

# **Aprenentatge computacional**

## **PAC2**

Luis Enrique Arribas Zapater 17745245D

## Exercici 1

a) Para construir el modelo importamos los ficheros train.csv y test.csv.

```
>train
ALCOHOL MALIC MAGNESIUM PHENOLS FLAVANOIDS COLOR CLASS
1 13.16 2.36 101 2.80 3.24 5.68 0
2 11.65 1.67 88 1.92 1.61 2.60 1
3 12.25 1.73 80 1.65 2.03 3.40 1
4 14.83 1.64 97 2.80 2.98 5.20 0
5 13.05 1.73 92 2.72 3.27 7.20 0
6 12.37 1.63 88 2.22 2.45 2.12 1
7 13.87 1.90 107 2.95 2.97 4.50 0
8 13.11 1.01 78 2.98 3.18 5.30 1
```

```
> test
ALCOHOL MALIC MAGNESIUM PHENOLS FLAVANOIDS COLOR CLASS
1 14.22 3.99 128 3.00 3.04 5.1 0
2 11.87 4.31 82 2.86 3.03 2.8 1
```

Vemos que los datos son numéricos y dispersos por lo que aplicamos estandarización restando la media de cada variable y dividiendo por la desviación típica.

```
> test <- scale(test)
> test
ALCOHOL MALIC MAGNESIUM PHENOLS FLAVANOIDS COLOR
CLASS
[1,] 0.7071068 -0.7071068 0.7071068 0.7071068 0.7071068 0.7071068
-0.7071068
[2,] -0.7071068 0.7071068 -0.7071068 -0.7071068 -0.7071068 -0.7071068
0.7071068
attr(,"scaled:center")
ALCOHOL MALIC MAGNESIUM PHENOLS FLAVANOIDS COLOR
CLASS
13.045 4.150 105.000 2.930 3.035 3.950
0.500
attr(,"scaled:scale")

> train <- scale(train)
>train
ALCOHOL MALIC MAGNESIUM PHENOLS FLAVANOIDS
COLOR CLASS
```

```

1.661700936 0.226274170 32.526911935 0.098994949 0.007071068
1.626345597 0.707106781
> train
      ALCOHOL      MALIC      MAGNESIUM      PHENOLS FLAVANOIDS      COLOR
CLASS
[1,] 0.12448676 1.75648455 0.96258595 0.5820066 0.8440076 0.6929745
-0.9354143
[2,] -1.39450324 -0.10451252 -0.33753014 -1.1541486 -1.7826890 -1.1158064
0.9354143
[3,] -0.79093105 0.05731332 -1.13760158 -1.6868326 -1.1058715 -0.6459932
0.9354143
[4,] 1.80442935 -0.18542543 0.56255023 0.5820066 0.4250253 0.4110866
-0.9354143
[5,] 0.01383186 0.05731332 0.06250558 0.4241743 0.8923517 1.5856196
-0.9354143
[6,] -0.67021661 -0.21239640 -0.33753014 -0.5622775 -0.4290540 -1.3976943
0.9354143
[7,] 0.83871385 0.51581984 1.56263953 0.8779421 0.4089106 0.0000000
-0.9354143
[8,] 0.07418908 -1.88459667 -1.33761944 0.9371292 0.7473193 0.4698132
0.9354143
attr(,"scaled:center")
      ALCOHOL      MALIC      MAGNESIUM      PHENOLS FLAVANOIDS      COLOR
CLASS
13.03625 1.70875 91.37500 2.50500 2.71625 4.50000
0.50000
attr(,"scaled:scale")
      ALCOHOL      MALIC      MAGNESIUM      PHENOLS FLAVANOIDS      COLOR
CLASS
0.9940816 0.3707690 9.9991071 0.5068671 0.6205513 1.7028044
0.5345225
>

```

De estos dos conjuntos ya normalizados haremos los cuatro conjuntos de datos siguientes:

ground\_truth= compuesto por 2 filas y la columna 7 (CLASS) del fichero test.csv.

```

> ground_truht
[1] 0 1

```

test= compuesto por el conjunto de prueba al que hemos quitado la variable CLASS

```

> test
      ALCOHOL      MALIC      MAGNESIUM      PHENOLS FLAVANOIDS      COLOR
[1,] 0.7071068 -0.7071068 0.7071068 0.7071068 0.7071068 0.7071068
[2,] -0.7071068 0.7071068 -0.7071068 -0.7071068 -0.7071068 -0.7071068

```

label= etiquetas de clasificación del training set. Corresponde a la columna CLASS de train.csv

```
> labels
[1] 0 1 1 0 0 1 0 1
```

Y train, que es el conjunto de entrenamiento ya preparado

```
> train
      ALCOHOL      MALIC      MAGNESIUM      PHENOLS FLAVANOIDS      COLOR
[1,]  0.12448676  1.75648455  0.96258595  0.5820066  0.8440076  0.6929745
[2,] -1.39450324 -0.10451252 -0.33753014 -1.1541486 -1.7826890 -1.1158064
[3,] -0.79093105  0.05731332 -1.13760158 -1.6868326 -1.1058715 -0.6459932
[4,]  1.80442935 -0.18542543  0.56255023  0.5820066  0.4250253  0.4110866
[5,]  0.01383186  0.05731332  0.06250558  0.4241743  0.8923517  1.5856196
[6,] -0.67021661 -0.21239640 -0.33753014 -0.5622775 -0.4290540 -1.3976943
[7,]  0.83871385  0.51581984  1.56263953  0.8779421  0.4089106  0.0000000
[8,]  0.07418908 -1.88459667 -1.33761944  0.9371292  0.7473193  0.4698132
>
```

Calculamos la matriz de distancias

```
> distance.matrix

      [1]      [2]
1 2.632731 3.648780
2 5.931878 4.038002
3 5.813856 3.924412
4 3.074636 4.588134
5 3.674779 4.326101
6 5.065638 3.084246
7 2.436503 4.021007
8 4.832219 3.650289
```

Para k=1 marcamos en rojo la distancia menor a cada uno de los miembros del training set, que son 6 y 7 respectivamente y corresponden a la categoría 0 y 1.

```
ground_truth
result 0 1
      0 1 0
      1 0 1
```

Cruzando la predicción con ground\_truth tenemos una **precisión del 100%** para k=1

```
> distance.matrix
```

```

      [1]      [2]
1 2.632731 3.648780
2 5.931878 4.038002
3 5.813856 3.924412
4 3.074636 4.588134
5 3.674779 4.326101
6 5.065638 3.084246
7 2.436503 4.021007
8 4.832219 3.650289

```

Para  $k=3$  vemos en verde los 3 valores más próximos a cada uno de los miembros del conjunto de prueba.

[1] tiene como distancias más cortas las correspondientes a las filas del training set {1, 4, 7} que corresponden en las categorías {0, 0, 0} por lo que lo clasificamos como 0

[2] tiene como distancias más cortas las correspondientes a las filas del training set {1, 6, 8} que corresponden en las categorías {1, 1, 1} por lo que lo clasificamos como 1

```

ground_truth
result 0 1
      0 1 0
      1 0 1

```

Cruzando la predicción con ground\_truth tenemos una **precisión del 100%** para  $k=3$

b) El modelo de clasificación supervisada basada en k-Means parte de la idea de clasificar mediante k-means las dos clases de el training set, obteniendo dos centroides por clase (4 centroides en total), de manera que la clasificación del test set mediante knn no se hará midiendo la distancia euclídea desde cada elemento del test set hasta cada elemento del training set, sino desde el primero a cada uno de los cuatro centroides obtenidos mediante C-means.

Para la construcción del modelo dividimos el conjunto de datos “train” en dos nuevos conjuntos en función del valor de su variable class y estandarizamos los datos.

```
> c0
  Z.ALCOHOL  Z.COLOR Z.FLAVANOIDS Z.MAGNESIUM  Z.MALIC Z.PHENOLS
1 0.12448676 0.6929745  0.8440076  0.96258595  1.75648455 0.5820066
4 1.80442935 0.4110866  0.4250253  0.56255023 -0.18542543 0.5820066
5 0.01383186 1.5856196  0.8923517  0.06250558  0.05731332 0.4241743
7 0.83871385 0.0000000  0.4089106  1.56263953  0.51581984 0.8779421

> c1
  Z.ALCOHOL  Z.COLOR Z.FLAVANOIDS Z.MAGNESIUM  Z.MALIC Z.PHENOLS
2 -1.39450324 -1.1158064  -1.7826890  -0.3375301 -0.10451252 -1.1541486
3 -0.79093105 -0.6459932  -1.1058715  -1.1376016  0.05731332 -1.6868326
6 -0.67021661 -1.3976943  -0.4290540  -0.3375301 -0.21239640 -0.5622775
8  0.07418908  0.4698132   0.7473193  -1.3376194 -1.88459667  0.9371292
```

Aplicamos k-means a ambos conjuntos y obtenemos la siguiente clasificación:

```
Cluster means:
  Z.ALCOHOL  Z.COLOR Z.FLAVANOIDS Z.MAGNESIUM  Z.MALIC Z.PHENOLS
1 0.4816003 0.3464872  0.6264591  1.2626127  1.13615219 0.7299743
2 0.9091306 0.9983531  0.6586885  0.3125279 -0.06405606 0.5030904
```

Clustering vector:

```
1 4 5 7
1 2 2 1
```

Within cluster sum of squares by cluster:

```
[1] 1.583267 2.569019
(between_SS / total_SS = 42.0 %)
```

Cluster means:

```
  Z.ALCOHOL  Z.COLOR Z.FLAVANOIDS Z.MAGNESIUM  Z.MALIC Z.PHENOLS
1 -0.95188363 -1.0531646  -1.1058715  -0.6042206 -0.08653187 -1.1344196
2  0.07418908  0.4698132   0.7473193  -1.3376194 -1.88459667  0.9371292
```

Clustering vector:

```
2 3 6 8
```

1 1 1 2

Within cluster sum of squares by cluster:

```
[1] 2.602227 0.000000
(between_SS / total_SS = 81.1 %)
```

Por tanto el conjunto de datos que utilizaremos como training set será el siguiente conjunto de centroides.

```
> centroids
  Z.ALCOHOL  Z.COLOR Z.FLAVANOIDS Z.MAGNESIUM  Z.MALIC  Z.PHENOLS CLASS
1 0.48160030 0.3464872 0.6264591 1.2626127 1.13615219 0.7299743 0
2 0.90913060 0.9983531 0.6586885 0.3125279 -0.06405606 0.5030904 0
3 0.95188363 -1.0531646 -1.1058715 -0.6042206 -0.08653187 -1.1344196 1
4 0.07418908 0.4698132 0.7473193 -1.3376194 -1.88459667 0.9371292 1
```

Ahora separamos la variable CLASS y elaboramos la matriz de distancias

```
      [5]      [6]
1 4.460282 5.952358
2 3.120707 5.146371
3 3.992652 3.585256
4 2.660603 4.234254
```

Para k=1 tenemos los centroides 4 y 3 respectivamente que corresponden ambos a la clase 1.

Cruzando con ground\_truth tenemos:

```
ground_truht
result 0 1
      0 1 0
      1 0 1
```

Donde vemos que la **precisión es del 100%**

Para k=3

```
      [5]      [6]
1 4.460282 5.952358
2 3.120707 5.146371
3 3.992652 3.585256
4 2.660603 4.234254
```

Para k=3 vemos en verde los 3 valores más próximos a cada uno de los miembros del conjunto de prueba.

[5] tiene como distancias más cortas las correspondientes a las filas del training set {2,3,4} que corresponden en las categorías {0, 1, 1} por lo que lo clasificamos como 1

[6] tiene como distancias más cortas las correspondientes a las filas del training set {2,3,4} que corresponden en las categorías {0, 1, 1} por lo que lo clasificamos como 1

```
ground_truth
result 0 1
      0 1 0
      1 0 1
```

Cruzando la predicción con ground\_truth tenemos una **precisión del 50%** para k=3



c) El conjunto de datos *train* (nuestro training set) tiene 6 variables numéricas (ALCOHOL, MALIC, MAGNESIUM, PHENOLS, FLAVANOIDS, COLOR) en base a las cuales debemos construir un conjunto de reglas de decisión que definan el valor de la variable CLASS.

train

ALCOHOL	MALIC	MAGNESIUM	PHENOLS	FLAVANOIDS	COLOR	CLASS
13,16	2,36	101	2,8	3,24	5,68	0
11,65	1,67	88	1,92	1,61	2,6	1
12,25	1,73	80	1,65	2,03	3,4	1
14,83	1,64	97	2,8	2,98	5,2	0
13,05	1,73	92	2,72	3,27	7,2	0
12,37	1,63	88	2,22	2,45	2,12	1
13,87	1,9	107	2,95	2,97	4,5	0
13,11	1,01	78	2,98	3,18	5,3	1

**Normalización:** Los datos no necesitan normalización, ya que van a ser evaluados independientemente unos de otros.

**Procedimiento:** en primer lugar deben fijarse umbrales para los valores numéricos en base a los cuales tabularemos, para cada variable, las frecuencias de pertenencia a los cuatro grupos siguientes:

Pertenece a CLASS 0 y está por debajo del umbral.

Pertenece a CLASS 1 y está por debajo del umbral.

Pertenece a CLASS 0 y es igual o mayor que el umbral.

Pertenece a CLASS 1 y es igual o mayor que el umbral.

En base a estas frecuencias, y de manera arbitraria, deberemos elegir, para cada variable, la hipótesis que mejor refleje la realidad de las observaciones.

<b>Hipótesis A</b>	Los valores por debajo del umbral indican CLASS=1
<b>Hipótesis B</b>	Los valores por encima del umbral indican CLASS=0

**Cálculo de los umbrales:** para cada variable, ordenada de menor a mayor, calcularemos los promedios de cada valor con el siguiente inmediatamente superior, de manera que podremos clasificar cualquier observación en uno de los dos grupos (bien inferior o bien mayor o igual que dicho umbral).

La primera tabla para cada variable representa el valor de la variable y su correspondencia con CLASS.

La segunda tabla para cada variable muestra las frecuencias de pertenencia CLASS=0 y CLASS=1 por debajo y por encima del umbral establecido entre cada par de valores consecutivos de la variable.

En las dos columnas de la derecha se muestran el porcentaje de aciertos frente a las hipótesis A y B.

La esquina superior derecha (en amarillo) muestra el indicador de bondad de la variable como criterio clasificador teniendo en cuenta (en base a la mayor bondad posible para cada hipótesis) la relación entre casos clasificados con una certeza del 100% y casos totales.

i	ALCOHOL	CLASS
2	11,65	1
3	12,25	1
6	12,37	1
5	13,05	0
8	13,11	1
1	13,16	0
7	13,87	0
4	14,83	0

	below threshold		above threshold		Bondad (hipotesis B)	42,86 %
Threshold	Class 0	Class 1	Class 0	Class 1	hipotesis A	hipotesis B
11,95	1	0	4	3	0,00 %	57,14 %
12,31	0	2	4	2	100,00 %	66,67 %
12,71	0	3	4	1	100,00 %	80,00 %
13,08	1	3	3	1	75,00 %	75,00 %
13,14	1	4	3	0	80,00 %	100,00 %
13,52	2	4	2	0	66,67 %	100,00 %
14,35	3	4	1	0	57,14 %	100,00 %

MALIC	CLASS	i
1,01	1	8
1,63	1	6
1,64	0	4
1,67	1	2
1,73	1	5
1,73	0	3
1,9	0	7
2,36	0	1

	below threshold		above threshold		Bondad (hipotesis B)	28,57 %
Threshold	Class 0	Class 1	Class 0	Class 1	hipotesis A	hipotesis B
1,32	0	1	4	3	100,00 %	57,14 %
1,64	0	2	4	2	100,00 %	66,67 %
1,66	1	2	3	2	66,67 %	60,00 %
1,70	1	3	3	1	75,00 %	75,00 %
1,73	1	3	3	1	75,00 %	75,00 %
1,815	2	4	2	0	66,67 %	100,00 %
2,13	3	4	1	0	57,14 %	100,00 %

i	MAGNESIUM	CLASS
8	78	1
3	80	1
2	88	1
6	88	1
5	92	0
4	97	0
1	101	0
7	107	0

	below threshold		above threshold		Bondad (hipótesis A)	57,14 %
MAGNESIUM	Class 0	Class 1	Class 0	Class 1	hipotesis A	hipotesis B
79	0	1	4	3	100,00 %	57,14 %
84	0	2	4	2	100,00 %	66,67 %
88	0	2	4	2	100,00 %	66,67 %
90	0	4	4	0	100,00 %	100,00 %
95	1	4	3	0	80,00 %	100,00 %
99	2	4	2	0	66,67 %	100,00 %
104	3	4	1	0	57,14 %	100,00 %

i	PHENOLS	CLASS
3	1,65	1
2	1,92	1
6	2,22	1
5	2,72	0
4	2,8	0
1	2,8	0
7	2,95	0
8	2,98	1

	below threshold		above threshold		Bondad (hipótesis A)	42,86 %
PHENOLS	Class 0	Class 1	Class 0	Class 1	hipotesis A	hipotesis B
1,79	0	1	4	3	100,00 %	57,14 %
2,07	0	2	4	2	100,00 %	66,67 %
2,47	0	3	4	1	100,00 %	80,00 %
2,76	1	3	3	1	75,00 %	75,00 %
2,8	1	3	3	1	75,00 %	75,00 %
2,875	3	3	1	1	50,00 %	50,00 %
2,97	4	3	0	1	42,86 %	0,00 %

i	FLAVANOIDS	CLAS
2	1,61	1
3	2,03	1
6	2,45	1
7	2,97	0
4	2,98	0
8	3,18	1
1	3,24	0
5	3,27	0

	below threshold		above threshold		Bondad (hipótesis A)	42,86 %
FLAVANOIDS	Class 0	Class 1	Class 0	Class 1	hipotesis A	hipotesis B
1,82	0	1	4	3	100,00 %	57,14 %
2,24	0	2	4	2	100,00 %	66,67 %
2,71	0	3	4	1	100,00 %	80,00 %
2,98	1	3	3	1	75,00 %	75,00 %
3,08	2	3	2	1	60,00 %	66,67 %
3,21	2	4	2	0	66,67 %	100,00 %
3,26	3	4	1	0	57,14 %	100,00 %

i	COLOR	CLASS
6	2,12	1
2	2,6	1
3	3,4	1
7	4,5	0
4	5,2	0
8	5,3	1
1	5,68	0
5	7,2	0

	below threshold		above threshold		Bondad (hipótesis A)	42,86 %
COLOR	Class 0	Class 1	Class 0	Class 1	hipotesis A	hipotesis B
2,36	0	1	4	3	100,00 %	57,14 %

	below threshold		above threshold		Bondad (hipótesis A)	42,86 %
COLOR	Class 0	Class 1	Class 0	Class 1	hipotesis A	hipotesis B
3,0	0	2	4	2	100,00 %	66,67 %
4,0	0	3	4	1	100,00 %	80,00 %
4,9	1	3	3	1	75,00 %	75,00 %
5,3	2	3	2	1	60,00 %	66,67 %
5,49	2	4	2	0	66,67 %	100,00 %
6,44	3	4	1	0	57,14 %	100,00 %

**Bondad del conjunto de entrenamiento:** una vez clasificadas las observaciones para cada variable calculamos el porcentaje de filas clasificadas para determinar la bondad de cada variable como clasificador. Vemos a continuación el resumen

Variable	bondad
MAGNESIUM	57,14 %
ALCOHOL	42,86 %
PHENOLS	42,86 %
FLAVANOIDS	42,86 %
COLOR	42,86 %
MALIC	28,57 %

Por tanto, el primer nodo de decisión del árbol de continuación será MAGNESIUM y el umbral será 95. Todos los casos con un valor inferior a 95 serán CLASS=1 y el resto, en la tabla a continuación, pasarán al siguiente nodo de decisión.

i	MAGNESIUM	CLASS
8	78	1
3	80	1
2	88	1
6	88	1
5	92	?

4	97	?
1	101	?
7	107	?

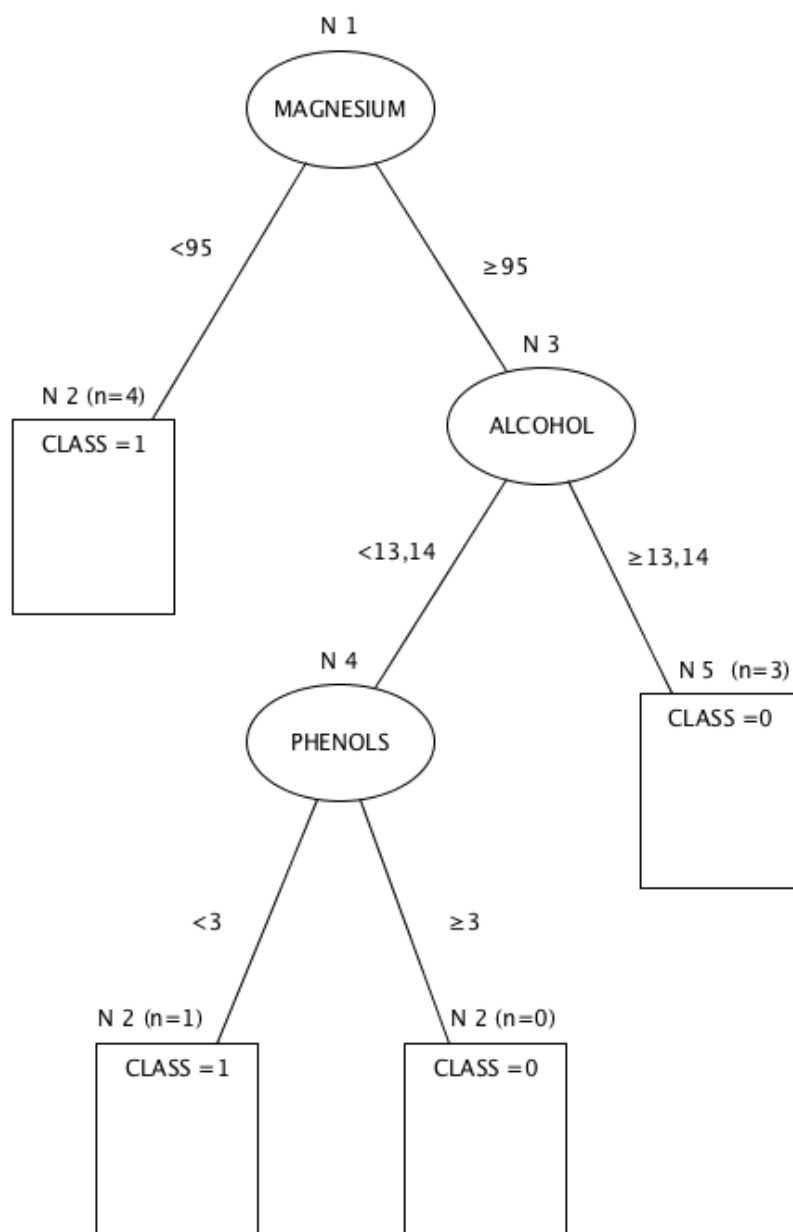
Las cuatro observaciones siguientes se clasificarán con ALCOHOL, que tiene un 42% de bondad, quedando como sigue:

i	ALCOHOL	threshold	CLASS
5	13,05	13,14	?
4	14,83	13,14	0
1	13,16	13,14	0
7	13,87	13,14	0

Queda una observación sin clasificar (índice 5) y usaremos la siguiente variable con mejor bondad para ello, que es PHENOLS. Como el umbral está en 2,76 y el valor es 2,72 asignamos el valor CLASS=1 (hipótesis A)

i	PHENOLS	threshold	CLASS
5	2,72	3	1

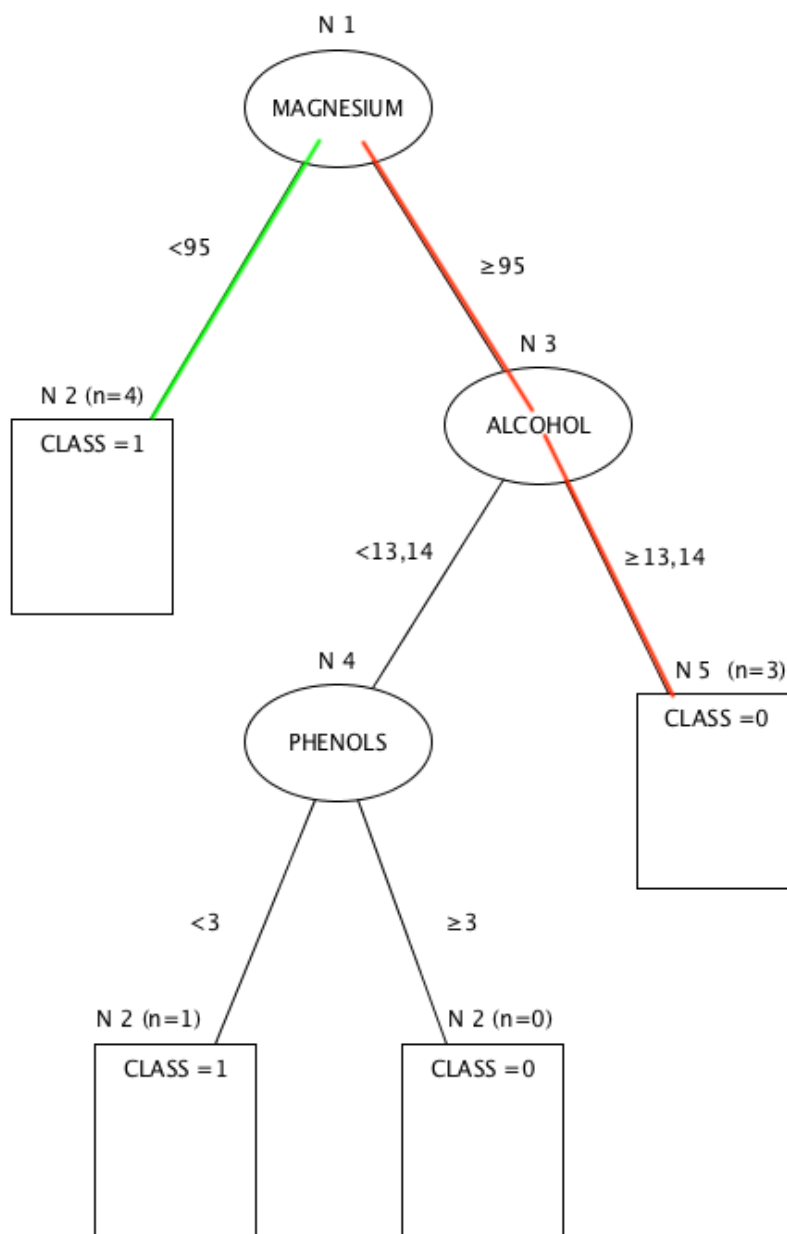
El modelo, por tanto se representa gráficamente como vemos a continuación:



**Clasificación del fichero test.csv:** ordenamos las variables para que sean evaluadas por el orden que indica el árbol y separamos CLASS como el valor ground\_truth que usaremos para evaluar la bondad del modelo.



La linea roja es el recorrido de i=1 y la linea verde i=2 (primer y segundo registros de test.csv respectivamente)



test

i	MAGNESIUM	ALCOHOL	PHENOLS	Prediction	Ground_truth	Accuracy
1	128	14,22	3	0	0	100 %
2	82	11,87	2,86	1	1	

d) El análisis de componentes principales tiene como objeto la reducción de dimensionalidad del conjunto de datos conservando la máxima información posible (varianza).

Primeramente preparamos los datos como en apartados anteriores separando ground truth del test set y quitándole a este último la variable CLASS

```
> test <- read.csv("~/Desktop/Aprendizaje computacional/PAC2/Materiales/test.csv")
> ground_truth <- test[,7]
> ground_truth
[1] 0 1
> test <- test[,-7]
> test
  ALCOHOL MALIC MAGNESIUM PHENOLS FLAVANOIDS COLOR
1  14.22  3.99      128    3.00      3.04    5.1
2  11.87  4.31      82    2.86      3.03    2.8
```

```
> train <- read.csv("~/Desktop/Aprendizaje computacional/PAC2/Materiales/train.csv")
```

Almacenamos en otro conjunto las etiquetas de clasificación que usaremos en kNN más adelante

```
> labels <- train[,2]
```

Y eliminamos esa información del training set

```
> train <- train[,-2]
```

Con los datos ya preparados realizamos el análisis de componentes obteniendo la matriz de covarianzas y multiplicándola por la matriz de vectores propios. La función prcomp() simplifica estas operaciones aportando además otra información de interés que veremos a continuación con la función summary().

```
> train.pca <- prcomp(train, scale= T, center=T)
> train.pca
Standard deviations:
[1] 1.8913735 1.2207959 0.7486862 0.4908162 0.3365002 0.1330403
```

Rotation:

	PC1	PC2	PC3	PC4	PC5
Z.ALCOHOL	-0.4474594	0.05304784	0.5498366	-0.66171827	0.1944556
Z.COLOR	-0.4300478	0.19075542	-0.6328226	-0.36372316	-0.4958270
Z.FLAVANOIDS	-0.4888875	0.19789144	-0.2221902	0.23440386	0.6034234
Z.MAGNESIUM	-0.3498976	-0.56363243	0.3000584	0.20941516	-0.4576829
Z.MALIC	-0.1303752	-0.75328839	-0.3641235	-0.08965254	0.3601042
Z.PHENOLS	-0.4862119	0.19108208	0.1588265	0.56832819	-0.1143419
	PC6				
Z.ALCOHOL	0.137701256				
Z.COLOR	0.008242405				
Z.FLAVANOIDS	-0.503383028				
Z.MAGNESIUM	-0.465324369				

```
Z.MALIC      0.381130397
Z.PHENOLS    0.604804766
```

```
> summary(train.pca)
Importance of components:
```

	PC1	PC2	PC3	PC4	PC5	PC6
Standard deviation	1.8914	1.2208	0.74869	0.49082	0.33650	0.13304
Proportion of Variance	0.5962	0.2484	0.09342	0.04015	0.01887	0.00295
Cumulative Proportion	0.5962	0.8446	0.93803	0.97818	0.99705	1.00000

```
>
```

Para conservar el 95% de la varianza debemos reducir la dimensionalidad como máximo a 4 componentes principales. De esta manera las 7 dimensiones de la matriz de datos que teníamos inicialmente quedará reducida a 4.

```
> C <- as.matrix(train.pca$rotation)
> C
```

	PC1	PC2	PC3	PC4	PC5
Z.ALCOHOL	-0.4474594	0.05304784	0.5498366	-0.66171827	0.1944556
Z.COLOR	-0.4300478	0.19075542	-0.6328226	-0.36372316	-0.4958270
Z.FLAVANOIDS	-0.4888875	0.19789144	-0.2221902	0.23440386	0.6034234
Z.MAGNESIUM	-0.3498976	-0.56363243	0.3000584	0.20941516	-0.4576829
Z.MALIC	-0.1303752	-0.75328839	-0.3641235	-0.08965254	0.3601042
Z.PHENOLS	-0.4862119	0.19108208	0.1588265	0.56832819	-0.1143419

	PC6
Z.ALCOHOL	0.137701256
Z.COLOR	0.008242405
Z.FLAVANOIDS	-0.503383028
Z.MAGNESIUM	-0.465324369
Z.MALIC	0.381130397
Z.PHENOLS	0.604804766

```
> is.matrix(C)
[1] TRUE
```

Eliminamos las componentes 5 a 7, ya que con las 4 primeras tenemos la información necesaria

```
> C <- C[,1:4]
> C
```

	PC1	PC2	PC3	PC4
Z.ALCOHOL	-0.4474594	0.05304784	0.5498366	-0.66171827
Z.COLOR	-0.4300478	0.19075542	-0.6328226	-0.36372316
Z.FLAVANOIDS	-0.4888875	0.19789144	-0.2221902	0.23440386
Z.MAGNESIUM	-0.3498976	-0.56363243	0.3000584	0.20941516
Z.MALIC	-0.1303752	-0.75328839	-0.3641235	-0.08965254
Z.PHENOLS	-0.4862119	0.19108208	0.1588265	0.56832819

```
>
```

Preparamos el conjunto de datos de entrenamiento para poder obtener un producto de matrices

```
> is.matrix(train)
[1] FALSE

> T <- as.matrix(train)
> T
      Z.ALCOHOL  Z.COLOR Z.FLAVANOIDS Z.MAGNESIUM  Z.MALIC  Z.PHENOLS
1  0.12448676  0.6929745  0.8440076  0.96258595  1.75648455  0.5820066
2 -1.39450324 -1.1158064 -1.7826890 -0.33753014 -0.10451252 -1.1541486
3 -0.79093105 -0.6459932 -1.1058715 -1.13760158  0.05731332 -1.6868326
4  1.80442935  0.4110866  0.4250253  0.56255023 -0.18542543  0.5820066
5  0.01383186  1.5856196  0.8923517  0.06250558  0.05731332  0.4241743
6 -0.67021661 -1.3976943 -0.4290540 -0.33753014 -0.21239640 -0.5622775
7  0.83871385  0.0000000  0.4089106  1.56263953  0.51581984  0.8779421
8  0.07418908  0.4698132  0.7473193 -1.33761944 -1.88459667  0.9371292

> is.matrix(T)
[1] TRUE
```

Y obtenemos una matriz de resultados que llamamos R. Esta matriz R representa el conjunto de entrenamiento con su dimensionalidad reducida a 4 componentes principales que conservan el 95% de la información.

```
> R <- T%*%C
> R
      PC1      PC2      PC3      PC4
1 -1.6151267 -1.4486588 -0.81591982  0.2382901
2  2.6682556 -0.5911665  0.08892174  0.1934944
3  2.3830940 -0.1083342 -0.41049984 -0.7029282
4 -1.6476239  0.1920652  0.96631383 -0.7787169
5 -1.3599224  0.4824365 -1.12882602 -0.1276871
6  1.5299074 -0.1442799  0.49806958  0.4800962
7 -1.6160814 -0.9761445  0.79080122  0.3208130
8 -0.3425028  2.5940822  0.01113931  0.3766384

>
```

Reducimos la dimensionalidad a continuación del conjunto de prueba (test) con el PCA modelado con el training set

```
> S <- as.matrix(test)
> S
      ALCOHOL  MALIC  MAGNESIUM  PHENOLS  FLAVANOIDS  COLOR
[1,]   14.22   3.99       128     3.00         3.04    5.1
[2,]   11.87   4.31        82     2.86         3.03    2.8
```

Efectuamos la proyección de las variables de test en sus componentes principales mediante el producto de las matrices.

```
> U <- S%*%C
> U
      PC1      PC2      PC3      PC4
```

```
[1,] -74.58208 23.83918 -22.54337 22.39698
[2,] -50.01076 14.31951 -14.22091 11.71747

U <-scale(U)
> U
      PC1      PC2      PC3      PC4
[1,] -0.7071068  0.7071068 -0.7071068  0.7071068
[2,]  0.7071068 -0.7071068  0.7071068 -0.7071068
attr(,"scaled:center")
      PC1      PC2      PC3      PC4
-62.29642 19.07935 -18.38214 17.05723
attr(,"scaled:scale")
      PC1      PC2      PC3      PC4
17.374545  6.731426  5.884868  7.551552

> is.matrix(U)
[1] TRUE
> result <- knn(R,U,labels, k=1)
```

A continuación aplicamos knn con k=1 al conjunto reducido de entrenamiento.

```
> result <- knn(R,U,labels, k=1)
> result
[1] 0 1
Levels: 0 1
>
```

Tabulamos los resultados para ver la precisión

```
> tabla <- table(result, ground_truth)
> tabla
      ground_truth
result 0 1
      0 1 0
      1 0 1
>
```

Tenemos por tanto una precisión del 100%

Aplicamos knn con k=3 al mismo conjunto.

```
> result <- knn(R,U,labels, k=3)
> result
[1] 0 1
Levels: 0 1
> tabla <- table(result, ground_truth)
> tabla
      ground_truth
result 0 1
      0 1 0
      1 0 1
```

Con lo que tenemos de nuevo una precisión del **100%**

Para reducir la dimensionalidad de ambos conjunto hemos utilizado el producto matricial de ambos conjuntos, train y test por la matriz de componentes principales obtenida del análisis del conjunto train.

Si aplicamos el PCA al conjunto test, obtenemos una matriz que conserva el 100% de la varianza en una única componente principal.

```
> test.pca <- prcomp(test, scale= TRUE, center=TRUE)
> test.pca
Standard deviations:
[1] 2.449490e+00 1.755417e-16
```

Rotation:

	PC1	PC2
Z.ALCOHOL	-0.4082483	0.9128709
Z.COLOR	-0.4082483	-0.1825742
Z.FLAVANOIDS	-0.4082483	-0.1825742
Z.MAGNESIUM	-0.4082483	-0.1825742
Z.MALIC	0.4082483	0.1825742
Z.PHENOLS	-0.4082483	-0.1825742

```
> summary(test.pca)
```

Importance of components:

	PC1	PC2
Standard deviation	2.449	1.755e-16
Proportion of Variance	1.000	0.000e+00
Cumulative Proportion	1.000	1.000e+00

```
>
```

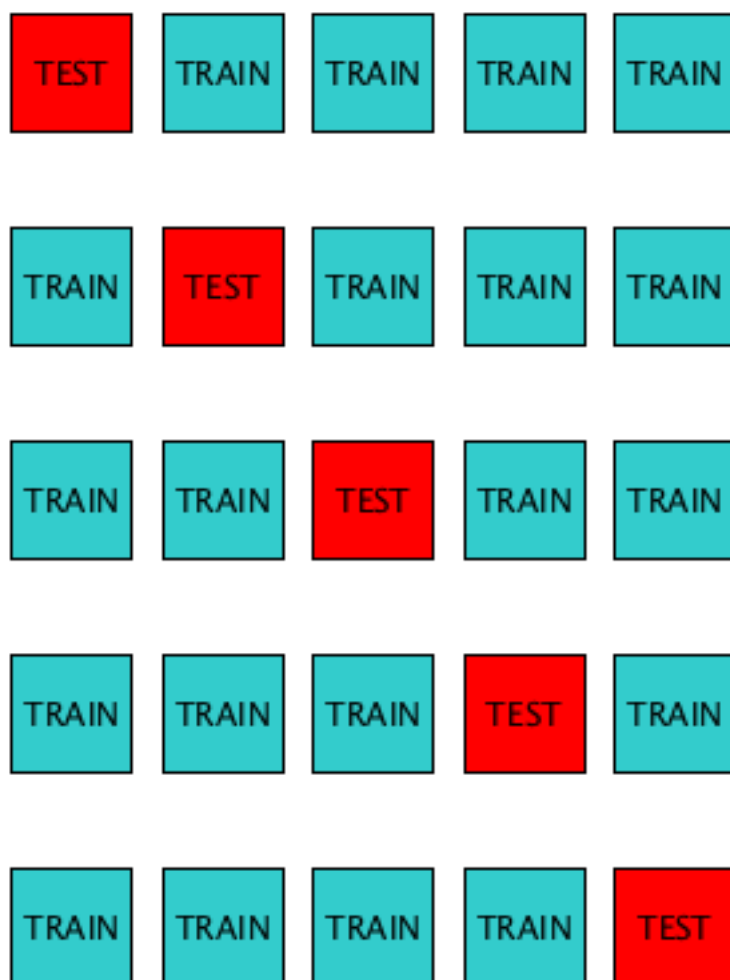
Esto es debido a que el conjunto de prueba, sus variables, una vez estandarizado, toman únicamente dos únicos valores (0.7071068, -0.7071068), por lo que toda la información puede reducirse a una componente principal.

```
> test
      Z.ALCOHOL      Z.COLOR Z.FLAVANOIDS Z.MAGNESIUM      Z.MALIC      Z.PHENOLS
1  0.7071068  0.7071068    0.7071068    0.7071068 -0.7071068  0.7071068
2 -0.7071068 -0.7071068   -0.7071068   -0.7071068  0.7071068 -0.7071068
>
```

## Exercici 2

Se ha escogido un clasificador de validación cruzada k Fold con  $k=10$  y el algoritmo de clasificación kNN con  $k=1$ .

El principio de funcionamiento de kFold cross validation es que, de forma iterativa, una parte del conjunto de entrenamiento es utilizada como conjunto de prueba en cada iteración como se indica en el dibujo a continuación y el resto compone el conjunto de entrenamiento.



Vemos a continuación la ejecución del clasificador en python:

```
runfile('/Users/luisarribas/Desktop/Aprendizaje computacional/PAC2/
Materiales/K-fold_test.py', wdir='/Users/luisarribas/Desktop/Aprendizaje
computacional/PAC2/Materiales')
```

```
* * * * *
Model construction time:0.00035500526428222656
* * * * *
```

```
* * * * *
Classification time:0.0006878376007080078
* * * * *
```

```
[STANDARDIZATION PARAMETERS]
- MEAN      : [-0.01 0.02 -0.04 -0.03 0.01 0.01 -0.01 0.01 0.01
-0.02 -0.03 0.00 -0.02
0.56]
- STD       : [0.98 1.05 1.00 0.99 1.03 1.02 1.03 1.01 1.02 1.03
1.02 1.00 1.01 0.50]
```

```
[CLASSES]
- KNN       : [0 0 0 1 0 1 0 1 1 0 1 0 0]
- GROUND TRUTH : [0 0 0 1 0 1 0 1 1 1 1 0 0]
- ACCURACY   : 92.3076923076923%
```

```
* * * * *
Model construction time:0.0003669261932373047
* * * * *
```

```
* * * * *
Classification time:0.0004949569702148438
* * * * *
```

```
[STANDARDIZATION PARAMETERS]
- MEAN      : [-0.06 0.05 -0.00 0.04 -0.02 -0.10 -0.09 0.09 -0.11
-0.08 -0.00 -0.04
-0.08 0.57]
- STD       : [0.98 1.04 1.02 1.01 1.01 0.95 0.99 1.01 0.97 0.98
1.03 1.02 0.97 0.50]
```

```
[CLASSES]
- KNN       : [0 1 0 1 0 0 1 0 1 0 0 0 0]
- GROUND TRUTH : [0 1 0 1 0 1 1 0 1 0 0 0 0]
- ACCURACY   : 92.3076923076923%
```

```
* * * * *
Model construction time:0.0004620552062988281
* * * * *
```

```
* * * * *
Classification time:0.0004849433898925781
* * * * *
```

```
[STANDARDIZATION PARAMETERS]
```



```

- MEAN          : [-0.03 -0.02 -0.03 0.03 -0.01 0.00 -0.01 -0.03 -0.00
-0.02 0.01 -0.01
-0.01 0.57]
- STD           : [1.02 1.01 1.01 1.02 1.02 0.95 1.00 0.98 1.04 1.00
1.03 0.97 1.02 0.49]
[CLASSES]
- KNN           : [0 0 0 1 1 1 0 1 0 0 0 0]
- GROUND TRUTH  : [0 0 0 1 1 1 0 1 0 0 0 0]
- ACCURACY      : 100.0%
* * * * *
Model construction time:0.0005130767822265625
* * * * *

* * * * *
Classification time:0.0004899501800537109
* * * * *

[STANDARDIZATION PARAMETERS]
- MEAN          : [-0.01 -0.04 0.01 0.03 0.02 0.02 0.03 -0.02 0.01 0.01
0.00 0.02 -0.03 0.55]
- STD           : [0.98 0.92 1.04 1.03 1.01 1.02 1.02 0.99 0.99 1.01
0.94 1.00 0.98 0.50]
[CLASSES]
- KNN           : [0 1 0 0 0 1 0 1 1 0 0 1]
- GROUND TRUTH  : [0 1 0 1 0 1 0 1 1 0 0 1]
- ACCURACY      : 91.66666666666666%
* * * * *
Model construction time:0.0006480216979980469
* * * * *

* * * * *
Classification time:0.0007359981536865234
* * * * *

[STANDARDIZATION PARAMETERS]
- MEAN          : [0.04 -0.03 0.04 -0.03 0.06 0.04 0.03 -0.03 0.05 0.03
0.08 0.02 0.06 0.52]
- STD           : [1.01 0.99 0.99 1.02 1.01 0.99 1.00 0.98 1.03 1.00
0.98 1.01 0.99 0.50]
[CLASSES]
- KNN           : [1 0 1 1 0 1 1 1 1 1 1 1]
- GROUND TRUTH  : [1 0 1 1 0 1 1 1 1 1 1 1]
- ACCURACY      : 100.0%
* * * * *
Model construction time:0.0003509521484375
* * * * *

* * * * *
Classification time:0.0005488395690917969
* * * * *
```

```
[STANDARDIZATION PARAMETERS]
- MEAN      : [0.01 0.04 -0.01 0.01 0.02 0.04 0.05 -0.07 0.07 0.01
-0.03 0.02 -0.01 0.55]
- STD       : [1.00 1.02 1.03 0.99 0.98 1.01 0.99 0.98 0.99 0.99
1.01 0.98 1.01 0.50]
[CLASSES]
- KNN       : [0 1 0 1 0 1 0 0 1 0 1 1]
- GROUND TRUTH : [0 1 0 1 0 1 0 0 1 0 1 1]
- ACCURACY   : 100.0%
* * * * *
Model construction time:0.00047206878662109375
* * * * *

* * * * *
Classification time:0.0004940032958984375
* * * * *

[STANDARDIZATION PARAMETERS]
- MEAN      : [0.02 -0.03 0.01 -0.00 -0.02 -0.01 0.01 0.03 -0.04
0.04 -0.03 -0.02 0.03
0.54]
- STD       : [0.99 0.99 1.01 1.01 0.97 1.02 1.00 0.99 0.99 1.00
1.00 1.02 0.99 0.50]
[CLASSES]
- KNN       : [0 1 0 0 1 1 1 1 0 1 1 1]
- GROUND TRUTH : [0 1 0 0 1 1 1 1 0 1 1 1]
- ACCURACY   : 100.0%
* * * * *
Model construction time:0.0005218982696533203
* * * * *

* * * * *
Classification time:0.0004432201385498047
* * * * *

[STANDARDIZATION PARAMETERS]
- MEAN      : [0.04 -0.04 -0.03 -0.09 -0.02 0.04 0.04 -0.04 -0.02
0.05 0.01 0.05 0.04
0.53]
- STD       : [1.01 0.96 0.96 0.94 0.99 0.99 1.02 1.00 0.96 1.00
0.97 0.98 1.01 0.50]
[CLASSES]
- KNN       : [1 0 1 1 1 0 1 1 1 1 1 0]
- GROUND TRUTH : [1 0 1 1 1 0 1 1 1 1 1 0]
- ACCURACY   : 100.0%
* * * * *
Model construction time:0.0004951953887939453
* * * * *
```

\* \* \* \* \*

Classification time:0.0004711151123046875

\* \* \* \* \*

[STANDARDIZATION PARAMETERS]

- MEAN : [-0.02 0.04 0.02 0.04 -0.03 -0.06 -0.04 0.06 0.00  
-0.03 0.02 -0.03 -0.01  
0.57]

- STD : [1.00 1.01 1.02 1.01 1.01 1.01 1.01 1.01 1.04 0.99  
1.01 1.01 1.01 0.49]

[CLASSES]

- KNN : [0 0 0 0 1 0 0 0 1 1 1 0]

- GROUND TRUTH : [0 0 0 0 1 0 0 0 1 1 1 0]

- ACCURACY : 100.0%

\* \* \* \* \*

Model construction time:0.0003578662872314453

\* \* \* \* \*

\* \* \* \* \*

Classification time:0.0004968643188476562

\* \* \* \* \*

[STANDARDIZATION PARAMETERS]

- MEAN : [0.02 0.02 0.04 0.00 -0.02 0.02 -0.00 0.02 0.03 -0.00  
-0.02 -0.00 0.02  
0.53]

- STD : [1.02 1.00 0.91 0.96 0.95 1.02 0.93 1.03 0.97 0.99  
1.02 0.99 1.01 0.50]

[CLASSES]

- KNN : [0 1 1 1 0 0 1 0 1 0 1 1]

- GROUND TRUTH : [1 1 1 1 1 0 1 0 1 0 1 1]

- ACCURACY : 83.33333333333334%

\*\*\*\*\*

Total aciertos=117

\*\*\*\*\*

\*\*\*\*\*

Total fallos=5

\*\*\*\*\*

\*\*\*\*\*

ERROR=4.038461538461535%

\*\*\*\*\*

\*\*\*\*\*

Model construction mean time:=0.00045430660247802734

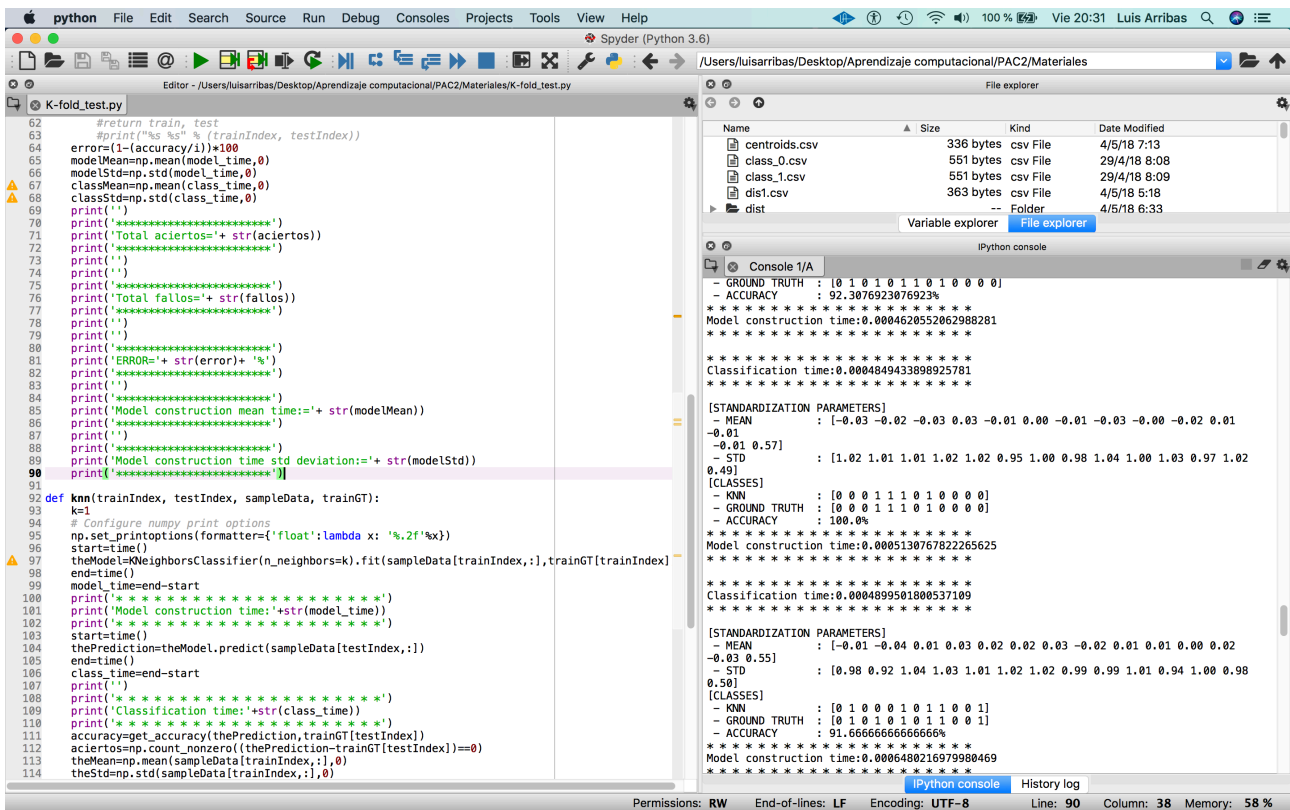
\*\*\*\*\*

```

*****
Model construction time std deviation:=9.223584112808984e-05
*****

```

En la ejecución vemos que el programa se comporta como se espera segmentando el conjunto de datos en un conjunto de entrenamiento y otro más pequeño de prueba.



The screenshot shows the Spyder Python IDE with a file named `K-fold_test.py` open. The script implements a K-fold cross-validation for a K-Nearest Neighbors (KNN) classifier. It calculates the mean and standard deviation of the model construction time across different folds. The console output shows the results for two different values of K (K=1 and K=5).

```

62 #return train, test
63 #print("%s %s" % (trainIndex, testIndex))
64 errors=(1-(accuracy/i))*100
65 modelMean=np.mean(model_time,0)
66 modelStd=np.std(model_time,0)
67 classMean=np.mean(class_time,0)
68 classStd=np.std(class_time,0)
69 print('')
70 print('*****')
71 print('Total aciertos='+ str(aciertos))
72 print('*****')
73 print('')
74 print('')
75 print('*****')
76 print('Total fallos='+ str(fallos))
77 print('*****')
78 print('')
79 print('')
80 print('*****')
81 print('ERROR='+ str(error)+ '%')
82 print('*****')
83 print('')
84 print('*****')
85 print('Model construction mean time:='+ str(modelMean))
86 print('*****')
87 print('')
88 print('*****')
89 print('Model construction time std deviation:='+ str(modelStd))
90 print('*****')
91
92 def knn(trainIndex, testIndex, sampleData, trainGT):
93     k=1
94     # Configure numpy print options
95     np.set_printoptions(formatter={'float':lambda x: '%.2f'%x})
96     start_time()
97     theModel=KNeighborsClassifier(n_neighbors=k).fit(sampleData[trainIndex,:],trainGT[trainIndex])
98     end_time()
99     model_time=end-start
100     print('*****')
101     print('Model construction time:'+str(model_time))
102     print('*****')
103     start_time()
104     thePrediction=theModel.predict(sampleData[testIndex,:])
105     end_time()
106     class_time=end-start
107     print('')
108     print('*****')
109     print('Classification time:'+str(class_time))
110     print('*****')
111     accuracy=get_accuracy(thePrediction,trainGT[testIndex])
112     aciertos=np.count_nonzero((thePrediction-trainGT[testIndex])==0)
113     theMean=np.mean(sampleData[trainIndex,:],0)
114     theStd=np.std(sampleData[trainIndex,:],0)

```

The console output shows the following results:

```

- GROUND TRUTH : [0 1 0 1 0 1 0 1 0 0 0]
- ACCURACY : 92.3076923076923%
*****
Model construction time:0.0004628552062988281
*****

[STANDARDIZATION PARAMETERS]
- MEAN : [-0.03 -0.02 -0.03 0.03 -0.01 0.00 -0.01 -0.03 -0.00 -0.02 0.01
-0.01]
- STD : [1.02 1.01 1.01 1.02 1.02 0.95 1.00 0.98 1.04 1.00 1.03 0.97 1.02
0.49]
[CLASSES]
- KNN : [0 0 0 1 1 1 0 1 0 0 0]
- GROUND TRUTH : [0 0 0 1 1 1 0 1 0 0 0]
- ACCURACY : 100.0%
*****
Model construction time:0.0005130767822265625
*****

[STANDARDIZATION PARAMETERS]
- MEAN : [-0.01 -0.04 0.01 0.03 0.02 0.02 0.03 -0.02 0.01 0.01 0.00 0.02
-0.03 0.55]
- STD : [0.98 0.92 1.04 1.03 1.01 1.02 1.02 0.99 0.99 1.01 0.94 1.00 0.98
0.50]
[CLASSES]
- KNN : [0 1 0 0 0 1 0 1 0 0 1]
- GROUND TRUTH : [0 1 0 1 0 1 0 1 0 0 1]
- ACCURACY : 91.66666666666666%
*****
Model construction time:0.0006480216979980469
*****

```

## Exercici 3

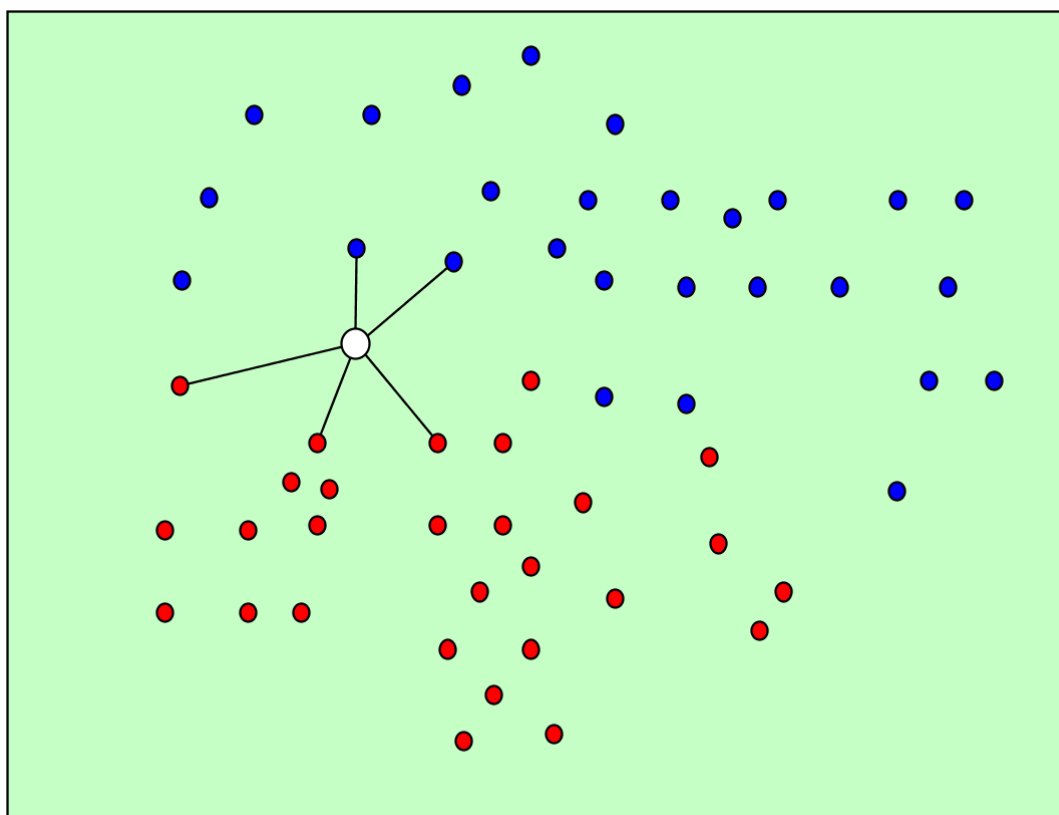
El método de clasificación kNN utilizado en el apartado a del ejercicio 1 es un método de clasificación sin parámetros que se basa en la probabilidad de pertenencia a un grupo de clasificación dado la distancia euclídea entre el punto n-dimensional definido por los valores de las variables del objeto y sus k vecinos más próximos.

Así para  $k=1$  será la pertenencia de ese vecino más próximo a un grupo la que dicte la clasificación correspondiente.

Para  $k>1$ , y siempre con valores impares, el método establecerá un sistema de votaciones de manera que la categoría con más votos será la asignada en la clasificación.

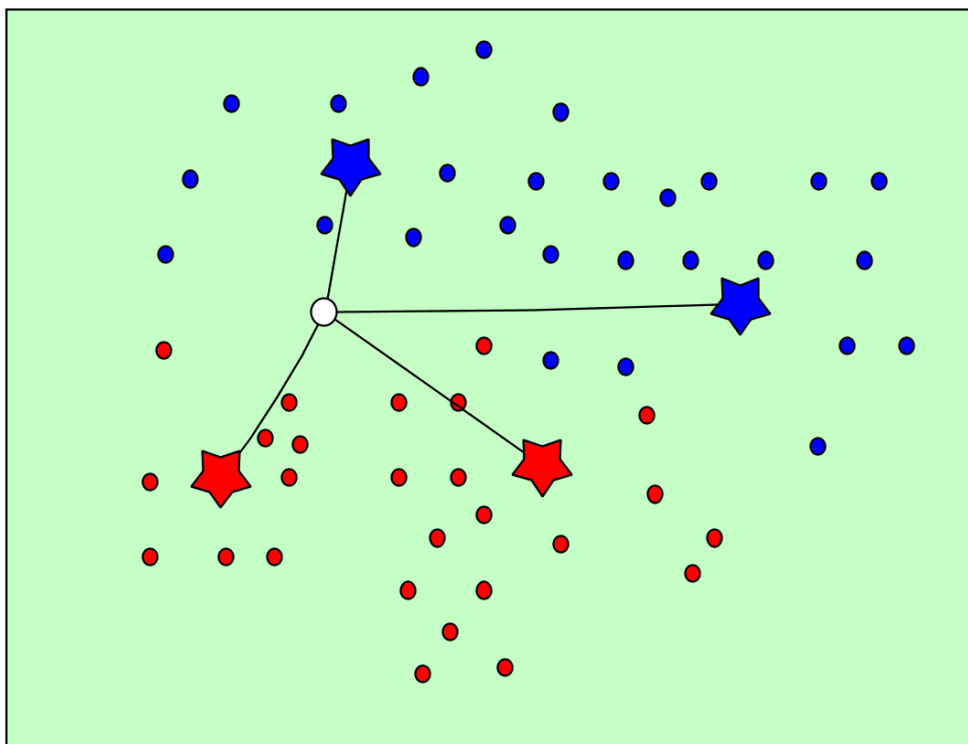
Computacionalmente costoso, para grandes conjuntos de datos o de grandes dimensionalidades, puede optimizarse mediante un agrupamiento previo por el método k-means (apartado b de la misma pregunta) o la reducción de su dimensionalidad mediante el análisis de componentes principales (apartado d. de la primera pregunta)

El sistema por kNN calcula para cada elemento a clasificar la distancia euclídea a cada uno de los miembros del training set



Si aplicamos previamente el k-means en  $k=2$  obtenemos dos centroides por clase, lo que sería como en el diagrama que vemos a continuación.

De esta manera el resultado de la clasificación en caso concreto del ejercicio sigue siendo del 100% de aciertos, pero el cote computacional se reduce notablemente.



La reducción del coste computacional mediante la reducción de dimensionalidad es otra técnica que hemos empleado en el apartado d previamente a kNN de manera que conservando la mayor parte de la información podemos reducir el tamaño del problema de clasificación.

En esencia podemos ver la matriz de componentes principales como un prisma a través del cual proyectar cualquier conjunto de datos  $n$ -dimensional transformándolo a otro sistema  $m$ -dimensional donde  $n > m$ .

La obtención de esta matriz está íntimamente ligada a la correlación entre las propias variables, a través de la matriz de covarianzas de las variables y el producto matricial con su matriz de eigen vectores.

En el apartado c) hemos visto como construir un árbol de decisión en base, primeramente, a la bondad de las variables a la hora de hacer clasificaciones certeras al 100%, de modo que aquella variable de mayor capacidad discriminante se ubica en el nodo raíz del árbol y así lo hacen de forma descendente.

Hemos visto cómo calcular los umbrales de decisión y la posibilidad de fijar la bondad de la variables en torno a si los valores superan o no el umbral en dos hipótesis.

Este sistema de clasificación es óptimo para conjuntos de datos con variables no numéricas, que representados en espacios n-dimensionales ocuparían posiciones extremas y con distribuciones poco representativas o de difícil manejo.

En el ejercicio 2 hemos construido un clasificador en python siguiendo el sistema de k fold cross validation.

El sistema de iteraciones con diferentes particiones del mismo conjunto de datos anula la posibilidad de que la clasificación quede definida por la selección de un conjunto de prueba no suficientemente representativo, ya que todo el conjunto de datos es conjunto de entrenamiento y conjunto de prueba en un momento u otro.

Una vez que se han ejecutado todos los “folds”, o particiones y cada una ha tenido su propia clasificación se realiza una media de los resultados con el fin de obtener un valor real e independiente de la elección del training set.