Curso : MACHINE LEARNING CON R

Tema : Preprocesamiento de Datos.

Docente : Victor Henostroza

**GUIA DE LABORATORIO 5**

**Objetivos del laboratorio:**

1. **Desarrollar un Modelo de Clustering con K-Means**
2. **Desarrollar un Modelo de Clustering Jeráquico.**
3. **Desarrollar un Modelo de Clustering con Densidad de Puntos.**
4. **Realizar una Reducción de Dimensiones con Análisis de Componentes Principales.**
5. **Realizar una Reducción de Dimensiones con Análisis de Componentes Principales con Kernel no Lineal.**
6. **Realizar una Reducción de Dimensiones con Análisis de Discriminante Lineal.**

**Objetivo 1:**

Empezamos el laboratorio desarrollando nuestro primer modelo de clustering con la técnica K-means, esta técnica se basa en generar centros de gravedad para cada cluster e ir asignando los puntos más cercanos a dicho centro formando clústeres cada vez más grandes (macroclusteres) hasta que los datos tengan suficiente similitud intra clústeres y suficiente diferencia entre clústeres. El dataset que usaremos se llama “protein.csv”, este contiene los datos de consumo de proteínas de varios países por tipo de proteína en cada columna.

**# Clustering con K-Means**

**# Importar los datos del centro comercial**

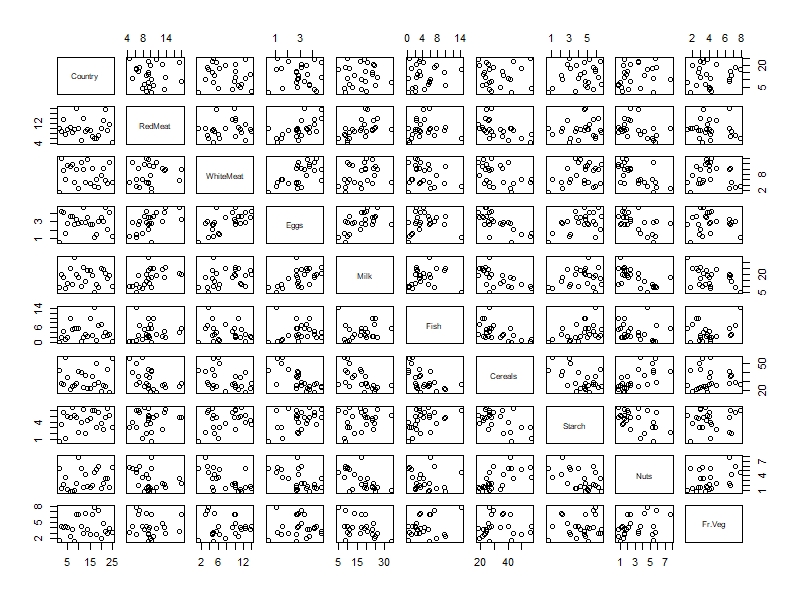
dataset = read.csv("protein.csv")

str(dataset)

head(dataset)

**# Ploteo de variables en 1Vs1**

pairs(dataset)

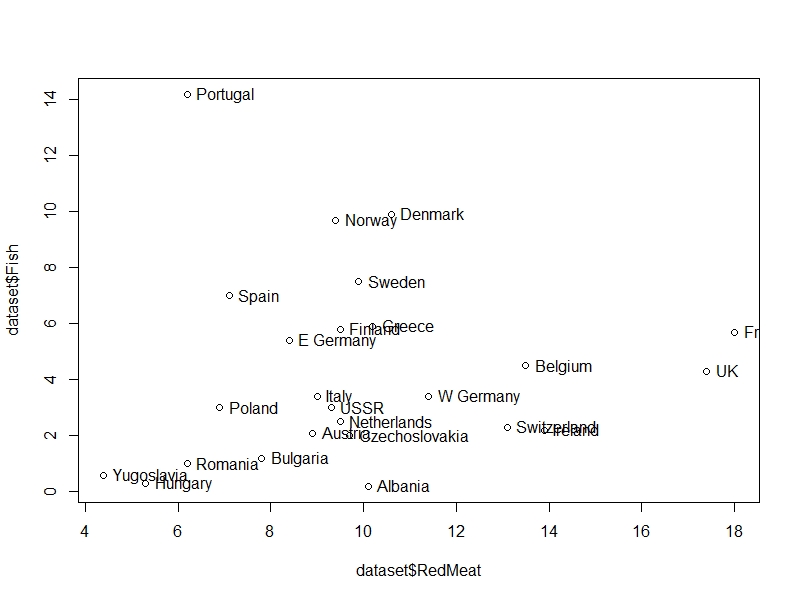


Del gráfico, apreciamos como se distribuyen los puntos en relación a dos variables independientes (todas las combinaciones posibles), en todas las combinaciones posibles de las variables independientes no se aprecian una tendencia marcada de ningún tipo de curva, por lo tanto, a simple vista no vemos que las variables estén correlacionadas fuertemente, no hay multicolinealidad.

**# Scatter plot de dos variables independientes**

plot(dataset$Fish~ dataset$RedMeat, data = dataset)

with(dataset,text(dataset$Fish ~ dataset$RedMeat, labels=dataset$Country,pos=4))



Del gráfico, vemos como en relación a las variables RedMeat (carnes rojas) y Fish (pescado) se están formando cuatro aglomeraciones de puntos de los países, abajo a la derecha (Belgium, Switzerland, etc) son un grupo de países que consumen mucha carne roja y poco pescado, arriba a la izquierda (Portugal, Spain, etc) son un grupo de países que consumen poca carne roja y mucho pescado, en el medio consumos intermedios de carne roja y pescado (Sweden, Germany, etc) y abajo a la izquierda con poco consumo de carne roja y poco consumo de pescado (Yugoslavia, Hungary, etc).

**# Normalizando las variables independientes numéricas**

datos\_originales = dataset

dataset[,2:9] = scale(dataset[,2:9])

**# Calculando la matriz de distancias entre casos (por defecto distancia euclidea)**

matriz\_distancias = dist(dataset)

print(matriz\_distancias, digits = 3)

> print(matriz\_distancias, digits = 3)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

2 6.86

3 6.59 2.60

4 3.65 5.17 5.52

5 5.78 2.26 2.34 4.17

6 7.02 3.59 3.02 6.56 3.82

7 6.94 2.79 2.26 5.73 2.01 3.10

8 6.20 4.99 4.34 6.61 4.78 2.91 4.71

9 7.87 4.25 3.19 6.20 4.17 5.28 4.74 6.59

10 6.15 5.78 5.41 4.45 5.58 6.91 6.45 7.33 4.79

11 5.39 3.46 4.21 3.53 2.91 5.54 3.91 6.19 5.62 4.78

12 7.20 3.14 2.00 6.65 3.47 3.01 3.26 3.66 4.58 6.79 5.20

13 6.10 4.45 4.58 3.73 4.25 6.29 5.30 6.96 4.01 2.27 3.98 6.10

14 6.57 1.30 2.39 5.47 2.33 2.91 2.67 4.10 4.35 5.96 3.71 2.57 4.90

15 5.83 4.32 3.32 5.72 3.89 2.12 3.54 2.46 5.31 5.91 5.34 3.81 5.49 3.65

16 7.55 3.57 3.84 5.13 3.18 5.48 3.88 6.30 3.79 4.65 3.84 5.10 3.29 3.88 5.19

17 8.84 7.57 6.87 7.11 6.75 **7.85** 6.70 8.92 6.09 5.19 6.83 8.64 5.03 7.66 6.81 5.20

18 2.99 5.08 5.13 2.34 3.90 5.84 5.09 5.48 6.67 5.01 2.88 5.91 4.74 4.96 4.94 5.34 7.43

19 7.60 5.75 5.06 5.74 5.20 6.84 5.34 7.70 4.73 3.32 4.86 6.73 3.06 5.98 5.90 3.62 3.15 5.91

20 5.97 3.70 3.24 6.01 3.86 **1.50** 3.52 2.24 5.64 6.57 5.28 3.11 6.00 2.94 1.70 5.72 8.07 5.17 6.82

21 6.09 2.38 2.59 4.77 2.88 4.01 3.93 4.83 2.92 4.55 4.11 3.45 3.48 2.26 4.01 3.57 6.97 4.95 5.23 3.81

22 6.42 4.05 2.14 6.16 4.08 3.75 4.15 4.42 3.90 5.62 5.45 2.40 5.33 3.72 3.78 5.56 7.99 5.73 6.03 3.49 3.31

23 4.70 4.56 3.47 4.20 3.02 4.41 3.65 3.89 5.47 5.36 3.79 4.11 5.02 4.15 3.44 4.47 6.92 2.91 5.37 4.05 4.37 4.24

24 6.95 1.79 1.50 5.92 2.31 2.80 2.00 4.39 3.90 6.13 4.13 2.06 5.05 1.34 3.61 4.00 7.41 5.44 5.70 3.04 2.59 3.08 4.18

25 3.37 5.82 5.95 2.28 4.63 6.74 5.83 6.31 7.25 5.05 3.31 6.82 4.87 5.82 5.72 5.60 7.40 1.10 5.96 6.10 5.69 6.61 3.55 6.31

De la tabla anterior, se muestra las distancias que hay entre las filas (países en nuestro caso). De color rojo, vemos que el país 6 y el país 17 tienen una distancia euclidea de 7.85 (están lejos, es poco probable que estén en el mismo cluster), en cambio, de color verde, apreciamos que el país 6 y el país 20 tienen de distancia 1.50 (están cerca, es muy probable que estén en el mismo cluster).

**# Generando el Cluster con K-Means**

clusters = kmeans(dataset[,2:9],4)

> clusters

**# Número de países aglomerados por clúster**

K-means clustering with 4 clusters of sizes 12, 6, 2, 5

**# Promedio de variables independientes (normalizadas) por clúster**

Cluster means:

RedMeat WhiteMeat Eggs Milk Fish Cereals Starch Nuts

1 0.6763112 0.4184171 0.69105398 0.8286637 0.2079823 -0.7492629 0.2084755 -0.6573728

2 -0.5610864 -0.9283320 -1.06118690 -0.7874744 -0.6565696 1.3593583 -1.3826291 1.1639997

3 -0.9494848 -1.1764767 -0.74802044 -1.4583242 1.8562639 -0.3779572 0.9326321 1.1220326

4 -0.5700494 0.5803879 -0.08589708 -0.4604938 -0.4537795 0.3181839 0.7857609 -0.2679180

**# Clúster asignado a cada país**

Clustering vector:

[1] 2 1 1 2 4 1 4 1 1 2 4 1 2 1 1 4 3 2 3 1 1 1 4 1 2

**# Variabilidad (en porcentaje) dentro de cada clúster**

Within cluster sum of squares by cluster:

[1] 39.395628 17.691265 4.225288 14.587826

(between\_SS / total\_SS = 60.5 %) **# Variabilidad explicada global**

Available components:

[1] "cluster" "centers" "totss" "withinss" "tot.withinss" "betweenss" "size" "iter"

[9] "ifault"

De la tabla, apreciamos que con 4 clústeres hemos conseguido explicar el 60.5% de la variabilidad entre clústeres en relación a la variabilidad entre el total de puntos, este es un monto relativamente bajo, pero recordamos que cuando hicimos el ploteo de “carne roja” vs “pescado” los grupos no estaban muy separados entre sí y eso se ve reflejado en nuestro resultado.

**# Visualización de los clústeres en dos variables independientes**

#install.packages("cluster")

x = as.data.frame(dataset$RedMeat)

x[,2] = dataset$Fish

library(cluster)

clusplot(x,

clusters$cluster,

lines = 0,

shade = TRUE,

color = TRUE,

labels = 2,

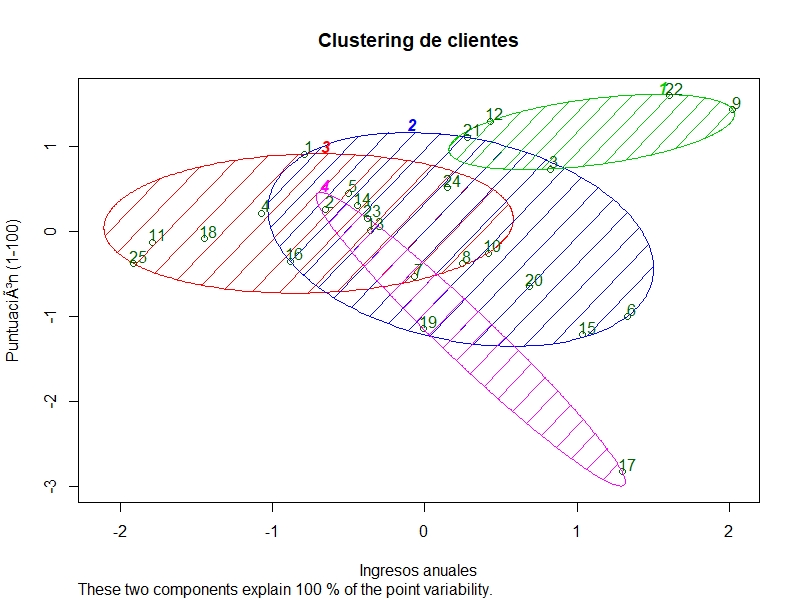
plotchar = FALSE,

span = TRUE,

main = "Clustering de clientes",

xlab = "Ingresos anuales",

ylab = "Puntuación (1-100)")



Del gráfico, vemos que se generan los 4 clústeres en cada región de elipse, los números de colores (rojo, azul, verde claro y morado) corresponde al número de clúster y los números en verde a los elementos (países) que pertenecen a cada uno de ellos. Vemos que la variabilidad del grupo 1 es la menor de todas por tener los puntos más cercanos entre sí dentro de dicho clúster. La manera de obtener el centro del cluster (baricentro) sería generando el promedio de las componentes en el eje “x” y en el eje “y” de cada uno de los puntos.

**Objetivo 2:**

En este apartado, realizaremos nuestro modelo de Clustering Jerárquico Aglomerativo, el cual se basa en ir juntando los datos según distancias más cercanas entre ellos hasta formar macro clusters que tengan suficiente similitud entre sus datos y suficiente diferencia entre clusters, dichas diferencias y similitudes son basadas en distancias euclideas para nuestro ejemplo. El dataset a utilizar será “utilities.csv” el cual muestra diferentes tipos de variables relacionadas a costos y ventas en relación a la ciudad donde se encuentra el local de la empresa.

**# Clustering Jerárquico**

**# Importar los datos**

dataset = read.csv2("utilities.csv")

str(dataset)

head(dataset)

**# Volviendo numéricas a las variables independientes que aparecen como factor**

dataset$Fixed\_charge = as.numeric(dataset$Fixed\_charge)

dataset$RoR = as.numeric(dataset$RoR)

dataset$Load = as.numeric(dataset$Load)

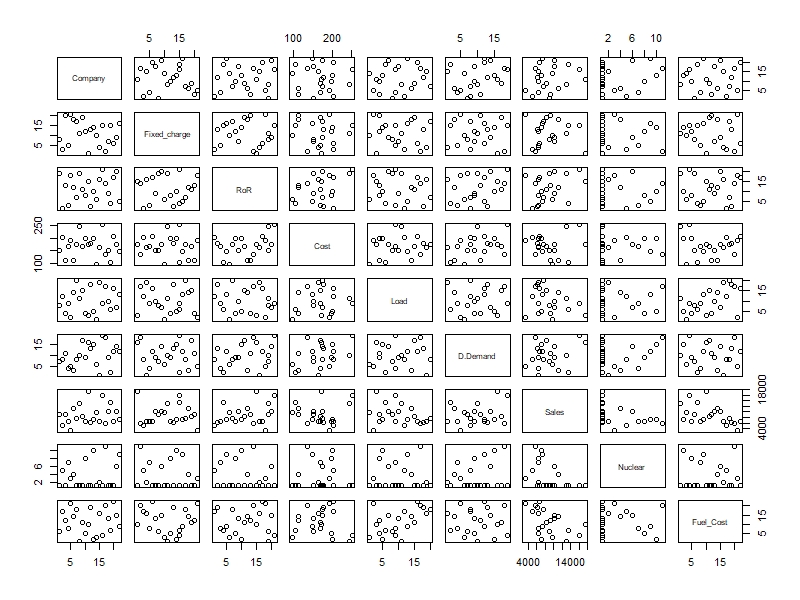
dataset$D.Demand = as.numeric(dataset$D.Demand)

dataset$Nuclear = as.numeric(dataset$Nuclear)

dataset$Fuel\_Cost = as.numeric(dataset$Fuel\_Cost)

**# Ploteo de variables en 1Vs1**

pairs(dataset)

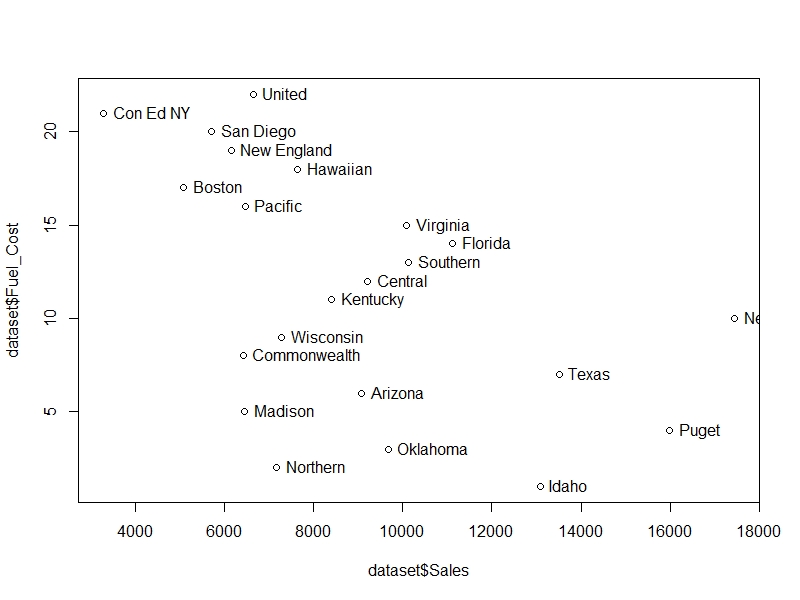
****

Del gráfico, apreciamos como se distribuyen los puntos en relación a dos variables independientes (todas las combinaciones posibles), con este gráfico podemos analizar tendencias como por ejemplo que la variable Nuclear crecer linealmente a la vez que la variable D.Demand lo hace), no queremos que entre grupos haya correlaciones fuertes, así que este es un mal indicativo visual para este par de variables cuando formen un clúster.

**# Scatter plot de dos variables independientes**

plot(dataset$Fuel\_Cost~ dataset$Sales, data = dataset)

with(dataset,text(dataset$Fuel\_Cost ~ dataset$Sales, labels=dataset$Company,pos=4))



Del gráfico, vemos como en relación a las variables “Sales” y “Fuel\_Cost” se están formando tres aglomeraciones de puntos de las ciudades, abajo a la derecha (Texas, Puget, etc) son un grupo de ciudades que venden mucho y tienen bajos costos de gasolina, arriba a la izquierda (Boston, Pacific, etc) son un grupo de ciudades que venden poco y tienen altos costos de gasolina y en el medio de ambas variables con ventas medias y costos intermedios de gasolina (Kentucky, Florida, etc).

**# Normalizando las variables independientes numéricas**

datos\_originales = dataset

dataset[,2:9] = scale(dataset[,2:9])

**# Calculando la matriz de distancias entre casos (por defecto distancia euclidea)**

matriz\_distancias = dist(dataset)

print(matriz\_distancias, digits = 3)

> print(matriz\_distancias, digits = 3)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

2 4.29

3 2.91 4.93

4 3.51 2.16 4.36

5 4.12 4.78 3.48 4.80

6 3.20 4.52 2.72 3.77 4.18

7 3.97 3.42 3.73 3.97 4.20 2.84

8 2.98 5.16 4.61 4.75 5.31 4.77 4.65

9 3.48 4.29 2.78 4.45 4.34 3.46 2.47 3.85

10 3.37 3.30 3.76 2.22 4.02 4.11 4.36 3.96 3.86

11 3.61 5.29 4.54 5.10 6.24 5.22 **5.66** 3.79 4.73 4.85

12 4.42 2.72 4.07 3.83 4.66 3.86 **1.61** 5.12 2.56 4.29 5.53

13 4.04 4.29 4.21 3.44 4.83 4.79 5.00 4.13 3.87 1.55 5.03 4.85

14 2.98 4.74 2.85 3.43 5.15 3.66 4.92 4.97 4.46 3.66 4.42 4.98 4.49

15 3.12 3.48 4.74 3.56 4.62 3.41 3.13 4.64 4.42 4.61 5.47 3.79 5.55 4.94

16 4.32 6.26 4.96 6.17 5.99 5.72 5.48 2.25 3.98 4.98 3.71 5.73 4.61 5.99 6.17

17 5.04 4.04 5.49 4.47 5.87 5.10 4.85 6.23 4.71 4.73 5.67 4.42 4.64 6.16 4.57 6.59

18 2.14 2.92 2.81 2.78 4.29 2.81 2.85 3.74 2.91 3.32 3.33 2.86 4.14 2.87 2.94 4.72 4.37

19 3.50 4.91 2.98 3.76 5.25 2.64 4.25 4.95 4.39 4.22 4.73 4.72 5.09 2.03 4.74 5.97 6.52 2.92

20 4.00 3.44 3.54 2.76 4.65 3.18 3.35 4.87 3.01 2.45 5.42 3.30 2.54 4.09 4.49 5.53 3.76 3.29 4.09

21 3.55 3.34 4.47 4.40 4.11 4.31 2.96 4.29 3.39 4.58 4.70 2.96 5.12 5.58 2.74 4.95 3.88 2.93 5.61 4.49

22 2.66 4.08 3.30 3.94 3.84 3.74 4.12 3.62 3.30 3.20 3.35 4.24 3.31 4.44 3.96 3.88 3.50 2.78 4.76 3.49 2.96

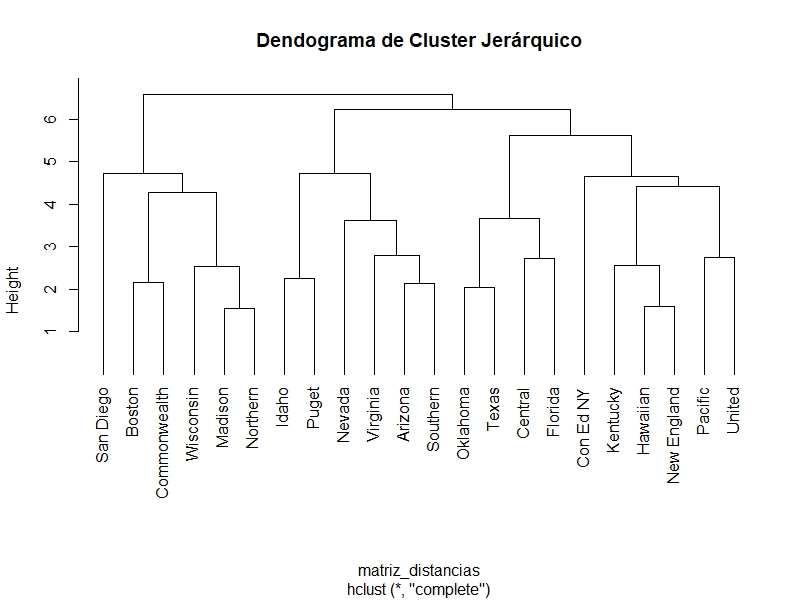
La tabla anterior muestra las distancias que hay entre las filas (ciudades en nuestro caso). De color rojo, vemos que la ciudad 7 y la ciudad 11 tienen una distancia euclidea de 5.66 (están lejos, es poco probable que estén en el mismo cluster), en cambio, de color verde, apreciamos que la ciudad 7 y la ciudad 12 tienen de distancia 1.61 (están cerca, es muy probable que estén en el mismo cluster).

**# Generando el Clúster Jerárquico**

clusters = hclust(matriz\_distancias)

**# Haciendoo un Dendograma para ver el número correcto de clúster a elegir**

plot(clusters,labels=dataset$Company,main='Dendograma de Cluster Jerárquico',hang=-1)



Del gráfico, vemos como se han ido formado los clústeres de abajo hacia arriba (modo aglomerativo). Si analizamos el gráfico por la izquierda, vemos que primero se forman los clusters de las ciudades Madison y Northern, luego este cluster nuevo se junta con la ciudad Wisconsin y así, sucesivamente hasta forman el macro clúster a la distancia (Height) 6,5 aproximadamente que incluya a todas las ciudades. Por obvias razones no queremos tener un solo clúster, eso no tiene sentido, las técnicas de clustering se usan con el objetivo de generar acciones de mercadeo, servicio, etc según las dierencias encontradas en las características de los individuos (casos), como regla práctica es bueno tener entre 3 a 5 clústeres como máximo.

**# Generación de grupos con 3 clústeres (a la altura 5 aproximada del dendrograma anterior)**

miembros = cutree(clusters,3)

**# Analizando promedios de variables independientes por grupos**

aggregate(datos\_originales[,2:9],list(miembros),mean)

> aggregate(datos\_originales[,2:9],list(miembros),mean)

Group.1 Fixed\_charge RoR Cost Load D.Demand Sales Nuclear Fuel\_Cost

1 1 8.500000 17.166667 190.8333 8.0 12.5 12637.333 1.833333 8.166667

2 2 8.666667 6.833333 175.0000 10.0 11.0 6357.667 8.333333 10.166667

3 3 14.300000 10.600000 150.5000 11.5 7.8 8213.900 1.600000 14.300000

De la tabla, apreciamos cuales son los valores promedio de cada una de las variables para nuestros tres grupos, esto puede ayudar a una empresa a clasificar en general a sus nuevos clientes en alguno de los tres grupos para tomas posteriores de decisiones en base a dichos valores esperados.

**# Visualizar los clústeres para las variables "Costo" y "Ventas"**

#install.packages("cluster")

x = as.data.frame(dataset$Cost)

x[,2] = dataset$Sales

library(cluster)

clusplot(x,

miembros,

lines = 0,

shade = TRUE,

color = TRUE,

labels = 2,

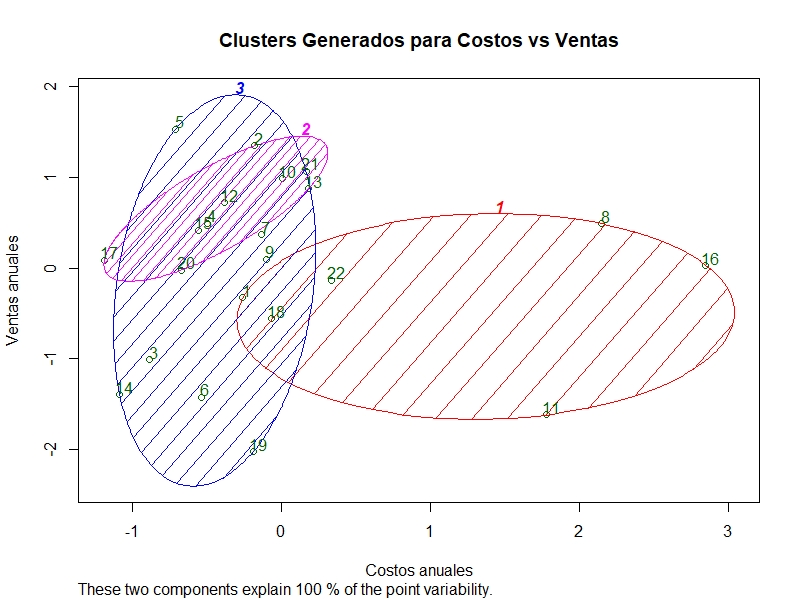
plotchar = FALSE,

span = TRUE,

main = "Clusters Generados para Costos vs Ventas",

xlab = "Costos anuales",

ylab = "Ventas anuales")

****

Del gráfico, vemos que se generan los 3 clústeres en cada región de elipse, los números en rojo corresponde al número de clúster y los números en verde a los elementos (ciudades) que pertenecen a cada uno de ellos. Vemos que la variabilidad del grupo 2 es la menor de todas, seguida por el grupo 3 y finalmente el grupo 1, esto debido al tamaño de las elipses en las que se encuentran inmersos los datos.

**Objetivo 3:**

Nuestro siguiente modelo de clasificación no supervisado es el de Clustering con Densidad de Puntos, este modelo tiene la ventaja de generar grupos de diferentes siluetas (no solo elipses o polígonos cerrados) y no ser sensible a los valores atípicos, la cantidad de clústeres generados se genera automáticamente según la densidad de puntos y distancias máximas indicadas entre ellos.

**# Clustering basado en Densidad de Puntos**

**# Importar los datos de la librería "factoextra"**

library(fpc)

library(factoextra)

data("multishapes", package = "factoextra")

head(multishapes)

> head(multishapes)

x y shape

1 -0.8037393 -0.8530526 1

2 0.8528507 0.3676184 1

3 0.9271795 -0.2749024 1

4 -0.7526261 -0.5115652 1

5 0.7068462 0.8106792 1

6 1.0346985 0.3946550 1

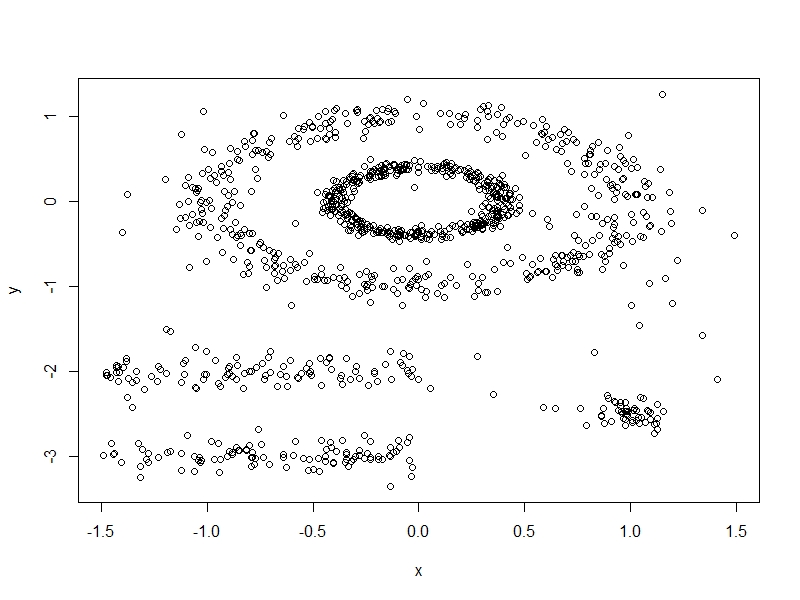
Apreciamos que el dataset tiene 3 variables, las dos primeras son coordenadas numéricas (x e y) y la tercera es un factor (shape) que debemos eliminar porque los clústeres no trabajan con este tipo de variables.

**# Eliminamos la tercera columna**

dataset <- multishapes[,1:2]

**# Analizando la distribución de los puntos en las variables independientes**

plot(dataset)

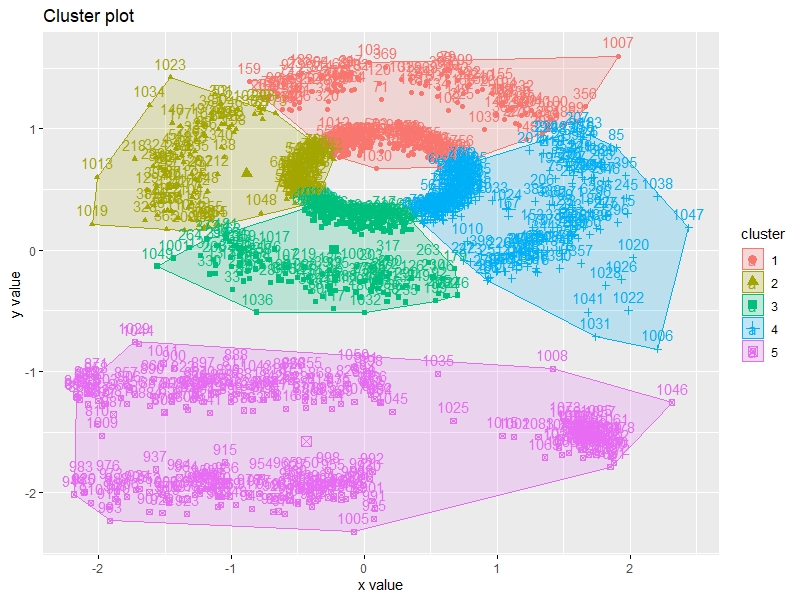


Del gráfico, apreciamos que existen dos circunferencias prácticamente concéntricas, podríamos estar tentados en hacer un solo clúster con ambas, pero lo más correcto es notar que las distancias entre puntos contemplan un radio diferente y que la densidad de puntos de la circunferencia interior es mucho mayor a la de la circunferencia exterior, por lo tanto, debería de haber dos clústeres, uno para la circunferencia de adentro y otro para la de afuera. En el caso de los demás puntos, apreciamos que abajo se forman tres grupos que si son separables en clústeres poligonales simples.

**# Generando el Cluster con K-Means (para ver el error en los clústeres formados)**

km <- kmeans(dataset, 5)

fviz\_cluster(km, data= dataset)



Del gráfico, apreciamos que con la técnica K-Means estamos generando los clústeres erróneamente, estos se forman como polígonos y para todos los casos se están aglomerando espacios vacíos demasiado amplios que aumentarán la variabilidad intra clústeres innecesariamente.

**# Generando el Cluster con Densidad de puntos (para ver la mejora de clústeres)**

dsFit <- dbscan(dataset, eps = 0.15, MinPts = 5)

fviz\_cluster(dsFit, dataset, geom = "point")

> dsFit

dbscan Pts=1100 MinPts=5 eps=0.15

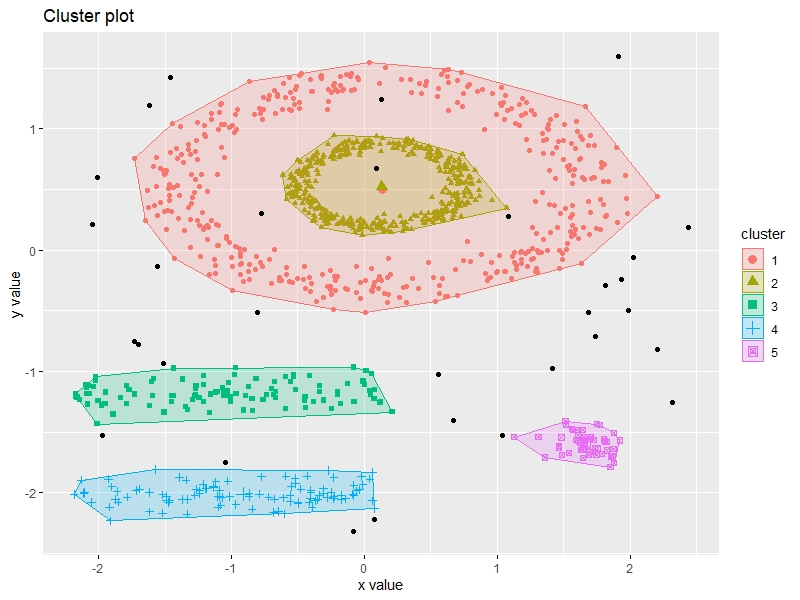
0 1 2 3 4 5

border 31 24 1 5 7 1

seed 0 386 404 99 92 50

total 31 410 405 104 99 51

De la tabla, apreciamos los puntos por cada clúster, la columna 0 contiene los puntos atípicos que quedan fuera de los clústeres (31 puntos), de la columna 1 hasta la columna 5 se muestran los puntos de los 5 clústeres formados, donde la primera fila (border) contiene los puntos del borde y la segunda fila (seed) contiene los puntos semilla. De manera práctica, el algoritmo decide formar 5 clústeres de acuerdo a la cantidad mínima de puntos indicada (MinPts) y el radio indicado (eps).



Del gráfico, vemos como ahora los clústers toman en cuenta mucho mejor las densidades de puntos en cada región del plano, los puntos negros son los que quedaron sin clasificar (outliers) por estar muy lejos (ser menos de 5 puntos en región con radio de 0.15).

**Objetivo 4:**

A continuación, haremos nuestro modelo de Análisis de Componentes Principales con el dataset “IRIS” que viene incluido dentro de los paquetes de RStudio, este dataset tiene 4 variables independientes numéricas: sepal lenght (longitud del sépalo), sepal width (ancho del sépalo), petal lenght (longitud de pétalo), petal with (ancho del pétalo) y una variable dependiente tipo factor species (especies)

**# Análisis de Componentes Principales**

**# Importar el dataset**

data("iris")

str(iris)

summary(iris)

> summary(iris)

Sepal.Length Sepal.Width Petal.Length Petal.Width Species

Min. :4.300 Min. :2.000 Min. :1.000 Min. :0.100 setosa :50

1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600 1st Qu.:0.300 versicolor:50

Median :5.800 Median :3.000 Median :4.350 Median :1.300 virginica :50

Mean :5.843 Mean :3.057 Mean :3.758 Mean :1.199

3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100 3rd Qu.:1.800

Max. :7.900 Max. :4.400 Max. :6.900 Max. :2.500

De la tabla, observamos los principales estadísticos (media, mediana, min, max, etc) de cada una de las variables independientes y cuantas clases hay por especie (50 por cada especie)

**# Dividir los datos en conjunto de entrenamiento y conjunto de test**

# install.packages("caTools")

library(caTools)

set.seed(123)

split = sample.split(iris$Species, SplitRatio = 0.8)

training\_set = subset(iris, split == TRUE)

testing\_set = subset(iris, split == FALSE)

**# Escalar las variables independientes**

training\_set[,-5] = scale(training\_set[,-5])

testing\_set[,-5] = scale(testing\_set[,-5])

**# Analizar las correlaciones de las variables independientes**

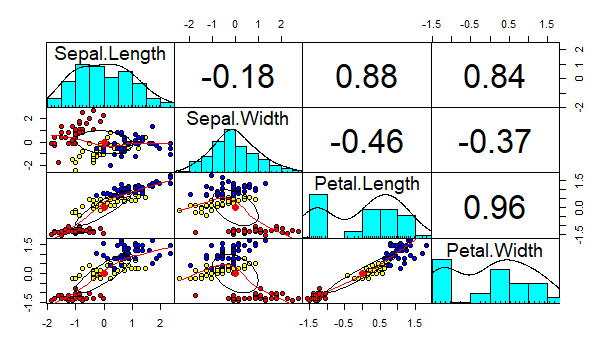
library(psych)

pairs.panels(training\_set[,-5],

gap = 0,

bg = c("red", "yellow", "blue")[training\_set$Species],

pch=21)

****

Del gráfico, apreciamos en la diagonal principal como se distribuyen las variables independientes, debajo de dicha diagonal vemos un ploteo bidimensional para cada combinación de las variables en cuestión y por encima de la diagonal, apreciamos la correlación entre cada par de variables, siendo las correlaciones más cercanas a 1 o -1 indeseadas puesto que eso indica multicolinealidad que le restaría al % de la varianza explicada de la variable dependiente. Cuando existe este tipo de problema es muy útil utilizar una técnica como el Análisis de Componentes principales, ya que así te quedarás con las proyecciones más explicativas y eliminarás las variables correlacionadas.

**# Realizando el Análisis de Componentes Principales**

pc = prcomp(training\_set[,-5])

summary(pc)

> pc

Standard deviations (1, .., p=4):

[1] 1.7219490 0.9340160 0.3768286 0.1431988

Rotation (n x k) = (4 x 4):

PC1 PC2 PC3 PC4

Sepal.Length 0.5223915 -0.36038936 0.7330731 0.2446026

Sepal.Width -0.2875942 -0.92615356 -0.2075963 -0.1281911

Petal.Length 0.5755887 -0.03224608 -0.1585619 -0.8015710

Petal.Width 0.5595483 -0.10639216 -0.6279855 0.5303024

De la tabla, apreciamos las desviaciones estándar de las 4 componentes principales, además, vemos como se han de construir dichas componentes principales con los coeficientes asignados a las variables independiente, por ejemplo:

**PC1 = 0.5223915(Sepal.Length) - 0.2875942(Sepal.Width) + 0.5755887(Petal.Length) +**

**0.5595483(Petal.Width)**

> summary(pc)

Importance of components:

PC1 PC2 PC3 PC4

Standard deviation 1.7219 0.9340 0.3768 0.14320

Proportion of Variance 0.7413 0.2181 0.0355 0.00513

Cumulative Proportion 0.7413 0.9594 0.9949 1.00000

De la tabla, vemos las desviaciones estándar de cada uno de las componentes principales, además, la proporción de varianza explicada para el modelo de cada una de ellas. La PC1 explica el 74.13% de la varianza y la PC2 explica el 21.81% de la varianza (ambos explican el 95.94% de la misma), por lo tanto, será suficiente quedarnos con dos componentes principales para el modelo.

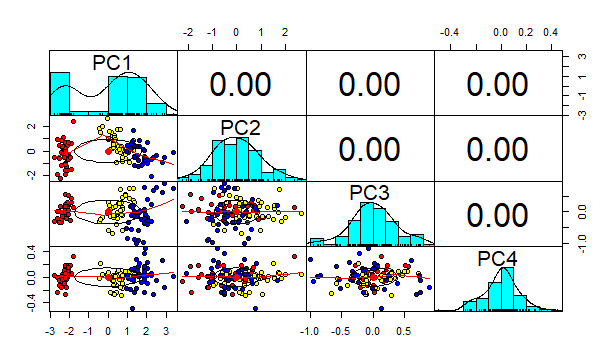
**# Analizar las correlaciones de las componentes principales**

pairs.panels(pc$x,

gap=0,

bg = c("red", "yellow", "blue")[training\_set$Species],

pch=21)

****

Del gráfico, ahora apreciamos que las correlaciones (valores pon encima de la diagonal principal) son cero, evitando con esto la multicolinealidad que restaba valor al modelo.

**# Actualización de conjuntos de entrenamiento y testeo con componentes principales**

trg <- predict(pc, training\_set)

trg <- data.frame(trg, training\_set[5])

tst <- predict(pc, testing\_set)

tst <- data.frame(tst, testing\_set[5])

> head(trg)

PC1 PC2 PC3 PC4 Species

1 -2.257701 -0.5208201 0.15200375 0.022400597 setosa

2 -2.046510 0.6626452 0.21672500 0.113639433 setosa

3 -2.343956 0.3154246 -0.05304469 0.038366010 setosa

6 -2.093041 -1.5628803 0.03659428 -0.007559562 setosa

7 -2.438912 -0.0936785 -0.33204478 -0.029154461 setosa

9 -2.300348 1.1035960 -0.18581427 -0.006746125 setosa

De la tabla, apreciamos los nuevos valores de nuestro conjunto de entrenamiento, ahora ya como componentes principales totalmente escalonadas. Vemos que en valor absoluto PC1 tiene más altos valores que PC2, PC2 más altos que PC3 y PC3 que PC4 (esto se corresponde con la explicación de varianza de cada uno de las componentes principales, PC1 y PC2 explican más).

**# Ajustar modelo Logístico Multinomial (elegido) con las dos primeras componentes**

library(nnet)

trg$Species <- relevel(trg$Species, ref = "setosa")

classifier <- multinom(Species~PC1+PC2, data = trg)

summary(classifier)

**# Predicción de resultados con el conjunto de testing**

p1 <- predict(classifier, tst)

> p1

[1] setosa setosa setosa setosa setosa setosa setosa

[8] setosa setosa setosa versicolor versicolor versicolor versicolor

[15] versicolor virginica versicolor versicolor versicolor versicolor virginica

[22] virginica versicolor virginica virginica virginica virginica virginica

[29] versicolor virginica

Levels: setosa versicolor virginica

**# Realizando la matriz de confusión**

cm <- table(p1, tst$Species)

> cm

p1 setosa versicolor virginica

setosa 10 0 0

versicolor 0 9 2

virginica 0 1 8

**# Hallando el ratio de predicción**

ratio =sum(diag(cm))/sum(cm)

> ratio

[1] 0.9

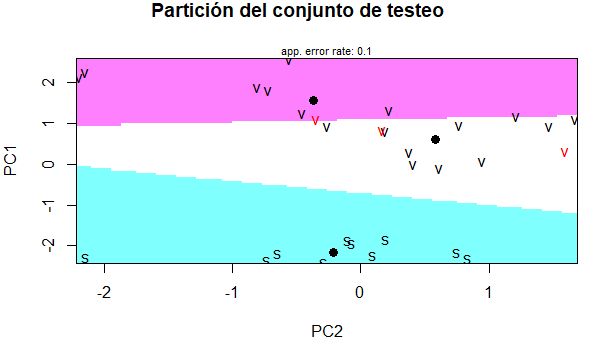
Apreciamos que el ratio de clasificación obtenido fue del 90% correctos.

**# Haciendo gráficos de parcela para el conjunto de testeo**

library(klaR)

partimat(Species ~ PC1 + PC2, data = tst, method = "lda",

main = "Partición del conjunto de testeo")



Del gráfico, vemos como se han clasificado correctamente casi todos los puntos, solo hay 3 puntos (ver matriz de confusión) de color rojo que han sido puestos en las zonas de clasificación incorrectas.

**Objetivo 5:**

En este apartado, realizaremos un Análisis de Componentes Principales para el caso en que los separadores lineales no son ideales, ya que la separación de datos requiere curvas distintas a una recta. Usaremos el dataset “Social\_Network\_Ads”, el cual contiene los datos de unos clientes sondeados en una red social con las variables independientes Gender (Género), Age (Edad) y EstimatedSalary (Salario Estimado) y la variable dependiente Purchased (Compra).

**# Kernel ACP**

**# Importar el dataset**

dataset = read.csv('Social\_Network\_Ads.csv')

dataset = dataset[, 3:5]

**# Dividir los datos en conjunto de entrenamiento y conjunto de test**

# install.packages("caTools")

library(caTools)

set.seed(123)

split = sample.split(dataset$Purchased, SplitRatio = 0.75)

training\_set = subset(dataset, split == TRUE)

testing\_set = subset(dataset, split == FALSE)

**# Escalado de valores**

training\_set[,1:2] = scale(training\_set[,1:2])

testing\_set[,1:2] = scale(testing\_set[,1:2])

**# Aplicar Kernel ACP**

#install.packages("kernlab")

library(kernlab)

kpca = kpca(~., data = training\_set[, -3],

kernel = "rbfdot", features = 2)

training\_set\_pca = as.data.frame(predict(kpca, training\_set))

training\_set\_pca$Purchased = training\_set$Purchased

testing\_set\_pca = as.data.frame(predict(kpca, testing\_set))

testing\_set\_pca$Purchased = testing\_set$Purchased

**# Ajustar el modelo de regresión logística con el conjunto de entrenamiento**

classifier = glm(formula = Purchased ~ .,

data = training\_set\_pca,

family = binomial)

**# Predicción de los resultados con el conjunto de testing**

prob\_pred = predict(classifier, type = "response",

newdata = testing\_set\_pca[,-3])

> prob\_pred

2 4 5 9 12 18 19

0.017229968 0.007244058 0.005928970 0.002746752 0.004943245 0.226564385 0.291928521

20 22 29 32 34 35 38

0.421529835 0.575472557 0.006618943 0.414534037 0.005555742 0.045159134 0.010018894

y\_pred = ifelse(prob\_pred> 0.5, 1, 0)

> y\_pred

2 4 5 9 12 18 19 20 22 29 32 34 35 38 45 46 48 52 66 69 74

0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

75 82 84 85 86 87 89 103 104 107 108 109 117 124 126 127 131 134 139 148 154

0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

**# Crear la matriz de confusión**

cm = table(testing\_set\_pca[, 3], y\_pred)

> cm

y\_pred

0 1

0 57 7

1 10 26

**# Ratio de predicción**

ratio = sum(diag(cm))/sum(cm)

> ratio

[1] 0.83

Con nuestro modelo de kernel ACP hemos logrado categorizar correctamente al 83% de los datos de la variable dependiente Purchased.

**# Visualización del conjunto de testing**

set = testing\_set\_pca

X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.05)

X2 = seq(min(set[, 2]) - 1, max(set[, 2]) + 1, by = 0.05)

grid\_set = expand.grid(X1, X2)

colnames(grid\_set) = c('V1', 'V2')

prob\_set = predict(classifier, type = 'response', newdata = grid\_set)

y\_grid = ifelse(prob\_set > 0.5, 1, 0)

plot(set[, -3],

main = 'Clasificación (Conjunto de Testing)',

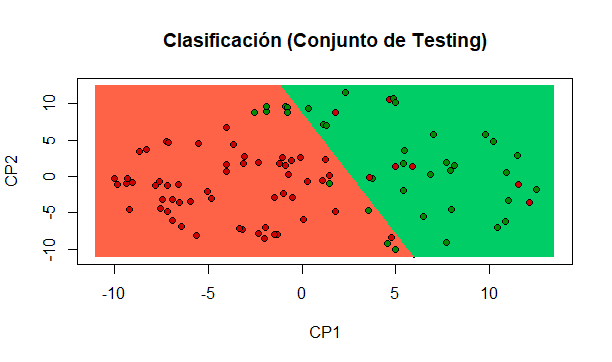
xlab = 'CP1', ylab = 'CP2',

xlim = range(X1), ylim = range(X2))

contour(X1, X2, matrix(as.numeric(y\_grid), length(X1), length(X2)), add = TRUE)

points(grid\_set, pch = '.', col = ifelse(y\_grid == 1, 'springgreen3', 'tomato'))

points(set, pch = 21, bg = ifelse(set[, 3] == 1, 'green4', 'red3'))



Del gráfico, apreciamos como se distribuyen las zonas de predicción (rojo = no compra y verde = si compra), los puntos correctamente predichos deben tener el mismo color que la zona en la que están, en caso contrario, los puntos han sido mal predichos.

**Objetivo 6:**

Ahora realizaremos nuestro modelo de reducción de dimensiones con Análisis de Discrimimante Lineal para el cual utilizaremos el dataset “Wine”, el cual muestra características de vinos como variables independientes y el tipo de cliente como variable respuesta, esta será la variable que deseamos predecir (clasificar por ser factor) luego de reducir las dimensiones (cantidad de variables independientes).

**# Análisis de Discriminante Lineal**

**# Importar el dataset**

dataset = read.csv('Wine.csv')

**# Dividir los datos en conjunto de entrenamiento y conjunto de test**

# install.packages("caTools")

library(caTools)

set.seed(1234)

split = sample.split(dataset$Customer\_Segment, SplitRatio = 0.8)

training\_set = subset(dataset, split == TRUE)

testing\_set = subset(dataset, split == FALSE)

# Escalado de valores

training\_set[,-14] = scale(training\_set[,-14])

testing\_set[,-14] = scale(testing\_set[,-14])

**# Aplicar LDA a los conjuntos de entrenamiento y de testeo**

library(MASS)

lda = lda(formula = Customer\_Segment ~ ., data = training\_set)

training\_set = as.data.frame(predict(lda, training\_set))

#Eliminamos las variables innecesarias en el conjunto de entrenamiento

training\_set = training\_set[, c(5, 6, 1)]

testing\_set = as.data.frame(predict(lda, testing\_set))

#Eliminamos las variables innecesarias en el conjunto de testeo

testing\_set = testing\_set[, c(5, 6, 1)]

> lda

Call:

lda(Customer\_Segment ~ ., data = training\_set)

**# Muestra la proporción de los 3 grupos de la variable dependiente**

Prior probabilities of groups:

1 2 3

0.3309859 0.4014085 0.2676056

**# Muestra las medias de las variables independientes para cada uno de los 3 grupos**

Group means:

Alcohol Malic\_Acid Ash Ash\_Alcanity Magnesium Total\_Phenols

1 0.9201051 -0.3054242 0.3317493 -0.7911875 0.40675516 0.85024073

2 -0.9052678 -0.4106957 -0.5039708 0.2191929 -0.32865624 -0.04152766

3 0.2198769 0.9938051 0.3456347 0.6497846 -0.01010755 -0.98932204

Flavanoids Nonflavanoid\_Phenols Proanthocyanins Color\_Intensity Hue

1 0.97623806 -0.5858085 0.51874322 0.1810301 0.4744574

2 0.05248446 0.0213819 0.09848304 -0.8523447 0.4201366

3 -1.28617902 0.6924798 -0.78932802 1.0546113 -1.2170339

OD280 Proline

1 0.8049077 1.1822726

2 0.2098551 -0.6847968

3 -1.3103264 -0.4350892

**# Muestra los coeficientes de las dos funciones discriminantes (escaladas)**

**# Siendo la primera función discriminante (LD1) una combinación lineal de los coeficientes:**

**# LD1 = - 0.30748036(Alcohol) + 0.23338178(Malic\_Acid) - 0.15960249(Ash) + …**

**# Muestra la probabilidad de pertenecer a alguno de los 3 grupos**

Coefficients of linear discriminants:

LD1 LD2

Alcohol -0.30748036 0.79536283

Malic\_Acid 0.23338178 0.46852468

Ash -0.15960249 0.60544816

Ash\_Alcanity 0.47491226 -0.52036860

Magnesium -0.05647592 0.02221126

Total\_Phenols 0.56827773 -0.19378015

Flavanoids -2.02732881 -0.23629702

Nonflavanoid\_Phenols -0.22716366 -0.22188397

Proanthocyanins 0.13247233 -0.19848658

Color\_Intensity 0.79619681 0.66394133

Hue -0.12489131 -0.38805096

OD280 -0.79839168 0.11713304

Proline -0.78827401 0.80064140

**# Muestra el porcentaje de separación logrado por las funciones discriminantes**

**# LD1 logra separar el 68.22% y LD2 logra separar el 31.78% en los grupos.**

Proportion of trace:

LD1 LD2

0.6822 0.3178

**# Ajustar el modelo de SVM (modelo elegido) con el conjunto de entrenamiento.**

library(e1071)

classifier = svm(formula = class ~ .,

data = training\_set,

type = "C-classification",

kernel = "linear")

**# Predicción de los resultados con el conjunto de testing**

y\_pred = predict(classifier, newdata = testing\_set[,-3])

> y\_pred

5 14 16 26 28 29 36 39 40 50 53 58 60 61 72 81 86 90 92 113 116

1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2

117 120 122 123 124 131 135 137 140 142 149 154 156 158 169

2 2 2 2 2 3 3 3 3 3 3 3 3 3 3

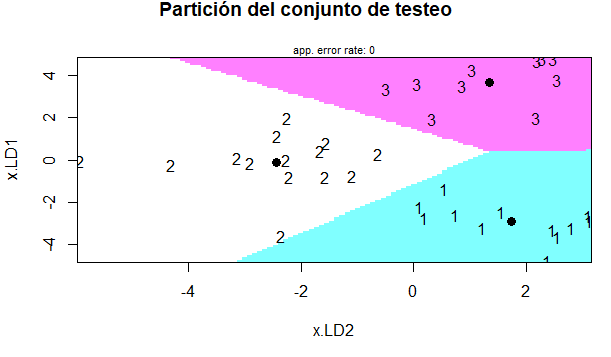
Levels: 1 2 3

**# Haciendo gráficos de parcela para el conjunto de testeo**

library(klaR)

partimat(class ~ ., data = testing\_set, method = "lda",

main = "Partición del conjunto de testeo")



Del gráfico, apreciamos en los ejes X e Y a las funciones discriminantes LD2 y LD1 respectivamente, las zonas coloreadas separar los grupos de manera disjunta y los puntos pintados representan los datos clasificados de cada grupo. No vemos a simple vista que exista algún dato mal clasificado por no estar en una zona equivocada.

**# Crear la matriz de confusión**

cm = table(testing\_set[, 3], y\_pred)

> cm

y\_pred

1 2 3

1 12 0 0

2 0 14 0

3 0 0 10

**# Calculando el ratio de predicción**

ratio = sum(diag(cm))/sum(cm)

> ratio

[1] 1

Apreciamos que el ratio obtenido fue del 100% de los datos correctamente clasificados.

**Conclusión:**

En el laboratorio descubrimos como reducir la cantidad de variables para poder generar un mayor control sobre unas proyecciones que expliquen la mayor cantidad de variabilidad posible de los modelos y vimos las técnicas de clustering (clasificación de modelos no supervisados) que sirven para generar grupos basados en características comunes y separar dichos grupos cuando las características difieren mucho, estas técnicas son ampliamente utilizadas en segmentación de clientes para generar diversas formas de mercadeo y servicio al cliente.