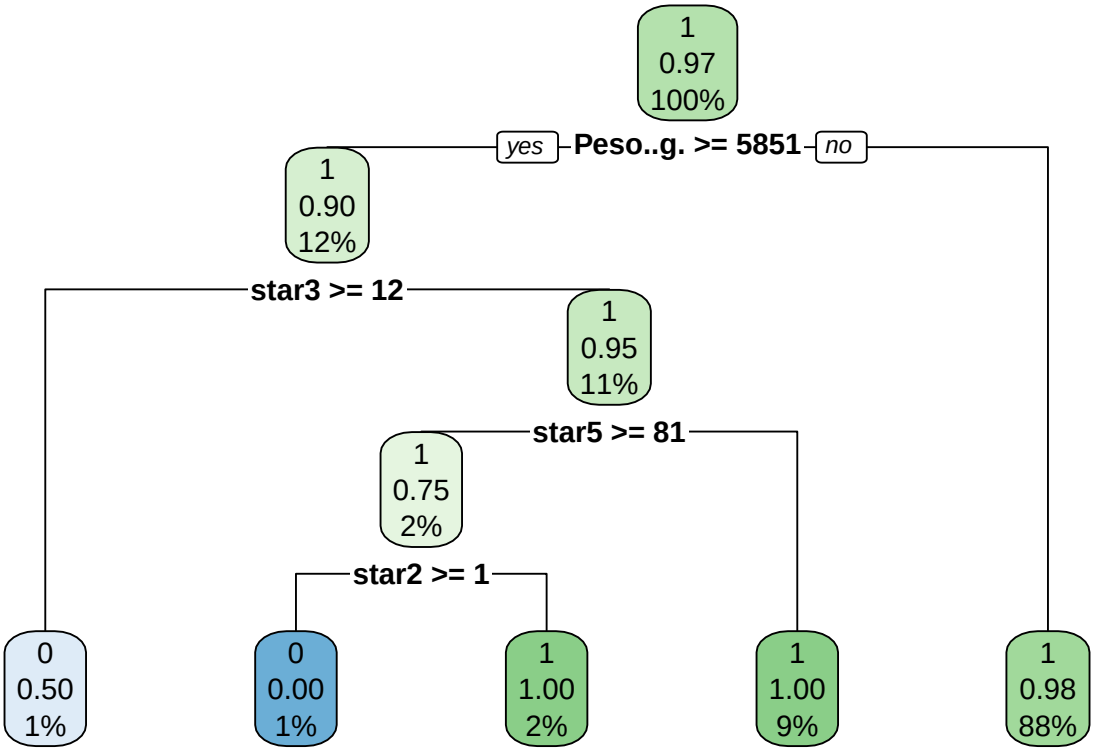
```
## [1] 0.04545455
```

Se extraen las reglas del modelo en formato texto y gráfico.

```
library(rpart)
library(rpart.plot)
arbol_podado <- prune(arbol, cp = 0.0308)
arbol_podado
```

```
## n= 351
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 351 11 1 (0.03133903 0.96866097)
##    2) Peso..g.>=5851.337 42  4 1 (0.09523810 0.90476190)
##      4) star3>=12 4  2 0 (0.50000000 0.50000000) *
##      5) star3< 12 38  2 1 (0.05263158 0.94736842)
##       10) star5>=80.5 8  2 1 (0.25000000 0.75000000)
##         20) star2>=0.5 2  0 0 (1.00000000 0.00000000) *
##         21) star2< 0.5 6  0 1 (0.00000000 1.00000000) *
##       11) star5< 80.5 30  0 1 (0.00000000 1.00000000) *
##    3) Peso..g.< 5851.337 309  7 1 (0.02265372 0.97734628) *
```

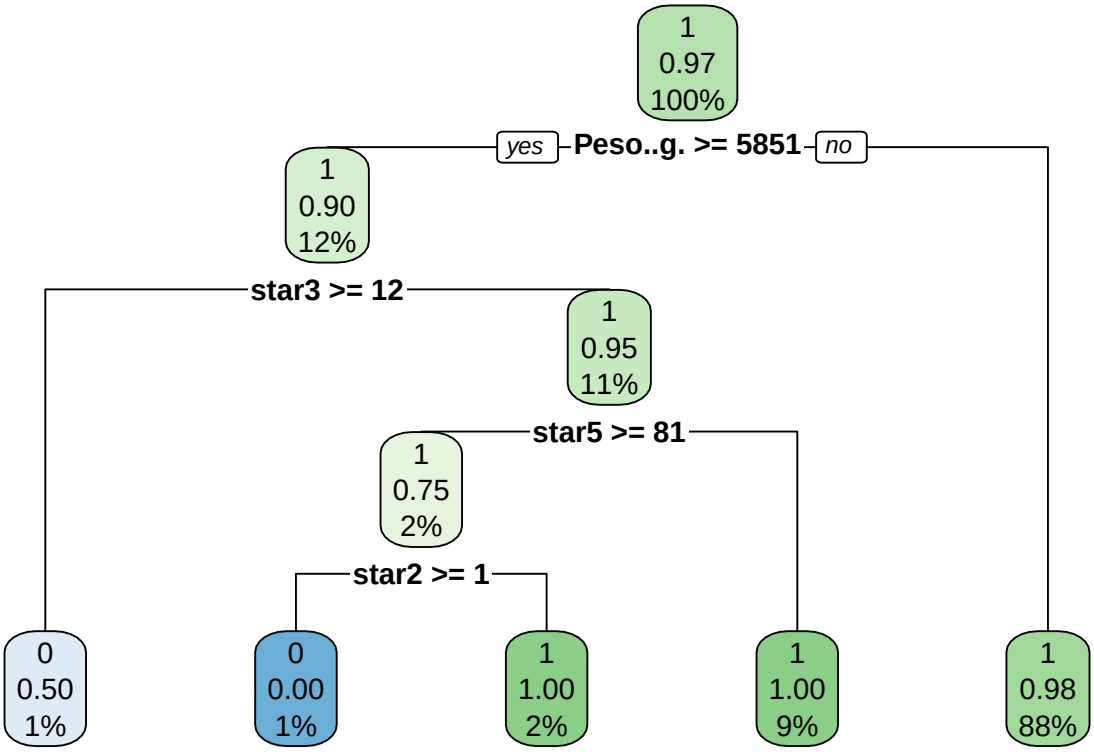
```
rpart.plot(arbol_podado)
```




```
arbol_podado <- prune(arbol, cp = 0.0307)
arbol_podado
```

```
## n= 351
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 351 11 1 (0.03133903 0.96866097)
##    2) Peso..g.>=5851.337 42  4 1 (0.09523810 0.90476190)
##      4) star3>=12 4  2 0 (0.50000000 0.50000000) *
##      5) star3< 12 38  2 1 (0.05263158 0.94736842)
##       10) star5>=80.5 8  2 1 (0.25000000 0.75000000)
##         20) star2>=0.5 2  0 0 (1.00000000 0.00000000) *
##         21) star2< 0.5 6  0 1 (0.00000000 1.00000000) *
##       11) star5< 80.5 30  0 1 (0.00000000 1.00000000) *
##    3) Peso..g.< 5851.337 309  7 1 (0.02265372 0.97734628) *
```

```
rpart.plot(arbol_podado)
```



Adicionalmente, se genera la matriz de confusión para medir la capacidad predictiva del algoritmo, teniendo en cuenta las distintas métricas asociadas a dicha matriz (precisión, sensibilidad, especificidad...). Alternativamente, si la variable objeto de estudio es cuantitativa pura se obtienen los criterios de error que nos permitan determinar la capacidad predictiva.

```
library(caret)
test_data$ingles <- factor(test_data$ingles)

pred <- predict(arbol, newdata = test_data, type = "class")

matriz_conf <- table(Prediccion = pred, Real = test_data$ingles)
matriz_conf
```

```
##           Real
## Prediccion   0   1
##          0   0   6
##          1   4 141
```

```
exactitud <- mean(pred == test_data$ingles)
exactitud
```

```
## [1] 0.9337748
```

```
confusionMatrix(data = pred, reference = test_data$ingles)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   0   6
##          1   4 141
##
##                Accuracy : 0.9338
##                  95% CI : (0.8816, 0.9678)
##     No Information Rate : 0.9735
##     P-Value [Acc > NIR] : 0.9976
##
##                   Kappa : -0.0328
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.00000
##             Specificity : 0.95918
##          Pos Pred Value : 0.00000
##          Neg Pred Value : 0.97241
##              Prevalence : 0.02649
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.03974
##       Balanced Accuracy : 0.47959
##
##        'Positive' Class : 0
##
```

```
pred <- predict(arbol_podado, newdata = test_data, type = "class")

matriz_conf <- table(Prediccion = pred, Real = test_data$ingles)
matriz_conf
```

```
##           Real
## Prediccion   0   1
##          0   0   1
##          1   4 146
```

```
exactitud <- mean(pred == test_data$ingles)
exactitud
```

```
## [1] 0.9668874
```

```
confusionMatrix(data = pred, reference = test_data$ingles)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
```

```
##          0   0   1
##          1   4 146
##
##                Accuracy : 0.9669
##                  95% CI : (0.9244, 0.9892)
##     No Information Rate : 0.9735
##     P-Value [Acc > NIR] : 0.7873
##
##                   Kappa : -0.0107
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.000000
##             Specificity : 0.993197
##          Pos Pred Value : 0.000000
##          Neg Pred Value : 0.973333
##              Prevalence : 0.026490
##          Detection Rate : 0.000000
##    Detection Prevalence : 0.006623
##       Balanced Accuracy : 0.496599
##
##        'Positive' Class : 0
##
```

Se comparan e interpretan los resultados (sin y con opciones de poda), explicando las ventajas e inconvenientes del modelo generado respecto a otro método de construcción.

Sin poda, el arbo tiende a sobreajustar los datos de entrenamiento, resultando en un menor de entrenamiento pero mayor error en el conjunto de test. Con poda el arbol se simplifica, y reduce el sobreajuste, y generalmente, puede mejorar la precision en el test. ### Se evalúa la tasa de error en cada nivel de árbol, la eficiencia en la clasificación (en las muestras de entrenamiento y test) y la comprensibilidad del resultado. La poda reduce la tasa de error en el conjunto del test, y mejora la eficiencia en la clasificación.

Se comentan las conclusiones.

La poda mejora el resultado en su generalización y reduce la complejidad del modelo, reduciendo el riesgo de sobreajuste.Además, simplifica el árbol, haciéndolo más comprensible y fácil de interpretar. En este caso, la poda no ha mejorado la precisión del modelo, ha tenido la misma precisión que sin poda. ***

Ejercicio 7

Se prueba con una variación u otro enfoque de algoritmo supervisado.

```
library(randomForest)
```

```
## randomForest 4.7-1.2
```

```
## 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
```

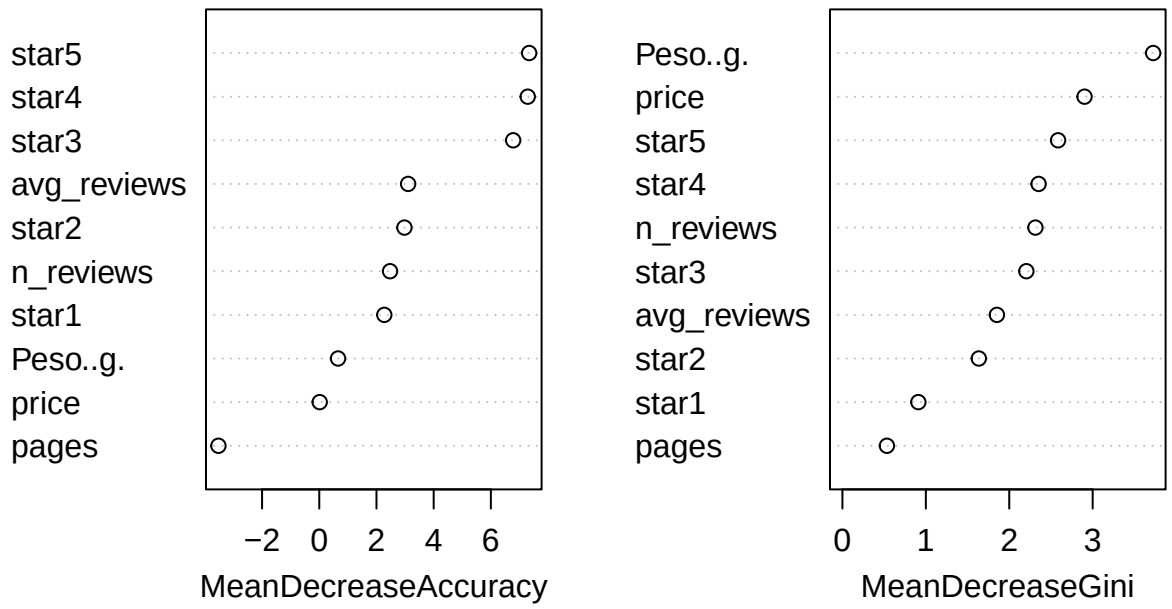
```
modelo_rf <- randomForest(
  ingles ~ ., data = train_data, ntree = 200, mtry = 2, importance = TRUE)

modelo_rf
```

```
##
## Call:
##  randomForest(formula = ingles ~ ., data = train_data, ntree = 200,      mtry = 2, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3.13%
## Confusion matrix:
##   0   1 class.error
## 0 0  11           1
## 1 0 340           0
```

```
importance(modelo_rf)
```

```
##                         0          1 MeanDecreaseAccuracy MeanDecreaseGini
## price       -0.090888970  0.1969783           0.01466902        2.9025783
## pages        0.005344436 -3.2195898          -3.52707181        0.5332945
## avg_reviews -1.916551470  3.4975444           3.10663575        1.8522070
## n_reviews    0.622760470  2.4390668           2.47532737        2.3137045
## star5        0.337521780  7.1391185           7.33953507        2.5856053
## star4       -0.086977243  7.1449280           7.28966527        2.3521238
## star3       -0.697374194  6.9613190           6.77990510        2.2052155
## star2        0.434627538  2.8887167           2.97789725        1.6354520
## star1       -1.072080458  2.3676402           2.27378467        0.9103217
## Peso..g.    -1.270442329  0.8162830           0.65640182        3.7254852
```

```
varImpPlot(modelo_rf)
```




```
modelo_rf
```

```
##
## Call:
##  randomForest(formula = ingles ~ ., data = train_data, ntree = 200,      mtry = 2, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 3.13%
## Confusion matrix:
##   0   1 class.error
## 0 0  11           1
## 1 0 340           0
```

Se detalla, comenta y evalúa la calidad de clasificación o del ajuste.

```
rf_pred <- predict(modelo_rf, newdata = test_data, type = "class")

rf_pred <- factor(rf_pred, levels = c(0, 1))
conf_rf <- confusionMatrix(data = rf_pred, reference = test_data$ingles)
conf_rf
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction   0   1
##          0   0   0
##          1   4 147
##
##                Accuracy : 0.9735
##                  95% CI : (0.9336, 0.9927)
##     No Information Rate : 0.9735
##     P-Value [Acc > NIR] : 0.6288
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.1336
##
##             Sensitivity : 0.00000
##             Specificity : 1.00000
##          Pos Pred Value :     NaN
##          Neg Pred Value : 0.97351
##              Prevalence : 0.02649
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00000
##       Balanced Accuracy : 0.50000
##
##        'Positive' Class : 0
##
```

Se comparan los resultados con el método supervisado del ejercicio 6.

Ejercicio 8

Se identifica qué posibles limitaciones tienen los datos que has seleccionado para obtener conclusiones con los modelos (supervisado y no supervisado)

Me faltaban tanto variables numéricas pero sobretodo variables categóricas, lo que complicó bastante el proceso. Trabajar con variables categóricas puede ser un desafío, especialmente cuando no hay suficiente variedad o cantidad para que el modelo encuentre patrones útiles. En mi caso, no pude realizar una clasificación con varias variables categóricas como habría sido ideal, ya que solo contaba con una, y esto limitó significativamente el análisis. Terminé haciendo una clasificación basada únicamente en una variable categórica que, además, no era muy representativa: se trataba de predecir si el libro estaba en inglés o no.

Esto no es muy útil en términos prácticos, ya que no permite capturar información más compleja sobre el conjunto de datos. Ademas, al tratarse de una sola variable binaria, el modelo tiene pocas posibilidades de aprender patrones enriquecedores, lo que limita tanto su precision como su capacidad para generalizar. Hubiera sido mejor contar con más variables categóricas que reflejaran características adicionales, como el género literario, la región de origen, que podrían complementar el análisis y permitir una clasificación más robusta y precisa.

Por otro lado, transformar variables categóricas en un formato útil para los algoritmos (como mediante técnicas de One-Hot Encoding o Label Encoding) es esencial, pero si solo se cuenta con una categoría binaria, el impacto de estas transformaciones es muy limitado. Esta experiencia me dejó claro que la falta

de diversidad y cantidad de variables en mi dataset puede limitar severamente las capacidades predictivas de los modelos y subraya la importancia de tener un conjunto de datos más completo y equilibrado.

Se identifican posibles riesgos del uso del modelo (mínimo 300 palabras).

El uso de modelos en entornos reales, aunque extremadamente útil, conlleva riesgos importantes que debemos considerar para garantizar su eficacia y equidad. En primer lugar, el sobreajuste es un problema común. Esto ocurre cuando un modelo se ajusta tanto a los datos de entrenamiento que pierde la capacidad de generalizar con datos nuevos. Por ejemplo, un modelo entrenado con datos específicos puede fracasar ante escenarios inesperados si no se trabaja con una muestra variada y representativa.

Otro desafío crítico es el desequilibrio de clases. Cuando los datos están desproporcionadamente representados entre categorías (en mi caso ingles), el modelo tiende a favorecer a la clase dominante. Esto no solo afecta la precisión, sino que puede generar injusticias, especialmente en casos donde hay grupos minoritarios subrepresentados.

Por otro lado, el data drift (o cambio en las distribuciones de los datos) es un riesgo a largo plazo. A medida que las preferencias de los usuarios o las condiciones del entorno cambian, los datos históricos pueden perder relevancia, disminuyendo la precisión de las predicciones. Esto exige una supervisión constante y actualizaciones periódicas del modelo.

También, los valores atípicos y los datos faltantes son problemas recurrentes que, si no se manejan correctamente, afectan la estabilidad y confiabilidad de los resultados. Esto puede requerir imputar valores o eliminar outliers.

En cuanto a las implicaciones éticas, la falta de cuidado al manejar datos sensibles puede derivar en discriminación o violaciones de privacidad. Además, los modelos complejos, como redes neuronales profundas, suelen ser poco interpretables, lo que dificulta explicar sus decisiones en aplicaciones críticas, como la medicina o las finanzas.

Por último, la falta de validación cruzada y de una supervisión adecuada puede inflar artificialmente las métricas de rendimiento, complicando su extrapolación a otros escenarios. Por ello, la combinación de monitoreo continuo, documentación clara y una preparación de datos rigurosa es clave para mitigar estos riesgos y garantizar un desempeño sostenible del modelo en entornos reales.
