# Predicción de Plantas

APRENENTATGE AUTOMÀTIC (APA)

**Joel Lopez Agudo**
joel.lopez

**Ivan Megias Durán**
ivan.megias

**Juan José Vázquez Giménez**
juan.jose.vazquez.gimenez

# Índice

# 1. Introducción

## 1.1 Descripción del problema

Reconocimiento de diferentes especies de planta a partir de la forma, la textura de sus hojas y sus bordes. La clasificación basada en las imágenes de las hojas de las plantas es una buena manera comparada a otros métodos como pueden ser a partir de las células u otros métodos biológicos. La gran ventaja es la facilidad y el bajo coste que tiene conseguir muestras, es decir las fotografías de las hojas. Una vez tenemos esto, hoy en día es bastante fácil obtener un análisis y transformar estos datos a datos informáticos a partir de técnicas de procesado de imágenes.

## 1.2 Objetivos

Conseguir un modelo que nos consiga clasificar una muestra de una hoja a partir de su forma, borde y textura. Que nos pueda decir a qué familia pertenece y a qué especie concreta de esta.

## 1.3 Datos iniciales

El conjunto de hojas consta de 100 especies de plantas diferentes. Más concretamente, existen diversos números de familias de plantas. Cada una de ellas contiene un número diferente de especies concretas. También hay algunas que solo cuentan con una especie concreta.

De cada una de ellas disponemos de un total de 16 muestras. De cada muestra tenemos una imagen en blanco y negro en la cual se aprecia la silueta de la hoja.

Además de estas imágenes, disponemos de 3 ficheros diferentes con datos de las plantas. Estos ficheros son: uno con la descripción de la forma de la hoja, otra con la descripción de los bordes de la hora y por último uno con la descripción de la textura. Estas descripciones son vectores de 64 enteros. Por lo que podemos concluir que de cada especie disponemos de un total de 192 variables.

# 2. Trabajo previo

## 2.1 Análisis datos iniciales

Antes de tratar las diferentes clases analizamos cada fichero de datos para ver si estos eran correctos. Entonces nos encontramos que en algunas columnas de los vectores, todas las muestras eran 0. Decidimos borrar estas columnas ya que no nos aportan información útil sobre nuestras clases. También nos encontramos que uno de los ficheros tenía una muestra menos, más concretamente una de las especie tenía 15 muestras en vez de 5. Lo que decidimos hacer fue obtener la media de las 15 muestras y añadirla como la 16.

## 2.2 Reducción de clases

Una vez solucionado esto pasamos al análisis de las especies.

Como hemos mencionado en la descripción de los datos iniciales, disponemos de demasiadas especies y con lo cual es muy difícil trabajar.

Para ello hacemos un trabajo previo para facilitar la obtención del modelo. A partir de las 100 especies, que serían nuestras clases a predecir, vamos a obtener “superclases” de ellas. Para ello disponemos de 2 formas para agruparlas:

- Por nombre de familia: cada nombre de especie viene dado por el nombre de la familia seguido de la especie. Un ejemplo: “Acer Campestre” y “Acer Capillipes” las agrupamos dentro de la familia de “Acer”.
- Clustering: a partir de los datos agrupamos los que están más cercanos en el número de clusters que a nosotros nos interese.

Estas dos técnicas decidimos aplicarlas por separado a cada vector descriptor y luego conjuntamente a los 3 vectores (forma, borde y textura). De esta forma podemos saber si uno de estos tiene mas influencia o si tienen relación entre ellos.

### 2.2.1 Nombre de familia

Para reducir el número de clases agrupando las especies, hemos utilizado directamente R. Por cada tabla que tenemos cargada de los ficheros de datos, utilizamos lapply sobre la primera columna de cada tabla aplicando la función:

```
function(x) {
   pos<-str_locate(x,' ')
   if (!is.na(pos[1,1]))
      substr(x,0,pos[1,1]-1)
   else x
}
```

Lo que hace esta función primero, es comprobar si la string x tiene un espacio. De ser así, devuelve x hasta el primer espacio, sino devuelve x. Con esto conseguimos tener como clase el nombre de todas las familias.

El resultado obtenido es el siguiente:

```
Numero de clases:  34
Muestras en cada clase:  16 176  80  16  32  16  16  16  16  48  16  16  16  48  16  16
32  16  16  32  32  16  16  48  32  16 608  16  32  16  48  16  32  16
```

Podemos ver como algunas familias solo tienen una especie con las 16 muestras. En cambio tenemos que la familia “Quercus” tiene  38 especies diferentes.

Con este método teníamos sospechas que no daría buenos resultados ya que hay especies dentro de una misma familia que son muy diferentes. Por lo tanto, será complicado predecir una clase que varien tanto.

### 2.2.2 Clustering

Aplicamos el algoritmo k-means para agrupar los datos en clusters. Decidimos poner como número de clusters 10. El resultado obtenido es el siguiente:

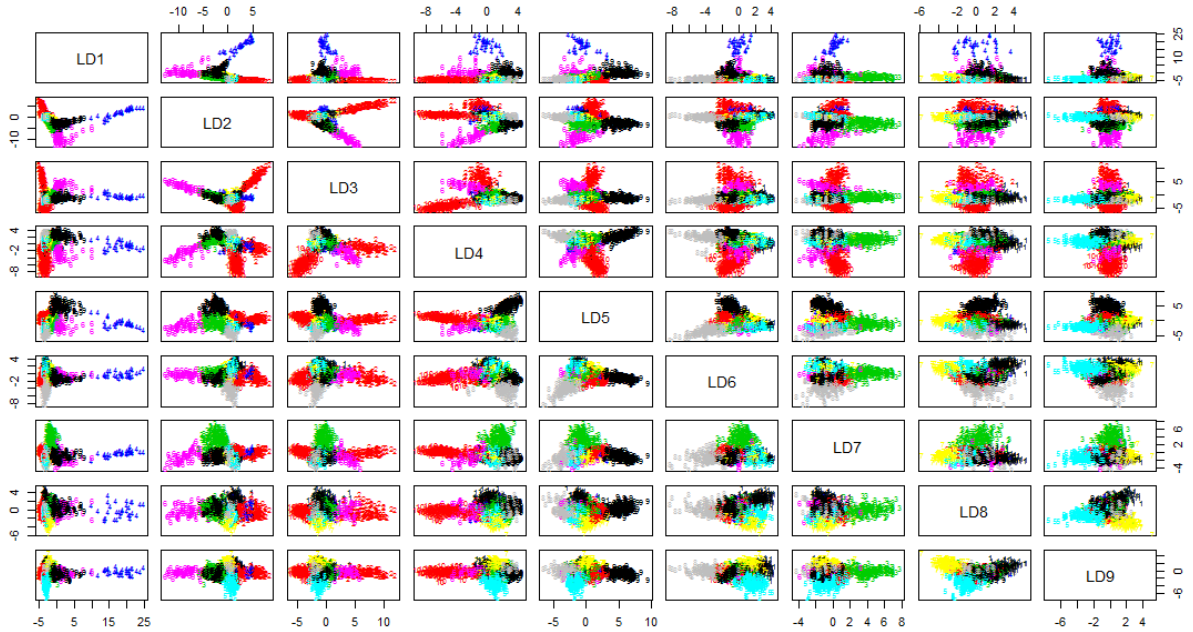
```
Numero de clases:  10
Muestras en cada clase:  130 97 91 149 287 178 252 233 35 148
```

Podemos ver el número de muestras que tendrá cada cluster. En este caso vemos que está más repartido el número de muestras de cada clase. Otra observación a hacer es que una misma especie puede tener muestras diferentes en clusters diferentes, ya que el número de muestras en cada cluster no es múltiplo de 16. Veremos más adelante si esto es un problema y si necesitaremos resolverlo.

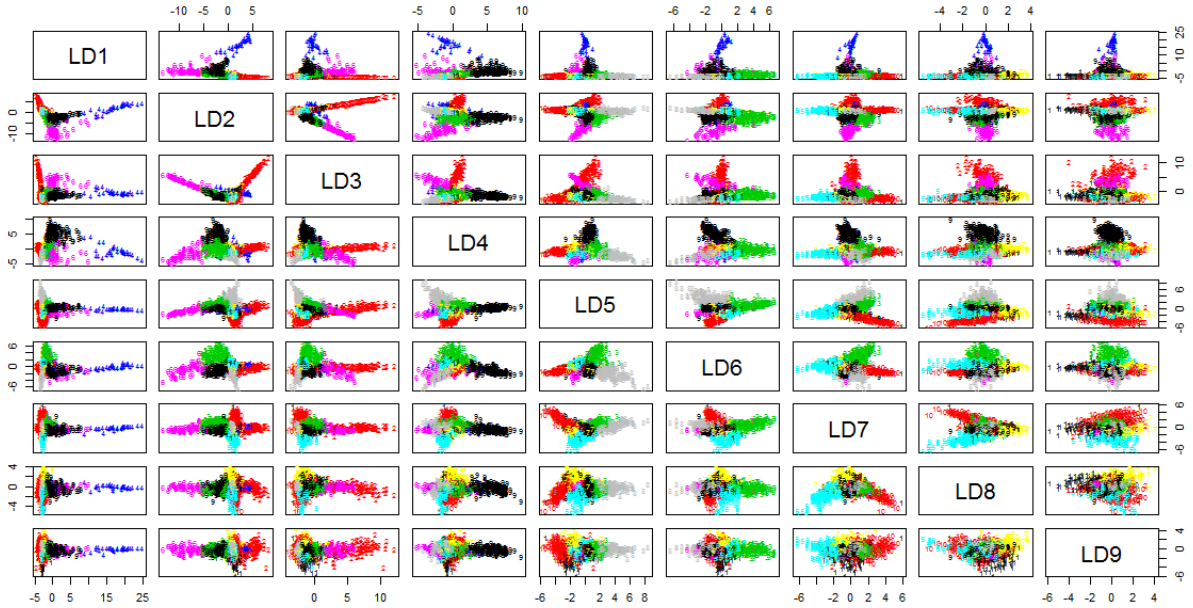
### 2.2.3 Visualización de los datos

Una vez que ya tenemos solucionado el problema de las clases podemos ver los datos que tenemos hasta ahora usando LDA. Como el número de clases de nombre de familia es muy grande no hemos podido obtener el plot de estos. Por ello hemos puesto solo los plot del método de clustering, tanto con todos los vectores como por separado.

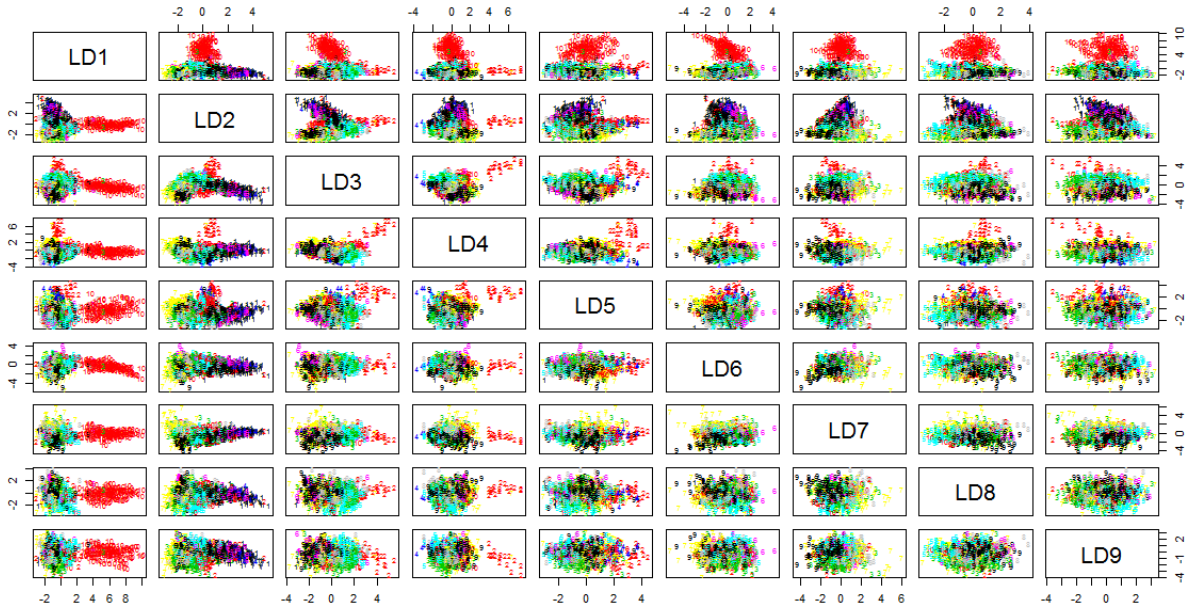
## Clustering - Todos los datos



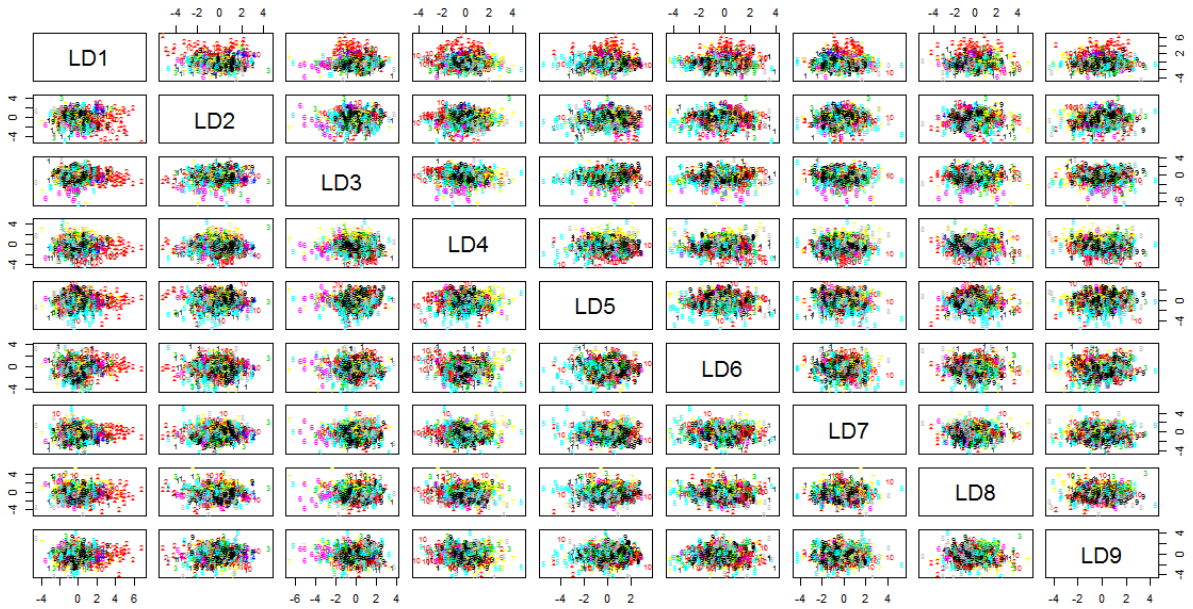

## Clustering - Textura



## Clustering - Bordes




## Clustering - Forma




Podemos observar a simple vista que el gráfico correspondiente a todos los vectores unidos tiene una gran influencia del vector descriptor de la textura. Podemos intuir entonces que la textura será el descriptor con más peso a la hora de predecir las clases y por lo tanto las especies.

### 2.3 Reducción de variables

Por último nos queda reducir el número de variables, para ello hemos probado 3 filtros diferentes:

1. Correlation filter + Random Forest (10 variables)
2. Correlation filter + ReliefF (10 variables)
3. Consistency filter

Aplicamos estos filtros a los 4 conjuntos diferentes que tenemos. A los conjuntos que tienen los 3 vectores por separado, se reducen las variables por separado y después se unirán.

Para cuantificar cómo de bueno o malos son los resultados obtenidos usaremos un evaluador de precisión. Tenemos 3 diferentes posibilidades:

1. **LDA + backward.search**: creará un modelo a partir de LDA y predecirá con todas las variables. Obtendrá el acierto y hará una media de estos. Luego irá quitando una variable en cada iteración y quedándose con la que mejor porcentaje tenga.
2. **LDA + forward.search**: la diferencia con el anterior es que irá añadiendo variables de una en una y probando todas las combinaciones que mejor porcentaje tengan.
3. **LDA + forward.search**: en este se utilizará QDA para crear el modelo y hacer las predicciones.

Finalmente solo usamos el método 2 ya que era con el que mejores resultados obteníamos. También después de utilizar los 3 filtros con los conjuntos de todas las variables decidimos solo usar el filtro 1 para los siguientes experimentos ya que era el que mejor resultados nos da.

La siguiente tabla muestra los resultados finales de la reducción de variables en cada caso. Los porcentajes nos indican la precisión que tendrían estas variables al predecir usando LDA.

| | Correlation + RF | Correlation + Relief | Consistency |
|---|---|---|---|
| Familias - Todo | 47,87% | 49,56% | 50,06% |
| Familias - Textura | 51,56% | - | - |
| Familias - Forma | 37,62% | - | - |
| Familias - Bordes | 53,94% | - | - |
| Familias - Unidas | 61,43% | - | - |
| Clustering - Todo | 83,44% | 49,5% | 73% |
| Clustering - Texturas | 38% | - | - |
| Clustering - Forma | 49,62% | - | - |
| Clustering - Bordes | 49,93% | - | - |
| Clustering - Unidas | 82,25% | - | - |

## 3. Clasificación

Una vez reducidas las variables debemos obtener el modelo. Para ello utilizamos dos métodos: LDA y SVM.

### 3.1 Clasificación con LDA

Para utilizar LDA necesitamos un conjunto de aprendizaje (data.learn) y otro de prueba (data.test). Seleccionamos aleatoriamente la mitad de las muestras de las que disponemos para formar el conjunto de aprendizaje. El resto formarán parte del conjunto de prueba.

Ejecutamos LDA con Cross Validation y lo vamos a entrenar con el conjunto de muestras que hemos definido anteriormente como data.learn.

Una vez que tenemos nuestro clasificador, probaremos su rendimiento. Esto lo haremos intentando predecir la clase de las muestras que hemos definido como data.test. A partir de los resultados, calculamos el porcentaje de fallos de training de nuestro clasificador.

Ahora pasamos hacer el reajuste y obtener el error de predicción. Para ello ejecutaremos la rutina predict con data.test.

Aquí tenemos las tablas de las predicciones y el error final que es del 0.146%:

```
> table(pred,t_true)
    t_true
pred   1   2   3   4   5   6   7   8   9  10
  1  101   3   0  13   0   0   9   7   1   0
  2    1  22   0   0   0   0   0   4   0   0
  3    0   0  47   0   0   4   0   0   0   1
  4    1   1   1 125   0   0   5   1   1   6
  5    0   0   0   0  18   0   0   0   4   1
  6    0   0   2   0   0  25   0   0   0   0
  7    2   0   0  13   0   0  98   0   0   0
  8    3   1   0   1   0   0   0  39   0   0
  9    1   5   0   0   2   0   0   5  84   3
  10   3   0   0   1   1   0   0   2   8 124
> (sum(pred != t_true)/length(t_true))
[1] 0.14625
```

### 3.2 Clasificación con SVM

En este caso también separamos los datos en los que usaremos para training y los de testeo.

Para usar SVM podemos usarlo con diferentes kernel. Probamos con el lineal, el polinómico, de 2 y 3 grados, y por último RBF Gaussian. El procedimiento que hemos seguido es escoger un modelo que tenga el menor error de CV y reajustarlo con todas los datos de learning, entonces volver a predecir con los datos de test y ver el error que obtenemos.

Estos son los datos del error de test obtenidos al ejecutar svm con los diferentes kernel:

Lineal: 7.5%
Poly.2: 7.625%
Poly.3: 6.875%
RBF: 8.875%

Vemos que el mejor es el polinómico de 3 grados. Ahora pasamos hacer el reajuste y obtener el error.

Aquí tenemos las tablas de las predicciones y el error final que es del 0.20%:

```
> table(pred,t_true)
    t_true
pred   1   2   3   4   5   6   7   8   9  10
  1  136   0   0   3   0   0   0   2   0   0
  2    6  35   0   2   0   0   0   0   0   0
  3    0   0  41   0   0   0   0   0   0   2
  4   13   0   7 110   0   0   6   0   0  11
  5    0   0   0   0   0   0   0   0   4   4
  6    0   0  21   0   0  43   0   0   0   0
  7    8   0   0   2   0   0  51   0   0   0
  8   16   5   1   4   0   0   0  18   1   0
  9    3   1  16   1   0   0   0   2  37   4
  10  10   0   2   1   0   1   0   4   2 164
> (sum(pred != t_true)/length(t_true))
[1] 0.20625
```

## 4. Conclusiones

Una vez obtenidos los dos modelos y testeados podemos concluir que dan un error de predicción bastante parecidos. Sería difícil decantarse por uno de los dos aunque con LDA tengamos un error un poco menor.

Finalmente después de todas las pruebas hechas hemos visto que para este problema lo más conveniente era usar un clustering para agrupar las diferentes muestras y poder predecirlas ya que agrupando por las mismas familias no hemos obtenido buenos resultados. Esto suponemos que es debido a que dentro de una misma familia hay especies muy diferentes entre ellas lo cual da problemas a la hora de obtener el modelo.

También hemos visto que era un poco mejor hacer la reducción de las variables a partir del conjunto formado por los tres vectores de variables. Aunque la diferencia tampoco sea muy grande hemos decidido crear el modelo a partir de esto, suponemos que no habría habido mucha diferencia de haberlo hecho con las variables de la unión de los vectores ya reducidos.

Con esta práctica hemos utilizado gran parte de lo visto en teoría Un ejemplo es clustering para obtener las nuevas clases. Random Forest y Relief para la reducción de variables. Dos métodos para obtener modelos como son LDA y SVM. También hemos podido aprender cómo a partir de un problema concreto dividirlo en las diferentes fases necesarias para obtener una resolución de este, partiendo del análisis de los datos iniciales, pasando por la resolución de problemas que estos pueden dar, hasta llegar a obtener el clasificador final de nuestro problema.

Por último, mencionar las cosas que por tiempo no nos ha dado tiempo a realizar pero que vemos interesantes realizar en un futuro. Principalmente seria conseguir un modelo que consiga clasificar las 100 especies diferentes de plantas. Partiendo de lo que tenemos, faltaría otro clasificador que dentro de cada cluster consiga seleccionar la especie que es. También se podría intentar obtener los modelos a partir de otros métodos como puede ser Naive Bayes y ver los resultados obtenidos y si conseguimos mejorar. Otra cosa que sería interesante sería conseguir nuevas muestras de las plantas y ver que realmente funcionan nuestros modelos.