MACHINE LEARNING EN RSTUDIO

ARBOLES DE DECISION

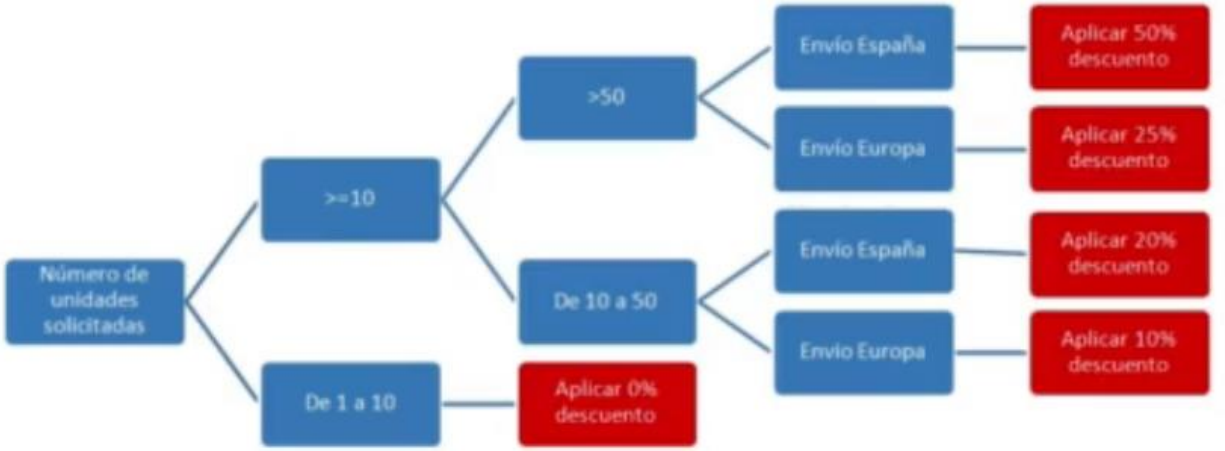
EDUARD LARA

1. INDICE

1. Introducción árboles de decisión
2. Random Forest
3. Maquina de vectores de soporte
4. Algoritmo de k-medias

1. ARBOLES DE DECISION

- Un árbol de decisión es un modelo de predicción utilizado en diferentes ámbitos que van desde la inteligencia artificial hasta la economía
- Dado un conjunto de datos, se fabrica unos diagramas de construcciones lógicas con lo que aparece en la figura, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva para la resolución de un problema.



1. EJEMPLO ARBOLES DE DECISION

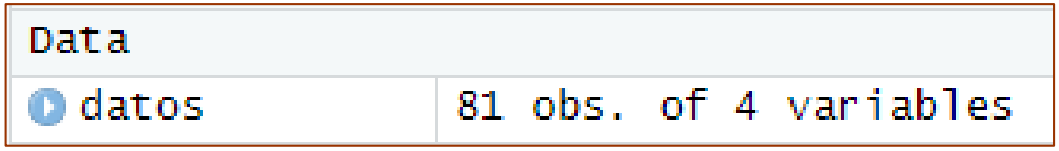
Paso 1. Para ver los árboles de decisión, instalamos primero el paquete repart, y lo cargamos en memoria con library.

```
> install.packages('rpart')
Installing package into 'C:/Users/eduar/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/rpart_4.1.23.zip'
Content type 'application/zip' length 710741 bytes (694 KB)
downloaded 694 KB

package 'rpart' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\eduar\AppData\Local\Temp\RtmpSclMQu\downloaded_packages
>
> library(rpart)
>
```

1. EJEMPLO ARBOLES DE DECISION

Paso 2. Vamos a cargar unos datos mediante el dataset llamado kyphosis. Hacemos un str() de los datos y observamos que contiene 81 observaciones de 4 columnas.

```
> datos = kyphosis
> str(datos)
'data.frame':   81 obs. of  4 variables:
 $ Kyphosis: Factor w/ 2 levels "absent","present": 1 1 2 1 1 1 1 1 1 2 ...
 $ Age     : int  71 158 128 2 1 1 61 37 113 59 ...
 $ Number  : int  3 3 4 5 4 2 2 3 2 6 ...
 $ Start   : int  5 14 5 1 15 16 17 16 16 12 ...
> |
```



Kyphosys (enfermedad relacionada con las vertebras) es la columna objetivo que queremos predecir, si está presente o está ausente, en función de la edad (en meses), el número de vértebras y en que vertebra comienza.

1. EJEMPLO ARBOLES DE DECISION

Paso 3. Construimos este árbol de decisión mediante la función rpart del paquete que hemos instalado.

```
> arbol = rpart(kyphosis, datos)
>
```

Indicamos la columna que queremos predecir y le pasamos los datos del dataset completo

1. EJEMPLO ARBOLES DE DECISION

Paso 4. Imprimiremos este modelo mediante la función printcp

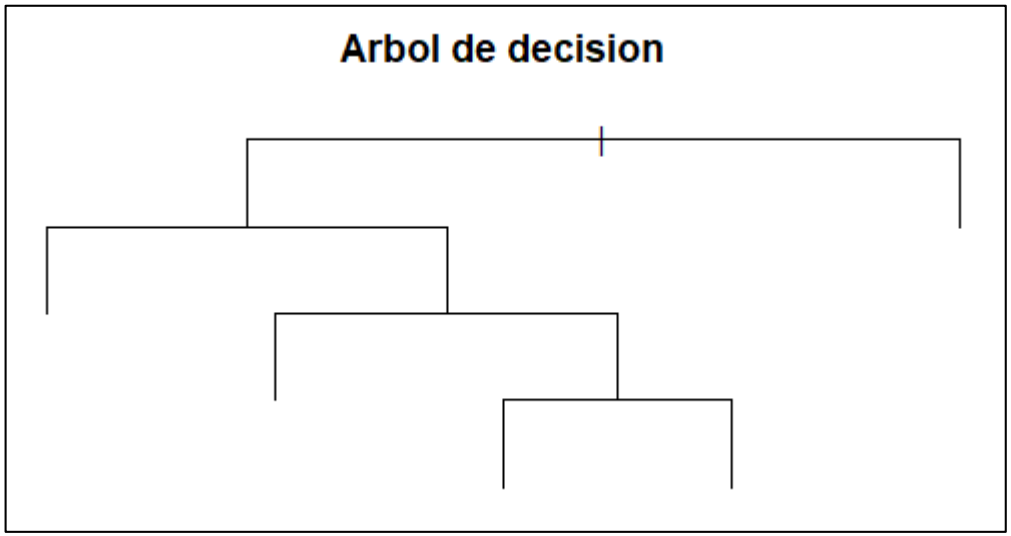
```
> printcp(arbol)

Classification tree:
rpart(formula = kyphosis, data = datos)

Variables actually used in tree construction:
[1] Age   Start

Root node error: 17/81 = 0.20988

n= 81

        CP nsplit rel error xerror    xstd
1 0.176471      0   1.00000 1.0000 0.21559
2 0.019608      1   0.82353 1.1176 0.22433
3 0.010000      4   0.76471 1.1176 0.22433
> 
```

Nos da información sobre el árbol de decisión/clasificación y las tasas de error de cada uno de los datos.

1. EJEMPLO ARBOLES DE DECISION

Paso 5. Dibujaremos este árbol de decisión mediante plot para que sea mejor. A la derecha pinta el árbol de decisión.

```
> plot(arbol, uniform=TRUE, main='Arbol de decision')
>
```

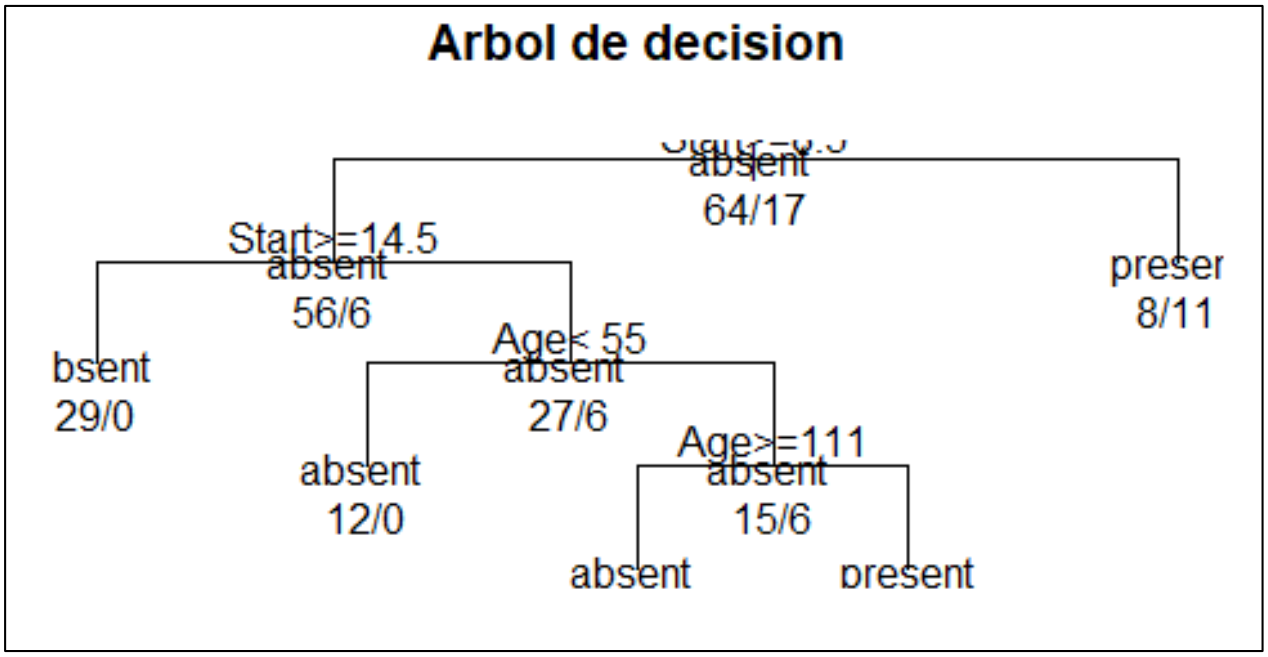


Nota: Nos faltan los textos.

1. EJEMPLO ARBOLES DE DECISION

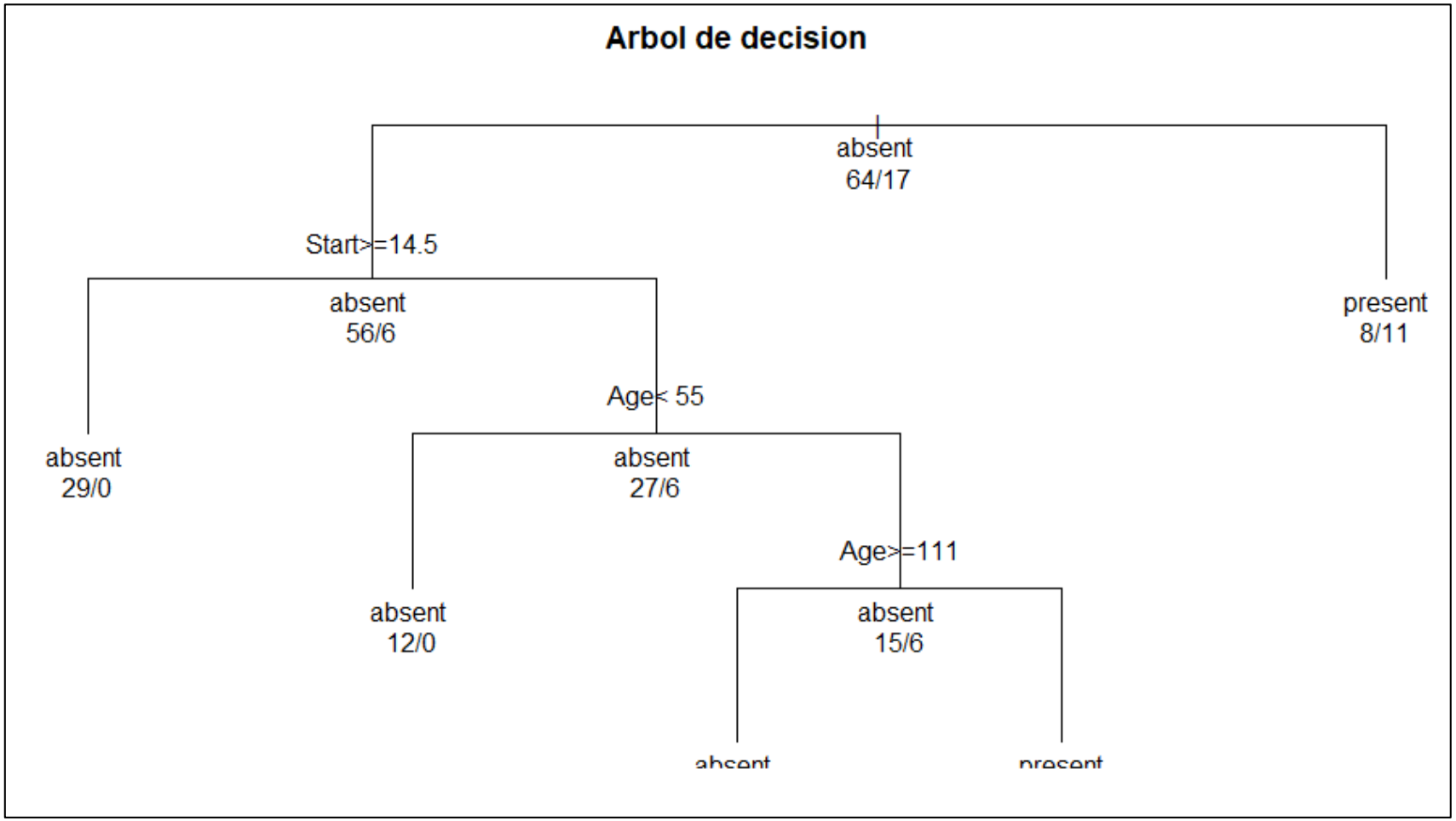
Paso 6. Agregamos los textos con la función text. Añade todas las etiquetas al árbol.

```
> text(arbol, use.n=TRUE, all=TRUE)
>
```



1. EJEMPLO ARBOLES DE DECISION

Paso 7. Para que se vea bien tenemos que darle al zoom.



1. EJEMPLO ARBOLES DE DECISION

Explicación

- Vemos el árbol de decisión en función de dónde comienza por ejemplo la columna start
- Si start es >=14.5 va por la izquierda indicando que esta ausente la enfermedad.
- Si start <14.5, entonces si la edad es <55, va por el lado derecho indicando que la enfermedad esta ausente.
- El árbol de decisión está en función de los valores de las columnas. Al final acaban dando resultados de si está presente o ausente la enfermedad, según los datos de las columnas

1. EJEMPLO ARBOLES DE DECISION

Paso 8. Utilizaremos otro gráfico donde se ve mejor el árbol de decisión: rpart.plot

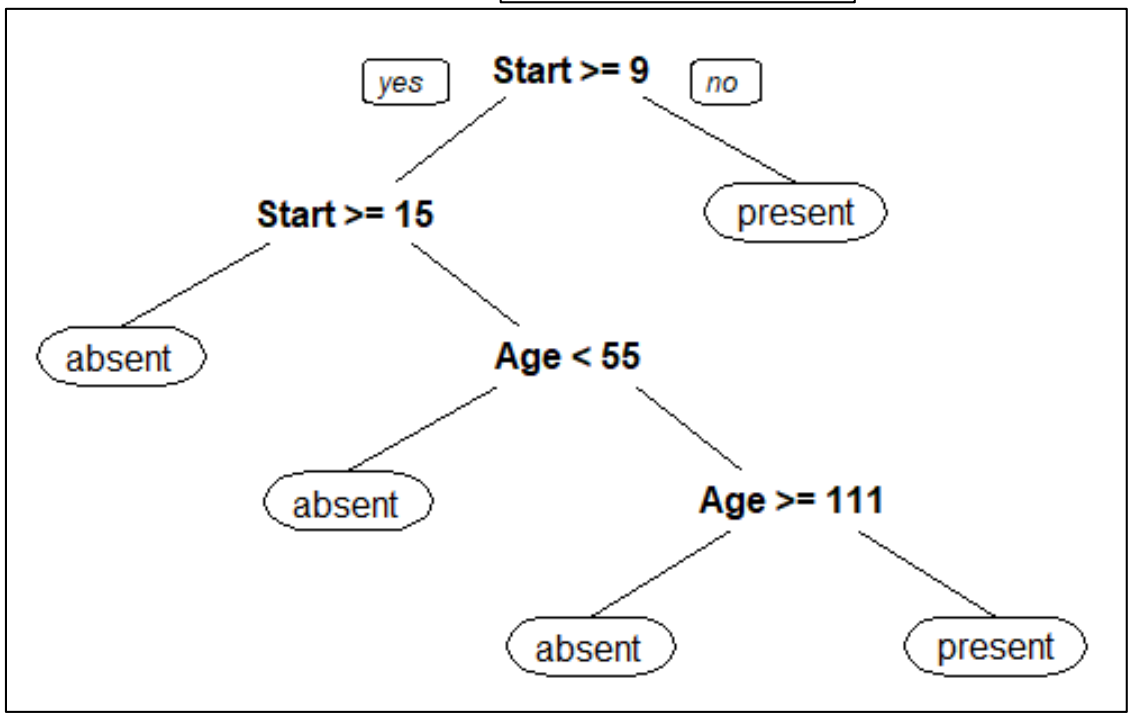
Lo instalamos y lo cargamos en memoria

```
> install.packages('rpart.plot')
Installing package into 'C:/Users/eduar/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/rpart.plot_3.1.1.zip'
Content type 'application/zip' length 1035076 bytes (1010 KB)
downloaded 1010 KB

package 'rpart.plot' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\eduar\AppData\Local\Temp\RtmpopVs11\downloaded_packages
> library(rpart.plot)
>
```

1. EJEMPLO ARBOLES DE DECISION

Paso 9. Visualizamos el árbol de decisión con la función prp.

```
> prp(arbol)
>
```

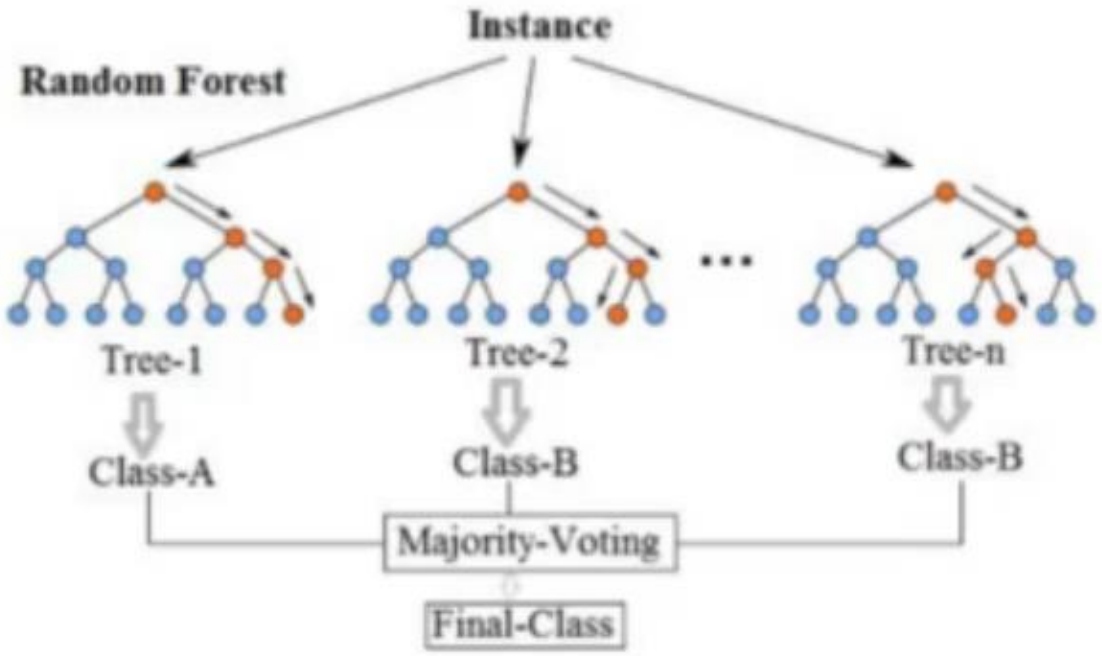


En función de la variable start, si es >=9, si es true va por la izquierda indicando que la enfermedad es presente, y si es false va por la derecha. Si start >=15, si es que si esta ausente de la enfermedad, etc

En función de start y edad se van evalúan las distintas opciones y nos daría el resultado de la predicción kyphosys

2. RANDOM FOREST

Random Forest (bosque aleatorio) es una combinación de árboles de decisión donde cada árbol selecciona una clase y luego se combinan las decisiones de cada árbol para seleccionar una clase final ganadora.



Es uno de los algoritmos de aprendizaje de clasificación con mayor precisión

Funciona de forma eficiente con base de datos grandes.

Además, puede manejar cientos de variables de entrada.

2. EJEMPLO DE RANDOM FOREST

Paso 1. Random Forest es un algoritmo de clasificación que combina diferentes árboles hasta encontrar el árbol óptimo. Instalamos el paquete de Random Forest, y lo cargamos en memoria

```
> install.packages('randomForest')
Installing package into 'C:/Users/eduar/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/randomFores
t_4.7-1.1.zip'
Content type 'application/zip' length 222252 bytes (217 KB)
downloaded 217 KB

package 'randomForest' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\eduar\AppData\Local\Temp\RtmpopVs11\downloaded_packages
> library(randomForest)
randomForest 4.7-1.1
Type rfNews() to see new features/changes/bug fixes.
>
```

2. EJEMPLO DE RANDOM FOREST

Paso 2. Ahora lo que hacemos es crear el modelo. Usamos la función randomForest a la que le pasamos la columna kyphosis, que es que queremos estimar, y datos.

Hacemos el print del modelo.

```
> modelo = randomForest(Kyphosis  ~ .  ,data=datos)
> print(modelo)

Call:
 randomForest(formula = Kyphosis ~ ., data = datos)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 23.46%
Confusion matrix:
        absent present class.error
absent      59       5   0.0781250
present     14       3   0.8235294
>
```

Tenemos el modelo de Random Forest para el dataset kyphosis

Indica el número de árboles utilizado, el tipo de randomforest de clasificación y nos da también la matriz de confusión.

2. EJEMPLO DE RANDOM FOREST

Paso 3. Para ver las predicciones que ha realizado este algoritmo de RandomForest, usamos **modelo$predicted**

```
> modelo$predicted
      1       2       3       4       5       6       7       8       9      10
present  absent  absent  absent  absent  absent  absent  absent  absent  absent
     11      12      13      14      15      16      17      18      19      20
 absent  absent  absent  absent  absent  absent  absent  absent  absent  absent
     21      22      23      24      25      26      27      28      29      30
 absent present  absent present  absent  absent  absent  absent  absent  absent
     31      32      33      34      35      36      37      38      39      40
 absent  absent  absent  absent  absent  absent  absent  absent  absent  absent
     41      42      43      44      45      46      47      48      49      50
 absent  absent present  absent  absent  absent  absent  absent  absent  absent
     51      52      53      54      55      56      57      58      59      60
 absent  absent  absent  absent  absent  absent  absent present present  absent
     61      62      63      64      65      66      67      68      69      70
 absent present present  absent  absent  absent  absent  absent  absent  absent
     71      72      73      74      75      76      77      78      79      80
 absent  absent  absent  absent  absent  absent  absent  absent  absent  absent
     81
 absent
Levels: absent present
> |
```

Nos indica que esta presente para la fila 1, ausente para la fila 2, etc, así hasta las 81 observaciones.

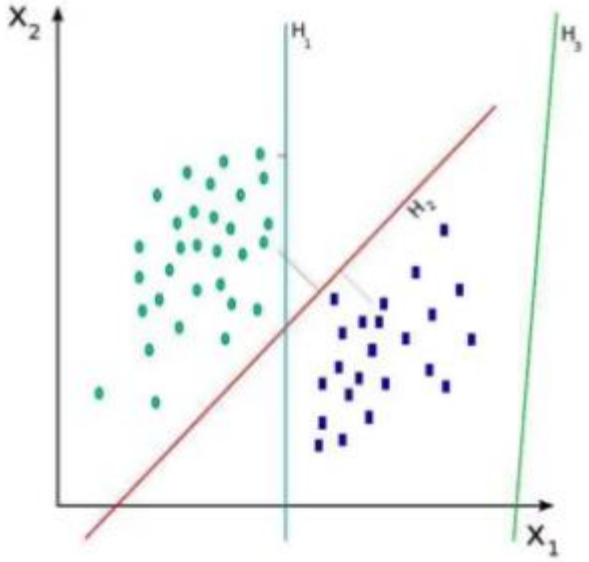
El algoritmo nos da las predicciones para todas las filas en función de los valores de las otras columnas.

3. MAQUINAS DE VECTORES DE SOPORTE

- Las máquinas de soporte vectorial (SVM) son un conjunto de algoritmos de aprendizaje supervisado para resolver problemas de clasificación y de regresión.
- Dado un conjunto de ejemplos de entrenamiento, podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra.
- Representan los puntos de muestra en el espacio, separando las clases en dos espacios de la forma más amplia posible mediante un hiperplano de separación, denominado vector de soporte

3. MAQUINAS DE VECTORES DE SOPORTE

- En la siguiente grafico el vector de soporte puede ser la línea roja que separa el grupo de puntos de muestra azules de los verdes.
- Así, una vez que tengamos una nueva muestra, podemos ver si está a la derecha de la raya roja o pertenecer al grupo azul, y si está a la izquierda pertenece al Grupo Verde.



3. EJEMPLO DE MAQUINAS DE VECTORES DE SOPORTE

Paso 1. Vamos a ver el algoritmo de máquinas de vectores de soporte SVM. Utilizaremos el dataset iris que viene en el paquete ISLR.

```
> library(ISLR)
> datos = iris
> str(datos)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ..
> |
```

Data	
datos	150 obs. of 5 variables

El dataset tiene 150 filas y 5 columnas: la longitud y el ancho del sépalo, la longitud y el ancho del pétalo, y luego, en función de sus características, la especie que es, que puede ser de tipos: setosa, versicolor y virginica

3. EJEMPLO DE MAQUINAS DE VECTORES DE SOPORTE

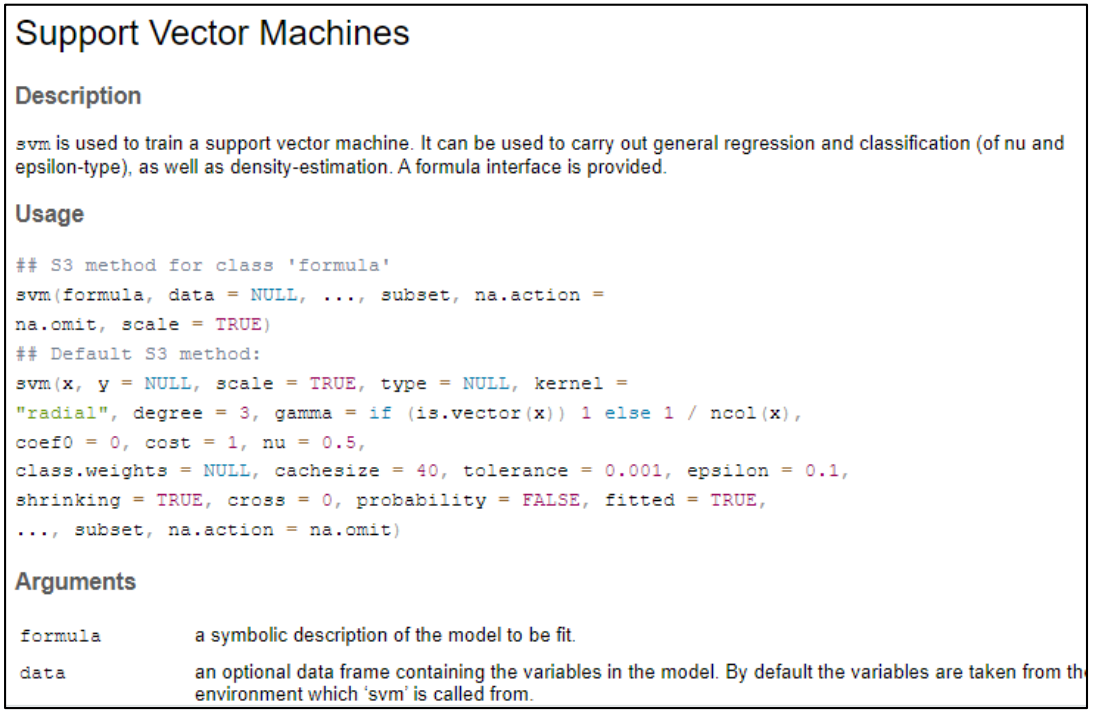
Paso 2. Instalaremos otro paquete para utilizar el SVM o el algoritmo de máquinas de vectores de soporte.

Entonces instalamos el paquete e1071.

```
> install.packages('e1071')
Installing package into 'C:/Users/eduar/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
also installing the dependency 'proxy'

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/proxy_0.4-27.zip'
Content type 'application/zip' length 179932 bytes (175 KB)
downloaded 175 KB

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/e1071_1.7-14.zip'
Content type 'application/zip' length 664440 bytes (648 KB)
downloaded 648 KB

package 'proxy' successfully unpacked and MD5 sums checked
package 'e1071' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\eduar\AppData\Local\Temp\Rtmp4YbTsm\downloaded_packages
> library(e1071)
>
```

3. EJEMPLO DE MAQUINAS DE VECTORES DE SOPORTE

Paso 3. Para obtener mas información sobre este algoritmo ponemos help('svm'), y nos sale la página de ayuda de rstudio donde podemos ver el algoritmo y la información de los distintos parámetros que podemos usar. Al final de todo vienen ejemplos de optimización



Support Vector Machines

Description

svm is used to train a support vector machine. It can be used to carry out general regression and classification (of nu and epsilon-type), as well as density-estimation. A formula interface is provided.

Usage

```
## S3 method for class 'formula'
svm(formula, data = NULL, ..., subset, na.action =
na.omit, scale = TRUE)
## Default S3 method:
svm(x, y = NULL, scale = TRUE, type = NULL, kernel =
"radial", degree = 3, gamma = if (is.vector(x)) 1 else 1 / ncol(x),
coef0 = 0, cost = 1, nu = 0.5,
class.weights = NULL, cachesize = 40, tolerance = 0.001, epsilon = 0.1,
shrinking = TRUE, cross = 0, probability = FALSE, fitted = TRUE,
..., subset, na.action = na.omit)
```

Arguments

formula	a symbolic description of the model to be fit.
data	an optional data frame containing the variables in the model. By default the variables are taken from th environment which 'svm' is called from.

3. EJEMPLO DE MAQUINAS DE VECTORES DE SOPORTE

Paso 4. Vamos a construir el modelo de la máquina de vectores de soporte.

```
> modelo = svm(Species ~ . , data=datos)
> predicciones = predict(modelo, datos[1:4])
>
```

Usamos el algoritmo SVM y vamos a predecir en este caso la columna Species a partir de todos los datos del dataset

Paso 5. A partir de este modelo vamos a hacer las predicciones. Formulamos una variable predicciones que va a ser el resultado del método Predict al que le vamos a pasar el modelo y las cuatro primeras columnas que contienen las características.

3. EJEMPLO DE MAQUINAS DE VECTORES DE SOPORTE

Paso 6. Si vemos proyecciones, veremos las predicciones que realiza nuestro modelo para cada una de las filas.

Las primeras filas son setosa

```
> predicciones
         1          2          3          4          5          6
    setosa     setosa     setosa     setosa     setosa     setosa
         7          8          9         10         11         12
    setosa     setosa     setosa     setosa     setosa     setosa
        13         14         15         16         17         18
    setosa     setosa     setosa     setosa     setosa     setosa
        19         20         21         22         23         24
    setosa     setosa     setosa     setosa     setosa     setosa
        25         26         27         28         29         30
    setosa     setosa     setosa     setosa     setosa     setosa
        31         32         33         34         35         36
    setosa     setosa     setosa     setosa     setosa     setosa
        37         38         39         40         41         42
    setosa     setosa     setosa     setosa     setosa     setosa
        43         44         45         46         47         48
    setosa     setosa     setosa     setosa     setosa     setosa
        49         50         51         52         53         54
    setosa     setosa versicolor versicolor versicolor versicolor
        55         56         57         58         59         60
versicolor versicolor versicolor versicolor versicolor versicolor
        61         62         63         64         65         66
versicolor versicolor versicolor versicolor versicolor versicolor
```

3. EJEMPLO DE MAQUINAS DE VECTORES DE SOPORTE

Paso 7. Si queremos visualizar el modelo de una forma más clara, podemos visualizar los datos junto con las predicciones. Construimos una tabla que va a ser un data.frame con los datos y las predicciones.

```
> tabla
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species predicciones
1            5.1         3.5          1.4         0.2     setosa        setosa
2            4.9         3.0          1.4         0.2     setosa        setosa
3            4.7         3.2          1.3         0.2     setosa        setosa
4            4.6         3.1          1.5         0.2     setosa        setosa
5            5.0         3.6          1.4         0.2     setosa        setosa
6            5.4         3.9          1.7         0.4     setosa        setosa
7            4.6         3.4          1.4         0.3     setosa        setosa
8            5.0         3.4          1.5         0.2     setosa        setosa
9            4.4         2.9          1.4         0.2     setosa        setosa
10           4.9         3.1          1.5         0.1     setosa        setosa
11           5.4         3.7          1.5         0.2     setosa        setosa
12           4.8         3.4          1.6         0.2     setosa        setosa
13           4.8         3.0          1.4         0.1     setosa        setosa
14           4.3         3.0          1.1         0.1     setosa        setosa
15           5.8         4.0          1.2         0.2     setosa        setosa
16           5.7         4.4          1.5         0.4     setosa        setosa
17           5.4         3.9          1.3         0.4     setosa        setosa
18           5.1         3.5          1.4         0.3     setosa        setosa
19           5.7         3.8          1.7         0.3     setosa        setosa
20           5.1         3.8          1.5         0.3     setosa        setosa
21           5.4         3.4          1.7         0.2     setosa        setosa
22           5.1         3.7          1.5         0.4     setosa        setosa
23           4.6         3.6          1.0         0.2     setosa        setosa
```

Vemos los valores de cada una de las columnas de especies y nuestras predicciones.

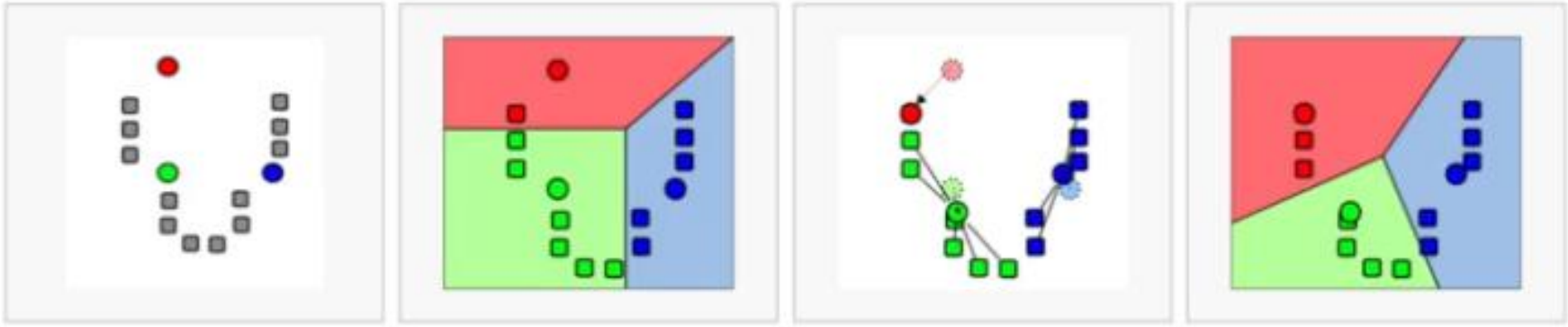
Las predicciones casi siempre aciertan con la especie. El modelo svm funciona muy bien para este dataset

4. ALGORITMO DE K-MEDIAS

- El algoritmo de medias es un algoritmo de aprendizaje no supervisado para resolver el problema de la clusterización o la formación de grupos
- Tiene como objetivo la partición de un conjunto de "n" observaciones en "k" grupos, en la que cada observación pertenece al grupo cuyo valor medio es más cercano.
- Es un método utilizado en la minería de datos.

4. ALGORITMO DE K-MEDIAS

- Diagrama de izquierda derecha que muestra las distintas fases por las que va pasando hasta que al final cada observación pertenece a un grupo concreto.
- En este caso K es igual a tres.
- Todas las formaciones están dentro o agrupadas dentro tres grupos.



4. EJEMPLO ALGORITMO DE K-MEDIAS

Paso 1. Vamos a ver un ejemplo del algoritmo de k-medias es un algoritmo de aprendizaje no supervisado

Primero cargaremos el paquete ISLR y utilizaremos el dataset irish. Si hacemos un dataset de datos, vemos que tiene 150 observaciones con 5 variables: longitud y ancho de los sépalos, longitud y ancho de los pétalos y según eso vamos a intentar identificar cuál es la especie.

```
> library(ISLR)
> datos = iris
> str(datos)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> |
```

4. EJEMPLO ALGORITMO DE K-MEDIAS

Paso 2. Vamos a hacer un gráfico para ver cómo se agrupan las especies en este caso por colores.

Cargaremos la librería ggplot2

```
> install.packages("ggplot2")
Installing package into 'C:/Users/eduar/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/ggplot2_3.4.4.zip'
Content type 'application/zip' length 4299762 bytes (4.1 MB)
downloaded 4.1 MB

package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\eduar\AppData\Local\Temp\RtmpoLYcWP\downloaded_packages
> library(ggplot2)
>
```

4. EJEMPLO ALGORITMO DE K-MEDIAS

Paso 3. Crearemos un gráfico mediante un ggplot.

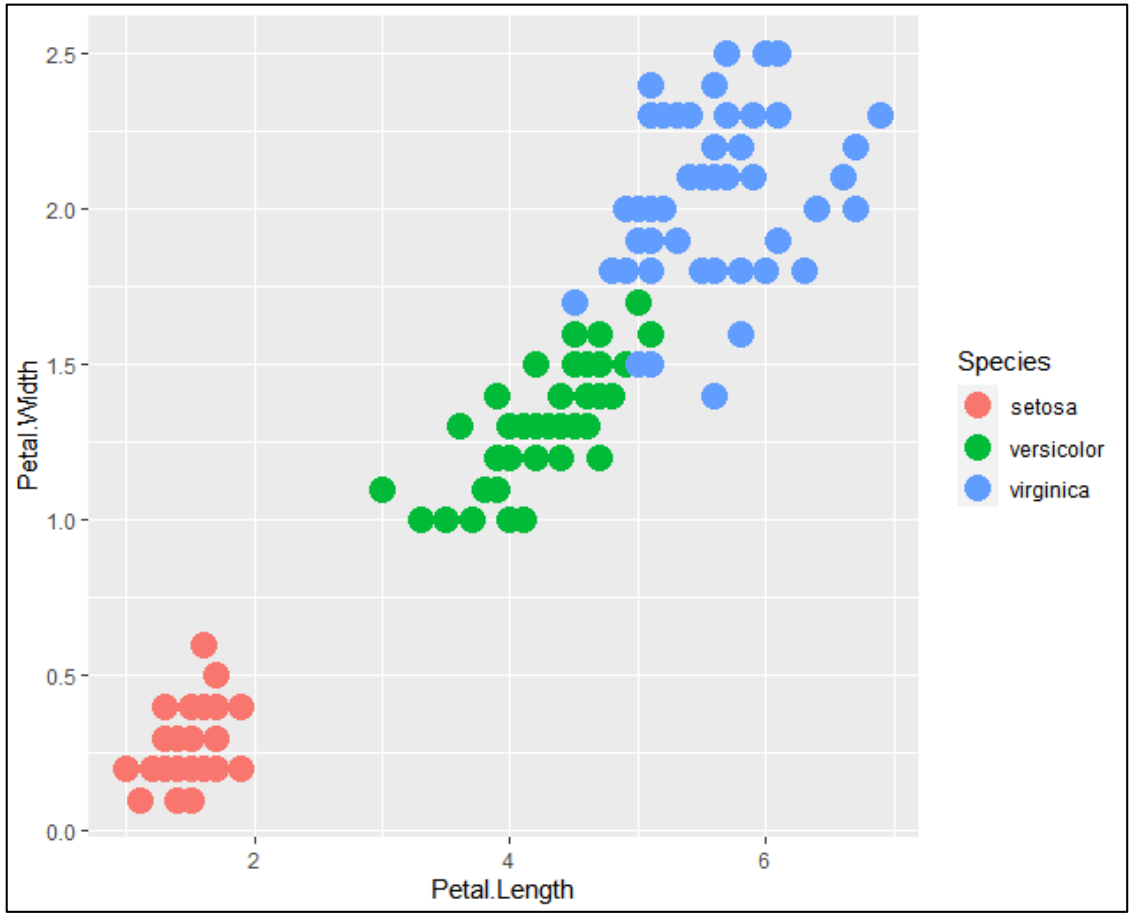
```
> grafico = ggplot(datos, aes(Petal.Length, Petal.Width, color=Species))
> grafico = grafico + geom_point(size=5)
> print(grafico)
>
```

Le pasaremos los datos: en el eje x → Petal.Length, en el eje y → Petal.Width y que ponga los colores según las especies, así podremos ver la agrupación entre especies.

Y por último, le decimos que añada el grafico tipo geom_point, donde le vamos a poner un tamaño de 5.

4. EJEMPLO ALGORITMO DE K-MEDIAS

Paso 4. Visualizamos el gráfico



En función de estos dos atributos Petal.Length y Petal.Width vemos que los 3 grupos de especies han quedado bastante bien agrupados:

color rojo → setosa

Color verde → versicolor

Color azul → virginica

4. EJEMPLO ALGORITMO DE K-MEDIAS

Paso 5. Vamos a crear el modelo o el algoritmo de k-medias. Primero ponemos la semilla y creamos una variable, que son los conjuntos de especies mediante kmeans

```
> set.seed(90)
> conjuntos = kmeans(datos[,1:4],3, nstart=20)
```

En este caso le pasamos las primeras 4 columnas de datos. Como hay 3 conjuntos de especies/colores ponemos un 3 y por último nstart lo ponemos a 20

4. EJEMPLO ALGORITMO DE K-MEDIAS

Paso 6. Si hacemos un print de los conjuntos para ver la información que ha generado el algoritmo de k-medias, vemos que he agrupado en tres clusters o conjuntos que tienen un tamaño de 38, 50 y 62 elementos

```
> print(conjuntos)
K-means clustering with 3 clusters of sizes 38, 50, 62

Cluster means:
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1     6.850000    3.073684     5.742105    2.071053
2     5.006000    3.428000     1.462000    0.246000
3     5.901613    2.748387     4.393548    1.433871

Clustering vector:
  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [44] 2 2 2 2 2 2 2 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3
 [87] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1 1 1 1 1 3 3 1 1 1 3 1 3 1 3 1 1 3 3 1
[130] 1 1 1 1 3 1 1 1 1 3 1 1 1 3 1 1 1 3 1 1 3

Within cluster sum of squares by cluster:
[1] 23.87947 15.15100 39.82097
 (between_SS / total_SS =  88.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
>
```

Aquí tenemos los valores que nos saca

4. EJEMPLO ALGORITMO DE K-MEDIAS

Paso 7. Si queremos ver esto en una tabla donde podamos ver cuántas especies hay en cada conjunto:

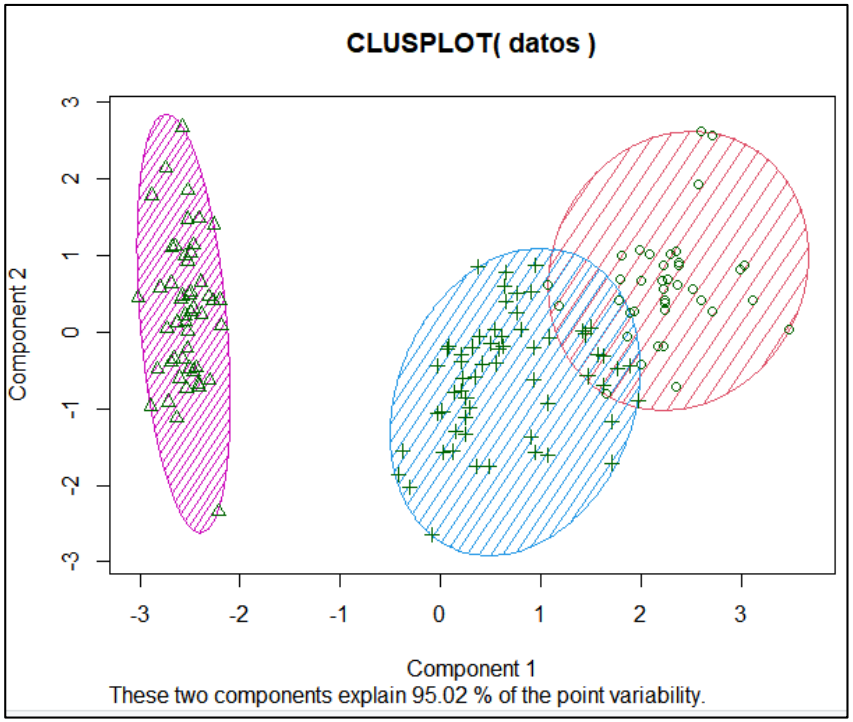
```
> table(conjuntos$cluster, datos$Species)

    setosa versicolor virginica
  1      0          2        36
  2     50          0         0
  3      0         48        14
> 
```

Vemos los datos para cada uno de los clusters o conjuntos. En el primer clúster aparece 0 (setosa), 2 (versicolor) y 36(virginica). En el segundo aparece 50, 0 y 0, y en el tercero 0, 48 y 14.

4. EJEMPLO ALGORITMO DE K-MEDIAS

Paso 8. También podemos realizar otro gráfico para ver los conjuntos representados por círculos. Cargamos el paquete cluster y lo vamos a utilizar mediante el cluspot

```
> library(cluster)
> clusplot(datos, conjuntos$cluster, color=TRUE, shade=TRUE, labels=0, lines=0)
>
```



Genera otro tipo de gráfico donde podemos agrupar la información que ha generado el algoritmo de k-medias, separándolo por círculos y colores.

4. EJEMPLO ALGORITMO DE K-MEDIAS

Paso 9. Si queremos más información del algoritmo de kmedias, simplemente tendríamos que ejecutar help y el nombre del algoritmo. Podemos ver los distintos parámetros que hay, y al final siempre viene un ejemplo que podemos utilizar.

```
> help('kmeans')
>
```

K-Means Clustering

Description

Perform k-means clustering on a data matrix.

Usage

```
kmeans(x, centers, iter.max = 10, nstart = 1,
       algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
                     "MacQueen"), trace = FALSE)
## S3 method for class 'kmeans'
fitted(object, method = c("centers", "classes"), ...)
```

Arguments

`x`	numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns).
`centers`	either the number of clusters, say k, or a set of initial (distinct) cluster centres. If a number, a random set of (distinct) rows in `x` is chosen as the initial centres.
`iter.max`	the maximum number of iterations allowed.
`nstart`	if `centers` is a number, how many random sets should be chosen?
`algorithm`	character: may be abbreviated. Note that "Lloyd" and "Forgy" are alternative