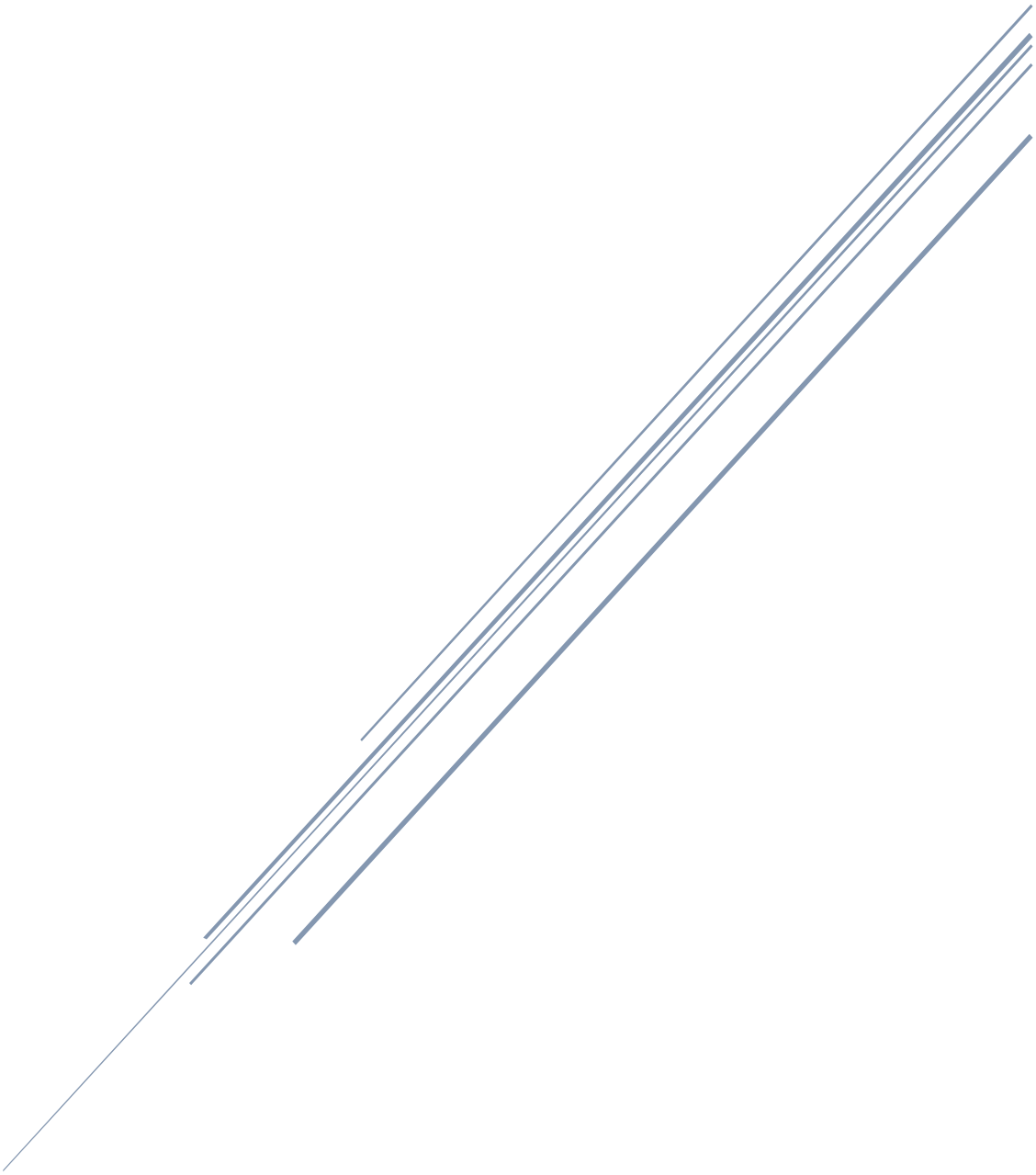
APRENDIZAJE AUTOMÁTICO Y BIG DATA

Proyecto final



FDI-UCM
Daniel García Moreno

Contenido

1. Presentación de los datos

En este proyecto final se va a trabajar con el siguiente conjunto de datos:

Abstract: Dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey.

Data Set Characteristics:	Multivariate	**Number of Instances:**	1473	**Area:**	Life
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	9	**Date Donated**	1997-07-07
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	94857

http://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice

Este conjunto de datos se ha obtenido mediante una encuesta por parte de la National Indonesia Contraceptive. A partir de estos datos se quiere saber la característica del método anticonceptivo que ha utilizado la mujer a partir del contexto de su vida (características demográficas y socio-económicas). Esto es, la duración del método anticonceptivo.

En total, se manejarán 1473 ejemplos. Cada ejemplo está formado por 9 atributos y por la clase. Los atributos son los siguientes:

- Edad de la mujer: edad de la mujer sobre la que se realizó la encuesta. Este atributo tendrá un valor numérico, tal cual la edad.
- Nivel de educación de la mujer: categoría de estudios que tiene la mujer encuestada. Este atributo puede tener cuatro posibles valores (del 1 al 4 incluidos). Cuanto más alto sea el valor, mayor nivel de educación tiene la mujer.
- Nivel de educación del marido: igual que el atributo anterior, pero referido al marido de la mujer encuestada.
- Número de hijos: número de hijos actual que tiene la mujer encuestada con su marido.
- Religión de la mujer: qué religión tiene la mujer encuestada. En este atributo solo se tiene en cuenta si la mujer es devota del Islam (1) o si no es devota del Islam (0).
- ¿Está activa la mujer?: este atributo almacena si trabaja la mujer encuestada actualmente. Se almacena con valor binario (0 si no trabaja y un 1 si trabaja).
- Categoría de trabajo del marido: este atributo tiene 4 posibles valores (del 1 al 4, ambos incluidos).

- Calidad de vida: calidad de vida de la mujer encuestada. Este atributo tiene 4 posibles valores (del 1 al 4, ambos incluidos). Cuanto mayor sea el valor, mejor calidad de vida presentará la mujer.

El resultado de la encuesta se almacena en la siguiente clase:

- Característica método anticonceptivo: se almacena la duración del método anticonceptivo utilizado por la mujer. La clase puede tener 3 posibles valores (del 1 al 3, ambos incluidos). El 1 corresponde a no utilizar método anticonceptivo, el 2 corresponde a utiliza un método anticonceptivo de larga duración y el 3 corresponde a utilizar un método anticonceptivo de corta duración.

ATRIBUTO	VALOR
Edad de la mujer	#Valor
Nivel de educación de la mujer	- [1 2 3 4] +
Nivel de educación del marido	- [1 2 3 4] +
Número de hijos actual del matrimonio	#Valor
Religión de la mujer	[0 1]
¿Trabaja la mujer?	[0 1]
Categoría del trabajo del marido	[1 2 3 4]
Calidad de vida	- [1 2 3 4] +
Exposición a los medios	+ [0 1] -

CLASE	VALOR
Característica método anticonceptivo	[1 2 3]

2. Técnicas de aprendizaje utilizadas

Las técnicas de aprendiza automático estudiadas en la asignatura que encajarían con los datos elegidos para hacer el proyecto serían las siguientes:

- Regresión logística multiclase
- Redes neuronales
- SVM

Los datos a analizar nos obligan a utilizar técnicas compatibles con varias clases, técnicas de clasificación.

En estas técnicas vamos utilizar los datos divididos en tres grupos. Por un lado tendremos los datos de entrenamiento, con los que entrenaremos los sistemas, por otro lado tendremos los datos de validación, con los que evaluaremos los diferentes valores de los parámetros, y los datos de prueba, con los que evaluaremos el porcentaje de éxito de cada técnica, que una vez hayamos encontrado los mejores valores de los parámetros de casa técnica, estarán entrenadas con los datos de entrenamiento más validación.

El Dataset elegido tiene ordenados los ejemplos según el valor de la clase. Por lo tanto, dispone de la siguiente distribución, la cual la podemos observar en la figura 1:

Valores clase | nº fila | nº ejemplos | total ejemplos

1--> 0001-0415 & 1001-1214 --> 415 (65,9%) + 214 (34,1%) = 629

2--> 0416-0643 & 1215-1320 --> 228 (68.2%) + 106 (31,8%) = 334

3--> 0644-1000 & 1321-1473 --> 357 (70%) + 153 (30%) = 510

1473

Figura 1

Teniendo en cuenta la siguiente distribución, para construir los tres grupos de datos con los que trabajaremos deberemos coger ejemplos aleatorios teniendo en cuenta la fila donde se encuentran. Tendremos un 70% de datos de entrenamiento, un 15% de datos de validación y un 15% de datos de prueba. Como los ejemplos vienen agrupados según el valor que tengan en la clase, y existen dos grupos por cada valor a lo largo del documento de datos, los tres grupos que utilizaremos estarán construidos tal y como podemos observar en la figura 2:

Para datos con valor de clase igual a 1:

629 → DE (70%) -> **441** → 1ra parte -> **291** (70% de 415 -> 290,5)
(70% de 629 -> 440,3) 2da parte -> **150** (70% de 214 -> 149.8)
DV (15%) -> **94** → 1ra parte -> **62** (15% de 415 -> 62,25)
(15% de 629 -> 94.35) 2da parte -> **32** (15% de 214 -> 32,1)
DP (15%) -> **94** → 1ra parte -> **62** (15% de 415 -> 62,25)
(15% de 629 -> 94.35) 2da parte -> **32** (15% de 214 -> 32,1)

Para datos con valor de clase igual a 2:

334 → DE (70%) -> **234** → 1ra parte -> **160** (70% de 228 -> 159.6)
(70% de 334 -> 233,8) 2da parte -> **74** (70% de 106 -> 74.2)
DV (15%) -> **50** → 1ra parte -> **34** (15% de 228 -> 34.2)
(15% de 334 -> 50.1) 2da parte -> **16** (15% de 106 -> 15.9)
DP (15%) -> **50** → 1ra parte -> **34** (15% de 228 -> 34.2)
(15% de 334 -> 50.1) 2da parte -> **16** (15% de 106 -> 15.9)

Para datos con valor de clase igual a 3:

510 → DE (70%) -> **358** → 1ra parte -> **251** (70% de 357 -> 249.9)
(70% de 510 -> 357) 2da parte -> **107** (70% de 153 -> 107.1)
DV (15%) -> **76** → 1ra parte -> **53** (15% de 357 -> 53.55)
(15% de 510 -> 76.5) 2da parte -> **23** (15% de 153 -> 22.95)
DP (15%) -> **76** → 1ra parte -> **53** (15% de 357 -> 53.55)
(15% de 510 -> 76.5) 2da parte -> **23** (15% de 153 -> 22.95)

Figura 2

En este anterior recuadro, DE es Datos de Entrenamiento, DV es Datos de Validación y DP es Datos de Prueba. 1ra parte y 2da parte indica que en el documento de datos de ejemplos, los ejemplos que tienen un valor igual a 1, por ejemplo, están colocados en dos partes del documento. En concreto, los datos están ordenados, según el valor de su clase, de la siguiente manera: [[1 2 3] (hasta aquí, llegamos a la fila del documento Dataset nº 1000) [1 2 3]]. La primera parte serían los tres primeros grupos de valores (1 2 3) y la segunda parte sería los tres siguientes grupos (1 2 3).

Aunque la explicación anteriormente descrita para general los grupos de valores con los que se va a trabajar pueda parecer ardua, extravagante y un poco complicada, el objetivo es alterar el documento Dataset lo menos posible, repartiendo los valores de manera equitativa en los tres diferentes grupos, habiendo un porcentaje de valores de clases similar.

Podemos visualizar la idea de "las dos partes del documento" en la figura 3.

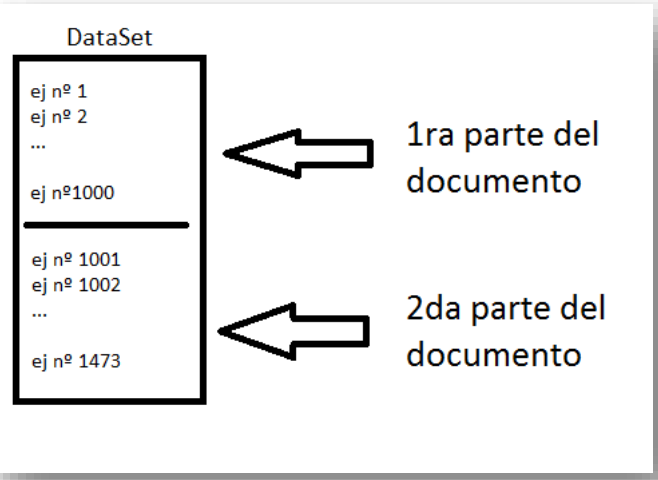


Figura 3

2.1. Regresión logística multiclase

Empezamos a trabajar con los datos utilizando, en primer lugar, la técnica de regresión logística multiclase.

Teniendo en cuenta la distribución anterior de los datos, comenzaremos eligiendo el valor de la variable "Lambda" que mejor trabaje con nuestros datos. Para ello, utilizaremos la función *pintarErrorLambda*. En esta función, trabajaremos con los datos de entrenamiento y con los de validación. Entrenaremos el sistema con distintos valores de lambda (0, 0.25, 0.5, 0.75, 1, 2, 3, 10, 20, 50, 200), y por cada valor de lambda utilizado, obtendremos los porcentajes de acierto, tanto para los datos de entrenamiento como los de validación.

Ejecutando, pues, la función anterior, tal y como podemos observar en la figura 4, obtendremos el lambda con el que entrenaremos el sistema posteriormente. Su implementación la podemos observar en la sección Anexo.

```
#ELEGIR EL MEJOR LAMBDA
pintarErrorLambda(Xent, Yent, Xval, Yval)
```

Figura 4

Como resultado se obtiene, que con 5000 iteraciones, los mejores valores lambdas de lambda con el que se obtienen mejor porcentaje de aciertos sobre los datos de validación son 1 y 50. Por lo tanto, para entrenar el sistema, utilizaremos un valor de lambda igual 1, ya que como podemos observar en la figura 5, 1 ha obtenido mejores porcentajes que 50.

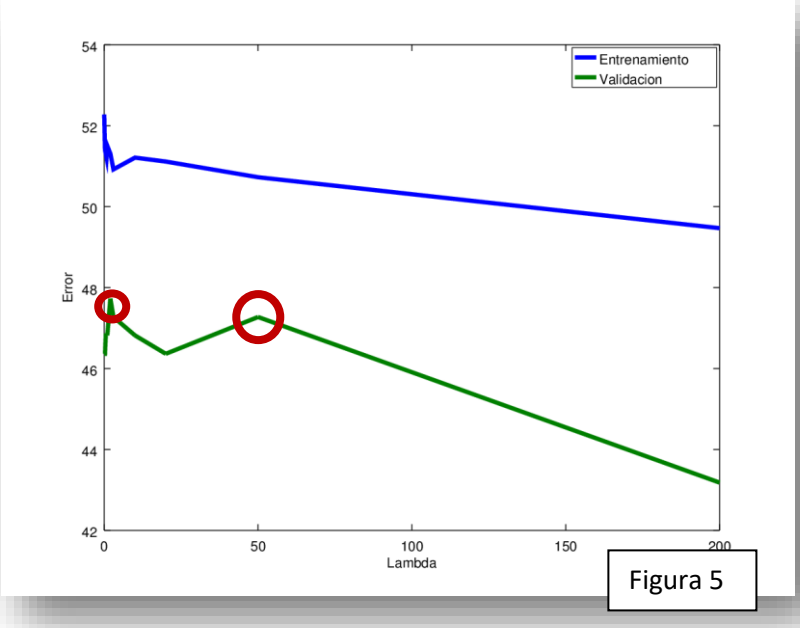


Figura 5

Se ha realizado el paso anterior con distintos valores de iteraciones (50, 100, 500, 1000, 5000 respectivamente al orden de las imágenes), obteniendo los resultados que podemos observar en la figura 6.

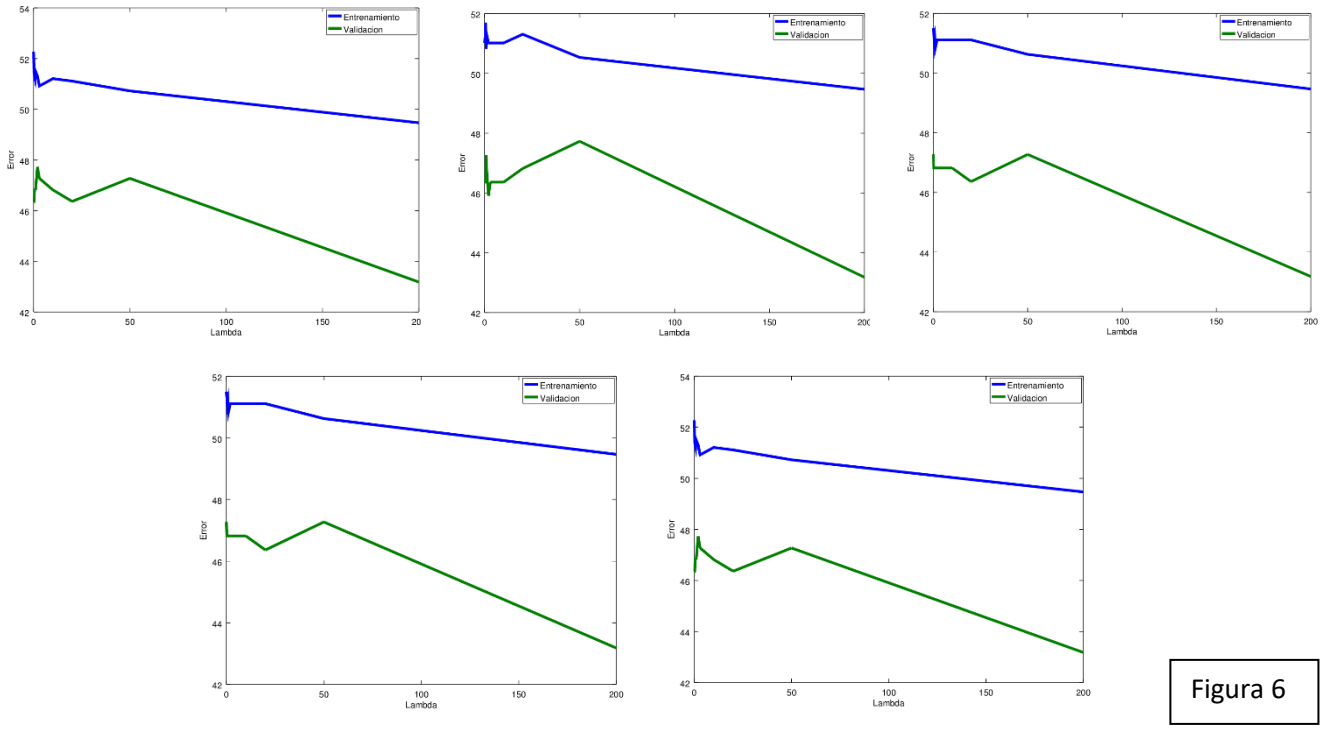


Figura 6

Elegido el valor de lambda 1, vamos a proceder a entrenar al sistema con este valor y con 5000 iteraciones, obteniendo el conjunto de valores de "theta". Para ello, utilizaremos la función *oneVsAll* (la misma utilizada en *pintarErrorLambda*), tal y como podemos observar en la figura 7. Su implementación la podemos ver en la sección Anexo.

```
#ENTRENAR VARIOS CLASIFICADORES CON LAMBDA
[theta] = oneVsAll(Xent_val, Yent_val, num_etiquetas, 1);
```

Figura 7

Entrenado el sistema, vamos a realizar una comparación de los datos obtenido. Se ha calculado el porcentaje de aciertos sobre los datos de entrenamiento, los datos de validación, los datos de entrenamiento y validación juntos, y los datos de prueba. Para ello, utilizamos la función *damePorcentaje*, tal y como podemos ver en la figura 8. Su implementación la podemos ver en la sección Anexo.

```
#OBTENER PORCENTAJE DE ACIERTOS DEL CLASIFICADOR
#DATOS DE ENTRENAMIENTO
p = damePorcentaje(theta, Xunos, Yent);

#DATOS DE VALIDACIÓN
Xunos = [ones(rows(Xval), 1) Xval];
p = damePorcentaje(theta, Xunos, Yval);

#DATOS DE ENTRENAMIENTO + VALIDACIÓN
Xunos = [ones(rows(Xent_val), 1) Xent_val];
p = damePorcentaje(theta, Xunos, Yent_val);

#DATOS DE PRUEBA
Xunos = [ones(rows(Xpru), 1) Xpru];
p = damePorcentaje(theta, Xunos, Ypru);
```

Figura 8

Podemos observar en la figura 9, que los **datos de prueba**, se ha obtenido un porcentaje de éxito del **53,636%**. Por otro lado, en los datos de entrenamiento se ha obtenido un porcentaje del 52,469%, en los datos de validación un 48,636% y en los datos de entrenamiento + evaluación un 51,796%.

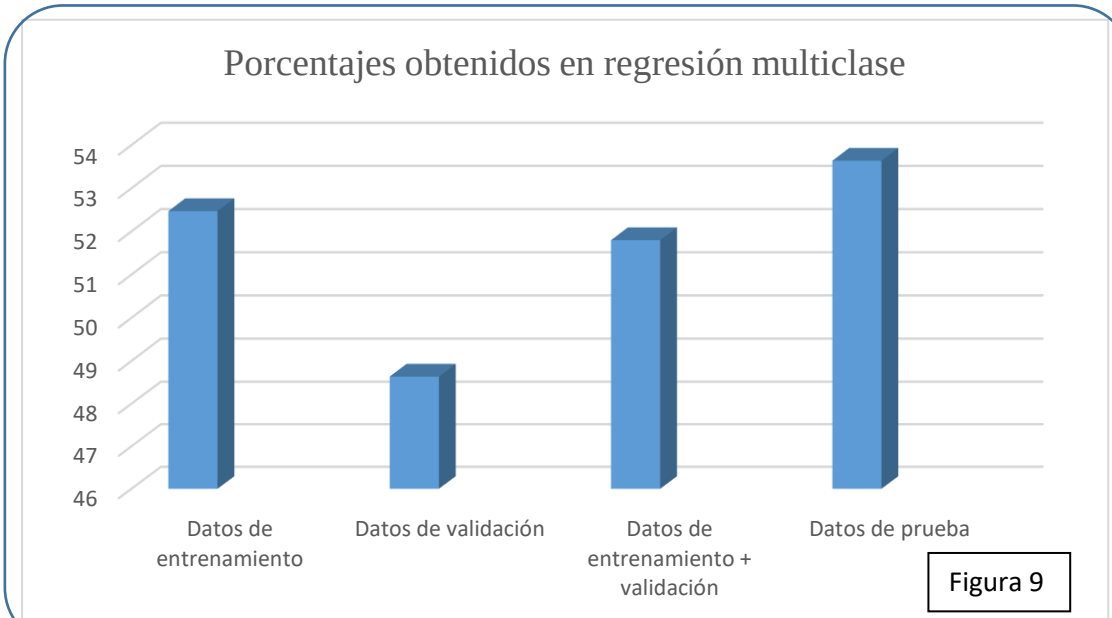


Figura 9

En general, en todas las pruebas que se han realizado, el porcentaje de éxito no es muy elevado, rondando siempre el 50% de éxito de acierto.

2.2. Redes neuronales

Al igual que ocurre en regresión logística multiclase, comenzaremos por elegir el valor de lambda que mejores resultados proporcione. Estudiaremos la evolución de los valores de lambda variando el número de capas ocultas de la red neuronal. Volveremos a hacer uso de la función *pintarErrorLambda*, adecuándola con las funciones de coste implementadas para las redes neuronales. Podemos ver su implementación en la sección Anexo. Ahora los diferentes valores de lambda son los siguientes: (0, 0.25, 0.5, 0.75, 1, 2, 3, 5).

Comenzaremos con 10 capas ocultas. Podemos observar la evolución de los valores de lambda en la figura 10.

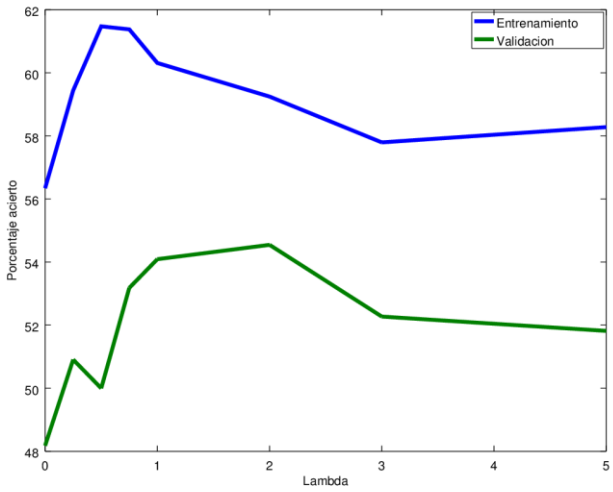


Figura 10

El mejor valor de lambda es 2. Entrenamos el sistema con este valor (entrenamos utilizando los datos de entrenamiento y validación) y los resultados son los siguientes: con los **datos de prueba** obtenemos un porcentaje de aciertos del **58,636%**. Por otro lado, con los datos de entrenamiento, conseguimos un porcentaje del 59,439%, con los datos de validación un 57,727% y con los datos de entrenamiento + validación un 59,138%. Podemos observar estos valores en modo resumen en la figura 11.

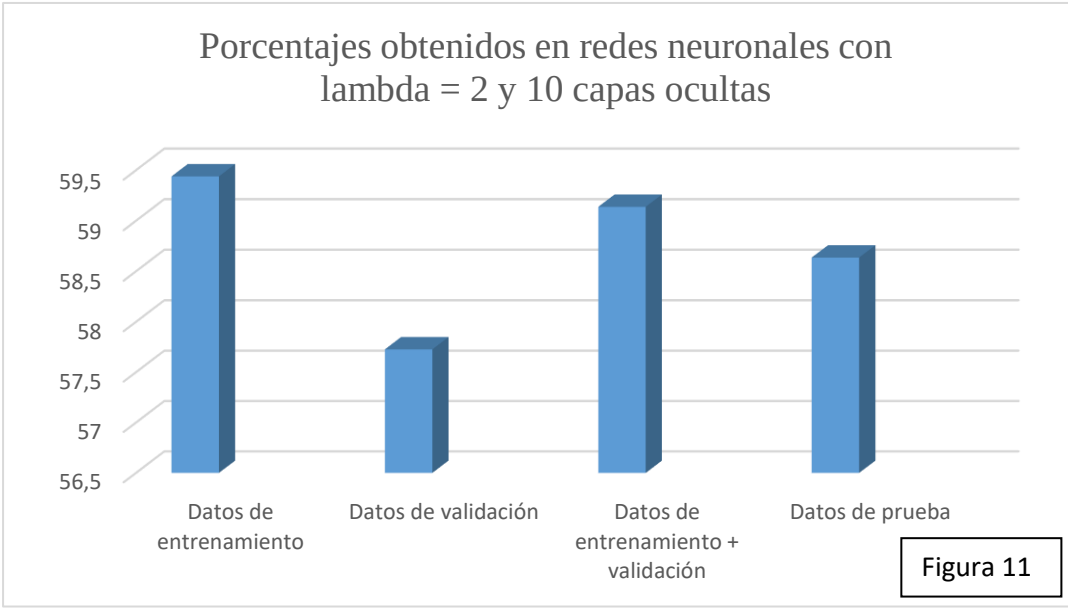


Figura 11

Modificamos el número de capas ocultas y ahora añadimos más, llegando a tener 50 capas. Volvemos a ejecutar la función *pintarErrorLambda*, y obtenemos el resultado que podemos observar en la figura 12.

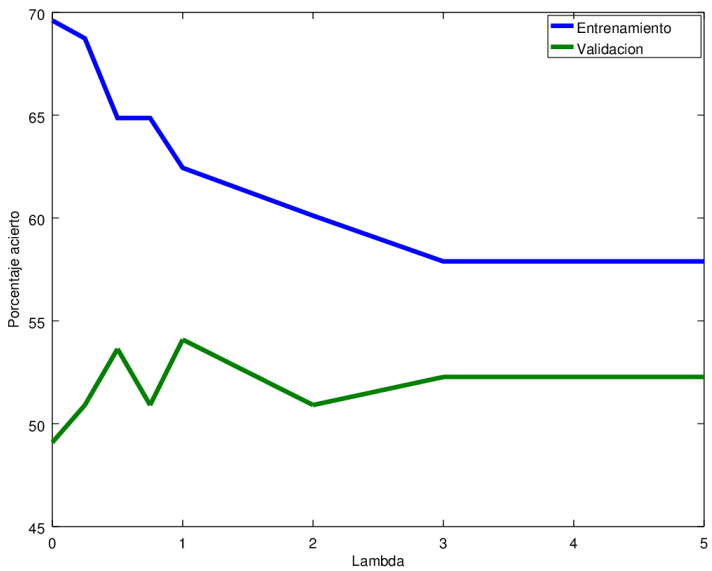


Figura 12

Observamos que el mejor valor de lambda es 1. Volvemos a entrenar a la red neuronal y vemos los siguientes resultados: el porcentaje de éxito con los **datos de prueba** es de **57,273%**, con los datos de entrenamiento se obtiene un porcentaje del 63,214%, con los datos de validación se obtiene un porcentaje del 60,909% y con los datos de entrenamiento y validación un 62,809%.

Podemos observar que al aumentar el número de capas ocultas se ha reducido ligeramente el porcentaje de aciertos. Hay que tener en cuenta que se utilizan pesos iniciales distintos. A modo resumen, podemos observar en la figura 13 la comparación de porcentajes obtenidos con los distintos grupos.

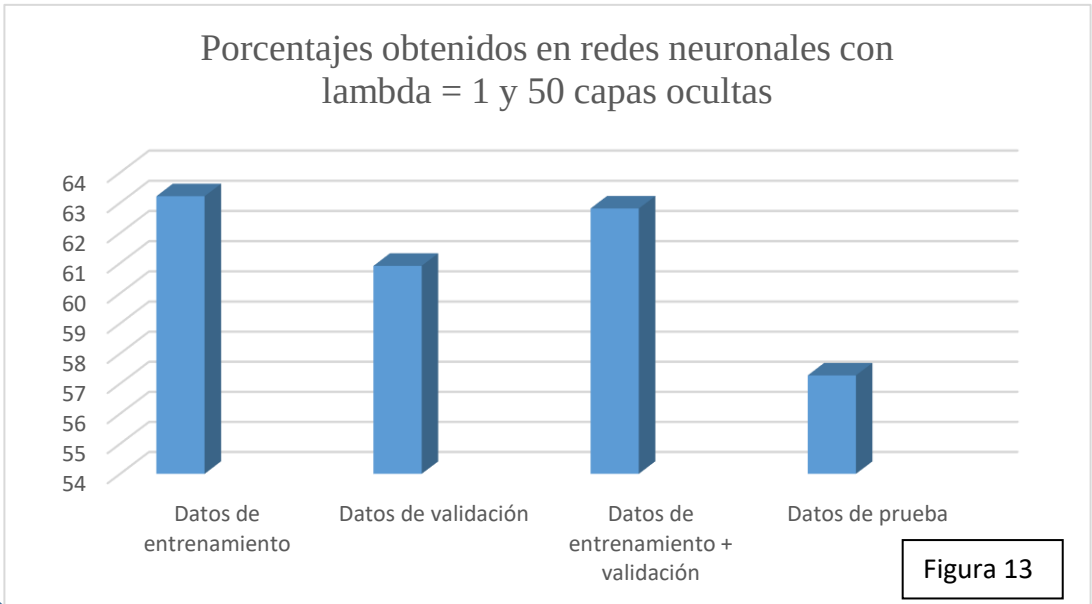


Figura 13

Como tercera prueba, aumentaremos el número de capas ocultas a 100. Con este nuevo valor conseguimos una evolución de lambda que podemos observar en la figura 14.

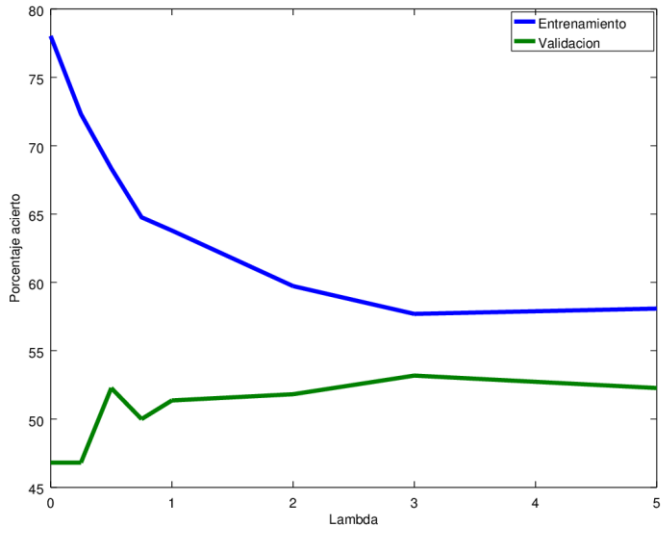


Figura 14

Podemos observar que el valor de lambda que mejores resultados ha obtenido es 3. Entrenamos, pues la red neuronal con este valor y obtenemos los siguientes resultados: con los **datos de prueba**, obtenemos un porcentaje de aciertos del **58,182%**, con los datos de entrenamiento se consigue un porcentaje del 58,083%, con los datos de validación se consigue un porcentaje del 55,909% y con los datos de entrenamiento + validación conseguimos un porcentaje del 57,702%. Al aumentar las capas ocultas se incrementa ligeramente el porcentaje de aciertos de los datos de prueba, que se recuerda, este grupo de datos no ha sido utilizado para entrenar la red neuronal. A modo resumen, podemos observar estos valores en la figura 15.

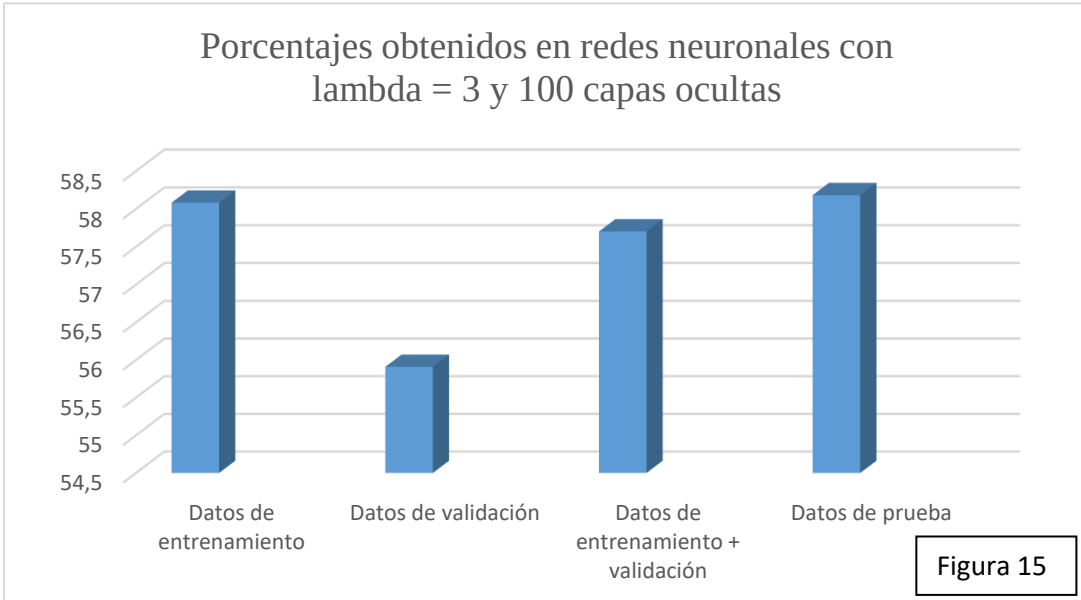


Figura 15

Como última prueba, realizaremos un entrenamiento de una red neuronal con 150 capas ocultas. Con este valor, conseguimos la evaluación de lambda que podemos observar en la figura 16.

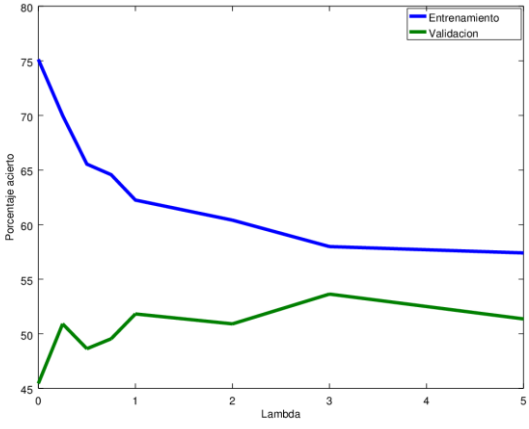


Figura 16

Observamos, que nuevamente, 3 es el valor de lambda que mejor resultados obtiene. Es por ello que entrenaremos la red neuronal con este valor. Una vez hecho esto, obtenemos los siguientes resultados: el porcentaje de aciertos con los **datos de pruebas** es de un **58,636%**, con los datos de entrenamiento se consigue un 57,986%, con los datos de validación se consigue un 57,273% y con los datos de entrenamiento + validación se consigue un porcentaje de acierto del 57,702%. A modo de resumen, podemos visualizar estos valores en la figura 17.

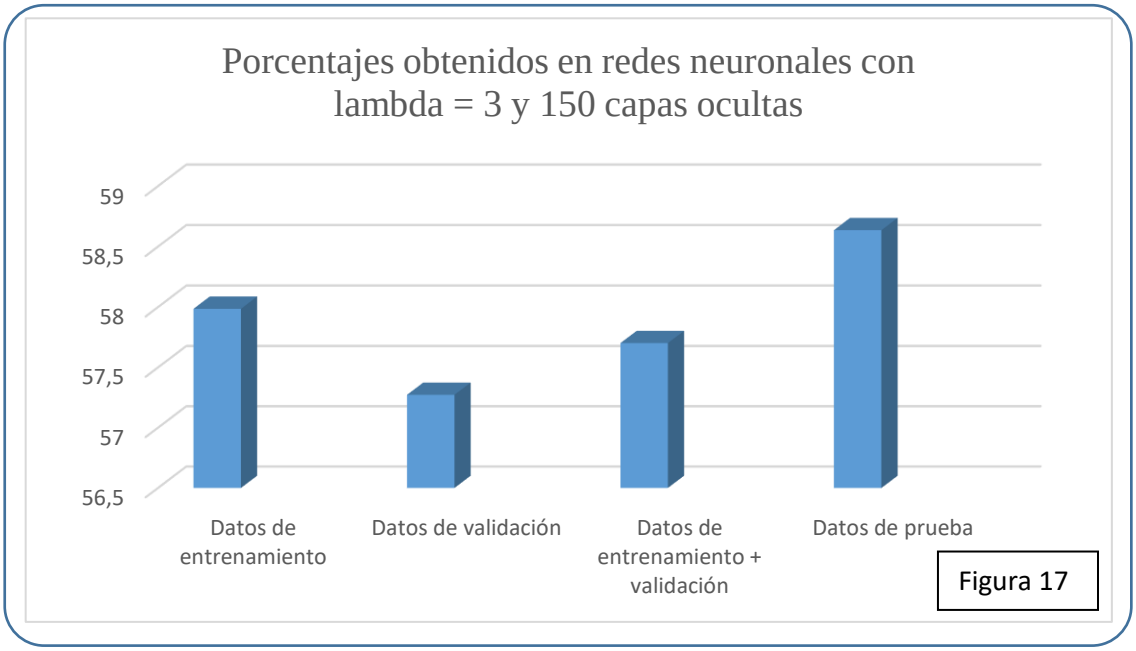


Figura 17

En general, en todas las pruebas, se ha obtenido entre un porcentaje de aciertos de entre 57-58%. Aumentar el número de capas ocultas ha conseguido obtener una ligera mejora de porcentajes de aciertos. Si bien, es cierto que los pesos iniciales han sido distintos en las diferentes pruebas, y este aspecto influye de manera notable en los resultados.

Respecto a la regresión logística multiclase, se ha obtenido unos resultados discretamente mejores, aumentando en un 4% aproximadamente el porcentaje de aciertos.

En la figura 18 podemos observar la evolución de los porcentajes obtenidos con los datos de prueba a los largo de las distintas pruebas. El tiempo de cálculo ha aumentado a medida que aumentaba las capas ocultas, empezando con 10 minutos en el caso de 10 capas hasta los 40 minutos en el caso de las 150 capas ocultas.

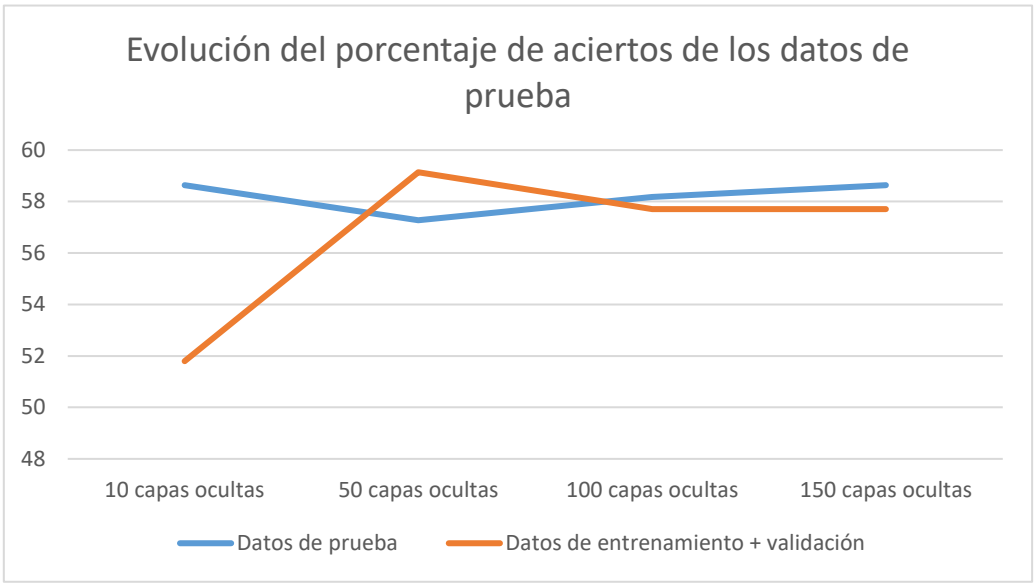

Evolución del porcentaje de aciertos de los datos de prueba
60
58
56
54
52
50
48
10 capas ocultas
50 capas ocultas
100 capas ocultas
150 capas ocultas
Datos de prueba
Datos de entrenamiento + validación

Figura 18

2.3. SVM

Comenzamos la prueba eligiendo el Kernel Gaussiano. Posteriormente, vamos a entrenar las SVM con distintos valores de C y Gamma (0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30). Para ello, haremos uso de la función *obtenerModelosGaussiano*. Su implementación la podemos observar en la sección Anexo. Una vez hemos obtenidos distintos modelos, vamos a calcular el porcentaje de aciertos de cada uno de ellos. Para ellos utilizaremos la función *damePorcentajeGaussiano*, cuya implementación la podemos observar en la sección Anexo.

Los resultados que obtenemos son un poco extraños, ya que para todas las combinaciones de C y Gamma, obtenemos el mismo resultado, un porcentaje de aciertos del **42,7272%.** Es por ello, que decidimos utilizar la funcionalidad de la librería LIBSVM.

La librearía LIBSVM nos proporciona una serie de herramientas con las que poder trabajar con SVM. Para poder utilizar estas herramientas se deben seguir una serie de pasos. En primer lugar, deberemos formatear los datos al formato con el que trabaja LIBSVM. Por un lado, se dará formato a un archivo formado por los datos de entrenamiento y validación, y por otro lado, se dará formato a un archivo formado por los datos de prueba.

Una vez tenemos los datos formateados, comenzaremos las pruebas entrenando la SVM con los valores de por defecto haciendo uso de la herramienta *svm-train*. Previamente, habremos hecho una normalización de los datos con la ayuda de la herramienta *svm*-scale. Realizado el entrenamiento, calcularemos el porcentaje de aciertos del sistema con los datos de prueba haciendo uso de la herramienta *svm-predict*. Se obtiene un porcentaje de aciertos del 45% (99 aciertos de 220 ejemplos de prueba).

Como segunda prueba, haremos uso de una herramienta python proporcionada en la LIBSVM con el que obtendremos los valores de C y Gamma que mejores resultados proporcionen. Para ello, haremos uso de la herramienta *grid.py*, junto a los datos de entrenamiento + validación. Una vez ejecutado el paso anterior, obtenemos que el mejor valor de C es 8 y que el mejor valor de Gamma es 0,5.

Una vez disponemos de los valores adecuados de C y Gamma, entrenaremos de nuevo la SVM indicando estos valores de manera manual. Para ello, volveremos a

utilizar la herramienta *svm-train*. Una vez entrenando, utilizamos los datos de prueba junto a la herramienta *svm-predict* observamos que el porcentaje de aciertos aumenta hasta llegar a un **58,634%.**

Podemos observar de manera detallada los comandos utilizados para la ejecución de los comandos anteriores en la sección Anexo.

3. Comparación de resultados obtenidos

En general, los resultados obtenidos en las tres técnicas utilizadas han sido similares, rondado el 58% de aciertos. La técnica que peores resultados ha obtenidos ha sido SVM. Por un lado, un 43% utilizando el código desarrollado en las prácticas. Utilizando la librería LIBSVM, se consiguen, a priori, unos resultados similares (42%). En cambio, cuando utilizamos unos valores óptimos de C y Gamma con las herramientas de la librería, este resultado aumenta hasta el 58%.

Con la regresión logística, los resultados son ligeramente inferiores a la SVM optimizada. Se queda con un 53%.

Los mejores resultados se los llevan las redes neuronales, con un 58%, gracias a un elevado número de capas ocultas.

En la figura 19 podemos observar un resumen con los resultados.

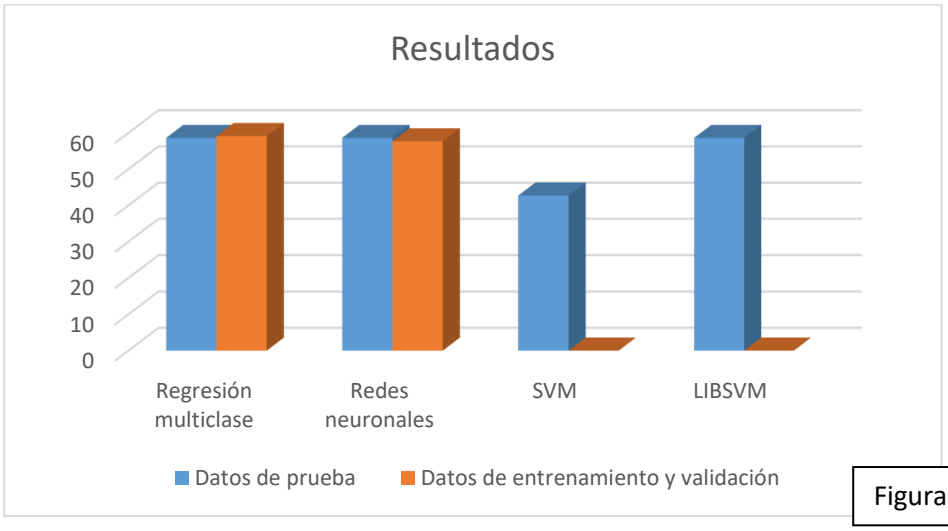


Figura 19

En general, el porcentaje de aciertos es relativamente bajo y ningún sistema ha conseguido unos resultados aceptables. Los datos presentas múltiples combinaciones que ningún sistema de aprendizaje automático de los utilizados consigue acertar un porcentaje amplio de valores.

4. Anexo

Para poder trabajar en el proyecto, lo primero que necesitamos es cargar los datos proporcionados para poder trabajar y volcarlos a las variables que utilizaremos a lo largo de este ejercicio. Atendiendo a la distribución de datos explicada en la sección de Técnica de Aprendizaje utilizadas (Figura 2), ejecutamos el código de la figura 20 para tener las variables de Datos de Entrenamiento, Datos de Validación y Datos de prueba.

El número de la fila donde empiezan los distintos valores de cada clase, que podemos observar, lo podemos contrastar con la Figura 1.

Para las distintas técnicas se ha utilizado el código desarrollado a lo largo de la asignatura, así como código proporcionado en las mismas. Además, en el caso de las SVM se ha hecho uso de una librearía externa, LIBSVM.

```
##CARGAR DATOS - COMÚN
#CARGAR DATOS
X = datos(:, 1:(columns(datos) -1));
Y = datos(:, (columns(datos)));
##CREAR GRUPOS DE DATOS
#1
Xent = X(1:291, :);
Yent = Y(1:291, :);
Xent = [Xent; X(1001:1150, :)];
Yent = [Yent; Y(1001:1150, :)];

Xval = X(292:353, :);
Yval = Y(292:353, :);
Xval = [Xval; X(1151:1182, :)];
Yval = [Yval; Y(1151:1182, :)];

Xpru = X(354:415, :);
Ypru = Y(354:415, :);
Xpru = [Xpru; X(1183:1214, :)];
Ypru = [Ypru; Y(1183:1214, :)];
#2
Xent = [Xent; X(416:575, :)];
Yent = [Yent; Y(416:575, :)];
Xent = [Xent; X(1215:1288, :)];
Yent = [Yent; Y(1215:1288, :)];

Xval = [Xval; X(576:609, :)];
Yval = [Yval; Y(576:609, :)];
Xval = [Xval; X(1289:1304, :)];
Yval = [Yval; Y(1289:1304, :)];

Xpru = [Xpru; X(610:643, :)];
Ypru = [Ypru; Y(610:643, :)];
Xpru = [Xpru; X(1305:1320, :)];
Ypru = [Ypru; Y(1305:1320, :)];
#3
Xent = [Xent; X(644:894, :)];
Yent = [Yent; Y(644:894, :)];
Xent = [Xent; X(1321:1427, :)];
Yent = [Yent; Y(1321:1427, :)];

Xval = [Xval; X(895:947, :)];
Yval = [Yval; Y(895:947, :)];
Xval = [Xval; X(1428:1450, :)];
Yval = [Yval; Y(1428:1450, :)];

Xpru = [Xpru; X(948:1000, :)];
Ypru = [Ypru; Y(948:1000, :)];
Xpru = [Xpru; X(1451:1473, :)];
Ypru = [Ypru; Y(1451:1473, :)];
```

Figura 20

4.1. Implementación para regresión logística multiclase

4.1.1. Código utilizado para la implementación de la regresión logística multiclase

El código utilizado para la regresión logística multiclase es similar al código de la práctica 3 de la asignatura. Se ha añadido la función *pinterErrorLambda* para poder elegir el mejor valor de lambda que mejor trabaje con nosotros.

Por lo tanto, para el proyecto se ha utilizado las siguientes funciones, incluyendo, además, la función *fmincg*, que se proporcionó en su momento con la práctica 3 (figura 21-26):

```
function [all_theta] = oneVsAll(X, y, num_etiquetas, lambda)
  n = columns(X);
  X = [ones(rows(X), 1) X];

  initial_theta = zeros (n + 1, 1);

  options = optimset ( "GradObj" , "on" , "MaxIter" , 100);

  for i = 1:num_etiquetas
    all_theta(i, :) = fmincg (@(t)(firstCostFunction(t , X, ( y == i ), lambda)),
initial_theta, options);
  endfor

endfunction
```

Figura 21

```
function [J, grad] = firstCostFunction(theta, X, y, lambda)

  numCols = columns(X);
  numFils = rows(X);

  grad = (1/numFils) * ((hipotesis(X', theta) - y') * X);
  grad = grad';
  grad(2:numCols, 1) = grad(2:numCols, 1) + ((lambda/numFils) *
theta(2:numCols));


  J = ((1/numFils) * sum((-y .* (log(hipotesis(X', theta)))') - ((1 - y) .* (log(1 -
hipotesis(X', theta)))'))) + ((lambda/(2*numFils)) * sum(theta.^2));

endfunction
```

Figura 22

```
function h = hipotesis(x, theta)
  h = sigmoide(theta' * x);
endfunction
```

Figura 23

```
function s = sigmoide(z)
  s = 1 ./ (1 + exp(-z));
endfunction
```

Figura 24

```
function pintarErrorLambda(X, y, Xval, yval)
  lambda = [0, 0.25, 0.5, 0.75, 1, 2, 3, 10, 20, 50, 200];

  m = rows(X);
  num_etiquetas = 3;

  Theta_ini= zeros(columns(X), 1);

  for i = 1:columns(lambda)
    theta = oneVsAll(X, y, num_etiquetas, lambda(i));

    XonesEnt = [ones(rows(X), 1) X];
    porcentajeEnt(i) = damePorcentaje(theta, XonesEnt, y);

    XonesVal = [ones(rows(Xval), 1) Xval];
    porcentajeVal(i) = damePorcentaje(theta, XonesVal, yval);
  endfor

  plot(lambda, porcentajeEnt, 'LineWidth', 3, lambda, porcentajeVal,
'LineWidth', 3)

  #axis([0 10 0 20])
  legend('Entrenamiento', 'Validacion')
  xlabel('Lambda')
  ylabel('Error')

end
```

Figura 25

```
function porcentaje = damePorcentaje(theta, X, y)
  numFils = rows(X);
  bienPredecidos = 0;
  h = hipotesis(X', theta');

  [maximo clase] = max(h);

  comparacion = (clase' == y);

  bienPredecidos = length(find(comparacion == 1));
  porcentaje = (bienPredecidos/length(comparacion)) * 100;

endfunction
```

Figura 26

4.1.2. Flujo de ejecución de regresión logística multiclase

En modo resumen, este el orden con el que se han ejecutado las funciones para poder obtener resultados (figura 27):

```
#COMANDOS UTILIZADOS -> REGRESION LOGISTICA MULTICLASE
#CARGAR DATOS
num_etiquetas = 3;

#ELEGIR EL MEJOR LAMBDA
pintarErrorLambda(Xent, Yent, Xval, Yval)
#MEJOR LAMBDA -> 1 y 50

#UNIR LOS DATOS DE ENTRENAMIENTO Y VALIDACIÓN
Xent_val = [Xent; Xval];
Yent_val = [Yent; Yval];

#ENTRENAR VARIOS CLASIFICADORES CON LAMBDA
[theta] = oneVsAll(Xent_val, Yent_val, num_etiquetas, 1);

#OBTENER PORCENTAJE DE ACIERTOS DEL CLASIFICADOR
#DATOS DE ENTRENAMIENTO
Xunos = [ones(rows(Xent), 1) Xent];
p = damePorcentaje(theta, Xunos, Yent);
# RESULTADO: p =  52.469

#DATOS DE VALIDACIÓN
Xunos = [ones(rows(Xval), 1) Xval];
p = damePorcentaje(theta, Xunos, Yval);
# RESULTADO: p =  48.636

#DATOS DE ENTRENAMIENTO + VALIDACIÓN
Xunos = [ones(rows(Xent_val), 1) Xent_val];
p = damePorcentaje(theta, Xunos, Yent_val);
# RESULTADO: p =  51.796

#DATOS DE PRUEBA
Xunos = [ones(rows(Xpru), 1) Xpru];
p = damePorcentaje(theta, Xunos, Ypru);
# RESULTADO: p =  53.636
```

Figura 27

4.2. Código utilizado para las redes neuronales

4.2.1. Código utilizado para la implementación de las redes neuronales

El código utilizado para la implementación de las redes neuronales es similar al utilizado en la práctica 4 de la asignatura. Se ha añadido la función *pintarErrorLambda* para poder evaluar que lambda trabaja mejor.

Por lo tanto, para el proyecto se ha utilizado las siguientes funciones, incluyendo, además, la función *fmincg*, que se proporcionó en su momento con la práctica 4 (figura 28-34):

```
function  W = inicializaPesos(L_in, L_out)
  eini = sqrt(6)/sqrt(L_in + L_out);
  sup = eini;
  inf = -eini;

  W = [inf + (sup - inf) * rand(L_out, 1 + L_in )];
endfunction
```

Figura 28

```
function porcentaje = damePorcentaje(Theta1, Theta2, X, y)
  m = rows(X);

  a1 = [ones(rows(X), 1) X];
  z2 = Theta1 * a1';
  a2 = sigmoide(z2);
  a2 = [ones(1, columns(a2)); a2];
  z3 = Theta2 * a2; %aqui
  a3 = sigmoide(z3);
  h = a3; %10x5000

  [maximo clase] = max(h);

  comparacion = (clase' == y);

  bienPredecidos = length(find(comparacion == 1));

  porcentaje = (bienPredecidos/m) * 100;

endfunction
```

Figura 29

```
function pintarErrorLambda(X, y, Xval, yval, Theta1_inicial, Theta2_inicial)
  lambda = [0, 0.25, 0.5, 0.75, 1, 2, 3, 5];

  m = rows(X);
  num_etiquetas = 3;
  num_entradas = 9;
  num_ocultas = 150;

  Theta_ini= zeros(columns(X), 1);

  params_rn_inicial = [Theta1_inicial(:) ; Theta2_inicial(:)];
  options = optimset('MaxIter', 5000);

  for i = 1:columns(lambda)
    #theta = oneVsAll(X, y, num_etiquetas, lambda(i));

    costFunction = @(t) costeRN(t,num_entradas, num_ocultas, num_etiquetas,
X, y, lambda(i));
    [params_rn, J] = fmincg(costFunction, params_rn_inicial, options);
    Theta11 = reshape(params_rn(1:num_ocultas * (num_entradas + 1)),
num_ocultas, (num_entradas + 1));
    Theta21 = reshape(params_rn((1 + (num_ocultas * (num_entradas +
1))):end), num_etiquetas, (num_ocultas + 1));
    #pred = damePorcentaje(Theta11, Theta21, X, y);

    #XonesEnt = [ones(rows(X), 1) X];
    #porcentajeEnt(i) = damePorcentaje(theta, XonesEnt, y);
    porcentajeEnt(i) = damePorcentaje(Theta11, Theta21, X, y);

    #XonesVal = [ones(rows(Xval), 1) Xval];
    #porcentajeVal(i) = damePorcentaje(theta, XonesVal, yval);
    porcentajeVal(i) = damePorcentaje(Theta11, Theta21, Xval, yval);
  endfor

  plot(lambda, porcentajeEnt, 'LineWidth', 3, lambda, porcentajeVal,
'LineWidth', 3)

  #axis([0 10 0 20])
  legend('Entrenamiento', 'Validacion')
  xlabel('Lambda')
  ylabel('Porcentaje acierto')

end
```

Figura 30

```
function  [J grad] = costeRN (params_rn, num_entradas, num_ocultas,
num_etiquetas, X, y, lambda)
  Theta1 = reshape(params_rn(1:num_ocultas * (num_entradas + 1)),
num_ocultas, (num_entradas + 1));
  Theta2 = reshape(params_rn((1 + (num_ocultas * (num_entradas + 1))):end),
num_etiquetas, (num_ocultas + 1));
  m = rows(X);

  yidentidad = eye(num_etiquetas); #generar matriz identidad
  ycodificada = yidentidad(y, :);

  a1 = [ones(rows(X), 1) X];
  z2 = Theta1 * a1';
  a2 = sigmoide(z2);
  a2 = [ones(1, columns(a2)); a2];
  z3 = Theta2 * a2; %aqui
  a3 = sigmoide(z3);
  h = a3;


  J = sum(sum((((-ycodificada) .* log(h)') - ((1 - ycodificada) .* log(1 - h)'))))/m;
  J = J + (lambda/(2 * m)) * (sum(sum(Theta1(:, 2:end) .^ 2)) +
sum(sum(Theta2(:, 2:end) .^ 2)));

  sig3 = a3' - ycodificada; %CUIDAO
  sig2 = (Theta2' * sig3') .* sigmoideDerivada([ones(1, columns(z2)); z2]);

  d1 = sig2(2:end, :) * a1;
  d2 = sig3' * a2';

  grad1 = ((1/m) * d1);
  grad2 = ((1/m) * d2);

  grad1(:, 2:end) += (lambda/m) * Theta1(:, 2:end);
  grad2(:, 2:end) += (lambda/m) * Theta2(:, 2:end);
  grad = [grad1(:); grad2(:)];

endfunction
```

Figura 31

```
function s = sigmoide(z)
  s = 1 ./ (1 + exp(-z));
endfunction
```

Figura 32

```
function sd = sigmoideDerivada(z)
  sd = sigmoide(z) .* (1 - sigmoide(z));
endfunction
```

Figura 33

```
function h = hipotesis(x, theta)
  h = sigmoide(theta' * x);
endfunction
```

Figura 34

4.2.2. Flujo de ejecución de las redes neuronales

En modo resumen, este el orden con el que se han ejecutado las funciones para poder obtener resultados (figura 35):

```
##########################################################################
##############################################
#COMANDOS UTILIZADOS -> REDES NEURONALES

##############################################
##10 CAPAS

#CARGAR DATOS
num_etiquetas = 3;
num_entradas = 9;
num_ocultas = 10;

#INICIALIZAR PESOS ALEATORIOS
Theta1_inicial = inicializaPesos(num_entradas, num_ocultas);
Theta2_inicial = inicializaPesos(num_ocultas, num_etiquetas);

#ELEGIR MEJOR LAMBDA
pintarErrorLambda(Xent, Yent, Xval, Yval, Theta1_inicial, Theta2_inicial)

params_rn_inicial = [Theta1_inicial(:) ; Theta2_inicial(:)];
options = optimset('MaxIter', 5000);

#UNIR LOS DATOS DE ENTRENAMIENTO Y VALIDACIÓN
Xent_val = [Xent; Xval];
Yent_val = [Yent; Yval];

#ENTRENAR LA RED NEURONAL CON MEJOR LAMBDA
costFunction = @(t) costeRN(t,num_entradas, num_ocultas, num_etiquetas,
Xent_val, Yent_val, 2);
[params_rn, J] = fmincg(costFunction, params_rn_inicial, options);
Theta11 = reshape(params_rn(1:num_ocultas * (num_entradas + 1)),
num_ocultas, (num_entradas + 1));
Theta21 = reshape(params_rn((1 + (num_ocultas * (num_entradas + 1))):end),
num_etiquetas, (num_ocultas + 1));

#OBTENER PORCENTAJES CON LOS DISTINTOS GRUPOS DE DATOS
pred = damePorcentaje(Theta11, Theta21, Xent, Yent)
#pred = 59.439
```

```
pred = damePorcentaje(Theta11, Theta21, Xval, Yval)
#pred = 57.727

pred = damePorcentaje(Theta11, Theta21, Xent_val, Yent_val)
#pred = 59.138

pred = damePorcentaje(Theta11, Theta21, Xpru, Ypru)
#pred = 58.636

##############################################
##50 CAPAS

#CARGAR DATOS
num_etiquetas = 3;
num_entradas = 9;
num_ocultas = 50;

#INICIALIZAR PESOS ALEATORIOS
Theta1_inicial = inicializaPesos(num_entradas, num_ocultas);
Theta2_inicial = inicializaPesos(num_ocultas, num_etiquetas);

#ELEGIR MEJOR LAMBDA
pintarErrorLambda(Xent, Yent, Xval, Yval, Theta1_inicial, Theta2_inicial)

params_rn_inicial = [Theta1_inicial(:) ; Theta2_inicial(:)];
options = optimset('MaxIter', 5000);

#UNIR LOS DATOS DE ENTRENAMIENTO Y VALIDACIÓN
Xent_val = [Xent; Xval];
Yent_val = [Yent; Yval];

#ENTRENAR LA RED NEURONAL CON MEJOR LAMBDA
costFunction = @(t) costeRN(t,num_entradas, num_ocultas, num_etiquetas,
Xent_val, Yent_val, 1);
[params_rn, J] = fmincg(costFunction, params_rn_inicial, options);
Theta11 = reshape(params_rn(1:num_ocultas * (num_entradas + 1)),
num_ocultas, (num_entradas + 1));
Theta21 = reshape(params_rn((1 + (num_ocultas * (num_entradas + 1))):end),
num_etiquetas, (num_ocultas + 1));

#OBTENER PORCENTAJES CON LOS DISTINTOS GRUPOS DE DATOS
pred = damePorcentaje(Theta11, Theta21, Xent, Yent)
#pred = 63.214

pred = damePorcentaje(Theta11, Theta21, Xval, Yval)
#pred = 60.909

pred = damePorcentaje(Theta11, Theta21, Xent_val, Yent_val)
#pred = 62.809
```

```
pred = damePorcentaje(Theta11, Theta21, Xpru, Ypru)
#pred = 57.273

##############################################
##100 CAPAS

#CARGAR DATOS
num_etiquetas = 3;
num_entradas = 9;
num_ocultas = 100;

#INICIALIZAR PESOS ALEATORIOS
Theta1_inicial = inicializaPesos(num_entradas, num_ocultas);
Theta2_inicial = inicializaPesos(num_ocultas, num_etiquetas);

#ELEGIR MEJOR LAMBDA
pintarErrorLambda(Xent, Yent, Xval, Yval, Theta1_inicial, Theta2_inicial)

params_rn_inicial = [Theta1_inicial(:) ; Theta2_inicial(:)];
options = optimset('MaxIter', 5000);

#UNIR LOS DATOS DE ENTRENAMIENTO Y VALIDACIÓN
Xent_val = [Xent; Xval];
Yent_val = [Yent; Yval];

#ENTRENAR LA RED NEURONAL CON MEJOR LAMBDA
costFunction = @(t) costeRN(t,num_entradas, num_ocultas, num_etiquetas,
Xent_val, Yent_val, 3);
[params_rn, J] = fmincg(costFunction, params_rn_inicial, options);
Theta11 = reshape(params_rn(1:num_ocultas * (num_entradas + 1)),
num_ocultas, (num_entradas + 1));
Theta21 = reshape(params_rn((1 + (num_ocultas * (num_entradas + 1))):end),
num_etiquetas, (num_ocultas + 1));

#OBTENER PORCENTAJES CON LOS DISTINTOS GRUPOS DE DATOS
pred = damePorcentaje(Theta11, Theta21, Xent, Yent)
#pred = 58.083

pred = damePorcentaje(Theta11, Theta21, Xval, Yval)
#pred = 55.909

pred = damePorcentaje(Theta11, Theta21, Xent_val, Yent_val)
#pred =  57.702

pred = damePorcentaje(Theta11, Theta21, Xpru, Ypru)
#pred = 58.182
```

```
##############################################
##150 CAPAS

#CARGAR DATOS
num_etiquetas = 3;
num_entradas = 9;
num_ocultas = 150;

#INICIALIZAR PESOS ALEATORIOS
Theta1_inicial = inicializaPesos(num_entradas, num_ocultas);
Theta2_inicial = inicializaPesos(num_ocultas, num_etiquetas);

#ELEGIR MEJOR LAMBDA
pintarErrorLambda(Xent, Yent, Xval, Yval, Theta1_inicial, Theta2_inicial)

params_rn_inicial = [Theta1_inicial(:) ; Theta2_inicial(:)];
options = optimset('MaxIter', 5000);

#UNIR LOS DATOS DE ENTRENAMIENTO Y VALIDACIÓN
Xent_val = [Xent; Xval];
Yent_val = [Yent; Yval];

#ENTRENAR LA RED NEURONAL CON MEJOR LAMBDA
costFunction = @(t) costeRN(t,num_entradas, num_ocultas, num_etiquetas,
Xent_val, Yent_val, 3);
[params_rn, J] = fmincg(costFunction, params_rn_inicial, options);
Theta11 = reshape(params_rn(1:num_ocultas * (num_entradas + 1)),
num_ocultas, (num_entradas + 1));
Theta21 = reshape(params_rn((1 + (num_ocultas * (num_entradas + 1))):end),
num_etiquetas, (num_ocultas + 1));

#OBTENER PORCENTAJES CON LOS DISTINTOS GRUPOS DE DATOS
pred = damePorcentaje(Theta11, Theta21, Xent, Yent)
#pred = 57.986

pred = damePorcentaje(Theta11, Theta21, Xval, Yval)
#pred = 57.273

pred = damePorcentaje(Theta11, Theta21, Xent_val, Yent_val)
#pred =  57.702

pred = damePorcentaje(Theta11, Theta21, Xpru, Ypru)
#pred = 58.636
```

Figura 35

4.3. Código utilizado para las SVM

4.3.1. Código utilizado para la implementación de las SVM

El código utilizado en la práctica es el obtenido al descargar la librería LIBSVM desde la web https://www.csie.ntu.edu.tw/~cjlin/libsvm/

Por otro lado, se ha utilizado el código de la práctica 6. Concretamente, se han utilizado las siguientes funciones (figura 36-38):

```
function model = obtenerModelosGaussiano(X, y)
  valores = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30];
  fila = 1;
  for i = valores
    for j = valores
      model(fila) = svmTrain(X, y, i, @(x1 , x2) gaussianKernel (x1 , x2 , j));
      fila++;
    endfor
  endfor

endfunction
```

Figura 36

```
function sim = gaussianKernel(x1, x2, sigma)
  sumatorio = sum((x1 - x2) .^ 2);
  sim = exp(-((sumatorio)/(2 * (sigma ^ 2))));
end function
```

Figura 37

```
function p = dameMejorPorcentajeGaussiano(models, Xval, yval)
  m = columns(models);
  valores = [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30];
  fila = 1;

  for i = 1:m
    bienPredecidos = length(find((svmPredict(models(i), Xval) == yval) == 1));
    p(i) = (bienPredecidos/rows(yval)) * 100;
  endfor

  for i = valores
    for j = valores
      fprintf(['C: %f; sigma: %f; porcentaje: %f\n'], i, j, p(fila));
      fila++;
    endfor
  endfor

  [maximo ind] = max(p);
  i = ceil(ind/columns(valores));
  j = ind - ((i - 1) * columns(valores));
  fprintf(['Valor optimo->C: %f; sigma: %f; porcentaje: %f\n'], valores(i),
valores(j), maximo);

endfunction
```

Figura 38

4.3.2. Flujo de ejecución de las SVM

Para la ejecución de las funciones desarrolladas en la práctica 6, se han realizado las siguientes llamadas. Podemos ver las llamadas en la figura 39.

```
#COMANDOS UTILIZADOS -> SVM

##KERNEL GAUSSIANO
#UNIR LOS DATOS DE ENTRENAMIENTO Y VALIDACIÓN
Xent_val = [Xent; Xval];
Yent_val = [Yent; Yval];
#OBTENER MODELOS Y PORCENTAJE
model = obtenerModelosGaussiano(Xent_val, Yent_val);
p = damePorcentajeGaussiano(model, Xpru, Ypru);
```

Figura 39

A partir de la guía que facilita LIBSVM en su web más la ayuda de mi compañero de prácticas en la asignatura para poder formatear los datos, se ejecutan los siguientes comandos para poder evaluar la eficacia de la técnica SVM. Podemos ver los pasos en la figura 40.

```
#UNIR LOS DATOS DE ENTRENAMIENTO Y VALIDACIÓN
Xent_val = [Xent; Xval];
Yent_val = [Yent; Yval];

formato = sparse(Xent_val);
libsvmwrite('Xent_val', Yent_val, formato);

formato = sparse(Xpru);
libsvmwrite('Xpru', Ypru, formato);

#EJECUCION
C:\Users\d_dan\Desktop\libsvm-3.22\windows>svm-scale -l 0 -u 1 -s range1
Xent_val > Xent_val.scale

C:\Users\d_dan\Desktop\libsvm-3.22\windows>svm-scale -r range1 Xpru >
Xpru.scale

C:\Users\d_dan\Desktop\libsvm-3.22\windows>svm-train Xent_val.scale
*
optimization finished, #iter = 407
nu = 0.693529
obj = -555.648353, rho = -0.517666
nSV = 577, nBSV = 561
*
optimization finished, #iter = 504
nu = 0.856019
obj = -794.606579, rho = -2.153614
nSV = 837, nBSV = 825
```

```
*
optimization finished, #iter = 356
nu = 0.773513
obj = -540.909093, rho = -0.549161
nSV = 559, nBSV = 549
Total nSV = 1156

C:\Users\d_dan\Desktop\libsvm-3.22\windows>svm-predict Xpru.scale
Xent_val.scale.model svmPrueba
Accuracy = 45% (99/220) (classification)

##OBTENER MEJORES VALORES DE C Y GAMMA
C:\Users\d_dan\Desktop\libsvm-3.22\tools>C:\Python27\python grid.py
Xent_val.scale
...
8.0 0.5 53.0726

##OBTENER PORCENTAJE CON EL MEJOR VALOR DE C Y GAMMA
C:\Users\d_dan\Desktop\libsvm-3.22\windows>svm-train -c 8 -g 0.5
Xent_val.scale
.*.*
optimization finished, #iter = 1778
nu = 0.589145
obj = -3648.093459, rho = -1.327679
nSV = 528, nBSV = 451
.*
optimization finished, #iter = 1772
nu = 0.687757
obj = -4967.717993, rho = -2.869739
nSV = 707, nBSV = 628
.*
optimization finished, #iter = 1425
nu = 0.684537
obj = -3727.674947, rho = -0.641048
nSV = 525, nBSV = 461
Total nSV = 1067

C:\Users\d_dan\Desktop\libsvm-3.22\windows>svm-predict Xpru.scale
Xent_val.scale.model svmPrueba
Accuracy = 58.6364% (129/220) (classification)
```

Figura 40