Práctica 3: Fundamentos de la Ciencia de Datos

Daniel López Moreno Alejandro Fernández Maceira

Álvaro Maestre Santa

November 5, 2019

1 Análisis de clasificación

1.1 Clasificación en árbol de una muestra

En este apartado vamos a realizar una clasificación en árbol de los datos de una muestra sobre calificaciones de alumnos utilizando para ello el algoritmo de Hunt con ganancia de información mediante la medida de impureza Gini.
En primer lugar, debemos de tener la librería **rpart**, si no la tenemos, tendremos que proceder a instalarla. También añadiremos la libreria **tree** para mostrar la clasificación de otra forma. Además, añadiremos la librería **rpart.plot** para visualizar el árbol de clasificación resultante.
El código para instalar las librerías es el siguiente.

```
> install.packages("rpart")
> install.packages("tree")
> install.packages("rpart.plot")
> library(rpart)
> library(tree)
> library(rpart.plot)
```

Los datos que se van a utilizar en la clasificación vienen dados en un fichero .txt llamado **calificaciones.txt**.

```
> calificaciones<-read.table("calificaciones.txt")
> calificaciones
```

```
  Teor Lab Prac Calif_Global
1    A   A    B           Ap
2    A   B    D           Ss
3    D   D    C           Ss
4    D   D    A           Ss
5    B   C    B           Ss
6    C   B    D           Ap
7    B   B    A           Ap
8    C   D    C           Ss
9    B   A    C           Ss
```

Una vez cargados los datos, antes de empezar a trabajar con ellos, lo primero que tenemos que hacer es generar una muestra, esto se realiza con la función **data.frame**.

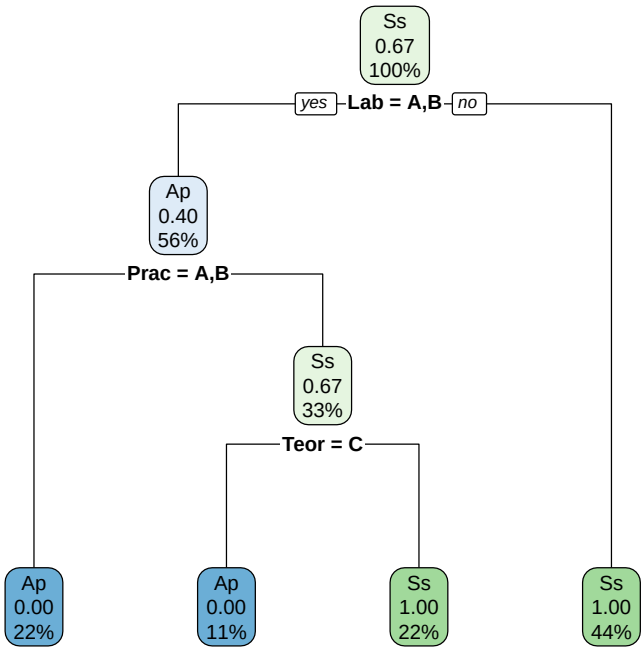
```
> muestraCalificaciones<-data.frame(calificaciones)
```

Ahora que tenemos la muestra ya podemos trabajar con los datos. Para poder clasificarlos, que es nuestro objetivo, hacemos uso de la función **rpart** del paquete **rpart** instalado previamente.

```
> clasificacion = rpart(Calif_Global~., data=muestraCalificaciones, method='class',
+ minsplit=1)
> clasificacion

n= 9

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 9 3 Ss (0.3333333 0.6666667)
   2) Lab=A,B 5 2 Ap (0.6000000 0.4000000)
     4) Prac=A,B 2 0 Ap (1.0000000 0.0000000) *
     5) Prac=C,D 3 1 Ss (0.3333333 0.6666667)
      10) Teor=C 1 0 Ap (1.0000000 0.0000000) *
      11) Teor=A,B 2 0 Ss (0.0000000 1.0000000) *
   3) Lab=C,D 4 0 Ss (0.0000000 1.0000000) *
```

Mostraremos el resultado de manera visual:

```
> rpart.plot(clasificacion)
```



En este comando, los distintos parámetros son los siguientes.

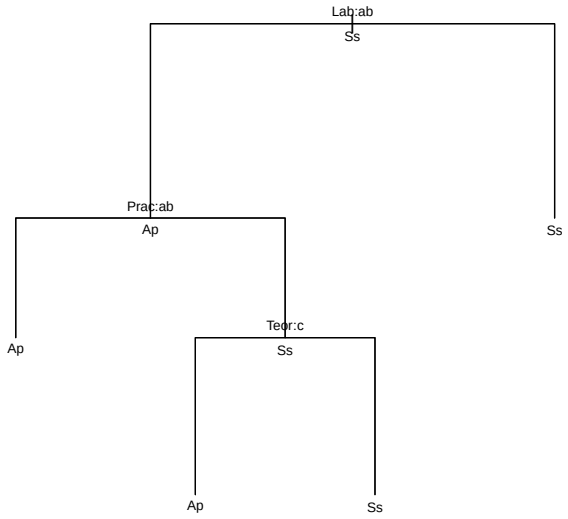
- **Calif-Global**:Selecciona cuál será la columna sobre la que clasificar los datos.
- **data=muestraCalificaciones**:Selecciona la muestra sobre la que clasificar.
- **method=class**:El método para realizar la partición, como es un problema de clasificación, se utiliza class.
- **minsplit=1**:Determina el número mínimo de observaciones por rama

No es necesario especificar que la impureza se calcule mediante **Gini**, ya que la función **rpart utiliza Gini por defecto**.
Una vez vista la clasificación con la función **rpart**, se prueba la clasificación de los mismos datos pero con la función **tree**, previamente instalado.

```
> clasificacionTree = tree(Calif_Global~., data=muestraCalificaciones, split="gini",
+ mincut=1, minsize=2)
> clasificacionTree

node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 9 11.460 Ss ( 0.3333 0.6667 )
   2) Lab: A,B 5  6.730 Ap ( 0.6000 0.4000 )
     4) Prac: A,B 2  0.000 Ap ( 1.0000 0.0000 ) *
     5) Prac: C,D 3  3.819 Ss ( 0.3333 0.6667 )
      10) Teor: C 1  0.000 Ap ( 1.0000 0.0000 ) *
      11) Teor: A,B 2  0.000 Ss ( 0.0000 1.0000 ) *
   3) Lab: C,D 4  0.000 Ss ( 0.0000 1.0000 ) *
```

Mostraremos el arbol:

```
> plot(clasificacionTree, main="Árbol de clasificación")
> text(clasificacionTree, splits=TRUE, all=TRUE, cex=.7)
```



En este comando, los distintos parámetros significan lo siguiente.

- **Calif-Global**:Selecciona cuál será la columna sobre la que clasificar los datos.
- **data=muestraCalificaciones**:Selecciona la muestra sobre la que clasificar.
- **split="gini"**: Selecciona Gini como método de cálculo de la impureza para clasificar.
- **mincut=1**:El método para realizar la partición,como es un problema de clasificación, se utiliza class.
- **minsize=2**:Determina el número mínimo de observaciones por rama

Como podemos observar tras realizar la clasificación utilizando ambas funciones, el resultado y el arbol mostrado es el mismo, necesitando en ambos casos las 3 columnas (Lab, Prac y Teor) para clasificar completamente la muestra. Sin embargo en los 2 casos utilizando el árbol de clasificación algunas veces somos capaces de averiguar la **Calif-Global** utilizando solamente la calificación de **Lab** o de **Lab y Prac**.

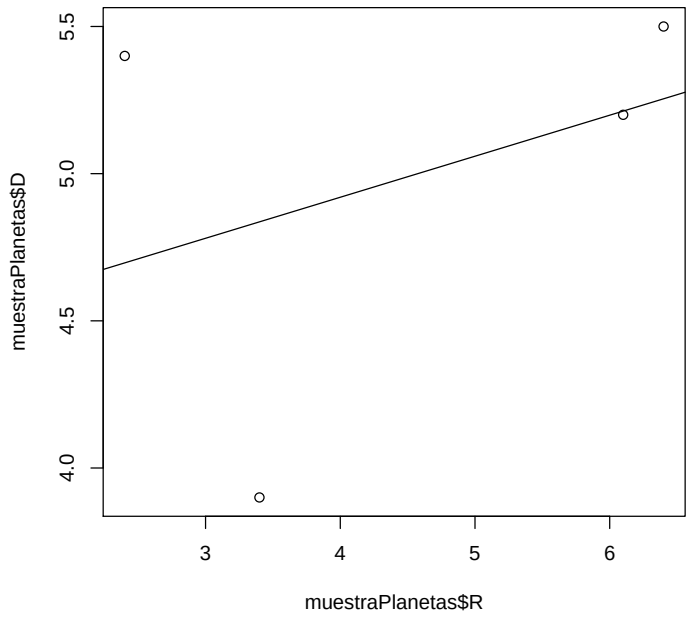
1.2 Función de regresión

En este apartado vamos a realizar una función de regresión sobre una muestra de datos que contiene datos del radio de varios planetas y su densidad. Lo primero que hay que hacer para poder hacer la función de regresión es cargar el documento con los datos de los planetas:

```
> muestraPlanetas = read.table("planetas.txt")
```

Después, simplemente hay que llamar a la función **lm**, la cual generará los coeficientes de la recta de regresión.

```
> regresion=lm(D~R, data=muestraPlanetas)
> summary(regresion)

Call:
lm(formula = D ~ R, data = muestraPlanetas)

Residuals:
Mercurio    Venus   Tierra    Marte
 0.70312 -0.01253  0.24566 -0.93624

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.3624     1.2050   3.620   0.0685 .
R             0.1394     0.2466   0.565   0.6289
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.846 on 2 degrees of freedom
Multiple R-squared:  0.1377,        Adjusted R-squared:  -0.2935
F-statistic: 0.3193 on 1 and 2 DF,  p-value: 0.6289

> plot(muestraPlanetas$R, muestraPlanetas$D)
> abline(regresion)
```



Donde **D R** indica cuál será la y(D) y la x(R) en la recta, y **data=muestra** indica los datos sobre los que hacer la regresión.

2 Analisis de archivo .txt, regresión y desarrollo por parte del grupo

2.1 Clasificación del tipo de vehículo

En este apartado del ejercicio vamos a realizar la clasificación de una muestra que contiene datos sobre distintos vehículos. Para realizar la clasificación serán necesarios los paquetes **rpart** y **tree**, pero puesto que ya los hemos instalado en el ejercicio anterior, no será necesario instalarlos de nuevo.
El clasificador será el la columna **TipoVehiculo**, y para clasificarlo utilizaremos las otras 3 columnas incluidas en nuestro archivo .txt llamadas **TipoCarnet**, **NumeroRuedas** y **NumeroPasajeros**.
Lo primero que haremos será leer el archivo "vehiculos.txt" y almacenarlo en la variable vehiculos:

```
> vehiculos<-read.table("vehiculos.txt")
> vehiculos

   TipoCarnet NumeroRuedas NumeroPasajeros TipoVehiculo
1           B            4               5        Coche
2           A            2               2         Moto
3           N            2               1    Bicicleta
4           B            6               4       Camion
5           B            4               6        Coche
6           B            4               4        Coche
7           N            2               2    Bicicleta
8           B            2               1         Moto
9           B            6               2       Camion
10          N            2               1    Bicicleta
```

Ahora, convertiremos nuestra muestra en un data.frame para poder trabajar con ella de manera adecuada:

```
> muestraVehiculos<- data.frame(vehiculos)
```

Una vez ya tenemos nuestra muestra de los vehiculos lista, ya podemos comenzar a clasificarlos utilizando la función **rpart** que ya hemos utilizado en el ejercicio anterior con las calificaciones.

```
> clasificacionVehiculos = rpart(TipoVehiculo~., data=muestraVehiculos, method='class',
+ minsplit=1)
> clasificacionVehiculos

n= 10

node), split, n, loss, yval, (yprob)
      * denotes terminal node
```

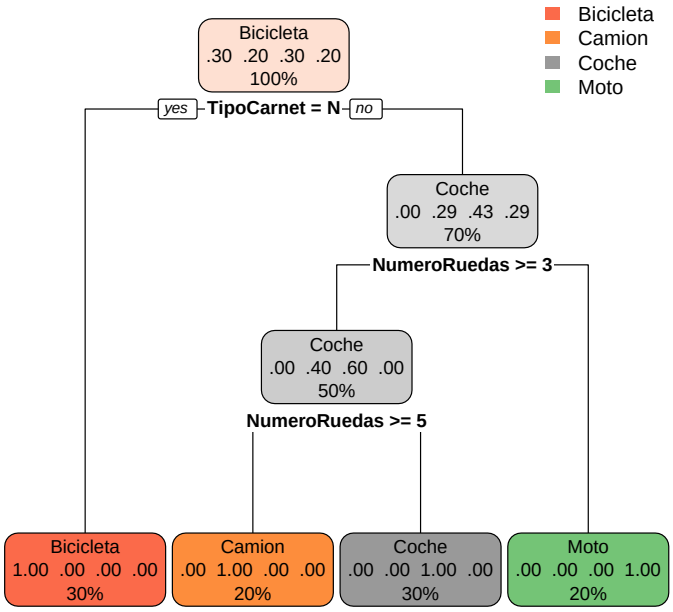
```
 1) root 10 7 Bicicleta (0.3000000 0.2000000 0.3000000 0.2000000)
   2) TipoCarnet=N 3 0 Bicicleta (1.0000000 0.0000000 0.0000000 0.0000000) *
   3) TipoCarnet=A,B 7 4 Coche (0.0000000 0.2857143 0.4285714 0.2857143)
     6) NumeroRuedas>=3 5 2 Coche (0.0000000 0.4000000 0.6000000 0.0000000)
      12) NumeroRuedas>=5 2 0 Camion (0.0000000 1.0000000 0.0000000 0.0000000) *
      13) NumeroRuedas< 5 3 0 Coche (0.0000000 0.0000000 1.0000000 0.0000000) *
     7) NumeroRuedas< 3 2 0 Moto (0.0000000 0.0000000 0.0000000 1.0000000) *
```

Como ya hemos explicado en el ejercicio 1.1, el primer parámetro de entrada de la función se utiliza para determinar la columna sobre la que se va a hacer la clasificación y el resto de parámetros determinan la muestra sobre la que trabajar, el tipo de partición y el mínimo de observaciones por rama.
Una vez hemos realizado la clasificación utilizando la función **rpart** vamos a comprobar si el resultado que obtenemos utilizando la función **tree** es similar:

```
> clasificacionVehiTree = tree(TipoVehiculo~., data=muestraVehiculos, split="gini",
+ mincut=1, minsize=2)
> clasificacionVehiTree

node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 10 27.320 Bicicleta ( 0.3000 0.2000 0.3000 0.2000 )
   2) NumeroPasajeros < 1.5 3  3.819 Bicicleta ( 0.6667 0.0000 0.0000 0.3333 )
     4) TipoCarnet: B 1  0.000 Moto ( 0.0000 0.0000 0.0000 1.0000 ) *
     5) TipoCarnet: N 2  0.000 Bicicleta ( 1.0000 0.0000 0.0000 0.0000 ) *
   3) NumeroPasajeros > 1.5 7 17.880 Coche ( 0.1429 0.2857 0.4286 0.1429 )
     6) NumeroPasajeros < 3 3  6.592 Bicicleta ( 0.3333 0.3333 0.0000 0.3333 )
      12) TipoCarnet: A 1  0.000 Moto ( 0.0000 0.0000 0.0000 1.0000 ) *
      13) TipoCarnet: B,N 2  2.773 Bicicleta ( 0.5000 0.5000 0.0000 0.0000 )
        26) TipoCarnet: B 1  0.000 Camion ( 0.0000 1.0000 0.0000 0.0000 ) *
        27) TipoCarnet: N 1  0.000 Bicicleta ( 1.0000 0.0000 0.0000 0.0000 ) *
     7) NumeroPasajeros > 3 4  4.499 Coche ( 0.0000 0.2500 0.7500 0.0000 )
      14) NumeroRuedas < 5 3  0.000 Coche ( 0.0000 0.0000 1.0000 0.0000 ) *
      15) NumeroRuedas > 5 1  0.000 Camion ( 0.0000 1.0000 0.0000 0.0000 ) *
```

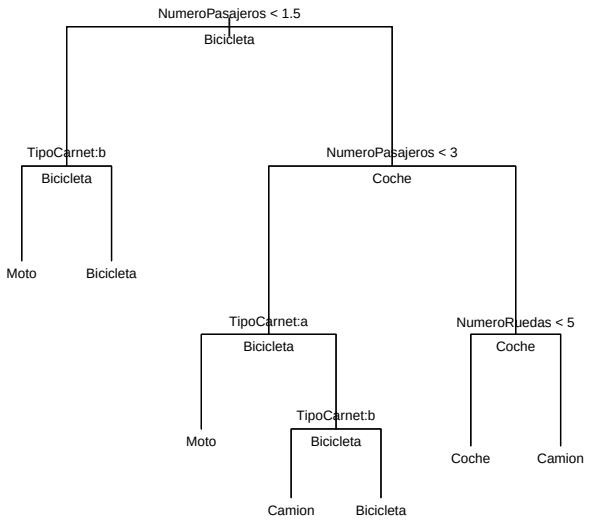
Mostramos los árboles resultantes:
Árbol de rpart:

```
> rpart.plot(clasificacionVehiculos)
```



Árbol de tree:

```
> plot(clasificacionVehiTree, main="Árbol de clasificación")
> text(clasificacionVehiTree, splits=TRUE, all=TRUE, cex=.7)
```



Esta vez, como se puede apreciar en los árboles mostrados, los resultados de rpart y tree son distintos.
En el caso de **rpart** escoge como nodo inicial el TipoCarnet, descartando así las Bicicletas. Despues utiliza el NumeroRuedas para descartar las Motos. Y por último, utiliza otra vez el NumeroRuedas para separar entre Coche y Camión.
En el caso de **tree** comienza utilizando **NumeroPasajeros** para separar aquellos vehiculos que tienen **menos de 1.5 pasajeros** y aquellos que **más de 1.5**. En el siguiente nivel, entre los que tenían menos de 1.5 pasajeros, utiliza el **TipoCarnet** para separar en Bicicleta o Moto. Mientras, en el otro hijo utiliza de nuevo el **NumeroPasajeros** para separar aquellos con **menos de 3 pasajeros** y aquellos con **más de 3 pasajeros**. Entre aquellos que tienen menos de 3, utiliza el **TipoCarnet** para clasificar la Moto y en el siguiente nivel vuelve a utilizar **TipoCarnet** para separar entre Bicileta y Camión. Por último, entre aquellos que tenían más de 3 pasajeros, utiliza el **NumeroRuedas** para separar Coche y Camión.

2.2 Análisis de regresión

En este apartado vamos a realizar la función de regresión de nuestra muestra de datos incluida en el enunciado, formada por 4 pares de datos. Como cada uno de los pares de datos es independiente del resto, pero queremos leerlos todos del mismo archivo .txt, procederemos a leer la muestra y despues dividiremos el data.frame en 4 partes de igual tamaño.Esta muestra estará guardada en el archivo **"muestra2_2.txt"**.

Nota importante: Al generar el archivo "muestra2_2.txt" podriamos haber generado un archivo con 44 filas(11 por muestra) y dos columnas X-Y y posteri-

ormente haber dividido en 4 muestras utilizando la función **"split(muestraDatos, factor(sort(rank(row.names(muestraDatos))%%4)))"**. Sin embargo, hemos optado por generar el archivo .txt como un archivo de 11 filas y 8 columnas numeradas (Un par X-Y por cada muestra) con el fin de facilitar la visibilidad y la comprensión, ya que tras realizar numerosas pruebas, hemos comprobado que el resultado es idéntico.
Comenzaremos por leer el archivo:

```
> muestraDatos <- read.table("muestra2_2.txt")
> muestraDatos
```

```
   X1    Y1 X2   Y2 X3    Y3 X4    Y4
1  10  8.04 10 9.14 10  7.46  8  6.58
2   8  6.95  8 8.14  8  6.77  8  5.76
3  13  7.58 13 8.74 13 12.74  8  7.71
4   9  8.81  9 8.77  9  7.11  8  8.84
5  11  8.33 11 9.26 11  7.81  8  8.47
6  14  9.96 14 8.10 14  8.84  8  7.04
7   6  7.24  6 6.13  6  6.08  8  5.25
8   4  4.26  4 3.10  4  5.39 19 12.50
9  12 10.84 12 9.13 12  8.15  8  5.56
10  7  4.82  7 7.26  7  6.42  8  7.91
11  5  5.68  5 4.74  5  5.73  8  6.89
```

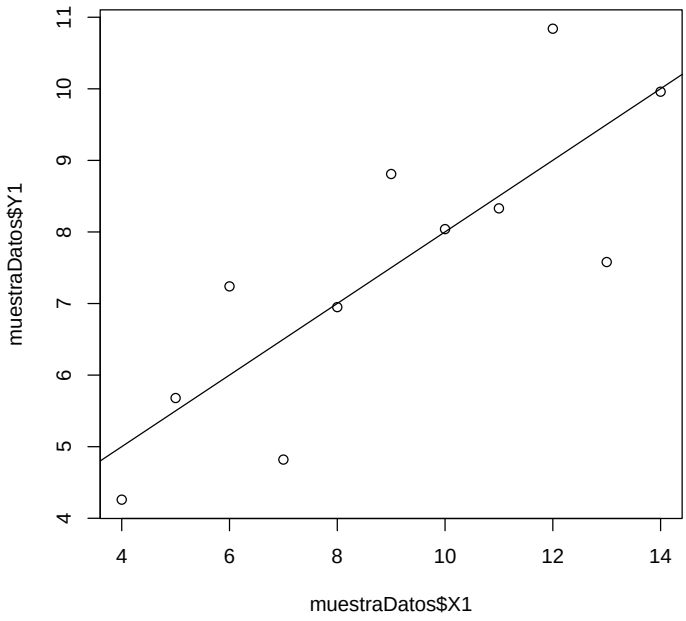
Una vez hemos leido los datos podemos calcular la regresión de cada una de las muestras utilizando sus respectivas columnas X-Y:

Regresión de la Muestra 1:

```
> regresion1=lm(Y1~X1, data=muestraDatos)
> summary(regresion1)
```

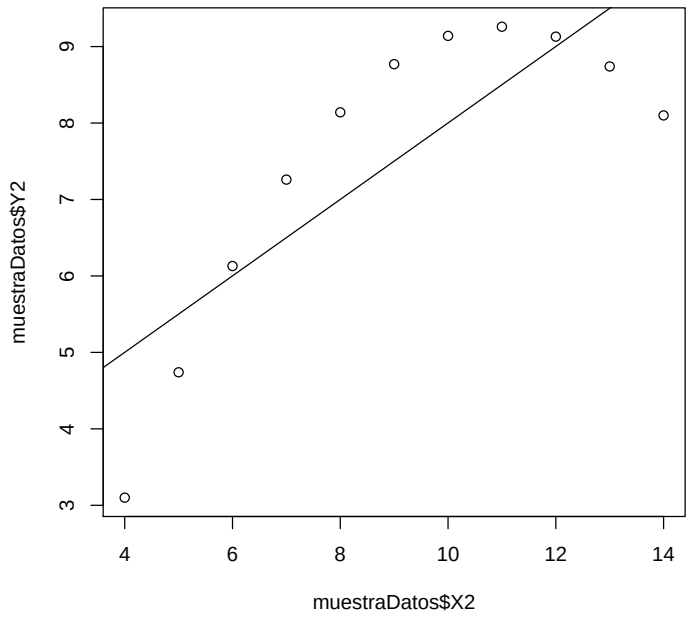
```
Call:
lm(formula = Y1 ~ X1, data = muestraDatos)

Residuals:
     Min       1Q   Median       3Q      Max
-1.92127 -0.45577 -0.04136  0.70941  1.83882

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0001     1.1247   2.667  0.02573 *
X1            0.5001     0.1179   4.241  0.00217 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.237 on 9 degrees of freedom
Multiple R-squared:  0.6665,        Adjusted R-squared:  0.6295
F-statistic: 17.99 on 1 and 9 DF,  p-value: 0.00217
```

```
> plot(muestraDatos$X1, muestraDatos$Y1)
> abline(regresion1)
```



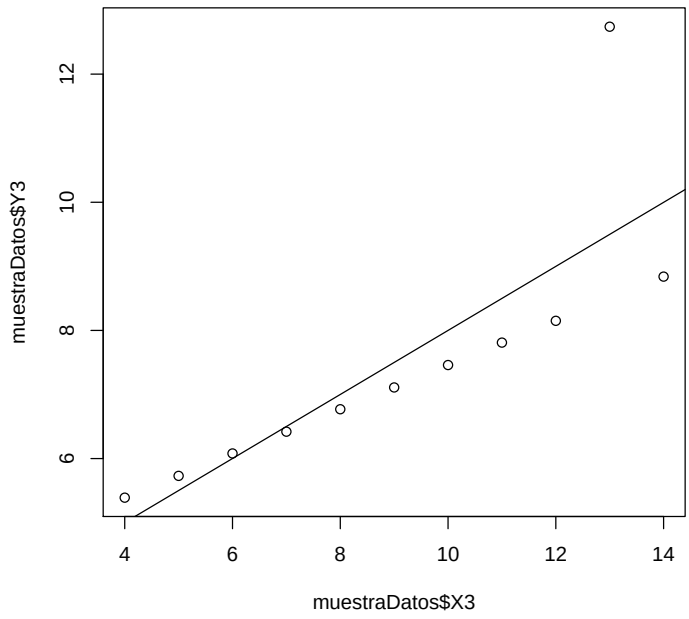
Regresión de la Muestra 2:

```
> regresion2=lm(Y2~X2, data=muestraDatos)
> summary(regresion2)

Call:
lm(formula = Y2 ~ X2, data = muestraDatos)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9009 -0.7609  0.1291  0.9491  1.2691

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    3.001      1.125   2.667  0.02576 *
X2             0.500      0.118   4.239  0.00218 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.237 on 9 degrees of freedom
Multiple R-squared:  0.6662,        Adjusted R-squared:  0.6292
F-statistic: 17.97 on 1 and 9 DF,  p-value: 0.002179

> plot(muestraDatos$X2, muestraDatos$Y2)
> abline(regresion2)
```



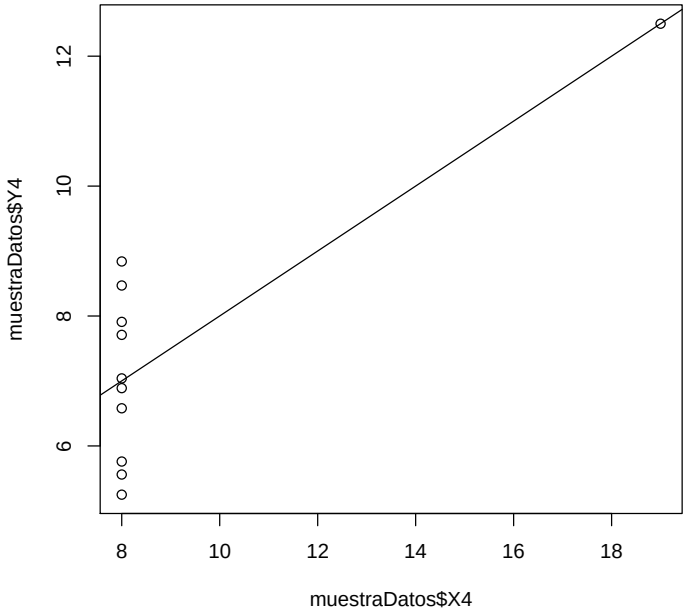
Regresión de la Muestra 3:

```
> regresion3=lm(Y3~X3, data=muestraDatos)
> summary(regresion3)

Call:
lm(formula = Y3 ~ X3, data = muestraDatos)

Residuals:
    Min      1Q  Median      3Q     Max
-1.1586 -0.6146 -0.2303  0.1540  3.2411

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0025     1.1245   2.670  0.02562 *
X3            0.4997     0.1179   4.239  0.00218 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.236 on 9 degrees of freedom
Multiple R-squared:  0.6663,        Adjusted R-squared:  0.6292
F-statistic: 17.97 on 1 and 9 DF,  p-value: 0.002176

> plot(muestraDatos$X3, muestraDatos$Y3)
> abline(regresion3)
```



Regresión de la Muestra 4:

```
> regresion4=lm(Y4~X4, data=muestraDatos)
> summary(regresion4)

Call:
lm(formula = Y4 ~ X4, data = muestraDatos)

Residuals:
   Min     1Q Median     3Q    Max
-1.751 -0.831  0.000  0.809  1.839

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.0017     1.1239   2.671  0.02559 *
X4            0.4999     0.1178   4.243  0.00216 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.236 on 9 degrees of freedom
Multiple R-squared:  0.6667,        Adjusted R-squared:  0.6297
F-statistic:    18 on 1 and 9 DF,  p-value: 0.002165

> plot(muestraDatos$X4, muestraDatos$Y4)
> abline(regresion4)
```



2.3 Desarrollo por parte del grupo

2.3.1 Clasificación en árbol de una muestra

En este apartado vamos a realizar una clasificación en árbol de los datos de una muestra sobre calificaciones de alumnos utilizando para ello el algoritmo de Hunt con ganancia de información mediante la medida de impureza Gini.

Los datos que se van a utilizar en este apartado viene dado de un fichero .csv llamado **basic_income2.csv**.

```
> data<-read.csv("basic_income2.csv")
```

Ahora procedemos a crear la muestra, para ello lo hacemos con el comando **data.frame**.

```
> muestraIncome = data.frame(data)
```

Una vez creada la muestra, ya podemos proceder a clasificarla, para ello debemos de saber, sobre que columna queremos clasificar, para ello, hacemos un **colnames()**.

```
> colnames(muestraIncome)
```

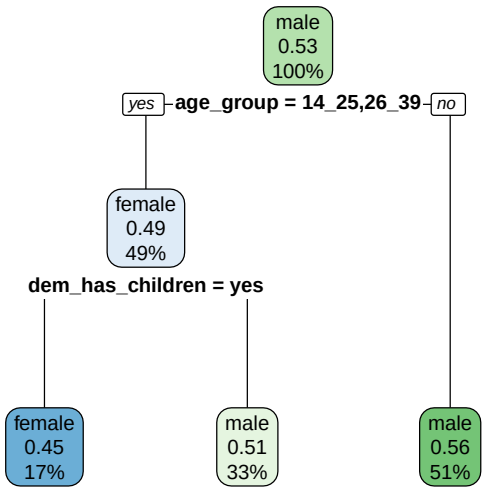
```
[1] "X"                   "gender"              "rural"
[4] "dem_education_level" "dem_has_children"    "age_group"
```

Una vez visualizadas las columnas, seleccionamos sobre cual queremos hacer la clasificación, en nuestro caso hemos elegido **gender**. Quedando el siguiente resultado:

```
> clasificacionIncome = rpart(gender~.,data=muestraIncome,method='class',minsplit=1)
> clasificacionIncome

n= 9649

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 9649 4555 male (0.4720696 0.5279304)
  2) age_group=14_25,26_39 4753 2336 female (0.5085209 0.4914791)
    4) dem_has_children=yes 1601  713 female (0.5546533 0.4453467) *
    5) dem_has_children=no 3152 1529 male (0.4850888 0.5149112) *
  3) age_group=40_65 4896 2138 male (0.4366830 0.5633170) *
```

Ahora lo visualizamos en forma de árbol, para ver de forma más clara los resultados:

```
> rpart.plot(clasificacionIncome)
```



Una vez visualizado el árbol de clasificación mediante **rpart**, ahora vamos a clasificarlos con **tree**, quedando el siguiente resultado:

```
> clasificacionTreeInc = tree(dem_education_level~.,data=muestraIncome,
+ mincut=1,minsize=2)
> clasificacionTreeInc
```

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

1) root 8986 21160 medium ( 0.36390 0.20198 0.39817 0.03594 ) *
```

Como podemos observar, la clasificación no nos ha salido igual, siendo necesario mediante la clasificación **rpart**, dos columnas, pero para la clasificación **tree**, no necesitamos ninguna.

2.3.2 Función de regresión

En este apartado vamos a calcular la función de regresión del archivo **appstore_games.csv**. Lo primero de todo es cargar las librerías necesarias, en este caso necesitamos solamente una, que es **ggplot2**, cuya función es representar los datos en una gráfica:

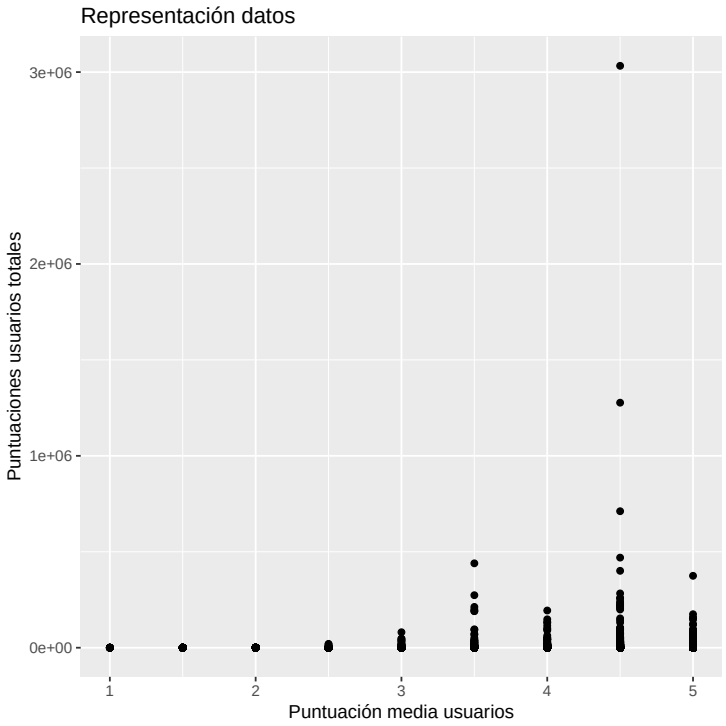
```
> install.packages("ggplot2")
> library(ggplot2)
```

Posteriormente cargamos la base de datos a utilizar:

```
> datosApps = read.csv("appstore_games.csv")
```

Ahora vamos a visualizar una gráfica que contendrá los datos, los cuales calcularemos la recta de regresión del archivo. Quedando el siguiente resultado:

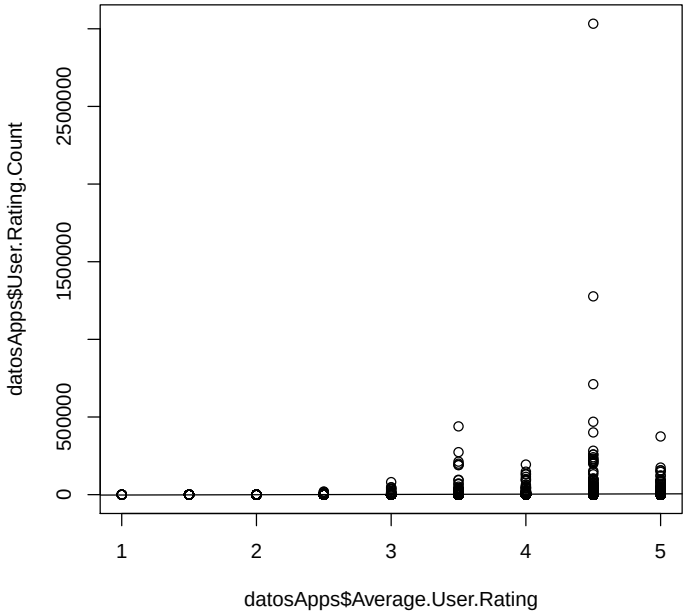
```
> ggplot()+geom_point(data = datosApps, aes(x = Average.User.Rating,
+ y = User.Rating.Count)) + xlab("Puntuación media usuarios") +
+ ylab("Puntuaciones usuarios totales") +
+ ggtitle("Representación datos")
```



Donde x es la media de valoración de usuarios(desde 0 hasta 5), y la y es el total de valoraciones que hay.

Una vez visualizada la gráfica, vamos a calcular los coeficientes de la recta regresión. Quedando el siguiente resultado:

```
> regresionApps=lm(User.Rating.Count~Average.User.Rating, data=datosApps)
> summary(regresionApps)

Call:
lm(formula = User.Rating.Count ~ Average.User.Rating, data = datosApps)

Residuals:
    Min      1Q  Median      3Q     Max
  -5048   -4072   -3154   -1745 3028611

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         -4247.0     2673.9  -1.588  0.11226
Average.User.Rating  1860.1      647.5   2.873  0.00408 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 42300 on 7559 degrees of freedom
  (9446 observations deleted due to missingness)
Multiple R-squared:  0.001091,        Adjusted R-squared:  0.0009585
F-statistic: 8.253 on 1 and 7559 DF,  p-value: 0.00408

> plot(datosApps$Average.User.Rating, datosApps$User.Rating.Count)
> abline(regresionApps)
```


datosApps$User.Rating.Count
0
500000
1500000
2500000
1
2
3
4
5
datosApps$Average.User.Rating