PROYECTO FINAL ASIGNATURA

Aprendizaje automático y Big data

24 DE ENERO DE 2019

VÍCTOR RAMOS Y FELIX VILLAR
Universidad Complutense de Madrid

Índice

Presentación de los datos

Los datos empleados en este proyecto son:

Abstract: Predict student performance in secondary education (high school).

Data Set Characteristics:	Multivariate	**Number of Instances:**	649	**Area:**	Social
Attribute Characteristics:	Integer	**Number of Attributes:**	33	**Date Donated**	2014-11-27
Associated Tasks:	Classification, Regression	**Missing Values?**	N/A	**Number of Web Hits:**	402219

Han sido extraídos de la siguiente URL:

http://archive.ics.uci.edu/ml/datasets/Student+Performance

Este conjunto de datos contiene la información de los estudiantes de 2 colegios portugueses.

La finalidad es conocer la nota final del alumno.

- 382 están matriculados en ambas asignaturas.
- 395 matriculados en matemáticas.
- 649 matriculados en portugués.

A continuación se muestra un listado con el significado de cada atributo en orden de aparición en el fichero "student.txt".

Además se incluye una tabla con el atributo y el intervalo de su posible valor.

Cada uno de los alumnos contiene 33 atributos:

1. Colegio: Sigla del colegio al que acuden los niños.
2. Sexo del alumno
3. Edad del alumno
4. Dirección de su casa
5. Tamaño familiar: si son más de 3 integrantes en la familia.
6. Situación de los padres: viven juntos o separados.
7. Nivel educativo de la madre
8. Nivel educativo del padre
9. Empleo de la madre
10. Empleo del padre
11. Razón para escoger el colegio
12. Tutor legal
13. Tiempo empleado en ir al colegio
14. Tiempo destinado al estudio por semana
15. Número de asignaturas suspensas
16. Apoyo extra escolar
17. Apoyo extra familiar
18. Clases de refuerzo ó academias
19. Actividades extraescolares
20. Servicio de enfermería en el colegio
21. Deseo del alumno por ir acceder a estudios superiores
22. Acceso a Internet desde casa
23. Relación sentimental
24. Relación entre los integrantes de la familia
25. Tiempo libre fuera del horario lectivo
26. Tiempo con los amigos fuera del horario escolar
27. Consumo de alcohol en días lectivos
28. Consumo de alcohol los fines de semana
29. Estado de salud
30. Ausencias en clase
31. Nota del primer período
32. Nota del segundo período
33. Nota final del curso

Atributo	Valor
Colegio	[0,1]
Sexo	[0,1]
Edad	#Valor
Dirección	[0,1]
Tamaño Familiar	[0,1]
Nivel educativo de la madre	[0,4]
Nivel educativo del padre	[0,4]
Empleo de la madre	[0,1]
Empleo del padre	[0,1]
Motivos elección colegio	[0,3]
Tutor	[0,2]
Tiempo casa-colegio	[0,4]
Tiempo estudio	[0,4]
Número suspensos	[1,4]
Apoyo extraescolar	[0,1]
Apoyo familiar	[0,1]
Clases de refuerzo	[0,1]
Actividades extraescolares	[0,1]
Servicio enfermería	[0,1]
Expectativas futuro	[0,1]
Acceso a Internet	[0,1]
Relación sentimental	[0,1]
Relación familiar	[0,1]
Tiempo libre	[0,5]
Tiempo libre con amigos	[0,5]
Consumo alcohol en día laboral	[0,5]
Consumo de alcohol en fin de semana	[0,5]
Estado de salud	[0,5]
Ausencias	[0,93]
Nota primer período	[0,20]
Nota segundo período	[0,20]
Nota final	[0,20]

Técnicas de aprendizaje utilizadas

En este proyecto se han utilizado 3 tipos de técnicas que han sido estudiadas durante el curso de esta asignatura:

- ❖ Regresión logística multiclase
- ❖ Redes Neuronales
- ❖ SVM

Durante el empleo de estas técnicas, se van a dividir los datos en tres grupos:

- ✓ Datos de entrenamiento
  - Utilizados en el entrenamiento de los sistemas
- ✓ Datos de validación
  - De cara a la evaluación de distintos valores de los parámetros
- ✓ Datos de prueba
  - Evaluación del porcentaje de éxito de cada técnica.

Tendremos un 70% de datos de entrenamiento, de los datos de validación y prueba, un 15% respectivamente.

Como aclaración, la información del Dataset está clasificada por colegio, con lo cual los datos se componen de la extracción de cada parte pertinente.

Regresión logística multiclase

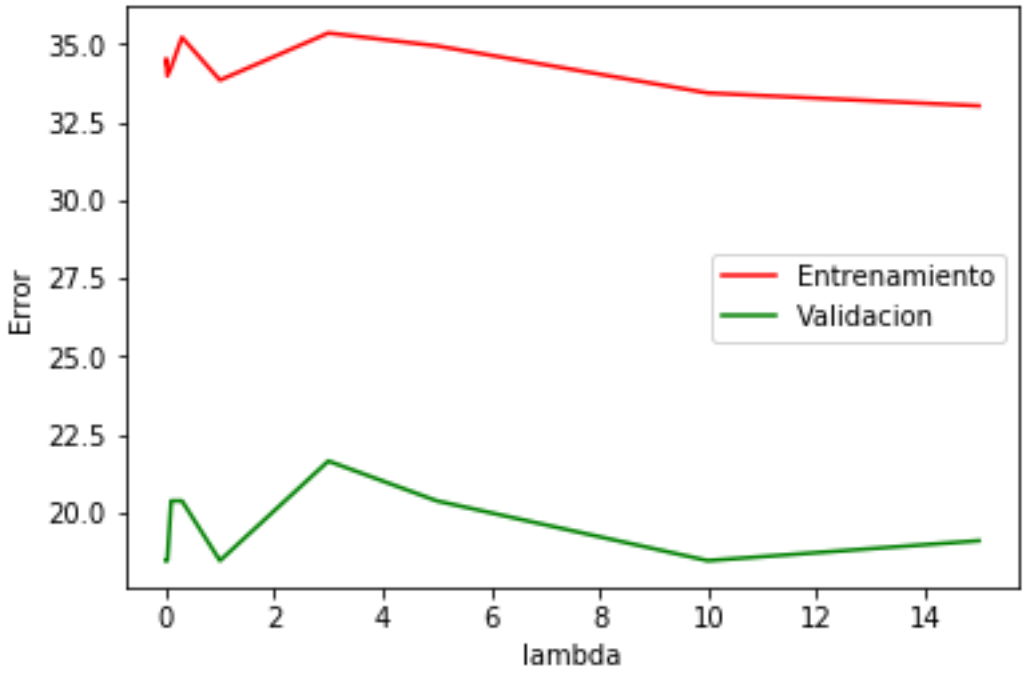
Sabiendo cómo están distribuidos los datos, procedemos primero a encontrar el mejor valor posible para lambda.

Utilizamos para ello la función "**errorlambda**" junto a los valores de entrenamiento y validación.

Lambda tendrá distintos valores, en este caso:

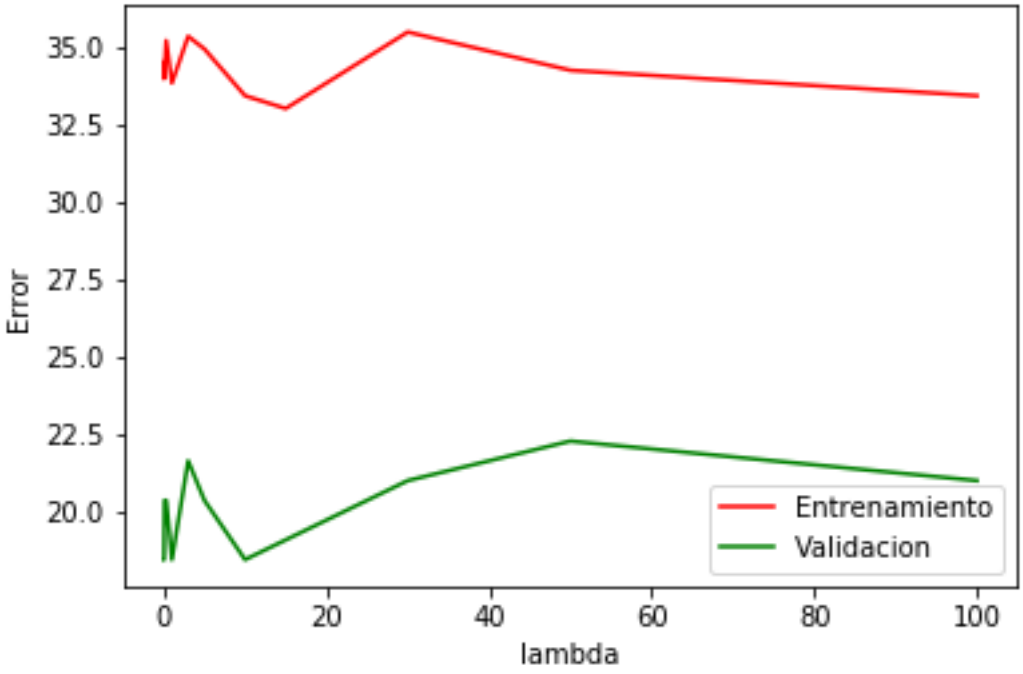
[0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 3, 5, 10, 15]

El resultado es el siguiente:



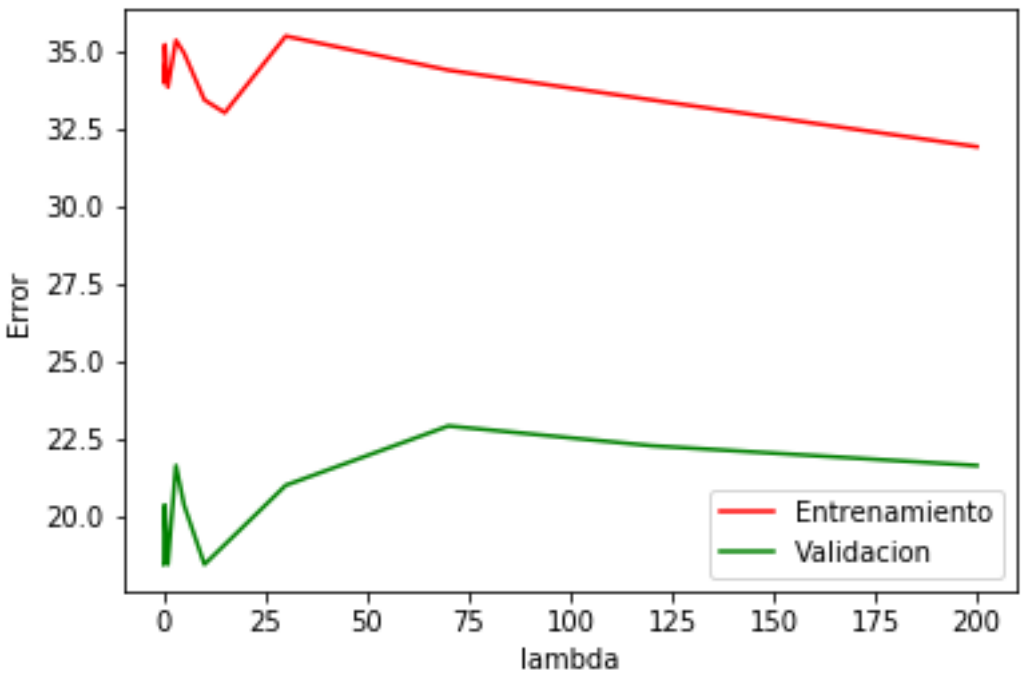
El valor que utilizaremos para lambda será de 3.

Se ha repetido el paso anterior otras dos veces más obteniendo los siguientes resultados:



Lambda tendrá distintos valores, en este caso:

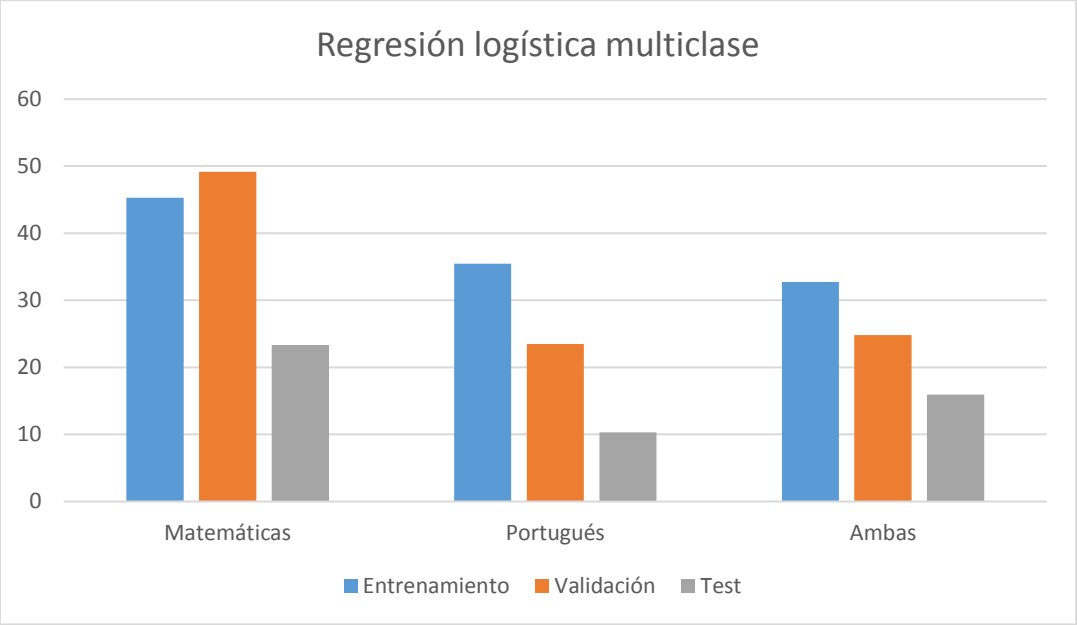
[0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, 30, 50, 70, 100]



Lambda tendrá distintos valores, en este caso:

[0.001,0.003,0.01,0.03,0.1,0.3,1,3,5,10,15,30,70,120,200]

Una vez introducido el valor de lamba, procedemos a calcular los porcentajes de entrenamiento, validación y test por cada asignatura y por el conjunto de ambas.



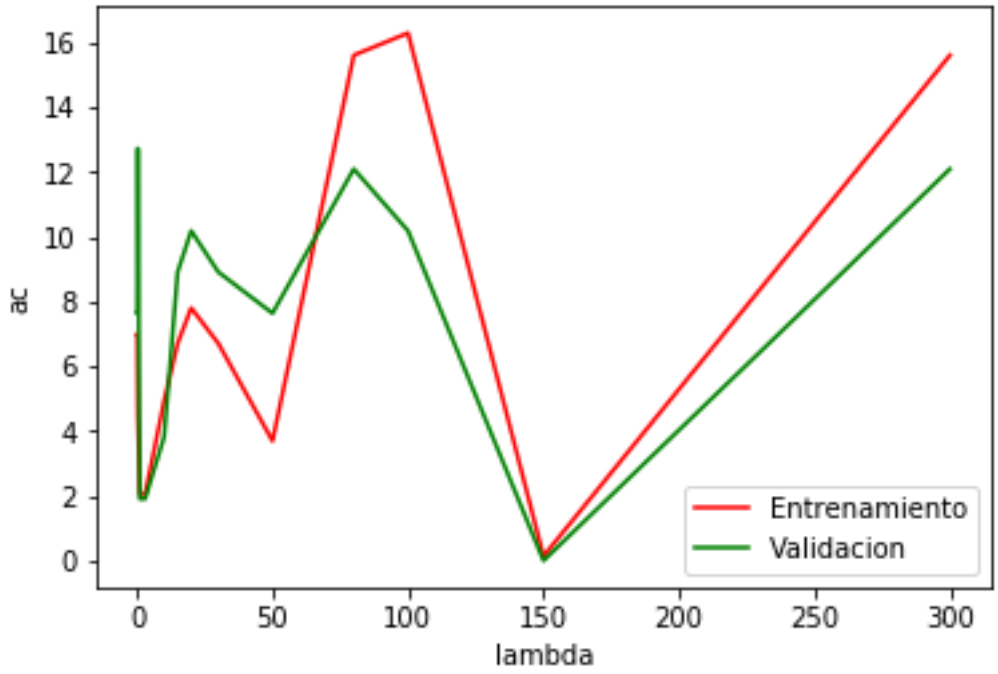
Redes Neuronales

Al igual que en la regresión logística multiclase, se empieza escogiendo el valor más adecuado para lambda.

La función escogida es "**errorlmdb**", los diferentes valores que puede adoptar lambda son:

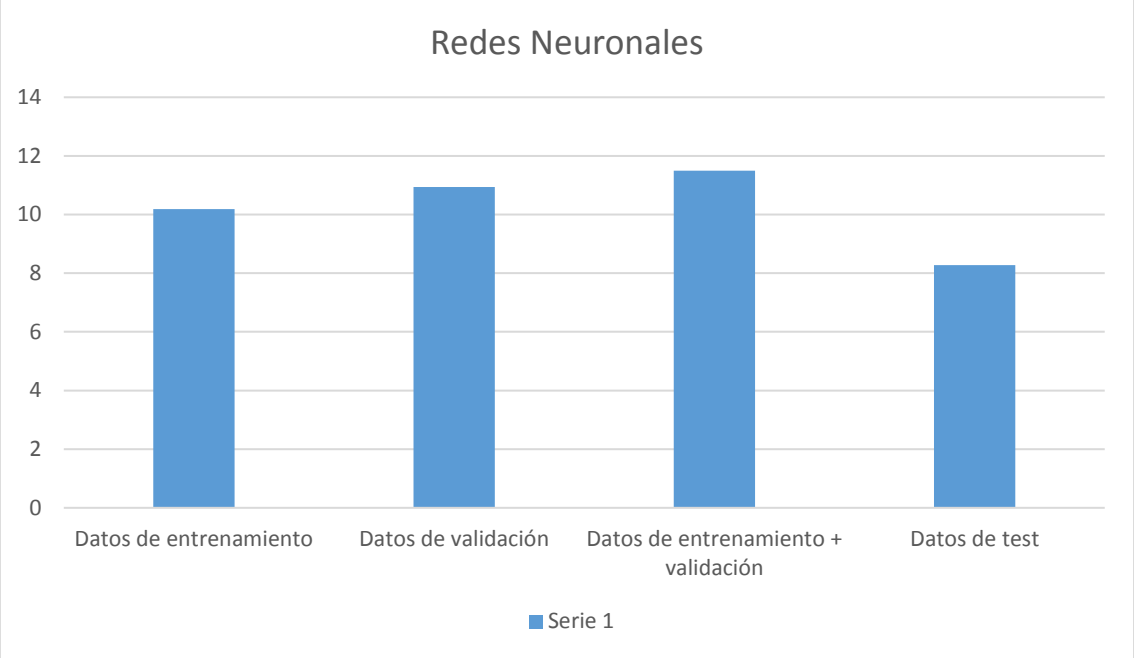
0.001,0.003,0.01,0.03,0.1,0.3,1,3,10,15,20,30,50,80,100,150,300

La gráfica resultante es:



El mejor valor para lambda es variable, la minimización a veces llega a un punto y a veces a otro, lo que conlleva que los pesos iniciales sean aleatorios.

A continuación se muestra una gráfica con los porcentajes de acierto de la red neuronal con los distintos tipos de datos:


Redes Neuronales
14
12
10
8
6
4
2
0
Datos de entrenamiento
Datos de validación
Datos de entrenamiento + validación
Datos de test
Serie 1

Support Vector Machine

Comenzamos eligiendo el mejor Kernel gaussiano para entrenar las SVM con distintos valores:

[0.01,0.03,0.05,0.1,0.3,0.5,1,3,5,10,15,30,50,100,150,300]

Así obtendremos la mejor solución lineal y gaussiana de distintas opciones:

- Solo matemáticas

En este apartado obtenemos: como mejor solución lineal un valor de C = 0.05, con un porcentaje de acierto del 51%.

Como mejor solución Gaussiana tenemos a C = 50, Gamma = 30 y un 53.33% de tasa de acierto.

- Solo matemáticas sin G1 y G2

Aquí obtenemos: como mejor solución lineal un valor de C = 0.05, con un porcentaje de acierto del 26,6%

Por otra parte, como mejor solución Gaussiana tenemos a C = 15, Gamma = 30 y un 28.3% de tasa de acierto

- Solo portugués

En este apartado obtenemos: como mejor solución lineal un valor de C = 0.1, con un porcentaje de acierto del 38,144%

La mejor solución Gaussiana obtenida es C = 300, Gamma = 50 y un 38% de tasa de acierto.

- Solo portugués sin G1 y G2

Aquí obtenemos: como mejor solución lineal un valor de C = 0.01, con un porcentaje de acierto del 18.56%.

Y, como mejor solución Gaussiana tenemos a C = 300, Gamma = 50 y un 20,62% de tasa de acierto.

- Todos los datos

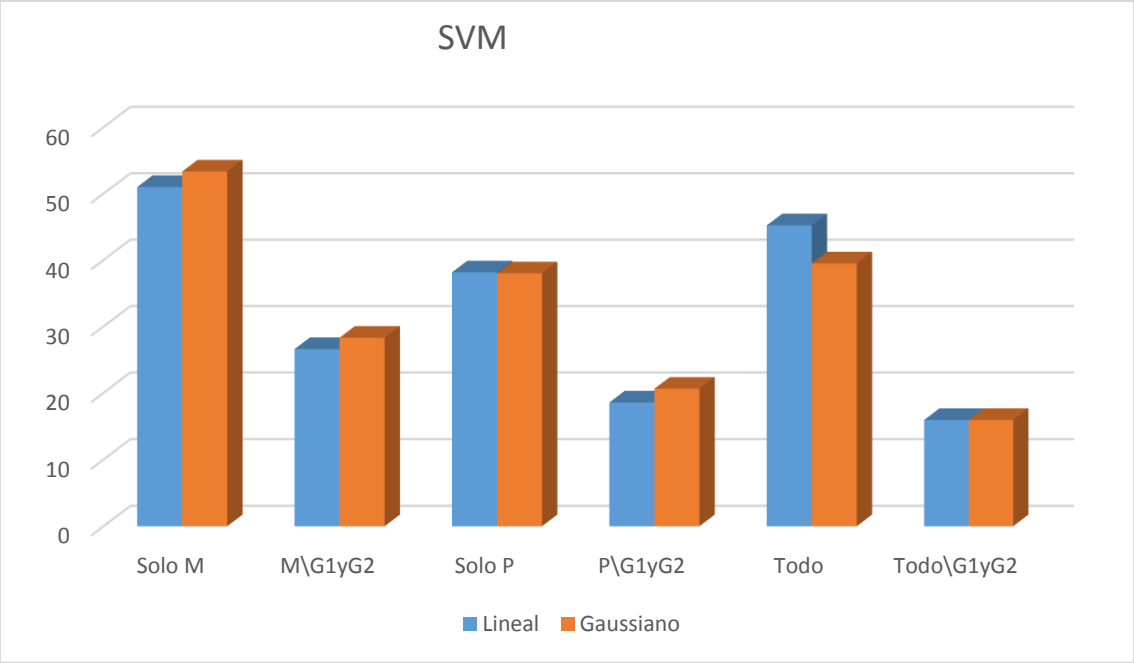
Obtenemos: como mejor solución lineal un valor de C = 10, con un porcentaje de acierto del 45,22%.

De mejor solución Gaussiana tenemos a C = 300, Gamma = 50 y un 39,49 % de tasa de acierto.

- Todos los datos sin G1 y G2

En este apartado obtenemos: como mejor solución lineal un valor de C = 0.03, con un porcentaje de acierto del 15,92 %

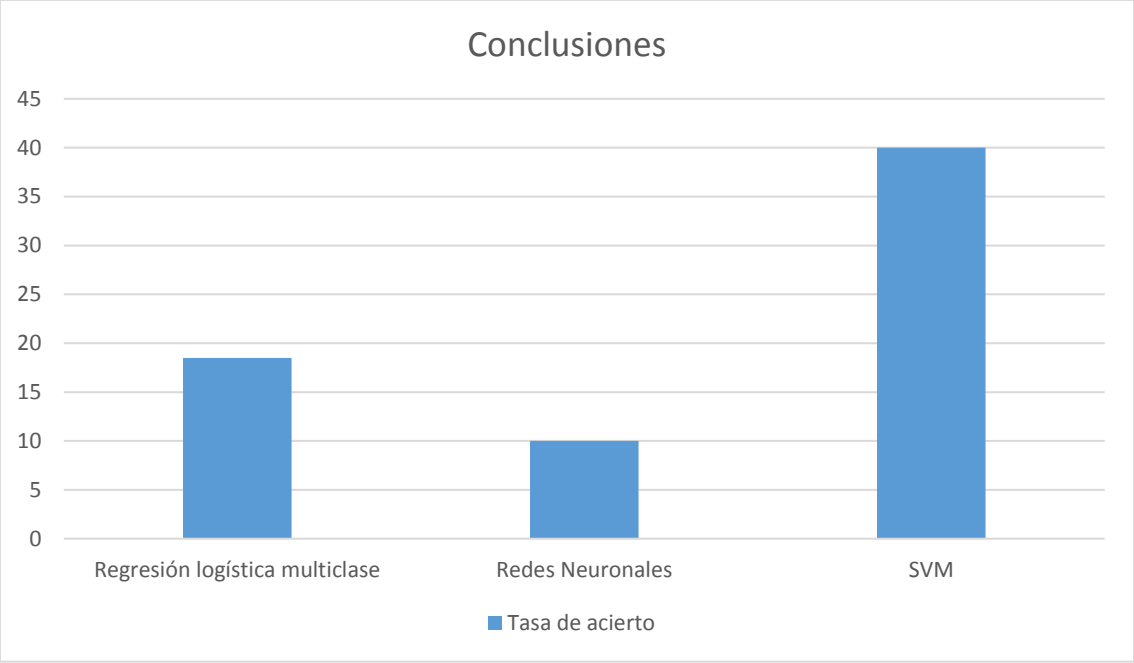
La mejor solución Gaussiana obtenida es C = 100, Gamma = 5 y un 15,92% de tasa de acierto.


SVM
60
50
40
30
20
10
0
Solo M
M\G1yG2
Solo P
P\G1yG2
Todo
Todo\G1yG2
Lineal
Gaussiano

Comparación de resultados obtenidos

Como conclusión, se puede corroborar que los resultados han sido dispares, si bien la mejor tasa de acierto le ha correspondido a la Support Vector Machine (SVM).

También se debe destacar que el porcentaje de la SVM se incrementa al incluir en los datos G1 y G2, es decir, las notas finales de cada parte del curso.



Anexo

Implementación de Regresión logística multiclase

Código empleado

```
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt

def sigmoide(x):
    return 1/(1+np.exp(-x))
def h(x,th):
    return sigmoide(np.matmul(x,th))

def costeGrad(th,X,y,lambd):
    n = X.shape[0]
    grad = (1/n)*(np.matmul((h(X,th)-y[:,0]).T,X))
    grad =grad.T
    reg =(lambd/n)*th
    reg[0]= 0
    grad = grad+reg
    a = ((-y*np.log(h(X,th))).T)
    b = (np.matmul((1-y).T,np.log(1-h(X,th))).T)
    coste= (1/n)*np.sum(a-b)
    reg = (lambd/(2*n))* np.sum(th**2)
    coste = coste+reg
    return coste,grad

def oneVsAll(X,y,num_etiquetas,lambd):
    Xaux = np.hstack((np.ones((X.shape[0],1)),X))
```

```
    entrenador = np.zeros((num_etiquetas,X.shape[1]+1))
    for i in range(0,num_etiquetas):
        entrenador[i]= opt.fmin_tnc(costeGrad,entrenador[i],args=(Xaux,(y==i)*1,lambd))[0]
        #entrenador[i] = opt.minimize(coste,entrenador[i],args=(X,(y==i)*1,reg), jac=gradiente).x
    return entrenador
def porcentaje(th,X,y):
    res = h(X,th.T)
    maximo = np.argmax(res, axis = 1)
    comp = (maximo==y[:,0])*1
    g= np.count_nonzero(comp)
    return (g/len(comp))*100
def errorlambda(X,y,Xval,yval):
    lmdb= np.array([0.001,0.003,0.01,0.03,0.1,0.3,1,3,5,10,15])
    n = X.shape[0]
    num_etiquetas = 21
    m=len(lmdb)
    porcent= np.zeros(m)
    porcval = np.zeros(m)
    for i in range(0,m):
        th=oneVsAll(X,y,num_etiquetas,lmdb[i])
        Xaux = np.hstack((np.ones((n,1)),X))
        porcent[i] = porcentaje(th,Xaux,y)
        Xaux = np.hstack((np.ones((Xval.shape[0],1)),Xval))
        porcval[i] = porcentaje(th,Xaux,yval)
    plt.xlabel('lambda')
    plt.ylabel('Error')
    plt.plot(lmdb,porcent,label="Entrenamiento", c='r')
    plt.plot(lmdb,porcval,label="Validacion", c= 'g')
    plt.legend()
```

```
datamat = np.genfromtxt('student-mat-Modificado.csv', delimiter = ';')
datapor = np.genfromtxt('student-por-Modificado.csv', delimiter = ';')

#datos ordenados
entmat = np.vstack((datamat[:244],datamat[349:381]))
entpor = np.vstack((datapor[:296],datapor[423:581]))
ent = np.vstack((entmat,entpor))
valmat = np.vstack((datamat[244:297],datamat[381:387]))
valpor = np.vstack((datapor[296:360],datapor[581:615]))
val = np.vstack((valmat,valpor))
testmat = np.vstack((datamat[297:349],datamat[387:]))
testpor = np.vstack((datapor[360:423],datapor[615:]))
test = np.vstack((testmat,testpor))

Xmat = entmat[:,:32]
Xpor = entpor[:,:32]
X = ent[:,:32]
ymat = entmat[:,32:]
ypor = entpor[:,32:]
y = ent[:,32:]

Xvalmat = valmat[:,:32]
Xvalpor = valpor[:,:32]
Xval = val[:,:32]
yvalmat = valmat[:,32:]
yvalpor = valpor[:,32:]
yval = val[:,32:]

Xtestmat = testmat[:,:32]
Xtestpor = testpor[:,:32]
```

```
Xtest = test[:,:32]
ytestmat = testmat[:,32:]
ytestpor = testpor[:,32:]
ytest = test[:,32:]
errorlambda(X,y,Xval,yval)

Xent_val = np.concatenate((X,Xval))
yent_val = np.concatenate((y,yval))
print("Todo")
th = oneVsAll(Xent_val,yent_val,21,3)
Xp = np.concatenate((np.atleast_2d(np.ones(X.shape[0])).T,X),axis=1)
p=porcentaje(th,Xp,y)
print("Porcentaje entrenamiento")
print(p)

Xpval = np.concatenate((np.atleast_2d(np.ones(Xval.shape[0])).T,Xval),axis=1)
pval=porcentaje(th,Xpval,yval)
print("Porcentaje validacion")
print(pval)

Xptest = np.concatenate((np.atleast_2d(np.ones(Xtest.shape[0])).T,Xtest),axis=1)
ptest=porcentaje(th,Xptest,ytest)
print("Porcentaje test")
print(ptest)

Xent_valmat = np.concatenate((Xmat,Xvalmat))
yent_valmat = np.concatenate((ymat,yvalmat))
print("Matematicas")
th = oneVsAll(Xent_valmat,yent_valmat,21,3)
Xpmat = np.concatenate((np.atleast_2d(np.ones(Xmat.shape[0])).T,Xmat),axis=1)
```

```
p=porcentaje(th,Xpmat,ymat)
print("Porcentaje entrenamiento")
print(p)

Xpvalmat = np.concatenate((np.atleast_2d(np.ones(Xvalmat.shape[0])).T,Xvalmat),axis=1)
pval=porcentaje(th,Xpvalmat,yvalmat)
print("Porcentaje validacion")
print(pval)

Xptestmat = np.concatenate((np.atleast_2d(np.ones(Xtestmat.shape[0])).T,Xtestmat),axis=1)
ptest=porcentaje(th,Xptestmat,ytestmat)
print("Porcentaje test")
print(ptest)

Xent_valpor = np.concatenate((Xpor,Xvalpor))
yent_valpor = np.concatenate((ypor,yvalpor))

print("Portugues")
th = oneVsAll(Xent_valpor,yent_valpor,21,3)
Xppor = np.concatenate((np.atleast_2d(np.ones(Xpor.shape[0])).T,Xpor),axis=1)
p=porcentaje(th,Xppor,ypor)
print("Porcentaje entrenamiento")
print(p)

Xpvalpor = np.concatenate((np.atleast_2d(np.ones(Xvalpor.shape[0])).T,Xvalpor),axis=1)
pval=porcentaje(th,Xpvalpor,yvalpor)
print("Porcentaje validacion")
print(pval)
```

```
Xptestpor = np.concatenate((np.atleast_2d(np.ones(Xtestpor.shape[0])).T,Xtestpor),axis=1)

ptest=porcentaje(th,Xptestpor,ytestpor)

print("Porcentaje test")

print(ptest)
```

Implementación de Redes Neuronales

Código empleado

```
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt



def sigmoide(x):
    return 1/(1+ np.exp(np.negative(x)))

def sigmoideDerivada(z):
    sd = sigmoide(z) * (1 - sigmoide(z));
    return sd

def porcentajeRedNeuronal(Theta1, Theta2, X, y):
   m = X.shape[0]
   a1=np.hstack((np.ones((m,1)),X))
   z2=np.matmul(Theta1,np.transpose(a1))
   a2=sigmoide(z2)
   a2=np.vstack((np.ones((1,a2.shape[1])),a2))
   z3=np.matmul(Theta2,a2)
   a3=sigmoide(z3)
   h=a3
   maximo=np.argmax(h,axis=0)
   comparacion=(maximo == y[:,0])*1

   bienPredecidos = np.count_nonzero(comparacion)

   porcentaje = (bienPredecidos/m)*100
```

```
    return porcentaje


def pesosAleatorios(L_in,L_out):
    ini =0.12
    pesos = np.random.rand((L_in+1)*L_out)*(2*ini) - ini
    pesos = np.reshape(pesos, (L_out,1+L_in))
    return pesos



def errorlmdb(X,y,Xval,yval,Theta1_ini,Theta2_ini):
    lmdb= np.array([0,0.001,0.003,0.01,0.03,0.1,0.3,1,3,10,15,20,30,50,80,100,150,300])
    num_etiquetas=21
    num_entradas=32
    num_ocultas=10 #Probar con distintos valores

    aux = np.reshape(Theta1_ini,(num_entradas+1)*num_ocultas)
    aux2 = np.reshape(Theta2_ini,(num_ocultas+1)*num_etiquetas)
    params_ini=np.concatenate((aux,aux2))

    porcentajeEnt = np.zeros(len(lmdb))
    porcentajeVal = np.zeros(len(lmdb))
    for i in range(0,len(lmdb)): #¿i == 1?
        res = opt.minimize(backprop,params_ini,args=(num_entradas,num_ocultas,num_etiquetas,X,y,lmdb[i]),jac=True)
        grad = res.jac

        Theta1 = np.reshape(grad[:num_ocultas*(num_entradas+1)],(num_ocultas,(num_entradas+1)))
        Theta2 = np.reshape(grad[num_ocultas*(num_entradas+1):],(num_etiquetas,(num_ocultas+1)))
```

```
        porcentajeEnt[i] = porcentajeRedNeuronal(Theta1, Theta2, X, y)
        porcentajeVal[i] = porcentajeRedNeuronal(Theta1, Theta2, Xval, yval)


    plt.xlabel('lambda')
    plt.ylabel('ac')
    plt.plot(lmdb,porcentajeEnt,label="Entrenamiento", c='r')
    plt.plot(lmdb,porcentajeVal,label="Validacion", c= 'g')
    plt.legend()



def debugInitializeWeights(fan_in, fan_out):
    """
    Initializes the weights of a layer with fan_in incoming connections and
    fan_out outgoing connections using a fixed set of values.
    """

    # Set W to zero matrix
    W = np.zeros((fan_out, fan_in + 1))

    # Initialize W using "sin". This ensures that W is always of the same
    # values and will be useful in debugging.
    W = np.array([np.sin(w) for w in
                range(np.size(W))]).reshape((np.size(W, 0), np.size(W, 1)))

    return W

def backprop(params_rn, num_entradas, num_ocultas, num_etiquetas, X, y, reg):
    Theta1 = np.reshape(params_rn[:num_ocultas*(num_entradas+1)],(num_ocultas,
(num_entradas+1)))
```

```
    Theta2 = np.reshape(params_rn[num_ocultas*(num_entradas+1):],(num_etiquetas, (num_ocultas+1)))
    m = X.shape[0]

    #Propagacion hacia delante
    a1 = np.vstack((np.ones(X.shape[0]),X.T))
    z2=np.matmul(Theta1,a1)
    a2=sigmoide(z2)
    a2 = np.vstack((np.ones(a2.shape[1]),a2))
    z3=np.matmul(Theta2,a2)
    a3=sigmoide(z3)
    h = a3

    etiqueta = np.identity(num_etiquetas)
    ycod = etiqueta[y[:,0].astype(int),:]
    J = np.sum(np.matmul((-ycod),np.log(h) )- np.matmul((1 - ycod),np.log(1 - h)))/m

    #Regularizacion
    regular = (reg/(2*m))*(np.sum(np.square(Theta1[:,1:]))+np.sum(np.square(Theta2[:,1:])))
    final = J+regular

    #Retro propagacion
    d3 = h.T - ycod
    d2 = np.matmul(Theta2.T,d3.T)[1:,:] *sigmoideDerivada(z2)
    grad1 = np.matmul(d2,a1.T)/m
    grad2 = np.matmul(d3.T,a2.T)/m

    #Regularizacion del gradiente
    reg1= (reg/m) * Theta1[:,1:]
    reg2= (reg/m) * Theta2[:,1:]
```

```
    #Regularizacion del gradiente
    fingrad1 = grad1
    fingrad1[:,1:] += reg1
    fingrad2 = grad2
    fingrad2[:,1:] += reg2

    #Fin del gradiente
    aux = np.reshape(fingrad1,fingrad1.shape[0]*fingrad1.shape[1])
    aux2 = np.reshape(fingrad2, fingrad2.shape[0]*fingrad2.shape[1])
    grad =np.concatenate((aux,aux2))
    return final,grad


def main():
    #DATOS INICIALES
    num_etiquetas=21
    num_entradas=32
    num_ocultas=10 #Probar con distintos valores

    datamat = np.genfromtxt('student-mat-Modificado.csv', delimiter = ';')
    datapor = np.genfromtxt('student-por-Modificado.csv', delimiter = ';')

    #Datos de entrada
    entmat = np.vstack((datamat[:244],datamat[349:381]))
    entpor = np.vstack((datapor[:296],datapor[423:581]))
    ent = np.vstack((entmat,entpor))

    #Datos de validacion
    valmat = np.vstack((datamat[244:297],datamat[381:387]))
```

```
valpor = np.vstack((datapor[296:360],datapor[581:615]))
val = np.vstack((valmat,valpor))

#Datos test
testmat = np.vstack((datamat[297:349],datamat[387:]))
testpor = np.vstack((datapor[360:423],datapor[615:]))
test = np.vstack((testmat,testpor))

#Datos de todos
Xent = ent[:,:32]
Yent = ent[:,32:]
Xval = val[:,:32]
Yval = val[:,32:]
Xtest = test[:,:32]
ytest = test[:,32:]
#Inicializacion de pesos aleatorios
Theta1_ini = pesosAleatorios(num_entradas,num_ocultas)
Theta2_ini = pesosAleatorios(num_ocultas,num_etiquetas)

#Calculamos el error de lambda y cogemos el mejor
errorlmdb(Xent,Yent,Xval,Yval,Theta1_ini,Theta2_ini)

# params_ini=np.concatenate((Theta1_ini,Theta2_ini))
#options = np.optimset('MaxIter', 5000);

#Unimos los datos de entrenamiento y validacion
Xent_val=np.concatenate((Xent,Xval))
yent_val=np.concatenate((Yent,Yval))

#Entrenamiento de la red neuronal
```

```
    #valorlambda = 1 #El que resulte de la grafica de errorlmbd


    aux = np.reshape(Theta1_ini,(num_entradas+1)*num_ocultas)

    aux2 = np.reshape(Theta2_ini,(num_ocultas+1)*num_etiquetas)

    params_ini=np.concatenate((aux,aux2))

    res = opt.minimize(backprop,params_ini,args=(num_entradas,num_ocultas,num_etiquetas,Xent_val,yent_val,50),jac=True)

    grad = res.jac


    Theta1 = np.reshape(grad[:num_ocultas*(num_entradas+1)],(num_ocultas, (num_entradas+1)))

    Theta2 = np.reshape(grad[num_ocultas*(num_entradas+1):],(num_etiquetas, (num_ocultas+1)))

    #Theta11 = np.reshape(params_rn(1:num_ocultas * (num_entradas + 1)), num_ocultas, (num_entradas + 1));

    #Theta21 = np.reshape(params_rn((1 + (num_ocultas * (num_entradas + 1))):end), num_etiquetas, (num_ocultas + 1));



    #Porcentaje

    num_por = porcentajeRedNeuronal(Theta1, Theta2, Xtest, ytest)

    print(num_por)



main()
```

Implementación de Support Vector Machine

Código empleado

```
import numpy as np
from sklearn.svm import SVC
import matplotlib.pyplot as plt


def supportv(Xent,yent,Xval,yval):
    val = np.array([0.01,0.03,0.05,0.1,0.3,0.5,1,3,5,10,15,30,50,100,150,300])
    maxilin = 0
    Csollin = 0
    for i in range(0,val.shape[0]):
        svm = SVC( kernel='linear', C=val[i])
        svm.fit(Xent,yent)
        w = svm.predict(Xval)
        t = (w==yval[:,0])
        p = (np.count_nonzero(t)/yval.shape[0])*100
        #text = 'C='+repr(val[i])+'.Porcentaje='+repr(p)
        if(p>maxilin):
            Csollin = val[i]
            maxilin = p
        #print(text)
    textlin = 'Mejor solucion lineal: C = '+ repr(Csollin)+ ' . % = ' +repr(maxilin)
    maxigaus = 0
    Csolgaus = 0
    sigmasolgaus= 0
    for i in range(0,val.shape[0]):
        for j in range(0,val.shape[0]):
            svm = SVC( kernel='rbf', C=val[i], gamma = 1/(2*val[j]**2))
            svm.fit(Xent,yent)
            w = svm.predict(Xval)
```

```
            t = (w==yval[:,0])
            p = (np.count_nonzero(t)/yval.shape[0])*100
            #text = 'C='+repr(val[i])+',sigma='+repr(val[j])+' .Porcentaje='+repr(p)
            if(p>maxigaus):
              Csolgaus = val[i]
              sigmasolgaus = val[j]
              maxigaus = p
            #print(text)
    text = 'Mejor solucion gaussiana: C = '+ repr(Csolgaus)+', Sigma = '+repr(sigmasolgaus)+ ' . % = ' +repr(maxigaus)
    print(textlin)
    print(text)



datamat = np.genfromtxt('student-mat-Modificado.csv', delimiter = ';')
datapor = np.genfromtxt('student-por-Modificado.csv', delimiter = ';')

entmat = np.vstack((datamat[:244],datamat[349:381]))
entpor = np.vstack((datapor[:296],datapor[423:581]))
ent = np.vstack((entmat,entpor))
valmat = np.vstack((datamat[244:297],datamat[381:387]))
valpor = np.vstack((datapor[296:360],datapor[581:615]))
val = np.vstack((valmat,valpor))
testmat = np.vstack((datamat[297:349],datamat[387:]))
testpor = np.vstack((datapor[360:423],datapor[615:]))
test = np.vstack((testmat,testpor))

print("Solo matematicas")
#lin C= 0.05 51,666%
#Gaus C=50 Sigma = 30 53,333%
X = entmat[:,:32]
```

```
y = entmat[:,32:]
Xval = valmat[:,:32]
yval = valmat[:,32:]
Xtest = testmat[:,:32]
ytest = testmat[:,32:]
Xr = np.vstack((X,Xval))
yr = np.vstack((y, yval))
yr = np.reshape(yr,yr.shape[0])
supportv(Xr,yr,Xtest,ytest)

print("Solo matematicas sin G1 y G2")
#lin C =0.05 26,666%
#Gaus C = 15 Sigma = 30 28,333%
X = entmat[:,:30]
Xval = valmat[:,:30]
Xtest = testmat[:,:30]
Xr = np.vstack((X,Xval))
supportv(Xr,yr,Xtest,ytest)

print("Solo portugues")
#lin C =0.1 39,17%
#Gaus C = 300 Sigma = 50 38,14%
X = entpor[:,:32]
y = entpor[:,32:]
Xval = valpor[:,:32]
yval = valpor[:,32:]
Xtest = testpor[:,:32]
ytest = testpor[:,32:]
Xr = np.vstack((X,Xval))
yr = np.vstack((y, yval))
```

```
yr = np.reshape(yr,yr.shape[0])
supportv(Xr,yr,Xtest,ytest)

print("Solo portugues sin G1 y G2")
#lin C =0.01 18,556%
#Gaus C = 300 Sigma = 50 20,62%
X = entpor[:,:30]
Xval = valpor[:,:30]
Xtest = testpor[:,:30]
Xr = np.vstack((X,Xval))
supportv(Xr,yr,Xtest,ytest)

print("Todo")
#lin C =10 45,22%
#Gaus C = 300 Sigma = 50 39,49%
X = ent[:,:32]
y = ent[:,32:]
Xval = val[:,:32]
yval = val[:,32:]
Xtest = test[:,:32]
ytest = test[:,32:]
Xr = np.vstack((X,Xval))
yr = np.vstack((y, yval))
yr = np.reshape(yr,yr.shape[0])
supportv(Xr,yr,Xtest,ytest)

print("Todo sin G1 y G2")
#lin C =0.03 15,92%
#Gaus C = 100 Sigma = 5 15,92%
X = ent[:,:30]
```

```
Xval = val[:,:30]
Xtest = test[:,:30]
Xr = np.vstack((X,Xval))
supportv(Xr,yr,Xtest,ytest)
```

Bibliografía

- https://es.stackoverflow.com/
- https://relopezbriega.github.io/blog/2015/10/10/machine-learning-con-python/
- http://www.aprendemachinelearning.com/principales-algoritmos-usados-en-machine-learning/
- https://www.kaggle.com/pablorr10/algoritmos-de-machine-learning-con-python-spa
- Diapositivas explicativas facilitadas por el profesor a través del campus virtual
- P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7. [Web Link]