Predicción de resultados académicos en el Open University Learning Analytics Dataset

Alberto Terceño Ortega

Aprendizaje Automático y Big Data

4º Doble grado en ingeniería informática y matemáticas

Índice

1. Descripción general del proyecto

Este proyecto consiste en el diseño y la implementación de un sistema de aprendizaje automático para la predicción de resultados académicos a partir de datos de alumnos de la Open University de Reino Unido, la cual posee 200.000 estudiantes y opera de manera similar a la UNED española.

El objetivo consiste en dar una tasa de efectividad para diversos métodos de aprendizaje automático vistos en la asignatura de Aprendizaje Automático y Big Data. La mayor parte del código utilizado para el proyecto ha sido obtenido del propio código desarrollado para las prácticas de la asignatura. También se han utilizado archivos y funciones proporcionados en las prácticas que han resultado de gran utilidad.

El problema que se ha pensado para el proyecto es el siguiente: dado un conjunto de alumnos con información demográfica asociada a los mismos, ¿es posible predecir su resultado académico para una determinada asignatura?

Este problema es un problema de clasificación, pues tenemos que predecir, tras haber entrenado los modelos, si un alumno aprobará o suspenderá una asignatura. Este problema de clasificación es binario (notas de los alumnos codificadas como 1 - aprobado y 0 - suspenso) y para resolverlo utilizaremos regresión logística y SVM.

Por otro lado, nos preguntamos si podemos afinar más este problema de clasificación. ¿Por qué no predecir de manera más detallada el resultado académico de cada alumno? Para ello, gracias a la información de nuestro dataset, podemos diseñar un problema de clasificación multiclase, donde podremos predecir si un alumno aprobará, sacará matrícula de honor, suspenderá o abandonará una asignatura. En esta parte utilizaremos regresión logística multiclase y redes neuronales.

La última parte del proyecto tiene que ver con la obtención de los distintos perfiles académicos y demográficos que se encuentran matriculados en una asignatura determinada. Podremos obtener un perfil aproximado utilizando clustering por medio del algoritmo k-means.

Por último, hay que mencionar que este proyecto da una pequeña noción de lo que el proyecto OU Analyse pretende abarcar. Este proyecto, dirigido por miembros de la Open University, pretende detectar alumnos con alta probabilidad de suspenso a partir de los datos proporcionados por los VLE (Virtual Learning Environments, como por ejemplo el Campus Virtual de la UCM) y plasmarlo en un dashboard.
El mismo equipo de OU Analyse es el que ha lanzado el dataset que utilizamos en el proyecto. Además, este grupo de investigadores de la Open University ha publicado unas cuantas publicaciones muy interesantes que pueden consultarse al final de este

documento en la sección de referencias. En una de ellas (link) se puede ver como se usan como predictores datos demográficos y datos de interacción con los VLE.
Esta publicación me ha servido de inspiración a la hora de trazar las líneas generales del proyecto.
En ella podemos observar resultados que afirmar que los datos de interacción son mejores predictores que los datos demográficos, como es de esperar.
La decisión de no incluir los datos de interacción en los datos de entrenamiento que utilizaré en el proyecto se debe a que no intento perseguir los mejores resultados y analizar la efectividad de los atributos, sino que el verdadero objetivo es utilizar un conjunto de atributos lo suficientemente amplio para poder ser usado por las distintas técnicas de la asignatura, así como poder aplicar sobre él diversas pruebas de error (sesgo y varianza).

El proyecto OU Analyse y el dataset son muy amplios, pero ambos exceden muchos ámbitos de la asignatura (por ejemplo, se podría diseñar un sistema que vaya entrenando un sistema de aprendizaje automático a medida que recibe información de interacciones con el VLE por medio de un streaming), por lo que he decidido ceñirme a un pequeño subconjunto del dataset tal y como detallo en la siguiente sección.

A continuación, en la figura 1, podemos observar un diagrama del diseño y el funcionamiento de nuestro proyecto. Las distintas partes del mismo se describen de manera más detallada en las secciones posteriores.

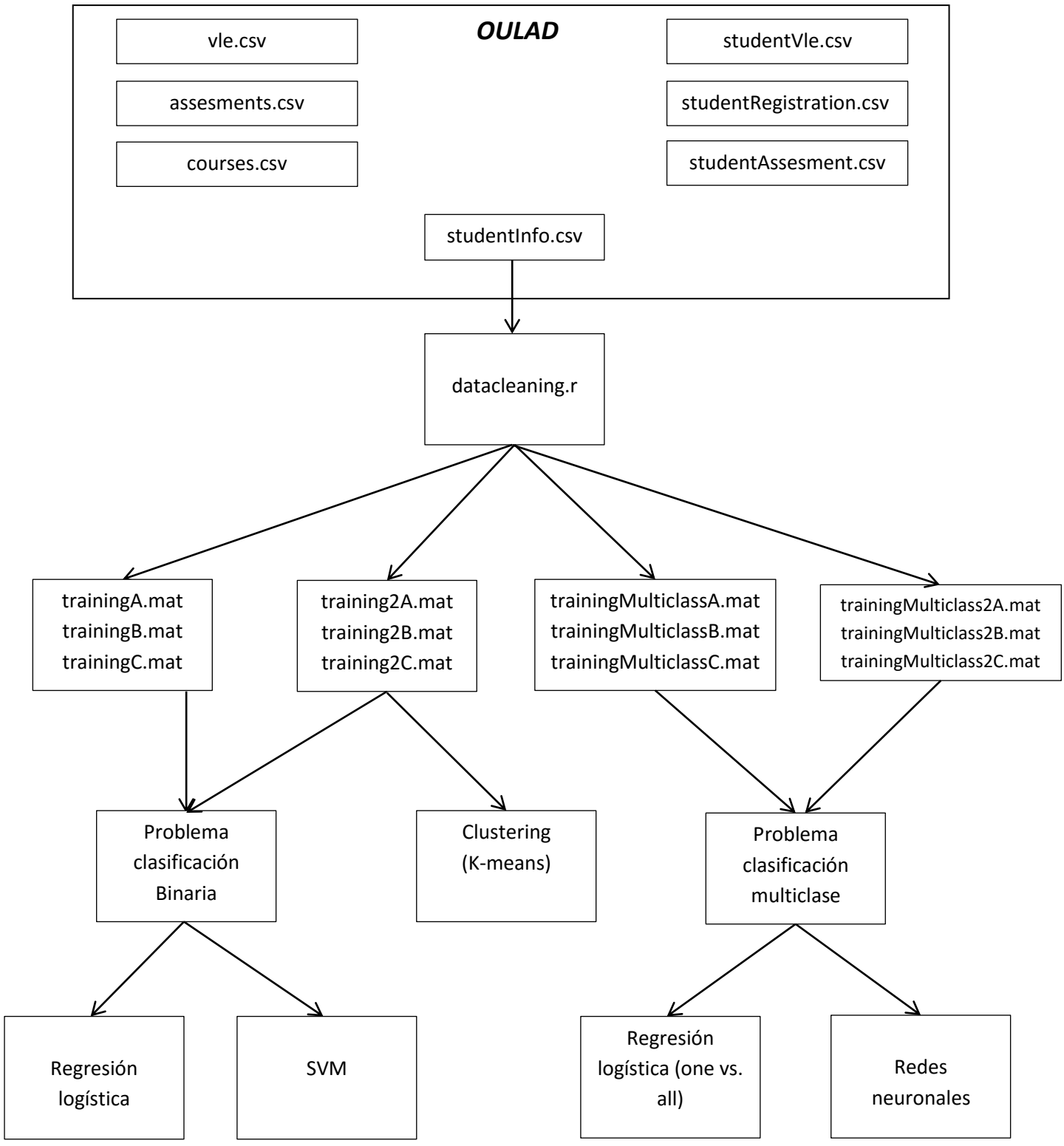


Figura 1: Diagrama del proyecto

2. Open University Learning Analytics Dataset

El Open University Learning Analytics Dataset (OULAD) contiene información sobre asignaturas, alumnos y diversas interacciones con el VLE. Todos estos datos están anonimizados y su origen está en el proyecto OU Analyse del Knowledge Media Institute, ubicado en la Open University de Reino Unido.
Las diversas tablas del dataset poseen formato csv y todas ellas poseen claves primarias (id's) que permiten identificar de manera única asignaturas, alumnos e interacciones entre otros entes.
En la figura 2 podemos apreciar un esquema de las diferentes tablas del dataset, así como las relaciones y dependencias entre ellas.
El dataset se divide en las tres zonas coloreadas de la imagen. La primera son las distintas interacciones de los estudiantes con las asignaturas (matriculación, entrega de prácticas e interacción con el VLE), la segunda se refiere a la información disponible por las distintas asignaturas y, por último, la tercera contiene la información demográfica sobre los estudiantes.
Esta última es la zona que he decidido usar para mi proyecto. Esto se debe a que la tabla 'studentInfo.csv' posee únicamente datos demográficos (y alguno de carácter académico) de los alumnos, lo cual otorga un gran número de atributos para poder trabajar en nuestro modelo de aprendizaje automático. Podemos ver una captura de este archivo .csv en la figura 3.
A pesar del gran tamaño del OULAD, trabajar solo con esta tabla me ha evitado realizar operaciones "join" en R, las cuales hubieran sido bastante costosas (como comentaremos más adelante, 'studentInfo.csv' tiene únicamente en torno a 32.000 filas). Además, la tabla con la que trabajamos tiene asignaturas etiquetadas como A, B, C…. hasta la letra G, lo cual nos permite tener un número suficiente de asignaturas con las que trabajar en el proyecto. No obstante, de cara a no dar un número abrumador de resultados para cada una de las asignaturas en las tablas de resultados, he optado por trabajar solo con las asignaturas A, B y C. En realidad cada asignatura se codifica como AAA en vez de A, pero yo he optado por esta recodificación para facilitar la legibilidad del código y de la memoria del proyecto.
Cada una de estas asignaturas se imparte en Febrero y Octubre, y el dataset nos proporciona datos para las convocatorias de 2013 y 2014.
Nosotros trabajaremos con los datos de ambas convocatorias en los meses de Febrero y Octubre, es decir, trabajaremos sobre los datos históricos que poseemos de los alumnos matriculados (los cuales también poseen una calificación final) en esas asignaturas por separado.
A partir de un script en R ('dataCleaning.r') del cual hablaré más adelante, he creado a partir de 'studentInfo.csv' dos conjuntos de entrenamiento distintos ('trainingX.mat' para la asignatura X referente al problema de clasificación binario y 'trainingMulticlassX.mat' para la asignatura X referente al problema de clasificación multiclase). No obstante, de cara a realizar pruebas en la primera parte del proyecto,

he obtenido otros tantos conjuntos de entrenamiento idénticos a los anteriores con la única diferencia de que ha eliminado el atributo de región (estos archivos tienen la etiqueta “2” en el nombre del archivo) a la que pertenece el alumno. Todos estos detalles se encuentran en el anexo, al final del documento.

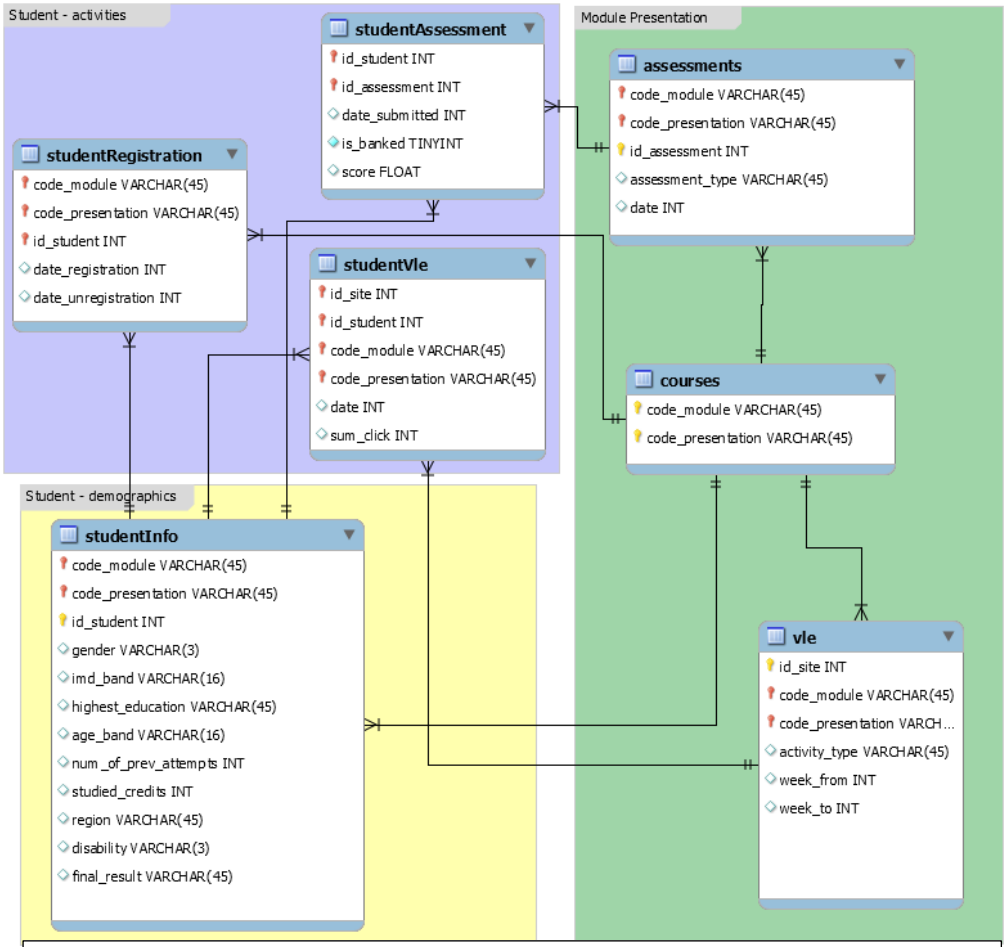


Figura 2: Esquema de las tablas del OULAD

Los atributos de los estudiantes que utilizaremos en nuestro proyecto son los siguientes: sexo, región en la que vivía el alumno mientras estudiaba el curso a distancia, nivel de estudios cuando se matriculó en la asignatura, rango de edad, indicador de discapacidad, número de convocatorias utilizadas previamente en la asignatura, número de créditos en los que el alumno está actualmente matriculado y la calificación final (aprobado, matrícula de honor, suspenso o abandono). La calificación final contendrá estas cuatro categorías en el output dirigido al problema de clasificación multiclase y solo dos categorías (aprobado y suspenso) para el output que utilizaremos en la clasificación binaria. He decidido que para la clasificación binaria las matrículas se consideren aprobados y los abandonos como suspensos. Aunque perdemos algo de información, creo que los datos resultantes siguen teniendo sentido. En cuanto a la codificación que he utilizado para pasar de los strings originales a los

valores numéricos de los atributos en las matrices .mat resultantes, el sexo se indica con 1 para los varones y 0 para las mujeres.
Para el número de convocatorias realizadas previamente y el número de créditos matriculados se realiza una conversión de string a número natural.
En el caso de la columna de calificaciones finales, en el caso del problema de clasificación binaria, los aprobados se indican con 1 y los suspensos con 0.
Para el caso de la clasificación multiclase, se realiza un mapeo de cada uno de las cuatro posibles calificaciones finales a un número entre 1 y 4.
También se incluyen indicadores binarios para la discapacidad (1 si el alumno sufre alguna discapacidad y 0 en caso contrario), la región, el nivel de estudios y rango de edad. Para estos tres últimos atributos se procede de la siguiente manera: para cada uno de los valores que pertenecen a esos atributos (por ejemplo para el valor "Escocia" en el atributo región) se crea una columna con un indicador binario que indica si un determinado alumno posee ese valor o no (si el alumno reside en Escocia tendremos un 1, mientras que 0 indicará lo contrario). En resumen, tendremos tantas columnas o atributos finales como valores distintos haya.
Esto provocará que haya en nuestro proyecto bastantes variables dummy (si una determinada región está a 1, el resto de regiones para esa fila en concreto serán 0), pero a cambio tendremos un parámetro theta para cada valor de estos atributos. Por ello, cada uno de estos valores se convertirá en un atributo (es decir, una columna) en la matriz de salida.
Todo este proceso se realiza en el archivo 'dataCleaning.r', el cual lee 'studentInfo.csv' a partir de la función 'read.csv' y genera estas columnas con valores binarios gracias a la función 'acm.disjonctif'. También se obtienen los distintos "levels" de estos atributos, los cuales son clasificados por R como columnas con datos categóricos.
Esto nos permite saber que hay 13 regiones distintas, 3 rangos de edad, 5 niveles académicos y 4 posibles tipos de calificaciones finales (las clasificaciones finales se han codificado como .
Como consecuencia, la matriz resultante tendrá 26 columnas (25 columnas de atributos y la columna de calificaciones finales, que actuará como el vector 'y' que teníamos en las prácticas). Para el otro output que resulta (aquel que incluye un "2" en el nombre del archivo .mat) de la ejecución del script, al no contar los distintos valores del campo región como atributos, tendremos 13 columnas en total (incluyendo la columna de calificaciones).
También hay que mencionar que la escritura de archivos .mat se realiza con la función 'writeMat', disponible en el paquete "R.matlab". Para realizar una escritura correcta realizamos un cast a matrices ('as.matrix') de los dataframes creados.
Finalmente, adjunto estadísticas relativas a 'studentInfo.csv':

Número de filas de ‘studentInfo.csv’: 32593
Número de alumnos en la asignatura AAA: 748
Número de alumnos en la asignatura BBB: 7909
Número de alumnos en la asignatura CCC: 4434

Además, ‘studentInfo.csv’ tiene información de las siguientes asignaturas: AAA, BBB, CCC, DDD, EEE, FFF, GGG.

He utilizado solo estas tres asignaturas para tratar de obtener distintos resultados y realizar varias comprobaciones. No obstante, el script en R no está preparado para sacar de manera automática archivos .mat para cada una de las distintas asignaturas, lo cual sería interesante de cara a una futura ampliación de este proyecto.

Por último, en las figuras 4, 5 y 6 se pueden ver las capturas de pantalla de los dataframes que dan lugar a ‘trainingA.mat’, ‘training2A.mat’ y ‘trainingMulticlass2A.mat’ respectivamente.

	code_module	code_presentation	id_student	gender	region	highest_education	imd_band	age_band	num_of_prev_attempts	studied_credits	disability	final_result
1	AAA	2013J	11391	M	East Anglian Region	HE Qualification	90-100%	55<=	0	240	N	Pass
2	AAA	2013J	28400	F	Scotland	HE Qualification	20-30%	35-55	0	60	N	Pass
3	AAA	2013J	30268	F	North Western Region	A Level or Equivalent	30-40%	35-55	0	60	Y	Withdrawn
4	AAA	2013J	31604	F	South East Region	A Level or Equivalent	50-60%	35-55	0	60	N	Pass
5	AAA	2013J	32885	F	West Midlands Region	Lower Than A Level	50-60%	0-35	0	60	N	Pass
6	AAA	2013J	38053	M	Wales	A Level or Equivalent	80-90%	35-55	0	60	N	Pass
7	AAA	2013J	45462	M	Scotland	HE Qualification	30-40%	0-35	0	60	N	Pass
8	AAA	2013J	45642	F	North Western Region	A Level or Equivalent	90-100%	0-35	0	120	N	Pass
9	AAA	2013J	52130	F	East Anglian Region	A Level or Equivalent	70-80%	0-35	0	90	N	Pass
10	AAA	2013J	53025	M	North Region	Post Graduate Qualification		55<=	0	60	N	Pass
11	AAA	2013J	57506	M	South Region	Lower Than A Level	70-80%	35-55	0	60	N	Pass
12	AAA	2013J	58873	F	East Anglian Region	A Level or Equivalent	20-30%	0-35	0	60	N	Pass
13	AAA	2013J	59185	M	East Anglian Region	Lower Than A Level	60-70%	35-55	0	60	N	Pass
14	AAA	2013J	62155	F	North Western Region	HE Qualification	50-60%	0-35	0	60	N	Pass
15	AAA	2013J	63400	M	Scotland	Lower Than A Level	40-50%	35-55	0	60	N	Pass
16	AAA	2013J	65002	F	East Anglian Region	A Level or Equivalent	70-80%	0-35	0	60	N	Withdrawn
17	AAA	2013J	70464	F	West Midlands Region	A Level or Equivalent	60-70%	35-55	0	60	N	Pass
18	AAA	2013J	71361	M	Ireland	HE Qualification		35-55	0	60	N	Pass
19	AAA	2013J	74372	M	East Anglian Region	A Level or Equivalent	10-20	35-55	0	150	N	Fail
20	AAA	2013J	75091	M	South West Region	A Level or Equivalent	30-40%	35-55	0	60	N	Pass
21	AAA	2013J	77367	M	East Midlands Region	A Level or Equivalent	30-40%	0-35	0	60	N	Pass
22	AAA	2013J	91265	M	North Western Region	HE Qualification	0-10%	0-35	0	60	N	Pass
23	AAA	2013J	94961	M	South Region	Lower Than A Level	70-80%	35-55	0	60	N	Withdrawn

Showing 1 to 24 of 32,593 entries

Figura 3: Captura de pantalla del archivo ‘studentInfo.csv’

	gender.M	region.East Anglian Region	region.East Midlands Region	region.Ireland	region.London Region	region.North Region	region.North Western Region	region.Scotland	region.South East Region	region.South Region	region.South West Region	region.Wales	region.West Midlands Region	region.Yorkshire Region	highest_education.A Level or Equivalent
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
6	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1
7	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
8	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
9	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
10	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
11	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0
12	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
13	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
15	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
16	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
17	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
18	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
19	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1
20	1	0	0	0	0	0	0	0	0	0	1	0	0	0	1
21	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1

highest_education.No Formal quals	highest_education.Post Graduate Qualification	age_band.0-35	age_band.35-55	age_band.55<=	disability.Y	attempts	studied_credits	final_result
0	0	0	0	1	0	0	240	1
0	0	0	1	0	0	0	60	1
0	0	0	1	0	1	0	60	0
0	0	0	1	0	0	0	60	1
0	0	1	0	0	0	0	60	1
0	0	0	1	0	0	0	60	1
0	0	1	0	0	0	0	60	1
0	0	1	0	0	0	0	120	1
0	0	1	0	0	0	0	90	1
0	1	0	0	1	0	0	60	1
0	0	0	1	0	0	0	60	1
0	0	1	0	0	0	0	60	1
0	0	0	1	0	0	0	60	1
0	0	1	0	0	0	0	60	1
0	0	0	1	0	0	0	60	1
0	0	1	0	0	0	0	60	0
0	0	0	1	0	0	0	60	1
0	0	0	1	0	0	0	60	1
0	0	0	1	0	0	0	150	0
0	0	0	1	0	0	0	60	1
0	0	1	0	0	0	0	60	1

Figura 4: Captura de pantalla del dataframe que genera ‘trainingA.mat’

	gender.M	highest_education.A Level or Equivalent	highest_education.HE Qualification	highest_education.Lower Than A Level	highest_education.No Formal quals	highest_education.Post Graduate Qualification	age_band.0-35	age_band.35-55	age_band.55<=	disability.Y	attempts	studied_credits	final_result
1	1	0	1	0	0	0	0	0	1	0	0	240	1
2	0	0	1	0	0	0	0	1	0	0	0	60	1
3	0	1	0	0	0	0	0	1	0	1	0	60	0
4	0	1	0	0	0	0	0	1	0	0	0	60	1
5	0	0	0	1	0	0	1	0	0	0	0	60	1
6	1	1	0	0	0	0	0	1	0	0	0	60	1
7	1	0	1	0	0	0	1	0	0	0	0	60	1
8	0	1	0	0	0	0	1	0	0	0	0	120	1
9	0	1	0	0	0	0	1	0	0	0	0	90	1
10	1	0	0	0	0	1	0	0	1	0	0	60	1
11	1	0	0	1	0	0	0	1	0	0	0	60	1
12	0	1	0	0	0	0	1	0	0	0	0	60	1
13	1	0	0	1	0	0	0	1	0	0	0	60	1
14	0	0	1	0	0	0	1	0	0	0	0	60	1
15	1	0	0	1	0	0	0	1	0	0	0	60	1
16	0	1	0	0	0	0	1	0	0	0	0	60	0
17	0	1	0	0	0	0	0	1	0	0	0	60	1
18	1	0	1	0	0	0	0	1	0	0	0	60	1
19	1	1	0	0	0	0	0	1	0	0	0	150	0
20	1	1	0	0	0	0	0	1	0	0	0	60	1
21	1	1	0	0	0	0	1	0	0	0	0	60	1
22	1	0	1	0	0	0	1	0	0	0	0	60	1
23	1	0	0	1	0	0	0	1	0	0	0	60	0

Showing 1 to 23 of 748 entries

Figura 5: Captura de pantalla del dataframe que genera ‘training2A.mat’

	gender.M	highest_education.A Level or Equivalent	highest_education.HE Qualification	highest_education.Lower Than A Level	highest_education.No Formal quals	highest_education.Post Graduate Qualification	age_band.0-35	age_band.35-55	age_band.55<=	disability.Y	attempts	studied_credits	final_result
1	1	0	1	0	0	0	0	0	1	0	0	240	1
2	0	0	1	0	0	0	0	1	0	0	0	60	1
3	0	1	0	0	0	0	0	1	0	1	0	60	2
4	0	1	0	0	0	0	0	1	0	0	0	60	1
5	0	0	0	1	0	0	1	0	0	0	0	60	1
6	1	1	0	0	0	0	0	1	0	0	0	60	1
7	1	0	1	0	0	0	1	0	0	0	0	60	1
8	0	1	0	0	0	0	1	0	0	0	0	120	1
9	0	1	0	0	0	0	1	0	0	0	0	90	1
10	1	0	0	0	0	1	0	0	1	0	0	60	1
11	1	0	0	1	0	0	0	1	0	0	0	60	1
12	0	1	0	0	0	0	1	0	0	0	0	60	1
13	1	0	0	1	0	0	0	1	0	0	0	60	1
14	0	0	1	0	0	0	1	0	0	0	0	60	1
15	1	0	0	1	0	0	0	1	0	0	0	60	1
16	0	1	0	0	0	0	1	0	0	0	0	60	2
17	0	1	0	0	0	0	0	1	0	0	0	60	1
18	1	0	1	0	0	0	0	1	0	0	0	60	1
19	1	1	0	0	0	0	0	1	0	0	0	150	3
20	1	1	0	0	0	0	0	1	0	0	0	60	1
21	1	1	0	0	0	0	1	0	0	0	0	60	1
22	1	0	1	0	0	0	1	0	0	0	0	60	1
23	1	0	0	1	0	0	0	1	0	0	0	60	2

Showing 1 to 23 of 748 entries

Figura 6: Captura de pantalla del dataframe que genera ‘trainingMulticlass2A.mat’

3. Métodos de aprendizaje automáticos utilizados

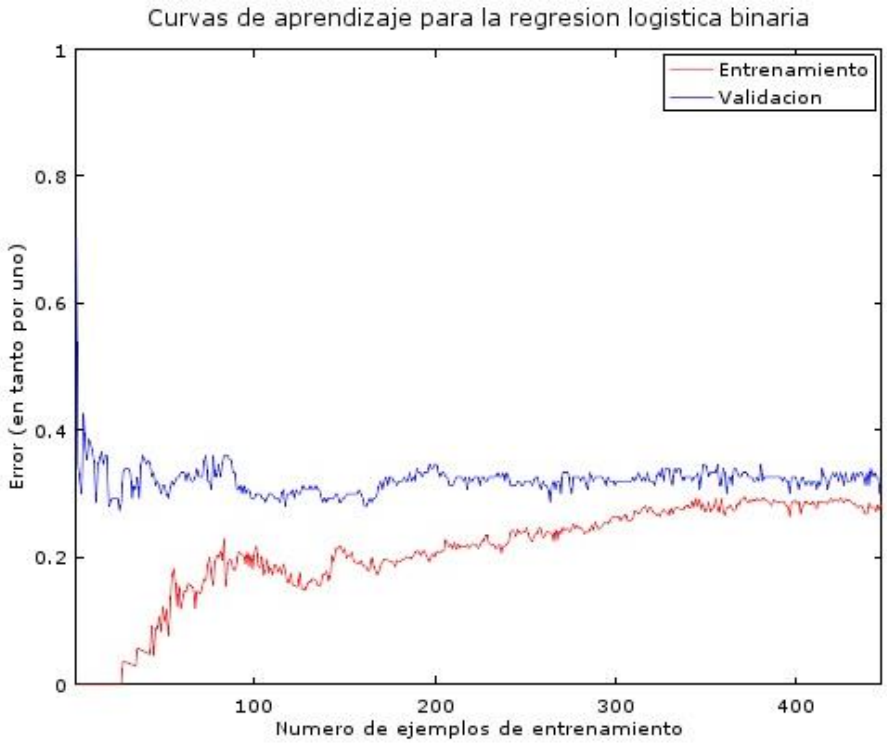
Parte 1: Análisis de errores (Sesgo y varianza. Precision y recall)

La primera parte del proyecto nos permitirá valorar el conjunto de atributos que hemos escogido y que luego utilizaremos en los distintos modelos de aprendizaje automático. Hemos escogido estos atributos por facilidad (solo hemos utilizado 'studentInfo.csv') por no abrumar con un número mucho mayor de ellos (había algún atributo más en ese archivo csv y en 'studentVLE.csv'), pues mi enfoque no ha sido usar todos los atributos que fueran posibles. Tampoco he querido ir probando con un gran número de subconjuntos de atributos, si no que he elegido un subconjunto (los atributos en 'trainingX.mat' menos el campo región) para ir realizando una serie de pruebas sobre él y así valorar de forma precisa los resultados de ese subconjunto respecto al conjunto original.
No se trata de obtener los mejores resultados (para ello hubiera incluido los atributos sobre la interacción con el VLE, lo cual hubiera mejorado de forma sustancial los resultados tal y como muestran los miembros del proyecto OU Analyse en sus publicaciones, las cuales se pueden encontrar en la sección de referencias), si no de deducir como de buenos son los atributos que tenemos y si alguno de ellos es redundante.

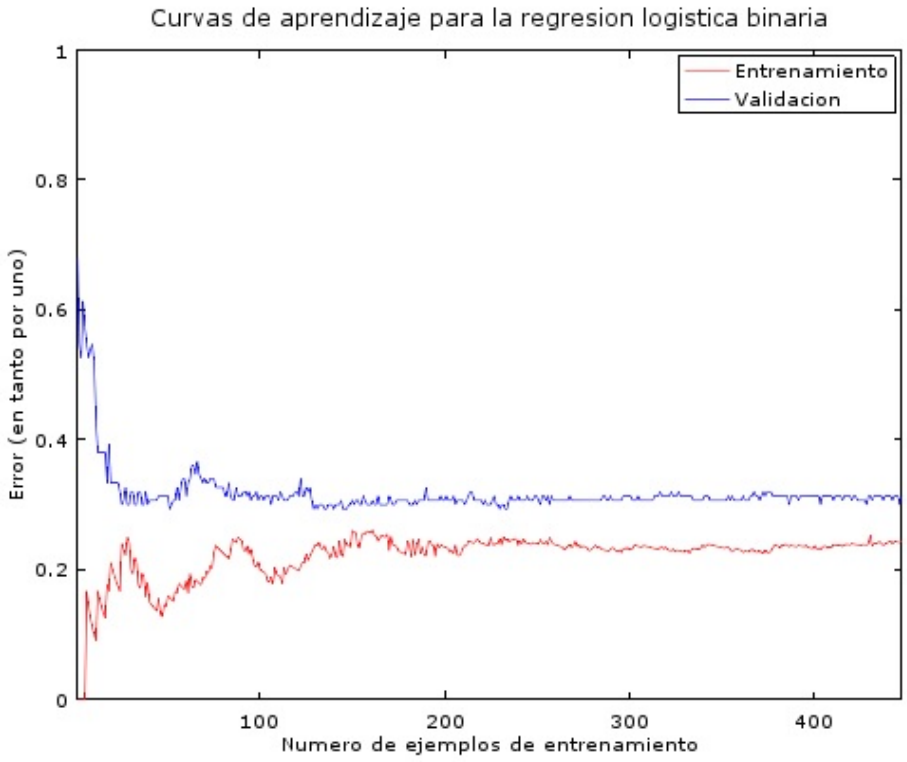
Realizaremos tres pruebas utilizando regresión logística binaria y dos pruebas utilizando regresión binaria multiclase, usando cada una de las matrices creadas por el script en R para finalmente comentar los resultados obtenidos. Para cada input mezclaremos primero los datos (puede haber pequeñas variaciones en los resultados si realizamos dos ejecuciones independientes de las pruebas) y luego escogeremos un 60% de los datos que servirán como datos de entrenamiento, un 20% destinados a validación (cross-validation) y el 20% restante serán datos de prueba (test data), los cuales devolverán un error final (habiendo usado un valor de lambda óptimo).
Sin embargo, para las demás pruebas realizadas en el proyecto, se utilizará un 70% de los datos para entrenamiento y un 30% para validación.
Las 3 pruebas se realizan de manera secuencial por medio de una función en el script 'analisisErrores.m' para la clasificación binaria y otra para la clasificación multiclase. Estos detalles se encuentran en el anexo del documento:

En primer lugar, la primera prueba nos permite saber si los algoritmos están sufriendo sesgo o varianza. Para ello, al igual que realizamos en la práctica 5 de la asignatura, vamos tomando subconjuntos de entrenamiento cada vez mayores para obtener las gráficas de las curvas de aprendizaje. El parámetro lambda utilizado en esta prueba ha sido 0.01.
Por otro lado, estas pruebas se han realizado iterando 200 veces en cada ejecución del modelo para el problema de clasificación binaria y multiclase.

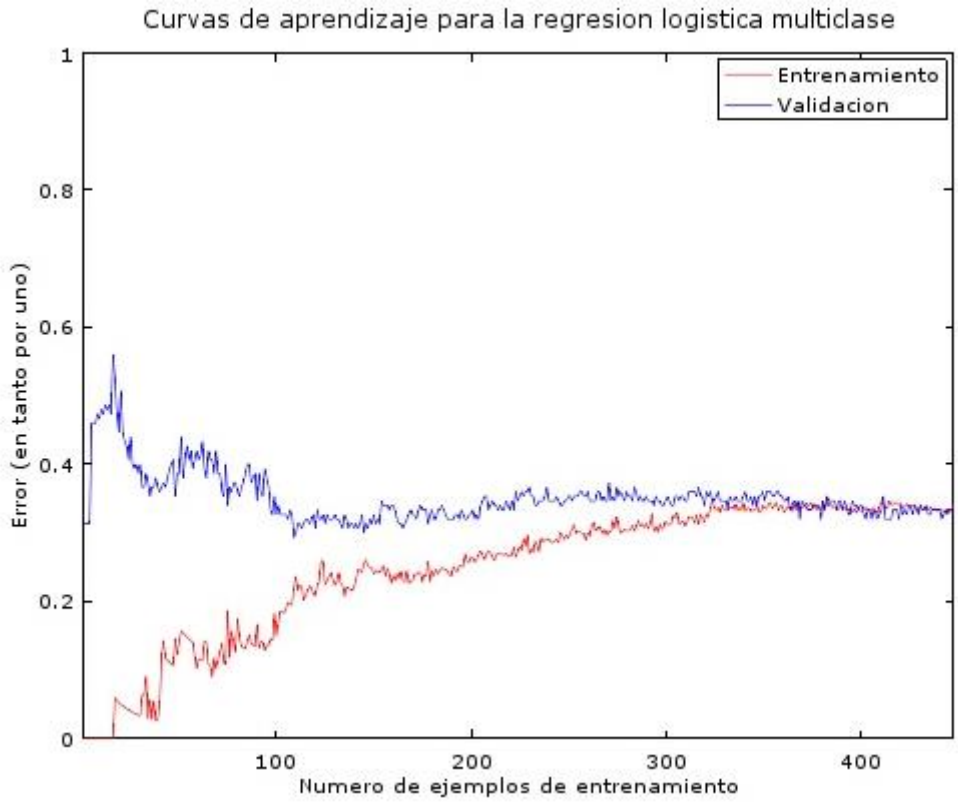
'trainingA.mat'



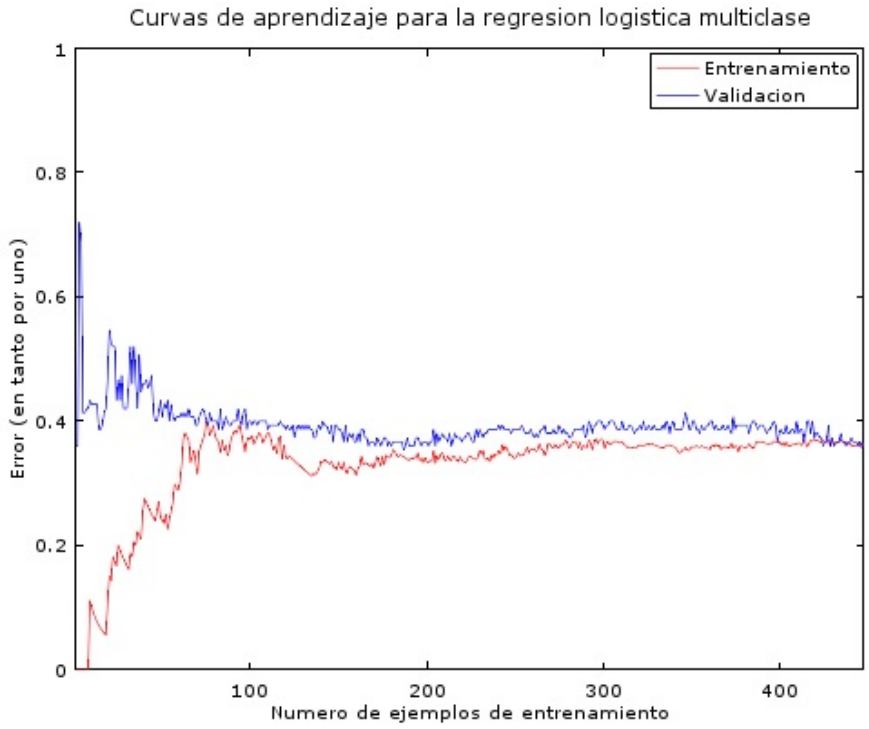
'training2A.mat'



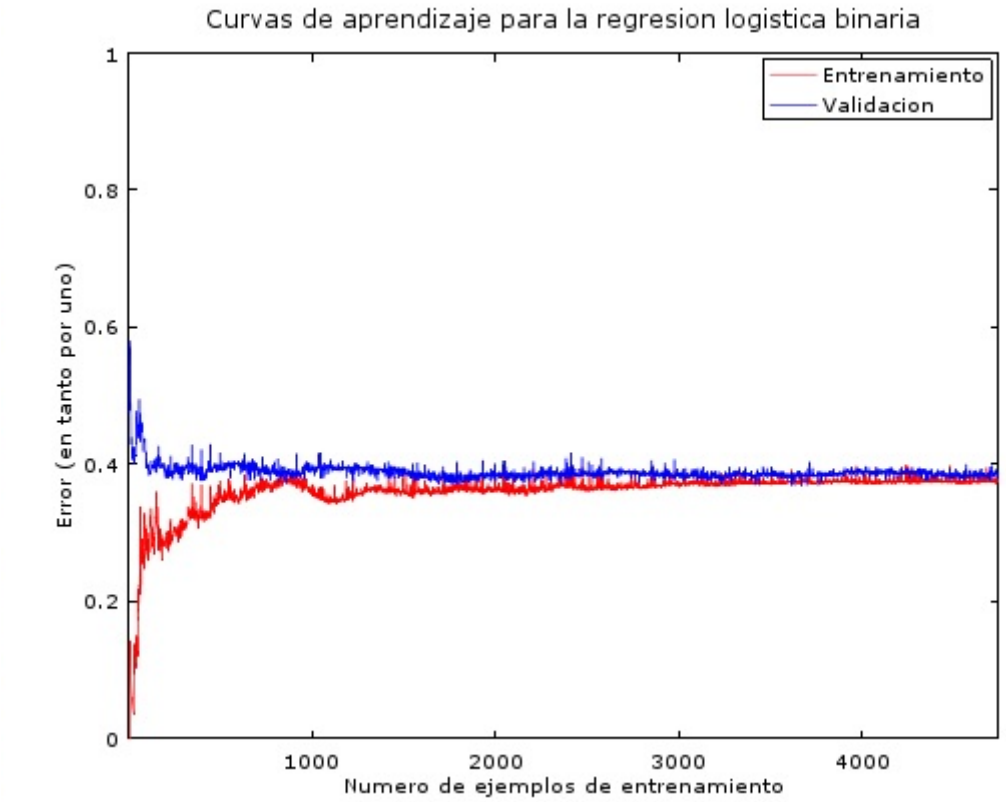
'trainingMulticlassA.mat'



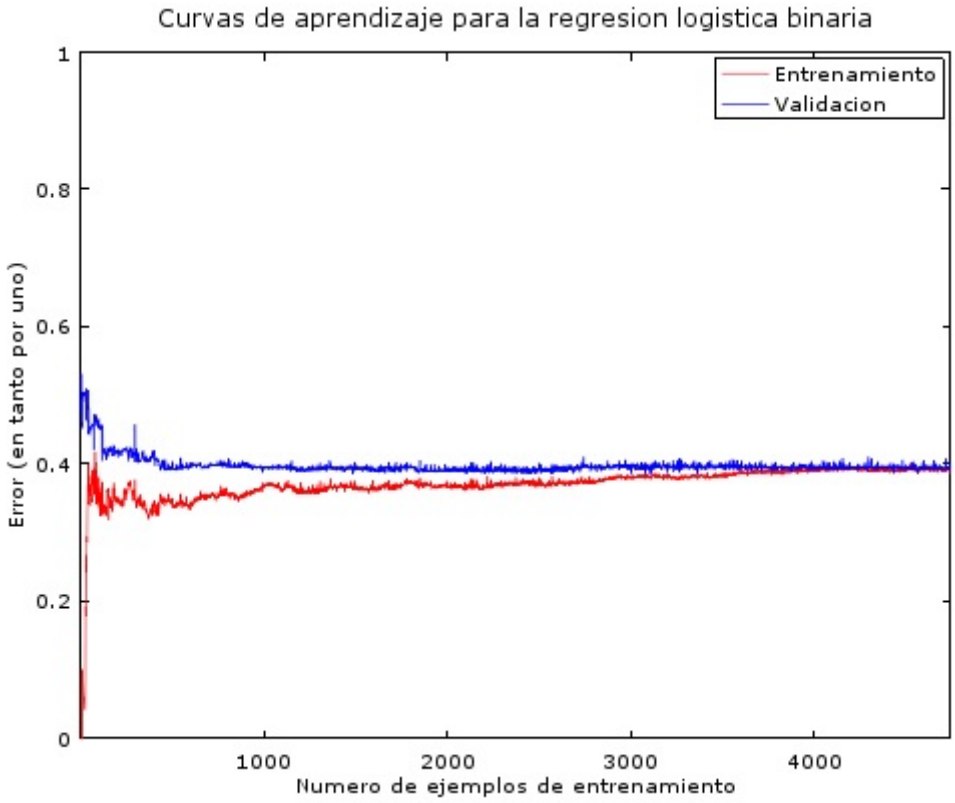
'trainingMulticlass2A.mat'



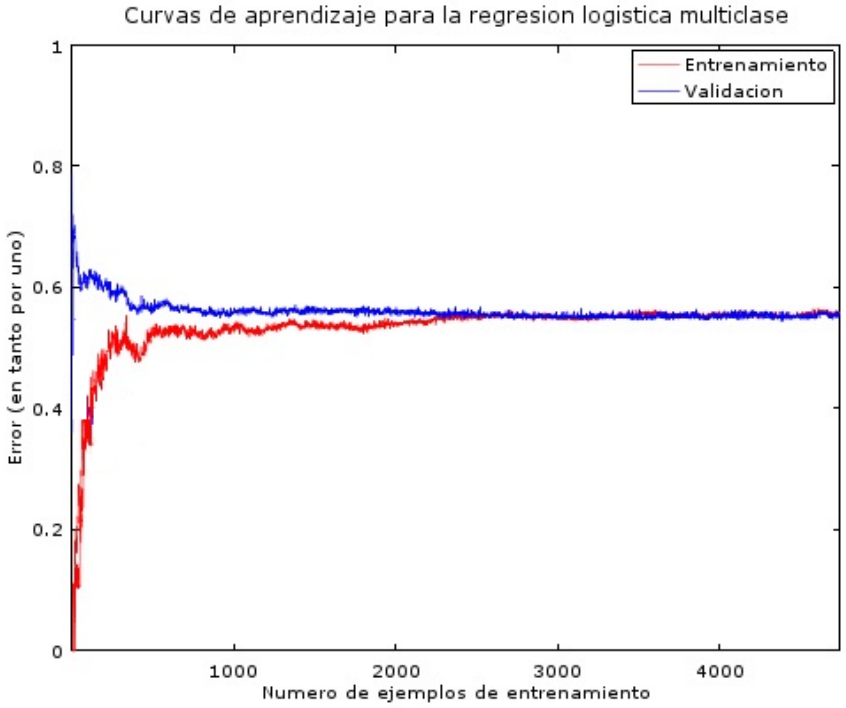
'trainingB.mat'



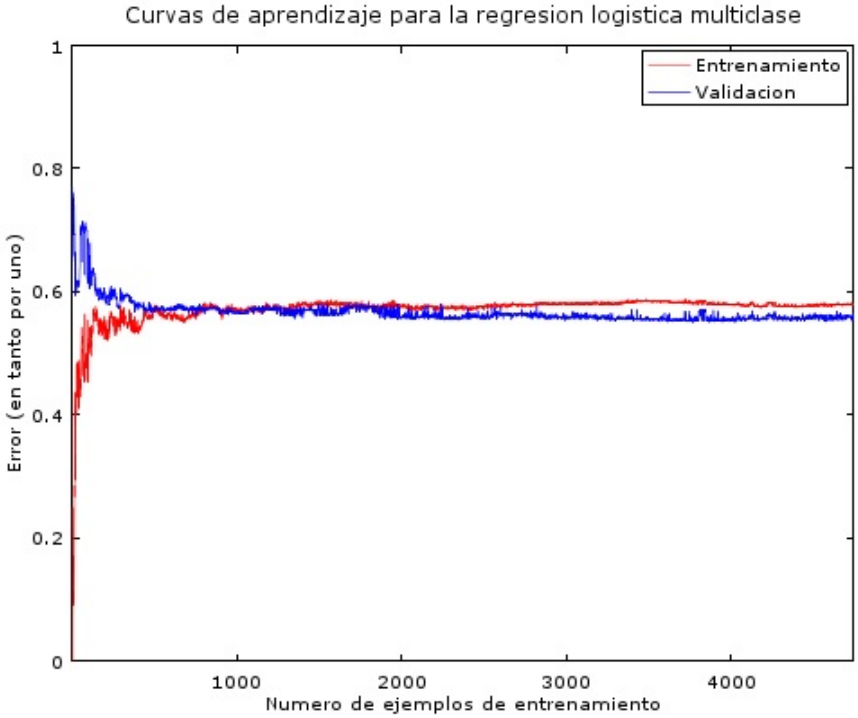
'training2B.mat'



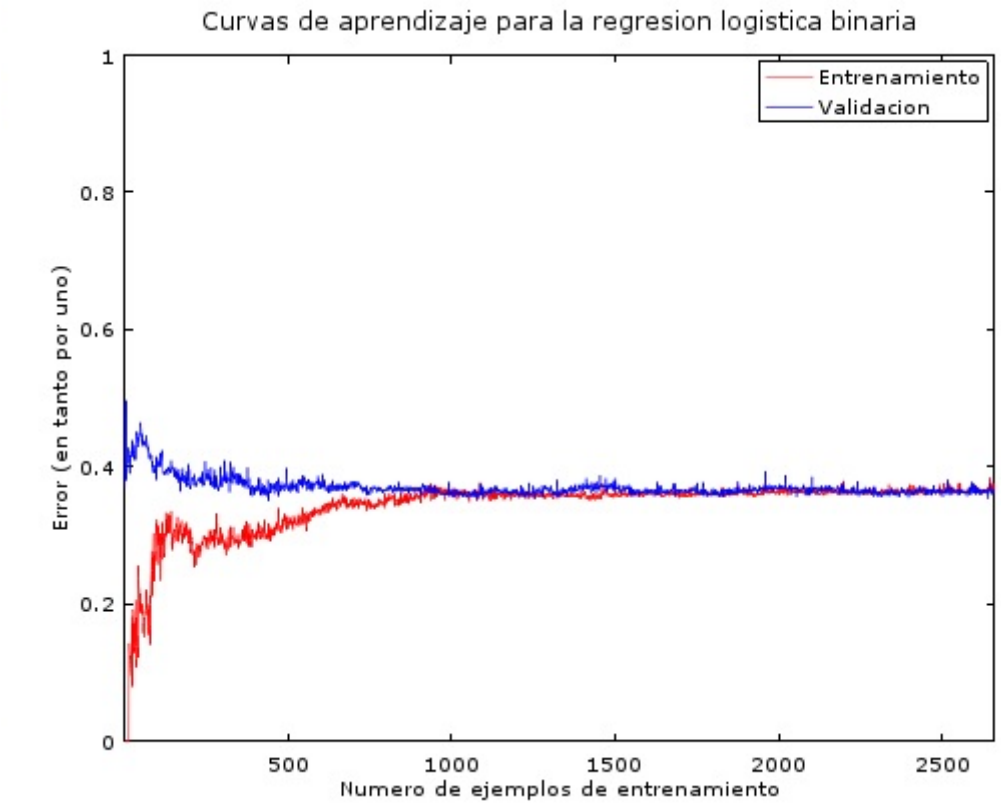
'trainingMulticlassB.mat'



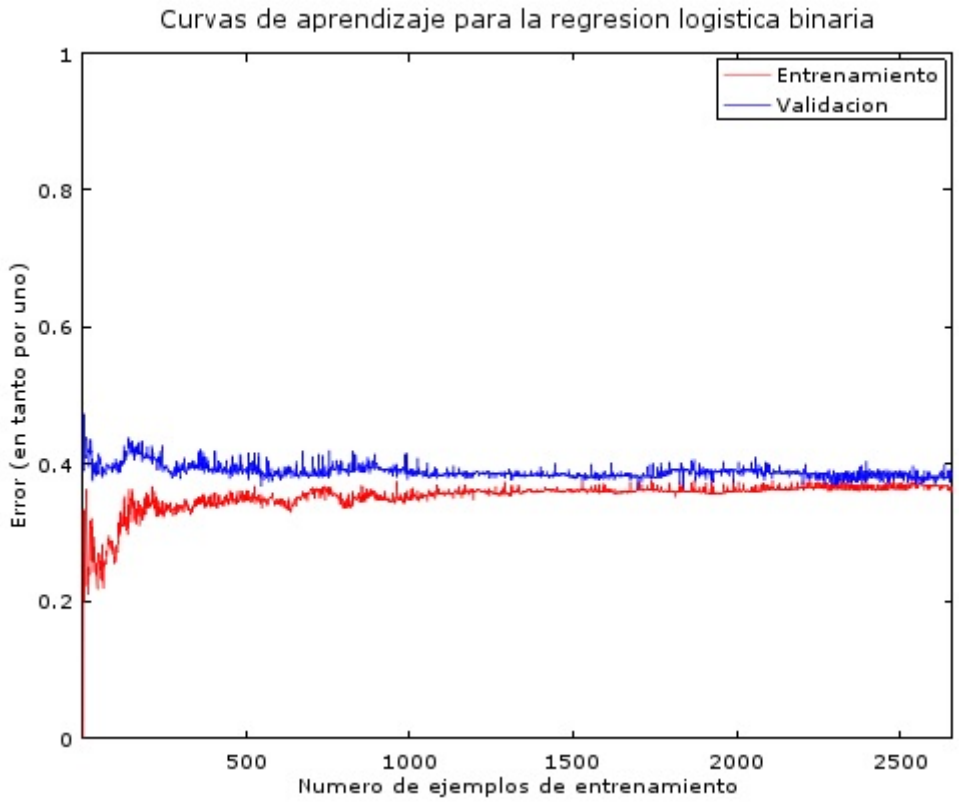
'trainingMulticlass2B.mat'



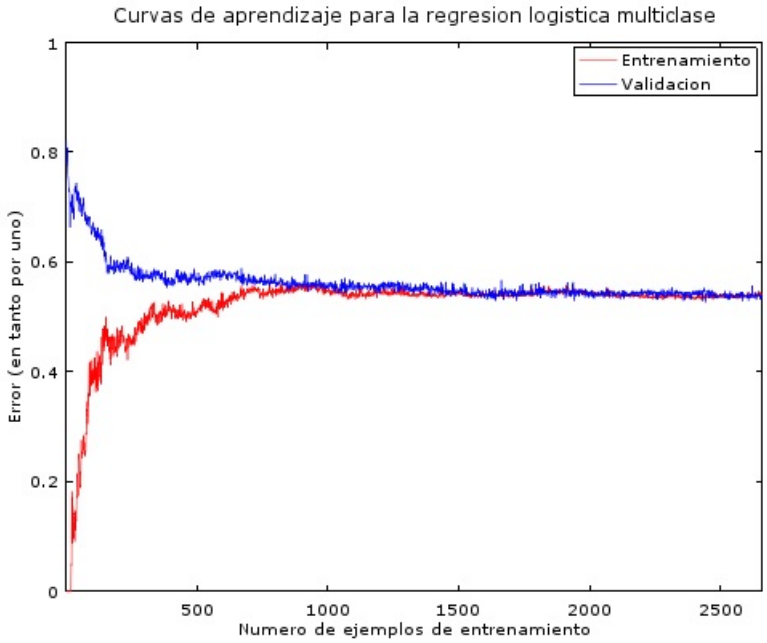
'trainingC.mat'



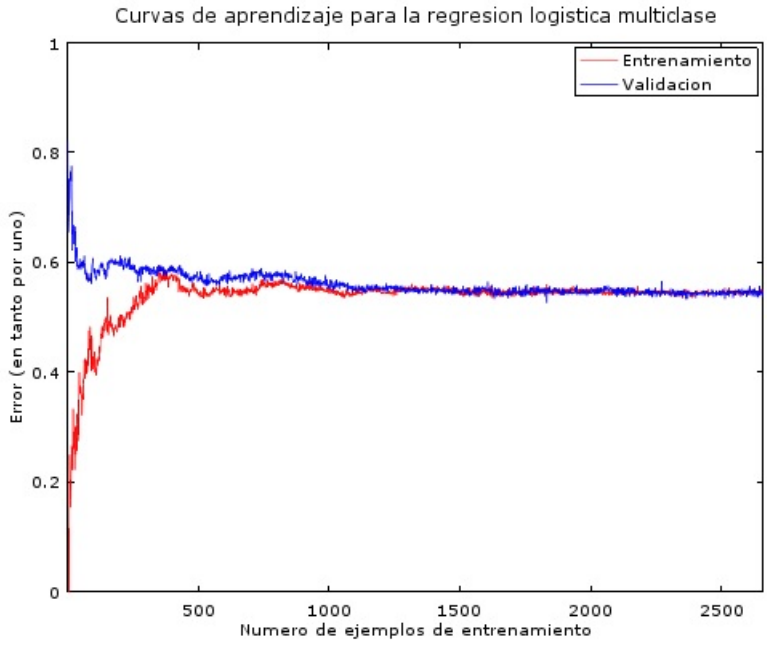
'training2C.mat'



'trainingMulticlassC.mat'



'trainingMulticlass2C.mat'



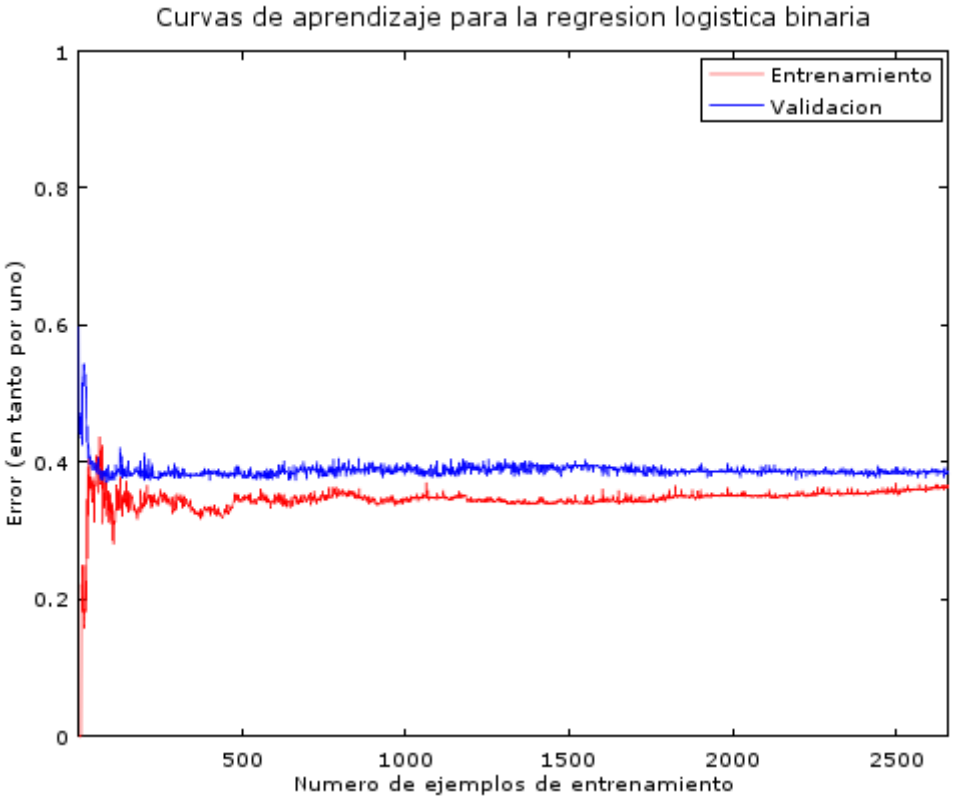
Estas graficas muestran que, tal y como esperábamos, tenemos alta varianza con los inputs 'trainingA.mat', 'trainingB.mat' y 'trainingC.mat', pues los 3 incluyen como atributos (13 en total) las distintas regiones en las que residen los alumnos. Mientras tanto, 'training2A.mat', 'training2B.mat' y 'training2C.mat' poseen algo de sesgo, pues podemos ver un pequeño gap constante entre los resultados de los datos de entrenamiento y los de validación.

Por último, podemos ver como el error cae unos diez puntos porcentuales en la

asignatura A a medida que vamos tomando más ejemplos de entrenamiento, pero para las asignaturas B y C el error permanece prácticamente constante. Por último, podemos notar que no hay diferencias de error entre las dos versiones de los inputs para cada asignatura, hecho que luego volveremos a ver en las secciones posteriores. A partir de esta observación podríamos trabajar únicamente con la versión 2 de cada input para el problema de clasificación binario, la cual tiene 13 columnas menos que la otra versión, aumentando la velocidad de ejecución notablemente sin grandes cambios en el porcentaje de aciertos.

Para las gráficas correspondientes al problema de clasificación multiclase: en las asignaturas A y B notamos como para 'trainingMulticlassA.mat' y 'trainingMulticlassB.mat' existe varianza (especialmente en la asignatura A, pues no la hemos podido corregir al tener menos datos de entrenamiento).
Por último, para la asignatura C, existe varianza para los dos inputs que tenemos, por lo que podemos afirmar que el campo región no ayuda a eliminar la varianza.

Por último, vamos a descomentar unas líneas en la función 'pruebasRegLogBin' dentro del script 'analisisErrores.m' para el archivo 'training2C.mat' y veamos si al aumentar el número de atributos (añadiendo valores polinómicos al atributo número de convocatorias utilizadas, el cual a priori es un dato académico que puede tener más peso que los datos demográficos de cara a la predicción de la calificación final) esto permite aumentar el porcentaje de aciertos:

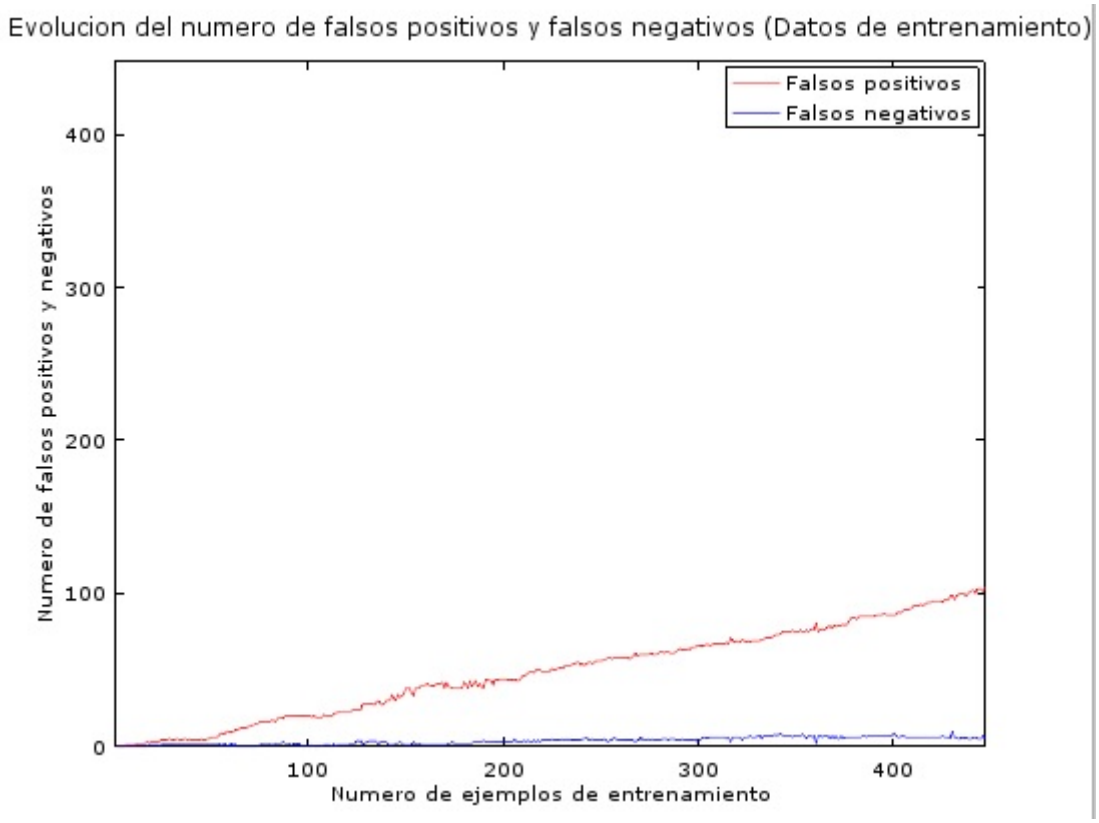


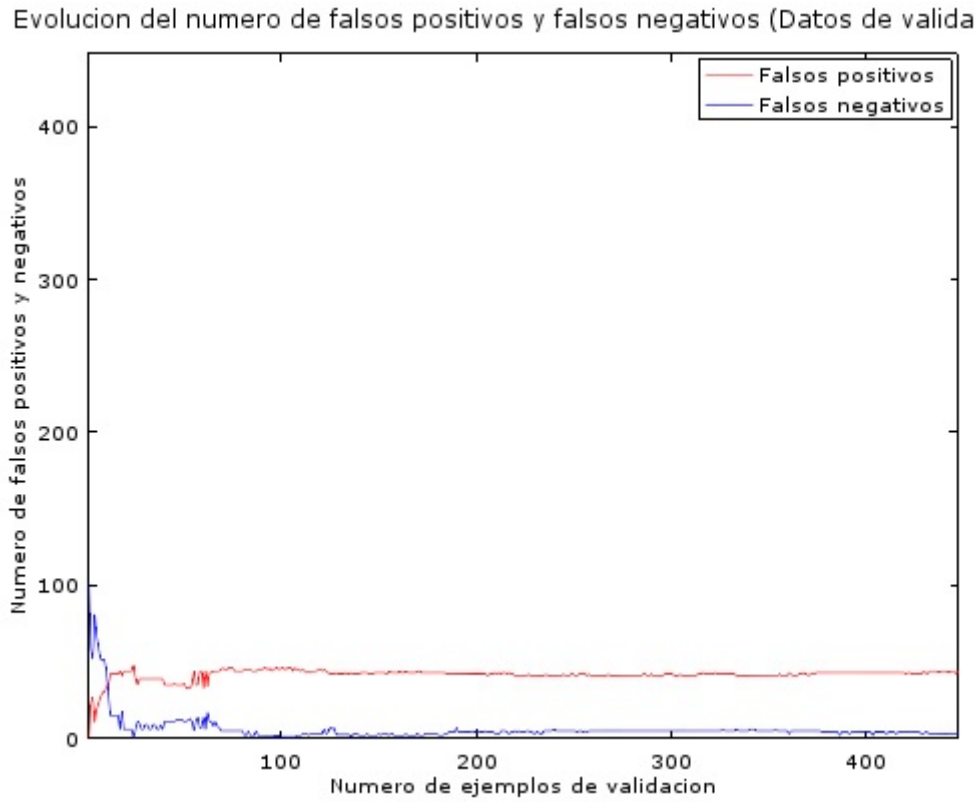
A pesar de que hemos introducido 8 atributos más (hemos creado atributos con potencias 1,2,3…8 para el atributo “número de convocatorias utilizadas”) no logramos mejoría en el resultado final (seguimos en torno al 40% de fallos) y además incluso vemos que el gap en la gráfica para ‘trainingC.mat’ es incluso algo mayor al que vimos anteriormente en la curva de aprendizaje para este input.
En la última parte del proyecto veremos que esto concuerda con los resultados del clustering, pues no existe para nada una correlación entre el número de convocatorias utilizadas y una mayor probabilidad de aprobar la asignatura C.

La segunda prueba (aunque computada al mismo tiempo que la primera) consiste en obtener una gráfica del número de falsos positivos y falsos negativos a medida que aumentamos el número de ejemplos de entrenamiento. Esta prueba solo se ha realizado para el problema de clasificación binaria.

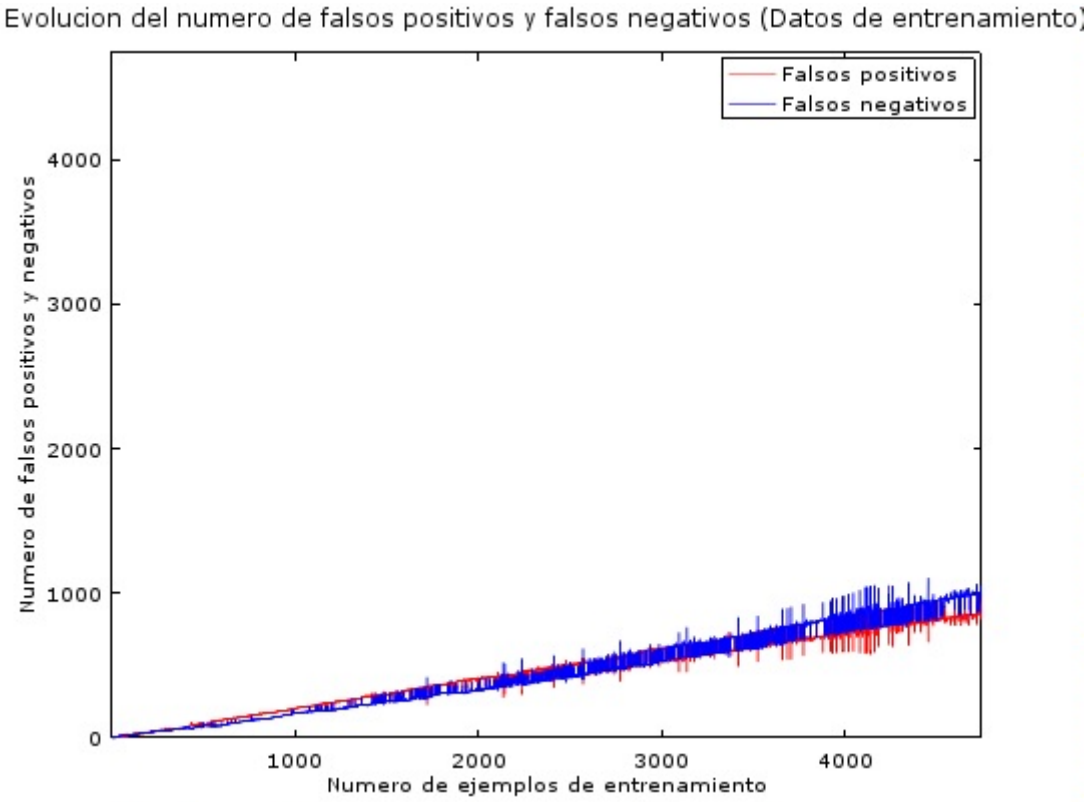
Debido a los resultados que hemos obtenido en la prueba anterior, es suficiente computar solo los resultados para ‘training2A.mat’, ‘training2B.mat’ y ‘training2C.mat’:

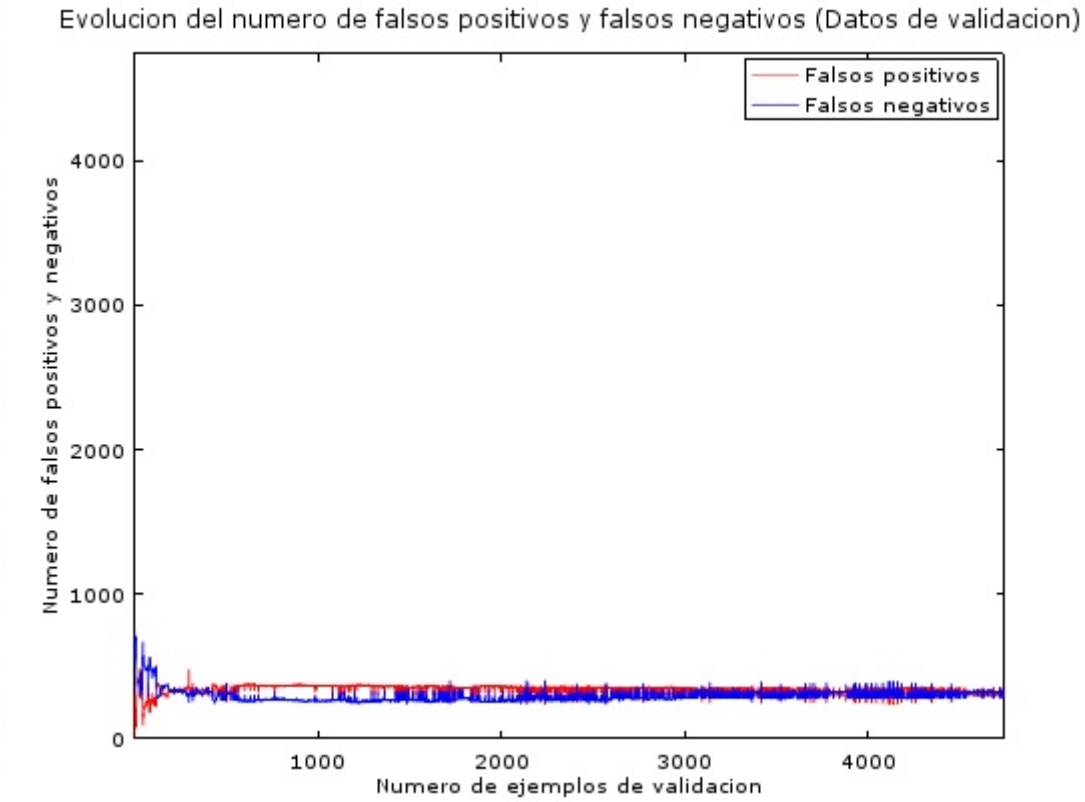
‘training2A.mat’



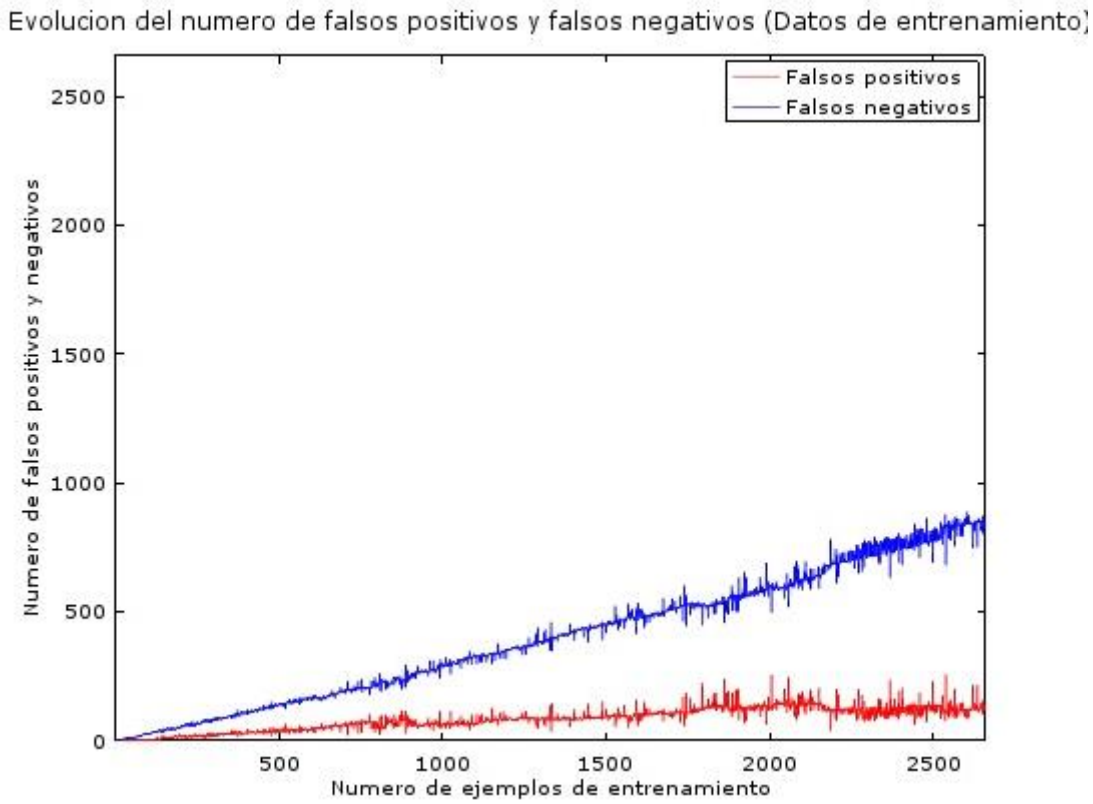


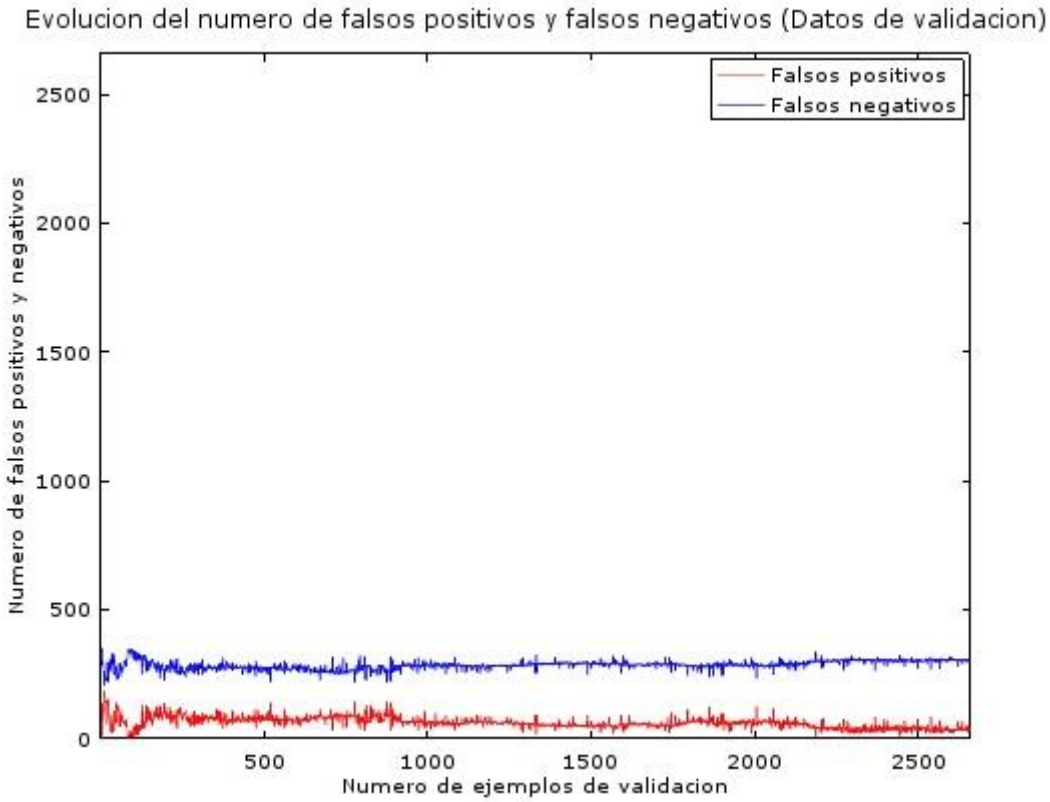
'training2B.mat'





'training2C.mat'

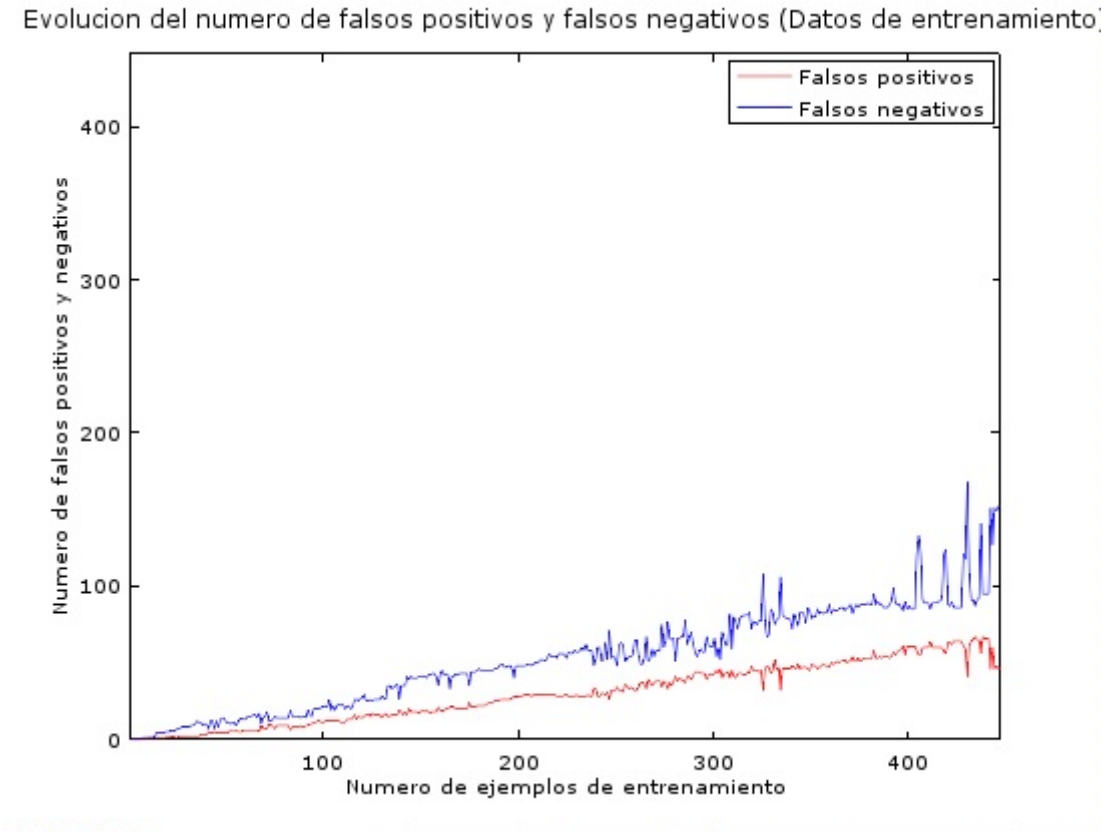


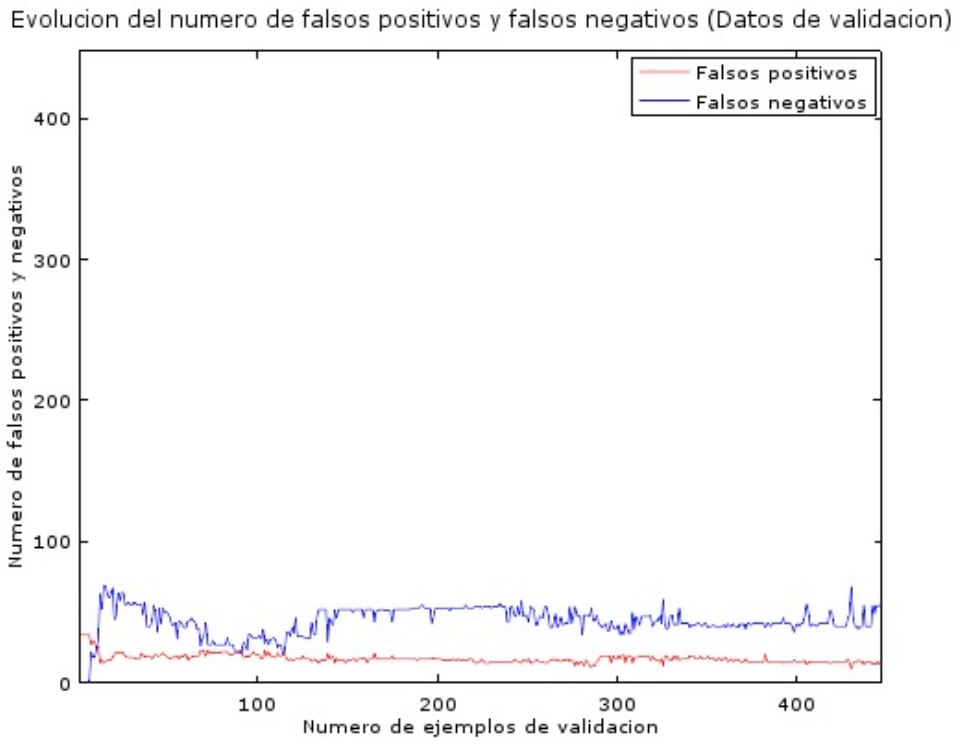


Como se ve, el número de falsos positivos (alumnos que considerábamos que iban a aprobar pero al final suspendían) es bastante superior al número de falsos negativos. Vemos en las 3 asignaturas como a pesar de que el número de falsos positivos va creciendo (en cambio para los falsos negativos el número es bastante constante) para los modelos entrenados con los datos de entrenamiento (posiblemente porque al haber tanta varianza y tantos datos no pueda darse un overfitting), el número de falsos positivos en los modelos con los datos de validación permanece constante. Esto parece indicar de nuevo que los datos no permiten aportar más información para poder corregir el error cometido al considerar alumnos que supuestamente aprobarían pero que finalmente acaban suspendiendo.

Para corregir esto, y dado que preferimos que haya un número mayor de falsos negativos (alumnos que preveíamos que suspenderían pero al final acaban aprobando) vamos a aumentar la precisión (y por tanto disminuir el recall) elevando el 'threshold' de la función 'h' de 0.5 a 0.7.

Realizamos este aumento para 'trainingA.mat' y observamos como ahora hay más falsos negativos que positivos:



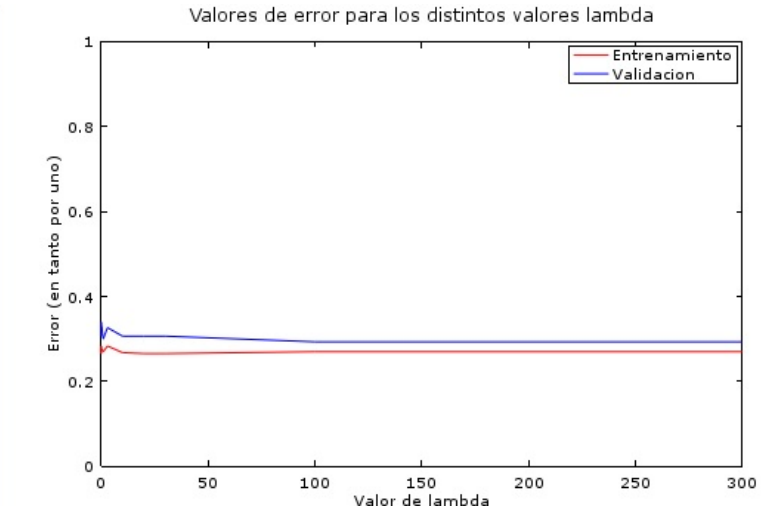


No obstante, nótese las altas variaciones (a partir de los 400 ejemplos) del número de falsos negativos en los modelos entrenados con los ejemplos de entrenamiento. Además, vemos como en estos modelos el número de falsos positivos y negativos

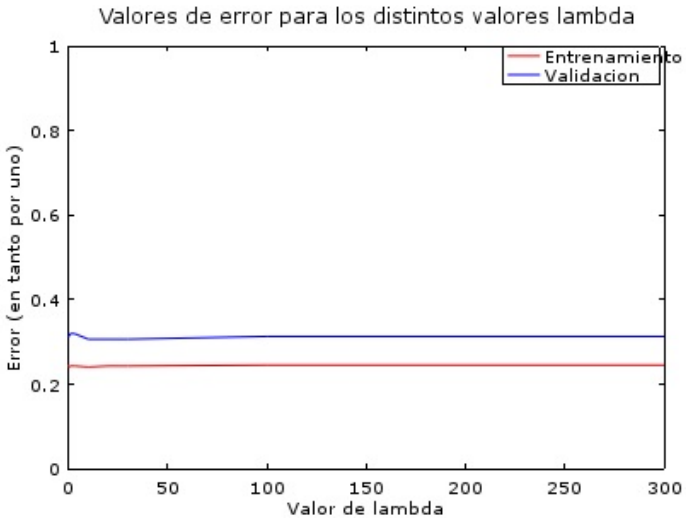
crece de manera conjunta, pero al igual que sucedía antes, el número de falsos positivos y negativos se mantiene más o menos constante para los modelos en los que usamos los datos de validación.
Por último, reseñar que los resultados finales fueron peores al elevar el threshold (los aciertos bajaron a menos del 60%).

La tercera de las pruebas (ahora sí, ya disponible para todos los inputs de los que disponemos) muestra la variación de los errores (sobre los datos de entrenamiento y validación) al escoger distintos valores de lambda.

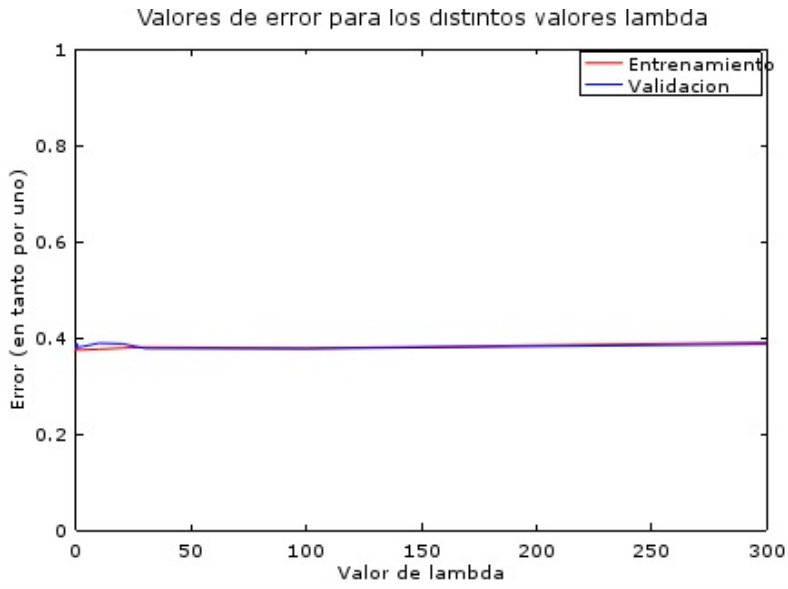
'trainingA.mat'



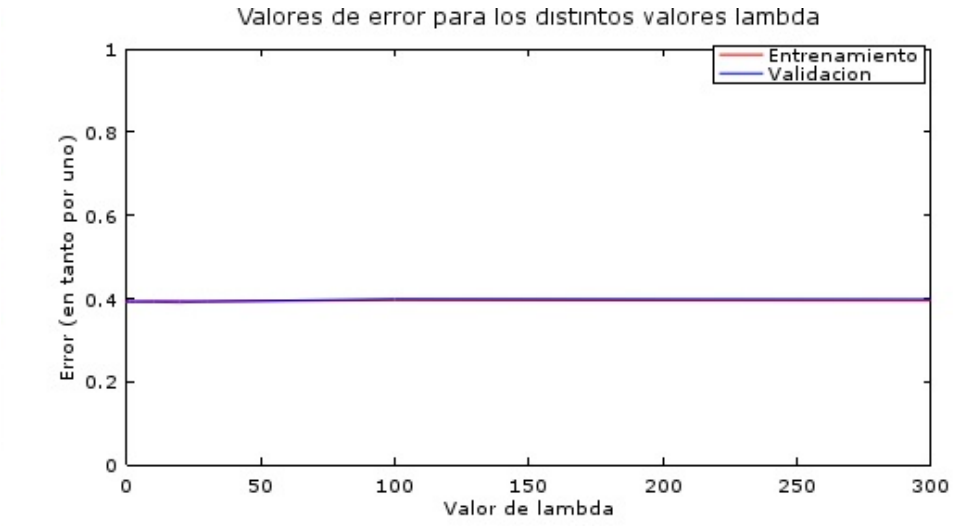
'training2A.mat'



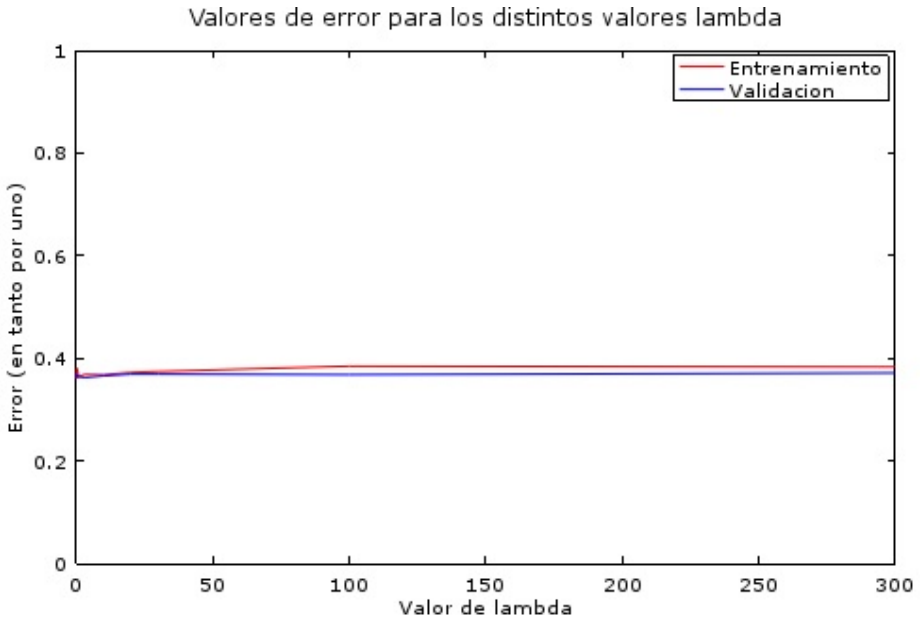
‘trainingB.mat’



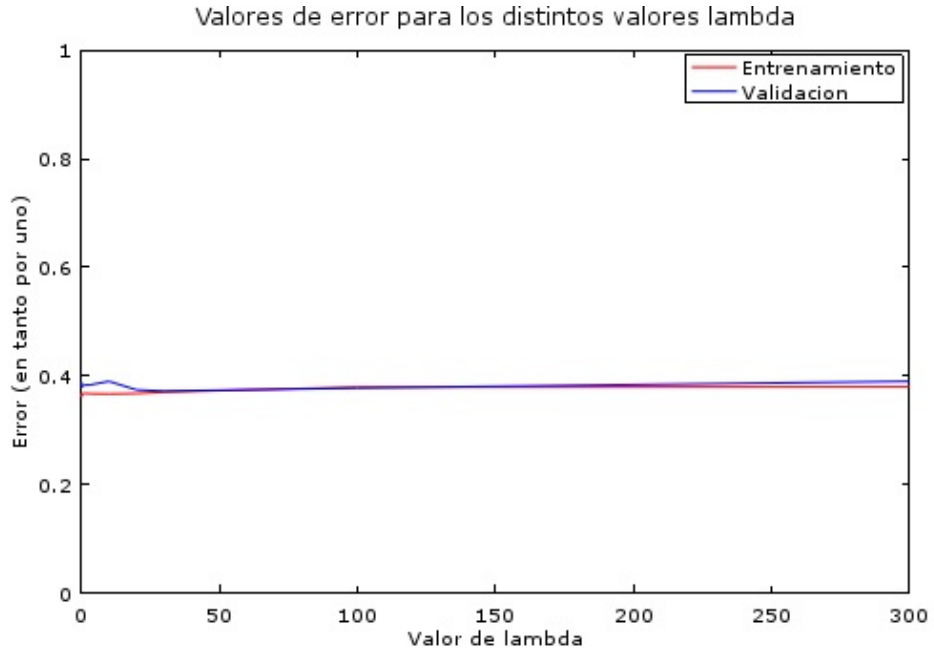
‘training2B.mat’



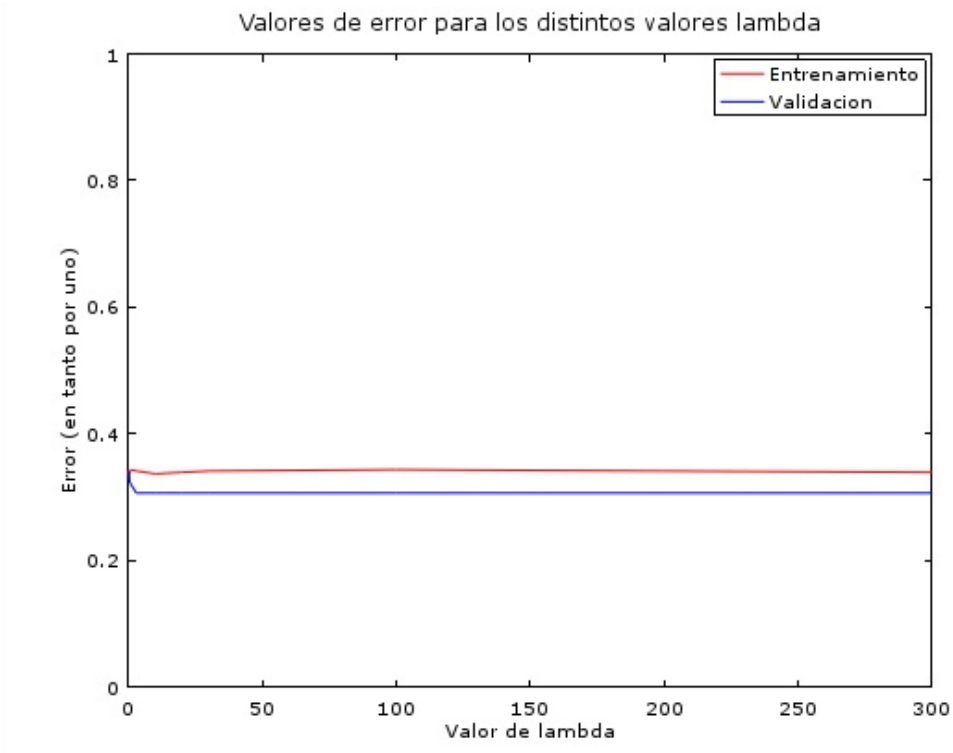
‘trainingC.mat’



‘training2C.mat’

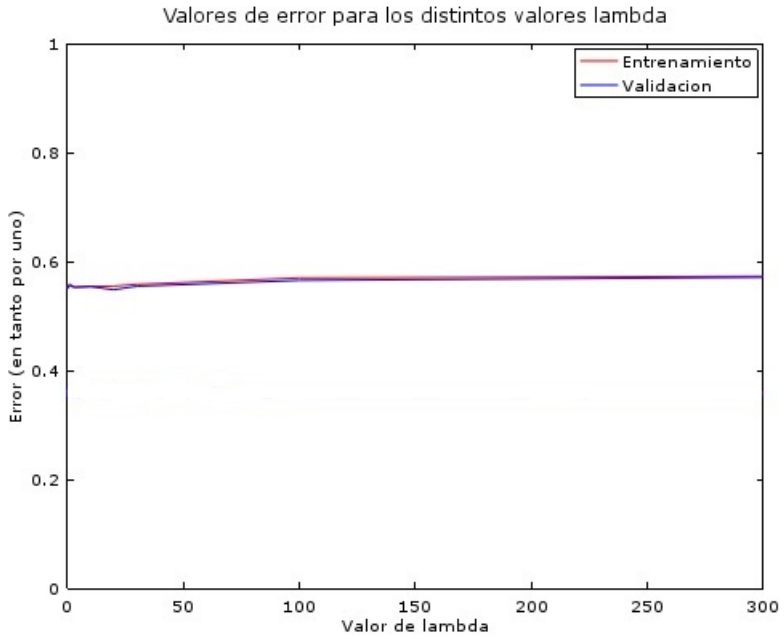


‘trainingMulticlassA.mat’

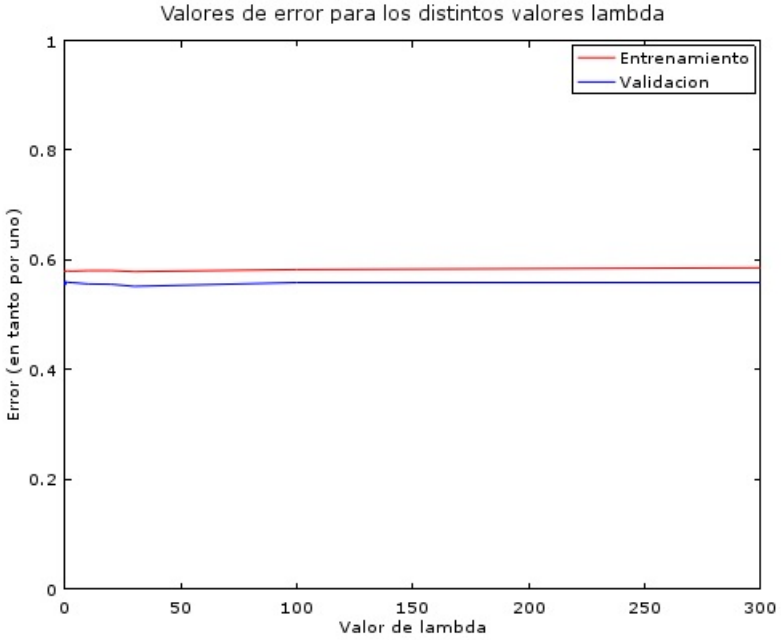


‘trainingMulticlass2A.mat’

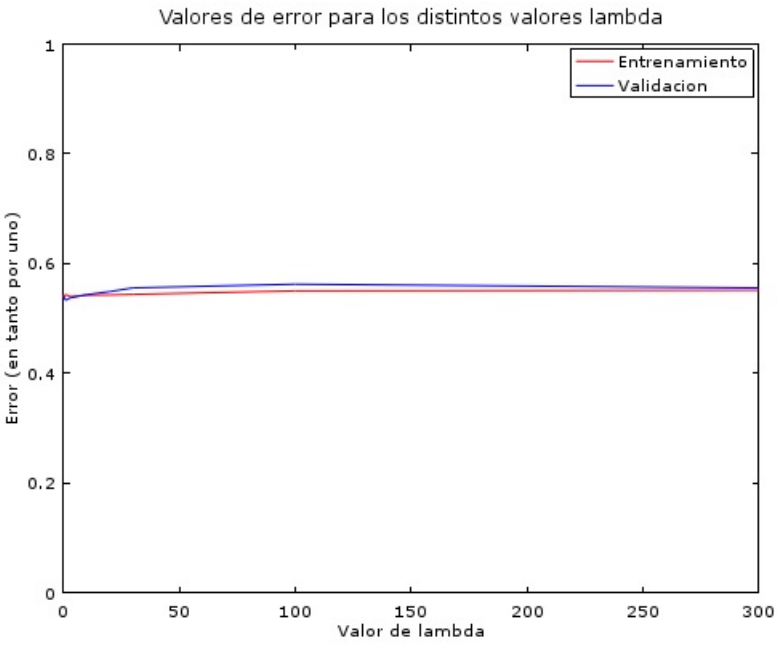
‘trainingMulticlassB.mat’



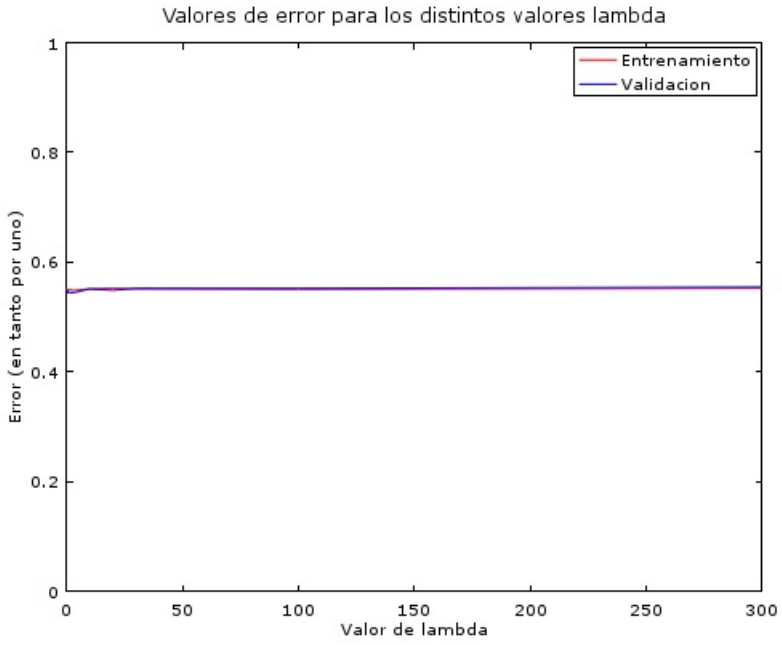
‘trainingMulticlass2B.mat’



‘trainingMulticlassC.mat’

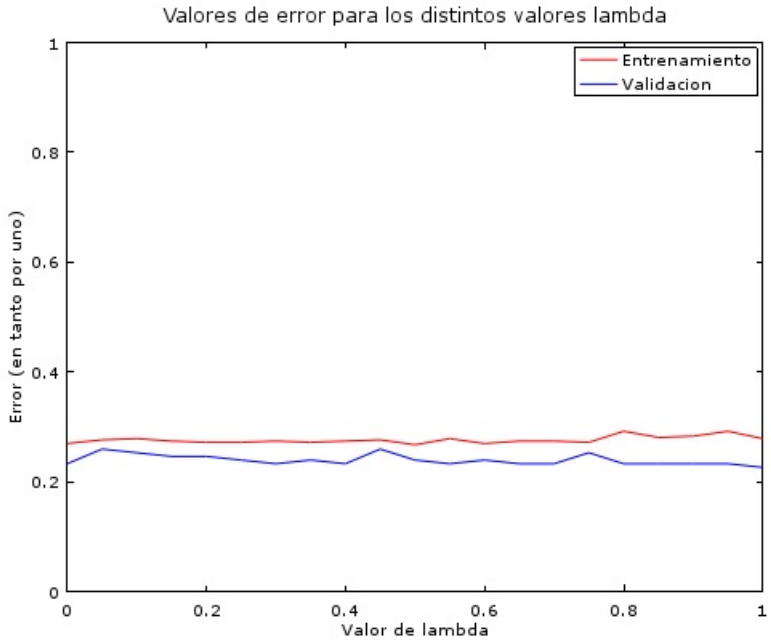


‘trainingMulticlass2C.mat’



Como vemos, no hay diferencias significativas, debido a que la suma de los parámetros ‘theta’ elevados al cuadrado (la cual multiplica al parámetro lambda) no es muy alta, lo que hace que la función de coste no varíe de manera notable.
En este gráfico podemos observar las pequeñas fluctuaciones del error para

‘trainingA.mat’ para valores de lambda entre 0 y 1:



En media, en este ejemplo y en el resto, los valores de lambda próximos a 0 y 1 tienen ligeramente un error menor, por lo que así podemos justificar porqué en las pruebas anteriores hemos usado lambda = 0.01. Por otro lado, a la hora de probar los modelos de manera individual en las secciones posteriores utilizaremos lambda = 1.

Para terminar esta sección, mostramos los errores finales que hemos obtenido utilizando los modelos entrenados para un conjunto de datos de prueba (test data):

Input	***Porcentaje de error***
trainingA.mat	30%
training2A.mat’	37,3%
trainingB.mat	39.3%
training2B.mat	39.3%
trainingC.mat	37.8%
training2C.mat	35.4%
trainingMulticlassA.mat	41.3%
trainingMulticlass2A.mat	30%
trainingMulticlassB.mat	59.7%
trainingMulticlass2B.mat	59.2%
trainingMulticlassC.mat	53.89%
trainingMulticlass2C.mat	55.24%

Parte 2: Regresión logística binaria y SVM (predicción de aprobados y suspensos)

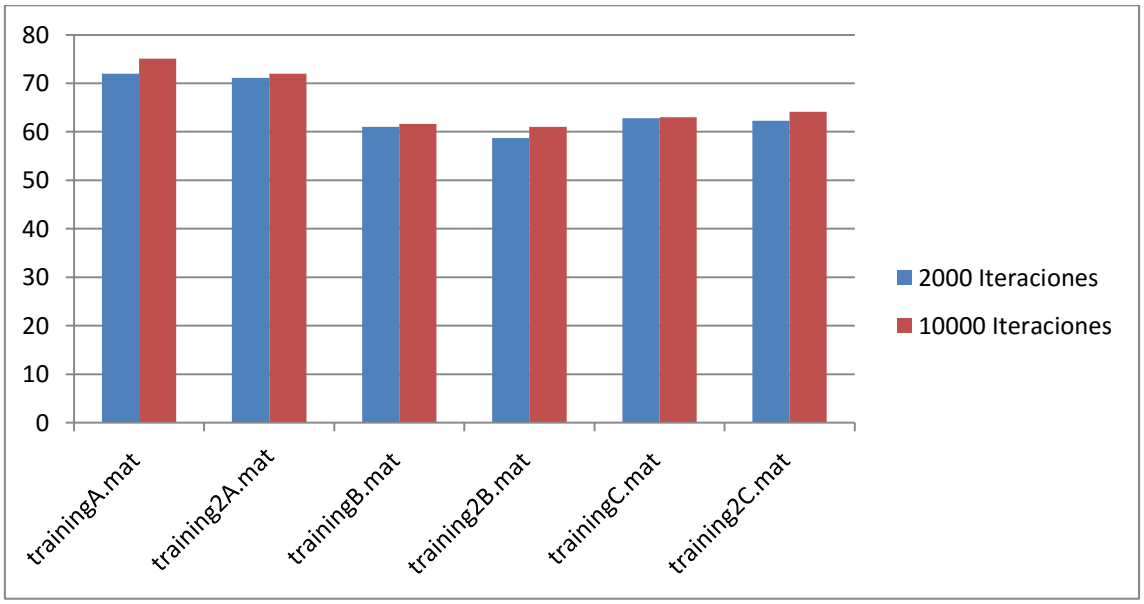
La segunda parte de nuestro proyecto pretende resolver un problema de clasificación binaria. A partir de las matrices con los distintos datos de entrenamiento, queremos predecir si un alumno aprobará o suspenderá una determinada asignatura. Por ello, la columna de calificación final es una variable binaria, la cual indica si un determinado alumno (un ejemplo de entrenamiento) ha aprobado (1) o no (0) una determinada asignatura.

En esta sección utilizaremos dos modelos que hemos aprendido a lo largo de la asignatura: la regresión logística y la SVM.

Los resultados devueltos por cada uno de estos modelos han sido los siguientes:

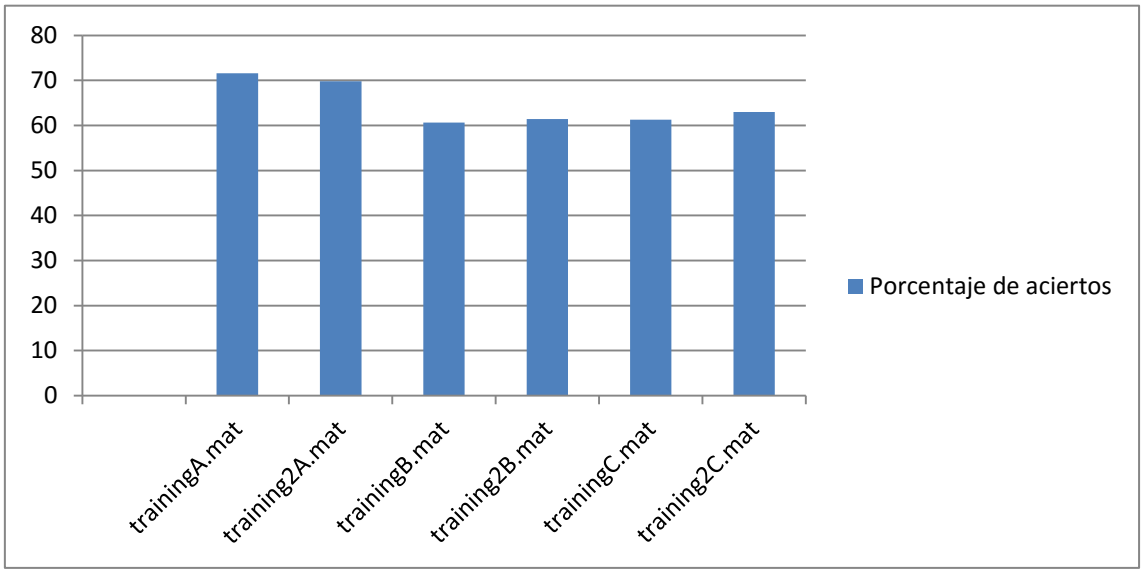
Resultados regresión logística (lambda = 1)

Input	***Número de iteraciones***	***Porcentaje de aciertos***
trainingA.mat	2000	72%
trainingA.mat	10000	75.1%
training2A.mat	2000	71.1%
training2A.mat	10000	72%
trainingB.mat	2000	61%
trainingB.mat	10000	61.6%
training2B.mat	2000	58.7%
training2B.mat	10000	61%
trainingC.mat	2000	62.8%
trainingC.mat	10000	63%
training2C.mat	2000	62.3%
training2C.mat	10000	64.1%



Resultados SVM

Input	***Valor óptimo C***	***Valor óptimo σ***	***Porcentaje de aciertos***
trainingA.mat	30	3	71.55%
training2A.mat	1	1	69.78%
trainingB.mat	1	3	60.64%
training2B.mat	0.01	1	61.41%
trainingC.mat	0.01	3	61.31%
training2C.mat	1	3	62.96%



Los resultados muestran resultados realmente similares entre los dos modelos para cada una de las 3 asignaturas. En primer lugar, observamos que no hay diferencia entre las dos versiones de los inputs de cada asignatura, por lo que esto refuerza la tesis de que el atributo región es redundante y no aporta casi nada de información sobre la predicción de si un alumno va a aprobar o suspender una determinada asignatura. En cuanto al número de iteraciones, vemos como al aumentarlo, este tampoco tiene un efecto significativo en el porcentaje de aciertos.

En resumen, para la asignatura A tenemos en torno al 70% de aciertos, y para las asignaturas B y C un 60% de aciertos.

Comentar en último lugar que el tiempo consumido por el modelo SVM para obtener estos resultados (en torno al par de horas para las asignaturas B y C) ha sido ostensiblemente muy superior al tiempo que tardó la regresión logística (Dos o tres minutos para las asignaturas B y C)

Parte 3: Regresión logística multiclase y redes neuronales (predicción de aprobados, suspensos, abandonos y matrículas de honor)

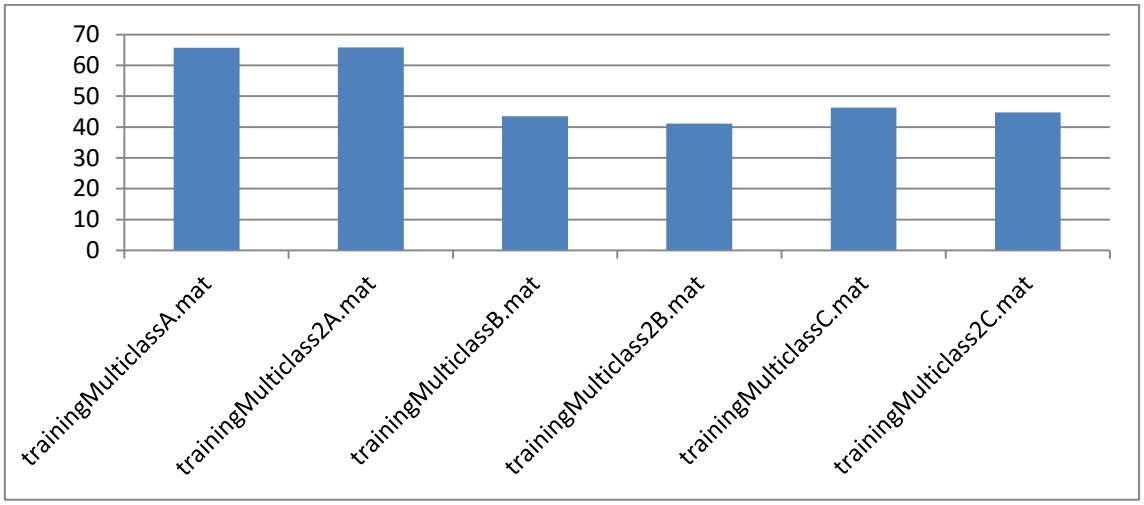
Esta sección pretende ser una ampliación de la sección anterior. En la sección anterior, al obtener los datos de alumnos aprobados y suspensos realizábamos una pequeña modificación en los datos originales del dataset. Este dataset nos proporciona una calificación final para cada alumno con los valores “Pass”, “Distinction”, “Fail” y “Withdrawn”. Sin embargo, considerábamos los alumnos que obtenían matrícula como alumnos aprobados y los alumnos que abandonaban una asignatura como suspensos.

En esta parte conservaremos las calificaciones originales del dataset para intentar resolver un problema de clasificación multiclase: predecir qué tipo de calificación obtendrán los alumnos que tenemos en el conjunto de datos de entrenamiento.
Para ello nos ayudaremos de dos técnicas que hemos visto en la asignatura: regresión logística con clasificación one vs. all y redes neuronales.
La primera de ellas es una ampliación de la regresión logística que usamos en la parte anterior. Ahora usaremos regresión logística para entrenar cada ejemplo de entrenamiento y decidir por separado si pertenece a cada una de las etiquetas o clases. Tras obtener los distintos valores de la función de predicción ‘h’ para cada una de estas clases, nos quedaremos con el mayor de todos ellos, etiquetando los ejemplos de entrenamiento de acuerdo a las clases que devuelven estos valores máximos.
Por otro lado, para la red neuronal se compone de una capa de entrada, una capa oculta y la capa de salida, pues el código utilizado ha sido el mismo que el de la práctica 4.

A continuación, adjunto los resultados obtenidos al aplicar los distintos inputs del dataset para cada uno de estos modelos (lambda = 1 en la regresión logística multiclase y lambda = 0 para las redes neuronales. Se ha elegido lambda = 0 debido a que esto sitúa el error entre el gradiente computado analíticamente y el gradiente computado por retropropagación por debajo de los límites recomendados):

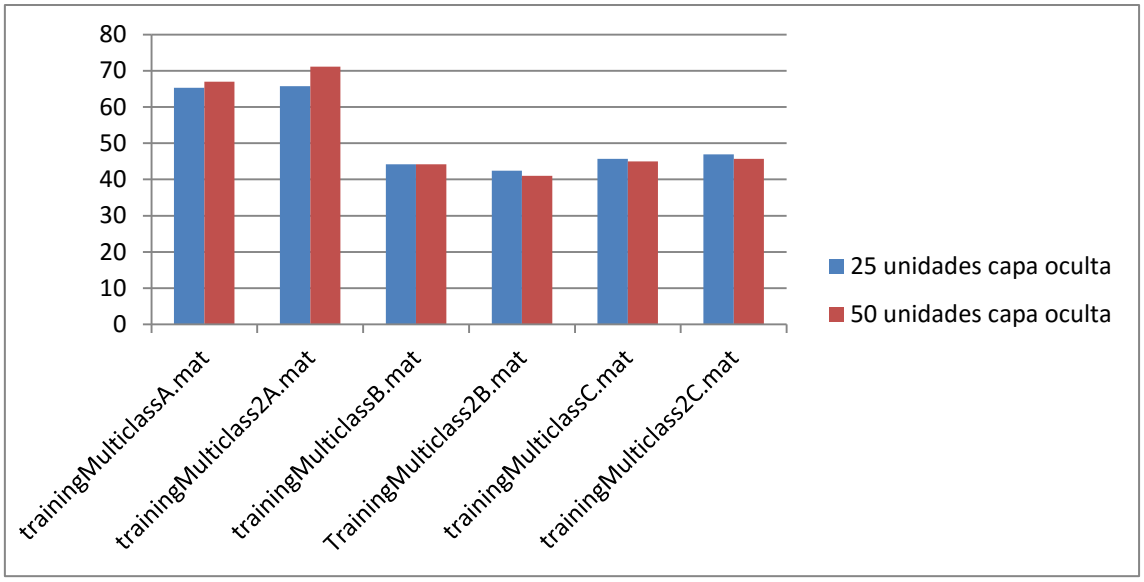
Regresión logística multiclase (one vs. all)

Input	***Iteraciones***	***Porcentaje de aciertos***
trainingMulticlassA.mat	10000	65.7%
trainingMulticlass2A.mat	10000	65.8%
trainingMulticlassB.mat	10000	43.5%
trainingMulticlass2B.mat	10000	41.1%
trainingMulticlassC.mat	10000	46.3%
trainingMulticlass2C.mat	10000	44.7%



Redes neuronales

Input	***Iteraciones***	***Unidades en la capa oculta***	***Porcentaje de aciertos***
trainingMulticlassA.mat	300	25	65.33%
trainingMulticlassA.mat	300	50	67%
trainingMulticlass2A.mat	300	25	65.78%
trainingMulticlass2A.mat	300	50	71.1%
trainingMulticlassB.mat	300	25	44.2%
trainingMulticlassB.mat	300	50	44.2%
TrainingMulticlass2B.mat	300	25	42.43
TrainingMulticlass2B.mat	300	50	41%
trainingMulticlassC.mat	300	25	45.7%
trainingMulticlassC.mat	300	50	45%
trainingMulticlass2C.mat	300	25	46.9
trainingMulticlass2C.mat	300	50	45.7%



Al igual que en la parte 2, los resultados que obtenemos entre los dos modelos que hemos usado son realmente similares. De hecho, no hay diferencias en el porcentaje de aciertos superiores al 5% (nótese que, además, al rejecutar estas pruebas podemos obtener resultados que pueden variar un poco, pues tras cargar las matrices, como comentábamos en la parte de análisis de errores, tomamos datos de manera aleatoria para entrenar y luego validar los modelos).
Tal y como comentábamos en la parte 2, el atributo de región no aporta variaciones positivas en el resultado final, por lo que confirmamos que podemos deshacernos de él con seguridad para poder realizar otras pruebas de manera más rápida en el futuro, pues recordemos que las matrices 'trainingMulticlass2X.mat' tienen 13 columnas menos que las existentes en 'trainingMulticlassX.mat'.

En la regresión multiclase se tarda aproximadamente 4 veces (recordemos que 4 es el número de etiquetas que disponemos) más que la regresión logística binaria, pero los tiempos de ejecución en ese modelo eran bastante pequeños, por lo que he optado por realizar 10000 iteraciones en esta parte.
No obstante, esto no ha sido así para la parte de redes neuronales. A pesar de que he escogido un número pequeño de iteraciones (300), la media del tiempo de ejecución para las asignaturas B y C (las cuales tienen aproximadamente unos 5000 y 8000 registros respectivamente, respecto a los 700 de la asignatura A) ha sido de en torno a los 20-30 minutos. He pensado que una de las razones de este problema es que la codificación de la función de coste de las redes neuronales la hice de manera iterativa, pues no conseguí visualizar (y por tanto programar) la fórmula de manera vectorial.

No obstante, a diferencia únicamente de la asignatura A, las asignaturas B y C disminuyen en porcentaje de aciertos respecto a los resultados del problema de

clasificación binario.
Esto a priori suena lógico, pues el problema que resolvemos en esta parte es un problema ampliado de la parte anterior. Si en esa parte ya comentábamos que los atributos no otorgaban mayor capacidad de mejorar los resultados, entonces en esta parte los resultados iban a ser peores de cierto modo, pues este problema es más complejo.
Sin embargo, a pesar de que contamos con 4 etiquetas y con la limitación de atributos que comentábamos, los resultados combinados de estas dos partes creo que son satisfactorios para el ámbito de nuestro proyecto.

Como decía los resultados que hemos obtenido para la asignatura A son diferentes, pues curiosamente, los porcentajes de acierto en los problemas de clasificación binaria y multiclase son bastante parecidos (en torno al 67-70%).
Esto, unido a los peores resultados que obtenemos en B y C nos hace sospechar que para estas dos asignaturas no es tan fácil predecir en detalle las calificaciones finales como en la asignatura A: El porcentaje de aciertos pasa del 60% en las asignaturas B y C en la clasificación binaria al 40-45% de aciertos en la clasificación multiclase. Esto precisa la idea que llevamos tiempo comentando: necesitamos más atributos para poder precisar mejor los resultados finales de los alumnos, especialmente en las asignaturas B y C.

Parte 4: Clustering (perfiles de estudiantes con resultados aprobado - suspenso)

En esta parte, el objetivo es obtener un perfil demográfico y académico de los estudiantes presentes en el dataset realizando un clustering por medio del algoritmo k-means. Para ello, utilizaremos los mismos datos que en la parte 2 (problema de clasificación binaria), es decir, que la calificación final indicará simplemente si el alumno ha aprobado o ha suspendido. Se podría haber utilizado la columna con las calificaciones detalladas (como la versión utilizada en la parte 3), pero para poder hacer esto se tendría que haber creado variables binarias para cada uno de los distintos tipos de calificaciones, lo cual he creído que no sería conveniente, pues habría aumentado aún más el número de atributos binarios en nuestras matrices de entrada. En esta parte se ha preferido utilizar las matrices que no tienen el atributo de región (aquellas que tienen un 2 en el nombre del archivo), pues esto hubiera dificultado la lectura de resultados de manera importante.
A la hora de interpretar los resultados, para los atributos con variable binaria, el resultado estará entre 0 y 1, el cual indicará el porcentaje de alumnos de cada cluster a tener el valor asociado a 1. Es decir, si por ejemplo en la columna de pertenencia a la región de Gales, el centroide final devuelve un valor de 0.7 esto quiere decir que de los alumnos asociados a ese cluster, el 70% tienen 1 en la columna de pertenencia a la región de Gales, lo cual indica que han residido en Gales mientras estudiaban la asignatura que estemos evaluando en ese momento.

La mayor parte del código utilizado en esta parte ha sido proporcionado o desarrollado para la práctica 7 de la asignatura.

A continuación mostramos los resultados obtenidos para esta parte (30 iteraciones para cada una de las ejecuciones del k-means):

'training2A.mat'

Cluster	Sexo	A Level education	HE Qualification	Lower tan A Level	No formal quals	Post Gruadate qual
1	0.61905	0.52381	0.19048	0.28571	0.00000	0.00000
2	0.61353	0.56039	0.24638	0.19324	0.00000	0.00000
3	0.50000	0.62000	0.23000	0.14500	0.00000	0.00500
4	0.60625	0.45000	0.26562	0.26250	0.00000	0.02188

0 -35 años	35 – 55 años	Mayores de 55	Discapacidad	Convocatorias utilizadas	Creditos matriculados	Calificación
0.57143	0.42857	0.00000	0.04762	0.04762	89	0.57143
0.68116	0.28986	0.02899	0.07246	0.05314	145	0.57488
0.99500	0.00000	0.00500	0.05500	0.03500	60	0.73500
0.00000	0.88125	0.11875	0.04688	0.06875	60	0.79063

'training2B.mat'

Cluster	Sexo	A Level education	HE Qualification	Lower tan A Level	No formal quals	Post Gruadate qual
1	0.11392	0.42198	0.13237	0.43241	0.01123	0.00201
2	0.11383	0.00000	0.00000	100.000	0.00000	0.00000
3	0.11959	0.77063	0.19847	0.00000	0.02763	0.00327
4	0.12016	0.41473	0.11240	0.44961	0.01938	0.00388

0 -35 años	35 – 55 años	Mayores de 55	Discapacidad	Convocatorias utilizadas	Creditos matriculados	Calificación
0.69635	0.30245	0.00120	0.11312	0.33293	134	0.41276
0.65476	0.34400	0.00125	0.10054	0.18031	60	0.41670
0.65358	0.34569	0.00073	0.06870	0.12868	60	0.58451
0.66279	0.33721	0.00000	0.11240	0.19767	87	0.44186

'training2C.mat'

Cluster	Sexo	A Level education	HE Qualification	Lower tan A Level	No formal quals	Post Gruadate qual
1	0.78623	0.46030	0.20822	0.29706	0.00888	0.02554
2	0.79205	0.32110	0.36391	0.24465	0.00000	0.07034
3	0.70503	0.49497	0.17821	0.29777	0.01117	0.01788
4	0.76744	0.43217	0.25969	0.27907	0.00194	0.02713

0 -35 años	35 – 55 años	Mayores de 55	Discapacidad	Convocatorias utilizadas	Creditos matriculados	Calificación
0.69795	0.28928	0.01277	0.07440	0.05497	60	0.41866
0.02446	0.97554	0.00000	0.03364	0.07951	31	0.42202
0.76536	0.22793	0.00670	0.10615	0.03408	117	0.31173
0.97287	0.00000	0.02713	0.08721	0.09496	30	0.44186

Hemos elegido 4 clusters, pues a partir de ese número, el algoritmo k-means devolvía para nuestros inputs varios campos 'NaN', lo cual sugiere que estos 4 cluster segmentan de la mayor manera posible los perfiles académicos y demográficos de los alumnos.

Antes de empezar compararemos los distintos niveles educativos británicos con los españoles, para que sea más fácil de comprender esta sección:

'A Level education' equivale a Educación secundaria
'HE Qualification' equivale a Titulado universitario (Educación superior)
'Lower than A Level' equivale a Eduación inferior a secundaria
'No formal qual' equivale a Sin estudios
'Post graduate qual' equivale a Posgraduado

Para la asignatura A vemos que 3 de los clusters tienen en torno a un 60% de estudiantes varones y el restante un 50%. Respecto al nivel académico, vemos que la mayoría de estudiantes en los 4 clusters son aquellos con educación secundaria. Aquellos con título universitario varían del 19 al 26% y en ocasiones son superados por aquellos que tienen educación menor que la secundara. Nótese como los (pocos)

posgraduados matriculados están situados en los dos clusters con mayor porcentaje de aprobados. Curiosamente, estos clusters (con buenos porcentaje de aprobados) tienen por un lado una casi totalidad de número de alumnos de menos de 35 años y por otro alumnos de 35 a 55 años en su mayoría, y mayores de 55
En cuanto al número de alumnos con discapacidad, estos se distribuyen de manera uniforme por todos los clusters, por lo que no parezca que tenga relación con el número de aprobados.
Observamos también que el número de convocatorias utilizadas se aproxima a 0 en los 4 centroides y que los créditos matriculados son 60 para dos de los clusters (los cuales se corresponden a los que tienen mejores resultados) , y una cantidad notablemente superior para el resto (que no tienen tan buenos resultados).
En general, vemos que en la asignatura A predomina el número de aprobados y que tomando por separado los datos demográficos y los académicos no podemos deducir de manera aproximada si acabaran aprobando o suspendiendo la asignatura.

En la asignatura B los 4 clusters tienen en torno al 90% de estudiantes femeninas. Aquí 3 de los clusters tienen en torno al 60% de suspensos, mientras que el cluster restante tiene un 60% de aprobados. Dentro de estos 3 clusters vemos que tienen alumnos con mayoría de título de educación secundaria o inferior o sin nivel académico. El otro cluster tiene en torno al 77% de alumnos con educación secundaria.
Nótese que a diferencia de la asignatura A, aquí tenemos estudiantes sin estudios, pero esto no parece tener relación con la calificación.
En los 4 clusters tenemos que aproximadamente el 60% tienen menos de 35 años y el 40% tienen entre 35 y 55. Nótese como prácticamente no hay estudiantes mayores de 55 años.
Respecto al número de estudiantes discapacitados, podemos observar que en 3 clusters estos suponen un 10% del total de cada uno de ellos.
El número de convocatorias utilizadas aumenta de manera importante respecto a la asignatura A y de hecho el cluster con el 60% de aprobados tiene 0.128 convocatorias utilizadas de media, el valor más bajo de los 4.
Por último, el número de créditos matriculados sigue un patrón similar al de la asignatura A.
Para la asignatura C los resultados del clustering muestran un 60% de suspensos en 3 clusters, pasando al 70% en el otro cluster.
A diferencia de la asignatura B, para los 4 clusters tenemos mayoría de estudiantes de sexo masculino (en torno al 70-80%). En esta asignatura hay más estudiantes posgraduados (en el segundo cluster hay un 7% de ellos) que en el resto de asignaturas, donde incluso no había ninguno. No obstante, estudiantes con el título de secundaria siguen dominando en los 4 clusters con valores entre el 32 y el 49%. Luego están los estudiantes con título inferior a secundaria (24-29%), con valores similares a graduados universitarios (18-36%). Los alumnos sin estudios poseen porcentajes casi

residuales en esta asignatura.
En cuanto a la edad de los estudiantes, un cluster posee prácticamente estudiantes menores de 35 años, mientras otro es igual pero para alumnos de entre 35 y 55 años, aunque curiosamente este último tiene un porcentaje de aprobados similar al primero. Por otro lado, los mayores de 55 años son prácticamente inexistentes en los 4 clusters. Por último, si nos fijamos en el tercer clúster, vemos que tenemos un 10% de alumnos con discapacidad, y un número bastante grande de créditos matriculados, 117. Este cluster es el que menor porcentaje de aprobados registra.
Es claro que aquí se rompe el patrón que habíamos visto en las otras dos asignaturas referente al número de créditos matriculados, pues ahora tenemos dos cluster con 30 y 31 créditos de media.

4. Conclusiones

En primer lugar, de cierto modo los resultados son satisfactorios, pues, aunque los resultados no son realmente buenos (de hecho, para el problema de clasificación multiclase se ha llegado a bajar del 50% en aciertos), tenemos que valorar que hemos tomados datos principalmente de carácter demográfico y alguno de carácter académico (pero que tampoco son especialmente relevantes, como el número de convocatorias utilizadas o el número de créditos matriculados).
Tal y como dije en las primeras secciones del documento, mi objetivo no era intentar conseguir los mejores resultados por todos los medios (para ello simplemente hubiera que tenido que incluir atributos como por ejemplo la media de interacciones del alumno con el VLE o las calificaciones de las entregas, lo cual es casi seguro que hubiera aumentado el porcentaje de aciertos para los modelos que hemos probado), si no diseñar un sistema de aprendizaje automático flexible y escalable para los distintos inputs del OULAD.
Aunque solo hemos probado con una pequeña parte de los datos de este dataset (nótese como el código del proyecto sería fácilmente adaptable a nuevos inputs), esto nos ha permitido explorar los datos del mismo mediante las conclusiones asociadas a la parte de clustering y a los resultados de los distintos modelos de aprendizaje automático.

El planteamiento que he seguido a la hora de diseñar los distintos códigos del proyecto ha sido el denominado “Large Data Rationale” en las transparencias de la asignatura. Con esto he tratado de incluir tantos atributos de carácter demográfico como he podido para intentar aumentar la varianza (el valor de la función de coste será bajo para los datos de entrenamiento debido al overfitting).No obstante, como el dataset es bastante amplio y registra un gran número de alumnos para cada una de las asignaturas (aunque quizás no tanto para la asignatura A en comparación con las otras dos) esto ayudará a reducir la varianza, evitando el overfitting (por lo que la función de coste devolverá un resultado para los datos de entrenamiento parecido al de los datos de validación) obteniendo así valores bajos de error para los datos de entrenamiento y validación.
Aunque a priori los datos demográficos no parecen algo realmente determinante para predecir una calificación final, hemos visto que esta información puede ser un buen punto de partida para ambos problemas de clasificación, especialmente en la clasificación binaria. Por ello, con los distintos resultados hemos deducido que el campo región puede ser eliminado sin causar grandes variaciones en el porcentaje de aciertos, ahorrándonos una importante cantidad de memoria y tiempo de ejecución.
No obstante, habría que ver qué resultados obtendrían de manera conjunta los atributos demográficos y los posibles nuevos atributos de carácter académico. Aquí es cuando podemos usar las exhaustivas pruebas que he diseñado en este proyecto y deducir los resultados con la ayuda de las distintas gráficas que se van obteniendo.

Si he de hacer una pequeña crítica al dataset sería que hay ciertos momentos en los que se precisaría algún detalle más sobre las asignaturas (quizás la rama a la que pertenecen o alguna lista de requisitos y conocimientos necesarios para cursarlas sería bastante interesante), pues hemos visto en la parte de clustering alguna dificultad para analizar los resultados, especialmente en la asignatura C.

Par finalizar, una posible mejora de cara al proyecto sería añadir la creación de una matriz de confusión para deducir y mostrar que clase de errores estamos teniendo en el problema multiclase.

5. Anexo (Listado y descripción de los archivos .r, .m y .mat. Código desarrollado para el proyecto)

dataCleaning.r

Script en R que tiene como input 'studentInfo.csv' y devuelve las matrices con los atributos necesarios para los problemas de clasificación binaria y multiclase. También se devuelven matrices para estos problemas de clasificación pero sin contar con los atributos de región.
Es necesario cambiar a mano la asignatura que queremos obtener del dataset inicial.

analisisErrores.m

La parte 1 del proyecto (análisis de errores) se ha realizado con este script. Las funciones 'pruebasRegLogBin' y 'pruebasRegLogMult' realizan las pruebas de detección de sesgo o varianza (curvas de aprendizaje), evolución del número de falsos positivos y negativos y la elección del parámetro lambda óptimo para el problema de clasificación binario y el multiclase respectivamente. Las pruebas se realizan de manera consecutiva utilizando la función 'figure' para realizar varios plot seguidos. Estas dos funciones toman como argumentos el nombre del archivo a cargar, el parámetro lambda con el que se computan las curvas de aprendizaje, el número máximo de iteraciones y el parámetro 'lambda_final' para calcular el error final.
Por último, comentar que en estas dos funciones hay sendos bloques de código comentado que al descomentarlas añaden de manera automática potencias hasta un cierto orden al atributo "número de convocatorias utilizadas". Para ello se utiliza la función 'genMatPol', la cual, al igual que gran parte del código de este script, esta reutilizado a partir del código de la práctica 5.

Otras funciones auxiliares en este script son 'error' y 'error2', las cuales computan el tanto por uno de errores cometidos en la predicción para el problema de clasificación binario y el multiclase respectivamente. En la función 'error' se devuelve también el número de falsos positivos y falsos negativos.

checkNNGradients.m

Función que compara los gradientes de la función de coste de una red neuronal: uno es el gradiente calculado numéricamente por 'computeNumericalGradient.m' y otro es el gradiente calculado por retro-propagación en 'redesNeuronales.m'. Este proceso se realiza construyendo una pequeña red neuronal.

computeCentroids.m

Dado un conjunto de entrenamiento, un array que indica a que centroide pertenece cada ejemplo de entrenamiento y un número de clusters, recalcula el valor de los centroides computando el valor medio de los ejemplos asociados a cada uno de ellos.

computeNumericalGradient.m

Cálculo numérico del gradiente de la función de coste de una red neuronal.

debugInitializeWeights.m

Función auxiliar de checkNNGradients.m que inicializa los pesos de la red

findClosestCentroids.m

Dada una matriz de ejemplos de entrenamiento y un conjunto de centroides, devuelve un vector con el índice del centroide más cercano para cada uno de los ejemplos de entrenamiento.

fmincg.m

Implementación de una función de minimización (similar a fminunc).

h.m

Función de predicción para la regresión logística (binaria y multiclase). Usa la función en 'sigmoide.m'

kMeansProy.m

Por medio de la función 'clusteringProyec' se realiza el clustering (utilizando la función 'runkMeans') para obtener los perfiles académicos y demográficos de los alumnos. Esta función toma como parámetros el archivo origen, el número de clusters o centroides y el número total de iteraciones a realizar. La función 'randomCentroids' inicializa los centroides con datos de estudiantes elegidos de manera aleatoria.

linearKernel.m

Kernel lineal para una SVM

lrCostFunction.m

Versión vectorizada de la función de coste y gradiente para la regresión logística (binaria y multiclase)

redesNeuronales.m

Script utilizado para la parte 3 del proyecto. Construye una red neuronal y mide su efectividad calculando el porcentaje de aciertos.
La función 'redesNeuronalesProy' toma como argumentos el nombre del archivo a cargar, el parámetro lambda, el número de entradas ocultas, el número de etiquetas (recordemos que en esta parte del proyecto tratamos de resolver un problema de clasificación multiclase), el número máximo de iteraciones y devuelve el porcentaje de ejemplos de entrenamiento correctamente clasificados. La función que calcula el coste y el gradiente ('costeRn'), la función 'pesosAleatorios' de inicialización de las matrices de pesos, la función de predicción 'h', 'toVec' (que convierte una etiqueta en un vector de ceros menos en el índice indicado por la etiqueta, donde se carga un 1), la función derivada de la función sigmoide ('dsigmoide') y la función sigmoide ('sigmoide') han sido encapsuladas en este script dado que no son reutilizadas en ninguna otra parte del proyecto.
Estas funciones fueron todas programadas para la práctica 4 de la asignatura.

regresionLogisticaBinaria.m

Este script se utiliza en la parte 2 del Proyecto, en el problema de clasificación binaria.
La función 'trainBinary' toma por argumento el archivo a utilizar en la regresión logística binaria, un parámetro lambda y muestra el porcentaje de ejemplos de entrenamiento correctamente clasificados.
La función 'regresiónLogística' optimiza la función de coste (la cual está definida en 'lrCostFunction.m') y devuelve el vector 'theta' óptimo. Por último, 'porcCorrectos' computa el número de ejemplos de entrenamiento correctamente clasificados a partir del vector 'theta' óptimo.

regresionLogisticaMulticlase.m

Este script es bastante similar a 'regresionLogisticaBinaria.m'. la función 'trainMulticlass' recibe el número de etiquetas, el parámetro lambda y el nombre del archivo a utilizar, carga este último y realiza una regresión logística multiclase sobre el mismo, devolviendo el porcentaje de ejemplos de entrenamiento correctamente clasificados. Para lograrlo se utiliza la función 'oneVsAll' que permite realizar una clasificación multiclase obteniendo tantos parámetros 'theta' como número de etiquetas.
Por último, la función 'porcCorrectos2' calcula el porcentaje de acierto al clasificar los distintos ejemplos de entrenamiento.

runkMeans.m

Ejecuta el algoritmo k-means utilizando las funciones 'findClosestCentroids' y 'computeCentroids'.

sigmoide.m

Función sigmoide utilizada al realizar las regresiones logísticas (binarias y multiclase)

svm.m

Modelo SVM utilizado en la parte 2 del proyecto (problema de clasificación binario). La función 'SVMProy' carga los datos de entrenamiento a partir del nombre del archivo que se pasa por argumento y lanza la ejecución de la SVM a través de la función 'chooseParam'. Esta función entrena una SVM con kernel Gaussiano probando distintas combinaciones de parámetros C y sigma. Finalmente, la función devuelve los parámetros óptimos C, sigma y el porcentaje de aciertos de la SVM.
También podemos encontrar en este script la función 'gaussianKernel', que calcula el kernel Gaussiano que utiliza el entrenador de la SVM, y la función 'porCorrect', que computa el porcentaje de datos de entrenamiento correctamente clasificados.

svmPredict.m

Función que utiliza una SVM para hacer predicciones

svmTrain.m

Función que entrena una SVM

trainingA.mat
trainingB.mat
trainingC.mat

Matrices con los ejemplos de entrenamiento para las asignaturas A, B y C respectivamente. El atributo calificación final es una variable binaria (aprobado – suspenso).

training2A.mat
training2B.mat
training2C.mat

Matrices con los mismos datos que trainingA.mat, trainingB.mat y trainingC.mat, pero sin los atributos correspondientes al campo región.

trainingMulticlassA.mat
trainingMulticlassB.mat
trainingMulticlassC.mat

Matrices con los ejemplos de entrenamiento para las asignaturas A, B y C respectivamente. El atributo calificación final toma 4 valores: “Pass”, “Distinction”, “Fail” y “Withdrawn” (Codificados como los números enteros 1,2,3 y 4).

trainingMulticlass2A.mat
trainingMulticlass2B.mat
trainingMulticlass2C.mat

Matrices con los mismos datos que trainingMulticlassA.mat, trainingMulticlassB.mat y trainingMulticlassC.mat, pero sin los atributos correspondientes al campo región.

En último lugar, adjunto el código que he desarrollado en las prácticas y que ha sido utilizado en el proyecto, así como el código desarrollado expresamente para el proyecto:

dataCleaning.r

```
# Filtrado y limpieza de datos para el proyecto de predicción de resultados académicos

# Open University Learning Analytics Dataset

# Aprendizaje Automático y Big Data

# Alberto Terceño Ortega - 4º Doble Grado Ingeniería Informática y Matemáticas

# Universidad Complutense de Madrid


library(R.matlab)

library(ade4)



x <- read.csv(file="B:/Documentos/UCM/Erasmus/AABD/proyecto/studentInfo.csv", header=TRUE, sep=",")


#Ejecutar cada una de las líneas comentadas para obtener los datos para otras asignaturas

x <- x[x$code_module == "AAA",]

# x <- x[x$code_module == "BBB",]

# x <- x[x$code_module == "CCC",]


# Levels de los factors del dataset original

# "Equivalente" a age_band <- levels(x$age_band), education <- levels(x$highest_education), regions <-
levels(x$region),

# results <- levels(x$final_result)


age_band <- unique(x$age_band)

education <- unique(x$highest_education)

regions <- unique(x$region)

results <- unique(x$final_result)



# Cada level del factor es una columna con indicador 1 o 0

# Los factors que consideramos son género, región, nivel de educación, banda de edad y discapacidad

# En df2 no consideramos el atributo de región

df <- acm.disjonctif(x[,c("gender", "region", "highest_education", "age_band", "disability")])

df2 <- acm.disjonctif(x[,c("gender", "highest_education", "age_band", "disability")])

drops <- c("gender.F", "disability.N")
```

```
df <- df[, !(names(df) %in% drops)]

df2 <- df2[, !(names(df2) %in% drops)]


# Añadimos número de intentos, créditos estudiados pero no como factors, sino como integers

# El resultado final será 1 si hay aprobado o matrícula y 0 en caso contrario (suspenso o abandono)

df$attempts <- x[,c("num_of_prev_attempts")]

df$studied_credits <- x[,c("studied_credits")]

df$final_result <- 0

df[x$final_result == "Pass" | x$final_result == "Distinction" , c("final_result")] <- 1


df2$attempts <- x[,c("num_of_prev_attempts")]

df2$studied_credits <- x[,c("studied_credits")]

df2$final_result <- 0

df2[x$final_result == "Pass" | x$final_result == "Distinction" , c("final_result")] <- 1


# Cast a matrices

a <- as.matrix(df)

b <- as.matrix(df2)


#Guardado de archivos .mat Cambiar nombre para otras asignaturas

writeMat("trainingA.mat", x=a)

writeMat("training2A.mat", x=b)


# Ahora para el resultado final vamos etiquetando según los distintos levels del factor final_result.

df$final_result <- 0

for (i in 1:length(results)){

  df$final_result[x$final_result == results[i]] <- i

}


df2$final_result <- 0

for (i in 1:length(results)){

  df2$final_result[x$final_result == results[i]] <- i

}


# Cast a matrices

df <- as.matrix(df)
```

```
df2 <- as.matrix(df2)


#Guardado de archivos .mat Cambiar nombre para otras asignaturas

writeMat("trainingMulticlassA.mat", x=df)

writeMat("trainingMulticlass2A.mat", x=df2)
```

analisisErrores.m

```
addpath(pwd)


function [err fp fn] = error(theta, X, y)


 m = length(y);


 y_pred = h(theta,X) >= 0.5;

 err = (1/m)*sum( abs( y_pred - y) );


 total_errores = sum( abs( y_pred - y) );

 balance = sum( y_pred - y );

 fp = (balance + total_errores)/2;

 fn = total_errores - fp;



endfunction



function err = error2(all_theta, X, y)



  all_theta = all_theta';

  m = length(y);


  res = h(all_theta,X);


  y_pred = zeros(m, 1);


  for i=1:m

    [maxV ind] = max(res(i,:));

    y_pred(i) = ind;

  endfor
```

```
  err = (1/m)*(m - sum( y == y_pred ));



endfunction


function M = genMatPol(v, p)

  M = v;

  for i = 2:p

    M = [M (v .^i)];

  endfor

endfunction



function pruebasRegLogBin(filename, lambda, max_iters, lambda_final)


    load(filename);

    n = columns(x);

    randidx = randperm(rows(x));

    x = x(randidx,:);


    X = x(:,1:(n-1));

    m = rows(X);

    X = [ ones(m,1) X];

    y = x(:,n);


    %%%%%%%% Descomentar para añadir valores polinómicos al atributo número convocatorias utilizadas

%    p = 6;

%    X = x(:,1:(n-3));

%    X2 = genMatPol(x(:,(n-2)), p);

%    X = [X X2 x(:,(n-1))];

%    n = columns(X);

    %%%%%%%%%%%%%%%%%%%%


    initial_theta = zeros(n,1);


    num_train = floor(m*0.6);
```

```
    num_val = num_train + 1 + floor(m*0.2);



    Xval = X((num_train+1):num_val,:);
    yval = y((num_train+1):num_val);

    Xtest = X((num_val+1):end,:);
    ytest = y((num_val+1):end);

    X = X(1:num_train,:);
    y = y(1:num_train);

    m = rows(X);



    errEnt = zeros(m,1);
    errVal = zeros(m,1);
    fpEnt = zeros(m,1);
    fpVal = zeros(m,1);
    fnEnt = zeros(m,1);
    fnVal = zeros(m,1);

   options = optimset( 'GradObj' , 'on' , 'MaxIter' , max_iters);

   for i = 1:m
         [theta] = fmincg(@(t)(lrCostFunction(t,X(1:i,:),y(1:i),lambda)), initial_theta, options);
         [errEnt(i) fpEnt(i) fnEnt(i)] = error(theta, X(1:i,:), y(1:i));
         [errVal(i) fpVal(i) fnVal(i)] = error(theta, Xval, yval);
   endfor



%Curvas de aprendizaje
    figure(1);
    hold on;
    axis([1,m,0,1]);
    title( "Curvas de aprendizaje para la regresion logistica binaria","fontsize", 12 );
```

```
    xlabel("Numero de ejemplos de entrenamiento", "fontsize", 10);

    ylabel("Error (en tanto por uno)","fontsize",10);

    x_axis = (1:1:m)';

    plot(x_axis, errEnt, "color", "red");

    plot(x_axis, errVal, "color", "blue");

    legend("Entrenamiento", "Validacion");


%Ploteo evolucion falsos positivos y falsos negativos

    figure(2);

    hold on;

    axis([1,m,0,m]);

    title( "Evolucion del numero de falsos positivos y falsos negativos (Datos de entrenamiento)","fontsize",
12 );

    xlabel("Numero de ejemplos de entrenamiento", "fontsize", 10);

    ylabel("Numero de falsos positivos y negativos","fontsize",10);

    x_axis = (1:1:m)';

    plot(x_axis, fpEnt, "color", "red");

    plot(x_axis, fnEnt, "color", "blue");

    legend("Falsos positivos", "Falsos negativos");




    figure(3);

    hold on;

    axis([1,m,0,m]);

    title( "Evolucion del numero de falsos positivos y falsos negativos (Datos de validacion)","fontsize", 12
);

    xlabel("Numero de ejemplos de validacion", "fontsize", 10);

    ylabel("Numero de falsos positivos y negativos","fontsize",10);

    x_axis = (1:1:m)';

    plot(x_axis, fpVal, "color", "red");

    plot(x_axis, fnVal, "color", "blue");

    legend("Falsos positivos", "Falsos negativos");


% Elección Lambda




  lambda_vec = [0; 0.001; 0.003; 0.01;  0.03; 0.1; 0.3; 1; 3; 10; 20; 30; 100; 300];
```

```
  l = length(lambda_vec);

  errEnt = zeros(l,1);
  errVal = zeros(l,1);

  initial_theta = zeros(n,1);

  for i = 1:l
    lambda = lambda_vec(i);
    [theta] = fmincg(@(t)(lrCostFunction(t,X,y,lambda)), initial_theta, options);
    [errEnt(i) dummy dummy2] = error(theta, X, y);
    [errVal(i) dummy dummy2] = error(theta, Xval, yval);
  endfor

  figure(4);
  hold on;
  axis([0,lambda_vec(l),0,1]);
  title( "Valores de error para los distintos valores lambda","fontsize", 12 );
  xlabel("Valor de lambda", "fontsize", 10);
  ylabel("Error (en tanto por uno)","fontsize",10);
  plot(lambda_vec, errEnt, "color", "red", "linewidth", 1);
  plot(lambda_vec, errVal, "color", "blue", "linewidth", 1);
  legend("Entrenamiento", "Validacion");

  lambda = lambda_final;
  [theta] = fmincg(@(t)(lrCostFunction(t,X,y,lambda)), initial_theta, options);
  [errorFinal fpFinal fnFinal] = error(theta, Xtest, ytest)


endfunction


function pruebasRegLogMult(filename, lambda, max_iters, num_etiquetas, lambda_final)

    load(filename);
    n = columns(x);
```

```
    randidx = randperm(rows(x));
    x = x(randidx,:);

    X = x(:,1:(n-1));
    m = rows(X);
    X = [ ones(m,1) X];
    y = x(:,n);
    %%%%%%%% Descomentar para añadir valores polinómicos al atributo número convocatorias utilizadas
%    p = 3;
%    X = x(:,1:(n-3));
%    X2 = genMatPol(x(:,(n-2)), p);
%    X = [X X2 x(:,(n-1))];
%    n = columns(X);
    %%%%%%%%%%%%%%%%%%%

    initial_theta = zeros(n,1);

    num_train = floor(m*0.6);
    num_val = num_train + 1 + floor(m*0.2);

    Xval = X((num_train+1):num_val,:);
    yval = y((num_train+1):num_val);

    Xtest = X((num_val+1):end,:) ;
    ytest = y((num_val+1):end);

    X = X(1:num_train,:);
    y = y(1:num_train);

    m = rows(X);

    initial_theta = zeros(n,1);
```

```
    all_theta = zeros(num_etiquetas,n);
    options = optimset( 'GradObj' , 'on' , 'MaxIter' , max_iters);

    errEnt = zeros(m,1);
    errVal = zeros(m,1);

    for i = 1:m
         for c=1:num_etiquetas
                  [theta] = fmincg(@(t)(lrCostFunction(t,X(1:i,:),(y(1:i)==c),lambda)), initial_theta,
options);
                  all_theta(c,:) = theta';
         endfor

         errEnt(i) = error2(all_theta, X(1:i,:), y(1:i));
         errVal(i) = error2(all_theta, Xval, yval);
    endfor



    %Curvas de aprendizaje
     figure(1);
     hold on;
     axis([1,m,0,1]);
     title( "Curvas de aprendizaje para la regresion logistica multiclase","fontsize", 12 );
     xlabel("Numero de ejemplos de entrenamiento", "fontsize", 10);
     ylabel("Error (en tanto por uno)","fontsize",10);
     x_axis = (1:1:m)';
     plot(x_axis, errEnt, "color", "red");
     plot(x_axis, errVal, "color", "blue");
     legend("Entrenamiento", "Validacion");



   %Elección Lambda

    lambda_vec = [0; 0.001; 0.003; 0.01;  0.03; 0.1; 0.3; 1; 3; 10; 20; 30; 100; 300];
    l = length(lambda_vec);
```

```
    errEnt = zeros(l,1);
    errVal = zeros(l,1);

    initial_theta = zeros(n,1);

    for i = 1:l
       lambda = lambda_vec(i);
       for c=1:num_etiquetas
           [theta] = fmincg(@(t)(lrCostFunction(t,X,(y==c),lambda)), initial_theta, options);
           all_theta(c,:) = theta';
       endfor

       errEnt(i) = error2(all_theta, X, y);
       errVal(i) = error2(all_theta, Xval, yval);
     endfor

    figure(2);
    hold on;
    axis([0,lambda_vec(l),0,1]);
    title( "Valores de error para los distintos valores lambda","fontsize", 12 );
    xlabel("Valor de lambda", "fontsize", 10);
    ylabel("Error (en tanto por uno)","fontsize",10);
    plot(lambda_vec, errEnt, "color", "red", "linewidth", 1);
    plot(lambda_vec, errVal, "color", "blue", "linewidth", 1);
    legend("Entrenamiento", "Validacion");

     lambda = lambda_final;
     for c=1:num_etiquetas
           [theta] = fmincg(@(t)(lrCostFunction(t,X,(y==c),lambda)), initial_theta, options);
           all_theta(c,:) = theta';
       endfor
     errorFinal = error2(all_theta, Xtest, ytest)


endfunction
```

computeCentroids.m

```
function centroids = computeCentroids(X, idx, k)

m = rows(X);

n = columns(X);

card = zeros(k,m);

lengths = ones(k,1);

for i = 1:m

  j = idx(i);

  card(j,lengths(j)) = i;

  lengths(j) = lengths(j) + 1;

endfor


lengths = lengths - ones(k,1);

centroids = zeros(k,n);

for i=1:k


  sum = zeros(1,n);

  l = lengths(i);

  for j = 1:l

    sum = sum + X(card(i,j),:);

  endfor


  centroids(i,:) = (1/l)*sum;


endfor
```

findClosestCentroids.m

```
function idx = findClosestCentroids(X,centroids)

  m = rows(X);

  k = rows(centroids);


  idx = zeros(m,1);


  for i = 1:m

    min = norm(X(i,:) - centroids(1,:))^2;

    idx(i) = 1;

    for j = 2:k

      val = norm(X(i,:) - centroids(j,:) )^2;

      if( val < min )

        min = val;

        idx(i) = j;

      endif

    endfor

  endfor
```

h.m

```
function res = h(theta,x)


  res = sigmoide(x*theta);
```

kMeansProy.m

```
addpath(pwd);

function centroids = randomCentroids(X,k)

  m = rows(X);

  randidx = randperm(m);

  centroids = X(randidx(1:k),:);


endfunction


function clusteringProyec(filename, num_centroids, max_iters)




  load(filename);

  initial_centroids = randomCentroids(x,num_centroids);

  plot_progress = false;

  [centroids, idx] = runkMeans(x, initial_centroids, max_iters, plot_progress);


  centroids

endfunction
```

lrCostFunction.m

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



  m = length(y);

  n = columns(X);



  J = (1/m)*sum(( -y .* log(h(theta,X)) ) .- ( ( 1 .- y) .* (log(1 .- h(theta,X))) )) +
(lambda/(2*m))*sum(theta(2:end) .^2);



  grad = (1/m)*X'*(h(theta,X) .- y);



  grad(2:n) = grad(2:n) .+ (lambda/m)*theta(2:n);



  #for j = 2:n

   # grad(j) = grad(j) + (lambda/m)*theta(j);

  #endfor
```

redesNeuronales.m

```
addpath(pwd);


function res = sigmoide(z)


  res = 1 ./ (1 .+ exp(-z));


endfunction


function res = dsigmoide(z)


  res = sigmoide(z).*(1 - sigmoide(z));


endfunction




function res = toVec(num_etiquetas,y)
  res = zeros(num_etiquetas,1);
  res(y) = 1;
endfunction


function res = h(Theta1, Theta2, X)
  m = rows(X);
  X = [ ones(m,1) X];


  X = X';
  z1 = sigmoide(Theta1 * X) ;


  z1 = [ones(1,m); z1 ];


  res = sigmoide(Theta2 * z1);


endfunction


function W= pesosAleatorios (L_in , L_out)
  epsIni = (sqrt(6/(L_in+L_out)));
```

```
  W = -epsIni .+ 2*epsIni .* rand(L_out, L_in+1);


endfunction


function [ J grad ] = costeRN(params_rn , num_entradas , num_ocultas , num_etiquetas , X, y , lambda )


  m = rows(X);

  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) ) ;


  h = h(Theta1,Theta2,X);


  J = 0;

  for i = 1:m

    y_aux = toVec(num_etiquetas,y(i));

    for k = 1:num_etiquetas

      J = J +((-y_aux(k)*log(h(k,i))) - ( 1 - y_aux(k))*log(1 - h(k,i)));

    endfor

  endfor

  J = (1/m)*J;


  sumReg = sum((Theta1 .^2)(:)) + sum((Theta2 .^2)(:));

  J = J + (lambda/(2*m))*sumReg;


  grad1 = zeros(rows(Theta1), columns(Theta1));

  grad2 = zeros(rows(Theta2), columns(Theta2));

  for i=1:m

    a1 = X(i,:);

    a1 = [ 1 a1];

    a1 = a1';

    a2 = sigmoide(Theta1 * a1) ;

    a2 = [ 1; a2 ];

    a3 = sigmoide(Theta2 * a2);


    d3 = a3 .- toVec(num_etiquetas,y(i));
```

```
    d2 = ((Theta2')*d3)(2:end) .* dsigmoide(Theta1 * a1);
    grad1 = grad1 .+ d2*((a1)');
    grad2 = grad2 .+ d3*((a2)');
  endfor




  grad = (1/m) * [grad1(:); grad2(:) ];



endfunction


function redesNeuronalesProy(filename, lambda, num_ocultas, num_etiquetas, max_iters)

  load(filename);
  randidx = randperm(rows(x));
  x = x(randidx,:);
  n = columns(x);
  X = x(:,1:(n-1));
  y = x(:,n);

  m = rows(X);

  num_train = floor(m*0.7);

  Xval = X((num_train+1):end,:);
  yval = y((num_train+1):end);
  X = X(1:num_train,:);
  y = y(1:num_train);

  checkNNGradients(lambda)

  m = rows(X);
```

```
  num_entradas = columns(X);

  % 25 ocultas en la practica 4

  % 4 etiquetas

  % 50 - 100 iteraciones



  initial_theta = [pesosAleatorios(num_entradas,num_ocultas)(:); pesosAleatorios(num_ocultas,
num_etiquetas)(:)];



  options = optimset( 'GradObj' , 'on' , 'MaxIter' , max_iters);



  [theta] = fmincg(@( t ) ( costeRN ( t , num_entradas, num_ocultas, num_etiquetas, X, y, lambda ) ) ,
initial_theta , options ) ;



  theta1 = reshape (theta (1: num_ocultas * ( num_entradas + 1) ) , num_ocultas , ( num_entradas + 1) ) ;

  theta2 = reshape (theta ((1 + (num_ocultas * ( num_entradas + 1) ) ) : end ) , num_etiquetas , (
num_ocultas+ 1) ) ;



  % Calculo porcentaje aciertos

  m = rows(Xval);

  v = zeros(m,1);

  h = h(theta1,theta2,Xval);

  for i=1:m

    [maxV ind] = max(h(:,i));

     v(i) = ind;

  endfor



  porc = (sum( v == yval )/m)*100



endfunction
```

regresionLogisticaBinaria.m

```
addpath(pwd);


function res = porcCorrectos(theta, X, y)


  m = rows(X);

  v = h(theta, X);

  v = v >= 0.5;

  res = (sum( v == y )/m)*100;


endfunction




function [theta] = regresionLogistica(X,y, lambda)


  n = columns(X);

  initial_theta = zeros(n,1);

  options = optimset('GradObj', 'on', 'MaxIter', 500);


  [theta] = fmincg(@(t)(lrCostFunction(t,X,y,lambda)), initial_theta, options);


endfunction


function trainBinary(filename, lambda)


  load(filename);

  randidx = randperm(rows(x));

  x = x(randidx,:);

  n = columns(x);

  X = x(:,1:(n-1));

  m = rows(X);

  X = [ ones(m,1) X];

  y = x(:,n);


  num_train = floor(m*0.7);
```

```
  Xval = X((num_train+1):end,:);

  yval = y((num_train+1):end);

  X = X(1:num_train,:);

  y = y(1:num_train);


  theta = regresionLogistica(X,y,lambda);


  porcCorrectos(theta,Xval,yval)


endfunction
```

regresionLogisticaMulticlase.m

```
addpath(pwd);


function res = porcCorrectos2(all_theta, X, y)



  etiquetas = rows(all_theta);
  total = length(y);
  res = zeros(etiquetas, total);
  for i=1:etiquetas
    res(i,:) = h((all_theta(i,:))',X);
  endfor


  v = zeros(total, 1);
  for i=1:total
    [maxV ind] = max(res(:,i));
    v(i) = ind;
  endfor



  res = (sum( y == v)/total)*100;


endfunction



function [all_theta] = oneVsAll(X,y, num_etiquetas, lambda)


n = columns(X);
initial_theta = zeros(n,1);
all_theta = zeros(num_etiquetas,n);
options = optimset('GradObj', 'on', 'MaxIter', 50);


for c=1:num_etiquetas


  [theta] = fmincg(@(t)(lrCostFunction(t,X,(y==c),lambda)), initial_theta, options);
  all_theta(c,:) = theta';
```

```
endfor

endfunction




function trainMulticlass(filename, lambda, num_etiquetas)

  load(filename);
  randidx = randperm(rows(x));
  x = x(randidx,:);
  n = columns(x);
  X = x(:,1:(n-1));
  m = rows(X);
  X = [ ones(m,1) X];
  y = x(:,n);

  num_train = floor(m*0.7);

  Xval = X((num_train+1):end,:);
  yval = y((num_train+1):end);
  X = X(1:num_train,:);
  y = y(1:num_train);
  all_theta = oneVsAll(X,y,num_etiquetas,lambda);
  porc = porcCorrectos2(all_theta, Xval, yval)

endfunction
```

sigmoide.m

```
function res = sigmoide(z)


  res = 1 ./ (1 .+ exp(-z));
```

svm.m

```
addpath(pwd);


function sim = gaussianKernel(x1,x2,sigma)

  sim = exp( -(sum( (x1-x2).^2 ) / (2*(sigma^2))) );


endfunction



function res = porCorrect(vSal,y)

  a = sum( vSal == y );

  b = length(y);

  res = (a/b)*100;

endfunction



function chooseParam(X,y,Xval,yval)

  params = [0.01;0.03;0.1;0.3;1;3;10;30];

  l = length(params);

  bestPorc = 0;


  for i= 1:l

    C = params(i);

    for j = 1:l

       sigma = params(j);

       model = svmTrain(X,y,C, @(x1,x2) gaussianKernel(x1,x2,sigma));

       vSal = svmPredict(model,Xval);

       porc = porCorrect(vSal,yval);

       if ( porc >= bestPorc )
```

```
        bestModel = model;
        bestPorc = porc;
        bestC = C;
        bestSigma = sigma;
       endif
    endfor

  endfor

  bestPorc
  bestC
  bestSigma

endfunction

function SVMProy(filename)

load(filename);
n = columns(x);
randidx = randperm(rows(x));
x = x(randidx,:);
X = x(:,1:(n-1));
y = x(:,n);

m = rows(X);

num_train = floor(m*0.7);

Xval = X((num_train+1):end,:);
yval = y((num_train+1):end);
X = X(1:num_train,:);
y = y(1:num_train);

chooseParam(X,y,Xval,yval)

endfunction
```

6. **Referencias**
    a. **Transparencias y material de la asignatura**
    b. **Open University Learning Analytics Dataset**
    c. **Open University Analyse (Project)**
    d. **Publicaciones del equipo de OU Analyse**
        i. Kuzilek, J., Hlosta, M., Zdrahal, Z. Open University Learning Analytics Dataset. Data Literacy For Learning Analytics Workshop at Learning Analytics and Knowledge (2016)
        ii. Kuzilek, J., Hlosta, M., Herrmannova, D., Zdrahal, Z. and Wolff, A. OU Analyse: Analysing At-Risk Students at The Open University. Learning Analytics Review, no. LAK15-1 (2015)
        iii. Herrmannova, D., Hlosta, M., Kuzilek, J., Zdrahal, Z. Evaluating weekly predictions of at-risk students at the Open University: results and issues. EDEN 2015