Sistema clasificador de jugadores de FIFA 18

Manuel Antonio Fernández Alonso

Gabriel Sellés Salvà

Grupo 15, AA 2018

1- Descripción del problema

FIFA 18 es uno de los videojuegos más vendidos en España y en el mundo entero. Según el medio digital VGChartz, en enero de 2018, vendió una totalidad de, aproximadamente, 12 millones de copias en todas las plataformas.

Los dos integrantes del grupo somos jugadores acérrimos de este videojuego y, al encontrar un dataset relacionado con él, quisimos usarlo para esta asignatura.

La idea inicial del problema era un sistema que, en función de las estadísticas (stats) de cada jugador, infiriera la posición en la que debería jugar. A pesar de la indudable utilidad y necesidad de este sistema, no fué posible su implementación debido a, principalmente, dos causas:

- La posición en la que juega un jugador, tanto en la vida real como en el videojuego, no depende tanto de sus estadísticas. Las diferencias entre un extremo y un lateral, en las estadísticas de FIFA, son mínimas, y por lo tanto no nos permitía ofrecer veredictos bien fundamentados.
- Las estadísticas globales de los jugadores dependen mucho de la liga y equipo en el que jueguen. Un jugador de primera división española que ocupe la posición de delantero tiene mejores estadísticas como defensa que un defensa de segunda división.

Al implementar esta primera idea, las tasas de acierto, por las causas descritas anteriormente, fueron inferiores al 45%.

Para resolver estos problemas, tratamos de clasificar jugadores de categorías similares, lo que implicaba desechar la gran mayoría de casos de prueba del dataset. Desechamos esta idea debido a los requisitos de la práctica y que los resultados no nos parecían interesantes.

Entonces, escogimos cambiar el problema, pero seguir utilizando el dataset. Actualmente, el sistema clasifica los jugadores en tres clases:

- **Jugadores “Normales”**: Aquellos que la nota global es menor o igual que 65.
- **Jugadores “Buenos”**: Aquellos que la nota global es menor que 85 pero mayor que 65.
- **Jugadores “Estrellas”**: Aquellos que la nota global es mayor o igual que 85.

En este videojuego, la nota global (Overall) se calcula teniendo en cuenta un número considerable de atributos del jugador (alrededor de 45) y no es un cálculo lineal. Dependiendo de su posición, se le da un mayor peso a las intercepciones que a los remates de cabeza, por ejemplo.

En nuestro dataset, tenemos todos estos atributos, pero no disponemos de la posición en la que juega ni de las estadísticas parciales de cada jugador en una posición. Más bien dicho, estos datos los hemos desechado para dotar de sentido al problema.

Entonces, en resumen, nuestro sistema infiere la “calidad” del jugador infiriendo el rango en el que se encuentra su nota global, todo esto en base a sus estadísticas como jugador.

2- Descripción y tratamiento de los datos

El dataset original, del cual se han obtenido los ejemplos de entrenamiento, de test y de validación, contiene los atributos asociados a, aproximadamente, 18.000 jugadores del videojuego FIFA 18.

El número de atributos del dataset es mayor que 70 y, muchos de estos, no son de utilidad para la resolución del problema. Por ejemplo, el nombre del jugador, su edad o su salario, no determinan de ninguna forma la “calidad” o la nota global del jugador.

Estos atributos irrelevantes, entre otros, se han desechado en el tratamiento del dataset porque no ofrecían información relevante.

Otros atributos se han obviado, debido a que simplifican en exceso la resolución del problema. Estos atributos son, por ejemplo, la posición que ocupa el jugador en el terreno de juego o las notas parciales de cada jugador en cada posición.

Los atributos que se han seleccionado para determinar la calidad del jugador suman un total de 30 atributos. Son los siguientes:

1. 'Acceleration'
2. 'Aggression'
3. 'Agility'
4. 'Balance'
5. 'Ball control'
6. 'Composure'
7. 'Crossing'
8. 'Curve'
9. 'Dribbling'
10. 'Finishing'
11. "Free kick accuracy"
12. 'GK diving'
13. 'GK reflexes'
14. 'GK handling’
15. 'GK kicking"
16. ‘GK positioning'
17. 'Heading accuracy'
18. 'Jumping'
19. 'Interceptions'
20. 'Long shots'

21. 'Long passing'
22. 'Marking'
23. 'Short passing'
24. 'Positioning'
25. 'Shot power'
26. 'Sliding tackle'
27. 'Volleys'
28. ''Sprint speed'
29. 'Strength'
30. 'Vision'

Obviamente, también almacena un atributo relevante que es la nota global de cada jugador.

No procederemos a realizar una explicación detallada de cada atributo debido a que los nombres son lo suficientemente descriptivos y a la cantidad de atributos que utiliza el sistema.

Utilizando el dataset original, hemos generado tres ficheros con extensión “csv” que contienen los ejemplos de entrenamiento, los ejemplos para la validación y los datos para el testing.

Para realizar esta tarea, hemos utilizado varias librerías python:

- **pandas**: librería Python destinada al análisis de datos, que proporciona estructuras de datos flexibles que permiten trabajar con ellas de forma muy flexible. Se utiliza para cargar en sus estructuras los datos del .csv original y moldearlos (eliminarlos, modificarlos,etc).
- **csv**: es la extensión de uso más extendido en lo referente a datasets. Hemos utilizado la librería de python que nos permite interactuar con los ficheros con esta extensión.

El código que realiza esta tarea se encuentra en el fichero “datasetCreator.ipynb”.
Los principales fragmentos del código que conviene destacar son los siguientes:

```
#IMPORTS NECESARIOS
import numpy as np
import csv
import pandas as pd
```

```
#Cargamos el df asociado al csv situado en path.
def loadDataset(path):
    names =['Name','Age','Photo','Nationality','Flag','Overall','Potential','Club',
            'Club Logo','Value','Wage','Special','Acceleration','Aggression',
            'Agility','Balance','Ball control','Composure','Crossing','Curve',
            'Dribbling','Finishing','Free kick accuracy','GK diving',
            'GK handling','GK kicking','GK positioning','GK reflexes',
            'Heading accuracy','Interceptions','Jumping','Long passing','Long shots',
            'Marking','Penalties','Positioning','Reactions','Short passing',
            'Shot power','Sliding tackle','Sprint speed','Stamina','Standing tackle'
            ,'Strength','Vision','Volleys','CAM','CB','CDM','CF','CM','ID','LAM','LB'
            ,'LCB','LCM','LDM','LF','LM','LS','LW','LWB','Preferred Positions',
            'RAM','RB','RCB','RCM','RDM','RF','RM','RS','RW','RWB','ST'];
    return pd.read_csv(path, names=names).dropna()    #Devolvemos el dataframe.
```

```
#Elimina atributos innecesarios.
#Algunos atributos, como "Aggression" podrían ser de utilidad pero, pero han sido eliminados para
#evitar tener demasiados.

def deleteFeatures(df):
    sel_features =['Acceleration','Aggression','Agility','Balance','Ball control',
                   'Composure','Crossing','Curve','Dribbling','Finishing',
                   "Free kick accuracy",'GK diving','GK reflexes','GK handling',
                   'GK kicking','GK positioning', 'Heading accuracy','Jumping',
                   'Interceptions','Long shots','Long passing','Marking','Short passing',
                   'Positioning','Shot power', 'Sliding tackle', 'Volleys',
                   'Sprint speed','Strength','Vision','Overall'];
    print(len(sel_features))
    df = df[sel_features]

    return df
```

```
#MAIN

df=loadDataset("CompleteDataset.csv")                #Cargamos el dataset.
df=deleteFeatures(df)                               #Eliminamos atributos inútiles.

#En el dataframe resultante, tenemos varios atributos y, en la última columna, la media
#general (overall). Esto será el objeto de estudio en este proyecto. En la parte en octave,
#separaremos los ejemplos de entrenamiento en tres posibles clases, en función de su media:
#             <=65 --> Normales.
#             >65 && <85 --> Buenos.
#             >=85 -->Estrellas.
#El sistema será capaz de clasificar los distintos jugadores en Normales, Buenos o Estrellas.

display(df)

#Diferenciamos entre ejemplos de entrenamiento, ejemplos de validación y ejemplos de test.
#Los primeros consistirán en el 60% de los ejemplos iniciales. Los segundos y terceros el 20%
#cada uno.
#ES IMPORTANTE SELECCIONAR EJEMPLOS ALEATORIOS DEL DF PORQUE LOS CASOS DE ENTRENAMIENTO
#VIENEN ORDENADOS POR OVERALL!!

#Cogemos 0.6*len(df) elementos aleatorios del df. Serán los ejemplos de entrenamiento.
x = df.sample(int(len(df)*0.6))
#Cogemos 0.2*len(df) elementos aleatorios del df. Serán los datos de validación.
xval= df.sample(int(len(df)*0.2))
#Cogemos 0.2*len(df) elementos aleatorios del df. Serán los datos de test.
xtest= df.sample(int(len(df)*0.2))


x.to_csv(path_or_buf="x.csv",header=False,index=False)    #Almacenamos los resultados.
xval.to_csv(path_or_buf="xval.csv",header=False,index=False)    #Almacenamos los resultados.
xtest.to_csv(path_or_buf="xtest.csv",header=False,index=False)    #Almacenamos los resultados.
```

Los resultados de la ejecución de este código son 3 ficheros de extensión csv que, como hemos dicho anteriormente, tienen extensión “csv”.

Si N es el número total de los ejemplos de entrenamiento del fichero original:

- El primer fichero contiene los ejemplos de entrenamiento, los cuales consisten en una selección aleatoria de N*0.6 elementos del dataset original, es decir, un 60% de los elementos del dataset original seleccionados de forma aleatoria.
- El segundo fichero contiene los ejemplos de validación, los cuales consisten en una selección aleatoria de N*0.2 elementos del dataset original, es decir, un 20% de los elementos del dataset original seleccionados de forma aleatoria.
- El tercer fichero contiene los datos de test, los cuales consisten en una selección aleatoria de N*0.2 elementos del dataset original, es decir, un 20% de los elementos del dataset original seleccionados de forma aleatoria.

Es muy importante matizar que **se han seleccionado de forma aleatoria** ya que, en el dataset original, los datos vienen ordenados por su nota global (Overall) y en el caso de que un fichero estuviera formado de varios ejemplos de entrenamiento que ocupan posiciones contiguas, este contendría ejemplos de entrenamiento de características similares, lo que repercute en el correcto funcionamiento del sistema.

Para terminar, también es importante remarcar que **los ficheros con extensión "csv" resultantes tienen, en la última columna, la nota global de cada jugador. Al cargar estos ficheros en el código octave, el valor de dicha columna se modifica por el de la clase a la que pertenece:**

- Si la nota es menor o igual que 65 : 1 (Jugador "Normal")
- Si la nota se encuentra entre 65 y 85, ambos sin incluir: 2 (Jugador "Bueno")
- Si la nota es mayor o igual que 85: 3 (Jugador "Estrella")

Muestra del dataset original

```
0,Cristiano Ronaldo,32,https://cdn.sofifa.org/48/18/players/20801.png,Portugal,https://cdn.sofifa.org/flags/38.png,94,94,Rea
1,L. Messi,30,https://cdn.sofifa.org/48/18/players/158023.png,Argentina,https://cdn.sofifa.org/flags/52.png,93,93,FC Barcelo
2,Neymar,25,https://cdn.sofifa.org/48/18/players/190871.png,Brazil,https://cdn.sofifa.org/flags/54.png,92,94,Paris Saint-Ger
3,L. Suárez,30,https://cdn.sofifa.org/48/18/players/176580.png,Uruguay,https://cdn.sofifa.org/flags/60.png,92,92,FC Barcelon
4,M. Neuer,31,https://cdn.sofifa.org/48/18/players/167495.png,Germany,https://cdn.sofifa.org/flags/21.png,92,92,FC Bayern Mu
5,R. Lewandowski,28,https://cdn.sofifa.org/48/18/players/188545.png,Poland,https://cdn.sofifa.org/flags/37.png,91,91,FC Baye
6,De Gea,26,https://cdn.sofifa.org/48/18/players/193080.png,Spain,https://cdn.sofifa.org/flags/45.png,90,92,Manchester Unite
7,E. Hazard,26,https://cdn.sofifa.org/48/18/players/183277.png,Belgium,https://cdn.sofifa.org/flags/7.png,90,91,Chelsea,http
8,T. Kroos,27,https://cdn.sofifa.org/48/18/players/182521.png,Germany,https://cdn.sofifa.org/flags/21.png,90,90,Real Madrid
9,G. Higuaín,29,https://cdn.sofifa.org/48/18/players/167664.png,Argentina,https://cdn.sofifa.org/flags/52.png,90,90,Juventus
10,Sergio Ramos,31,https://cdn.sofifa.org/48/18/players/155862.png,Spain,https://cdn.sofifa.org/flags/45.png,90,90,Real Madr
11,K. De Bruyne,26,https://cdn.sofifa.org/48/18/players/192985.png,Belgium,https://cdn.sofifa.org/flags/7.png,89,92,Manchest
12,T. Courtois,25,https://cdn.sofifa.org/48/18/players/192119.png,Belgium,https://cdn.sofifa.org/flags/7.png,89,92,Chelsea,h
13,A. Sánchez,28,https://cdn.sofifa.org/48/18/players/184941.png,Chile,https://cdn.sofifa.org/flags/55.png,89,89,Arsenal,htt
14,L. Modrić,31,https://cdn.sofifa.org/48/18/players/177003.png,Croatia,https://cdn.sofifa.org/flags/10.png,89,89,Real Madri
15,G. Bale,27,https://cdn.sofifa.org/48/18/players/173731.png,Wales,https://cdn.sofifa.org/flags/50.png,89,89,Real Madrid CF
16,S. Agüero,29,https://cdn.sofifa.org/48/18/players/153079.png,Argentina,https://cdn.sofifa.org/flags/52.png,89,89,Manchest
17,G. Chiellini,32,https://cdn.sofifa.org/48/18/players/138956.png,Italy,https://cdn.sofifa.org/flags/27.png,89,89,Juventus,
18,G. Buffon,39,https://cdn.sofifa.org/48/18/players/1179.png,Italy,https://cdn.sofifa.org/flags/27.png,89,89,Juventus,https
19,P. Dybala,23,https://cdn.sofifa.org/48/18/players/211110.png,Argentina,https://cdn.sofifa.org/flags/52.png,88,93,Juventus
20,J. Oblak,24,https://cdn.sofifa.org/48/18/players/200389.png,Slovenia,https://cdn.sofifa.org/flags/44.png,88,93,Atlético M
21,A. Griezmann,26,https://cdn.sofifa.org/48/18/players/194765.png,France,https://cdn.sofifa.org/flags/18.png,88,91,Atlético
22,Thiago,26,https://cdn.sofifa.org/48/18/players/189509.png,Spain,https://cdn.sofifa.org/flags/45.png,88,90,FC Bayern Munic
23,P. Aubameyang,28,https://cdn.sofifa.org/48/18/players/188567.png,Gabon,https://cdn.sofifa.org/flags/115.png,88,88,Borussi
24,L. Bonucci,30,https://cdn.sofifa.org/48/18/players/184344.png,Italy,https://cdn.sofifa.org/flags/27.png,88,88,Milan,https
```

Muestra de alguno de los ficheros csv resultantes

```
73,33,61,66,66,37,64,65,70,56,59,13,14,13,16,15,42,47,21,62,61,22,69,58,57,31,42,71,50,71,66
73,69,63,50,65,66,54,55,62,61,49,7,11,8,9,15,65,64,32,62,43,17,58,64,74,23,62,72,90,50,67
76,79,68,66,68,68,44,53,60,71,46,10,15,11,11,14,74,80,28,70,39,25,58,72,81,24,78,79,77,54,73
77,43,76,66,73,71,77,54,78,78,58,10,14,12,12,8,68,68,21,69,56,13,68,82,74,18,67,72,75,52,76
67,69,71,61,60,66,48,44,45,45,40,13,13,10,15,8,48,60,65,58,55,59,58,46,57,59,36,64,74,51,63
84,68,82,84,66,51,61,56,70,67,44,6,5,5,6,8,64,79,12,61,61,16,65,66,63,18,50,83,61,63,69
67,39,74,70,76,53,54,54,75,67,36,9,10,15,9,7,62,51,20,65,64,25,72,67,61,24,58,68,58,70,71
78,69,77,68,74,64,70,65,75,67,51,8,13,7,16,11,48,45,36,67,51,35,66,70,77,46,59,79,68,69,72
87,42,86,83,76,84,74,86,75,73,74,8,8,15,15,13,45,58,32,68,75,20,75,70,74,21,77,74,57,77,77
84,55,91,86,83,82,73,80,85,84,77,6,7,13,9,7,74,85,44,69,68,28,77,83,78,26,76,83,57,76,81
72,82,58,57,68,72,35,47,54,26,63,7,11,10,14,7,77,80,74,57,59,75,69,52,73,69,29,68,83,38,76
69,37,73,79,56,43,43,59,49,41,39,5,15,10,7,9,33,63,27,32,53,34,55,47,41,23,38,66,46,55,54
86,42,93,88,75,79,40,76,73,71,64,12,8,9,11,12,61,64,26,63,43,28,61,77,62,29,75,85,32,65,71
78,60,83,82,70,70,66,67,73,62,56,7,11,14,12,16,71,60,65,54,60,45,67,65,62,40,60,70,57,64,70
70,38,68,76,61,60,43,54,64,56,34,8,5,11,7,7,56,62,23,52,42,16,58,54,62,13,64,75,68,51,61
78,70,81,84,59,50,57,69,65,53,61,12,9,6,9,10,32,54,32,56,56,31,59,60,49,39,44,71,34,58,56
63,51,74,75,51,48,44,35,53,27,38,12,6,12,7,8,47,76,46,32,43,42,55,35,40,48,26,60,56,44,51
79,42,78,68,58,61,56,45,58,47,46,15,7,10,10,11,59,70,64,41,55,63,59,55,56,62,49,78,60,51,66
60,77,62,53,67,62,49,43,46,34,40,11,12,10,16,8,66,54,76,44,72,71,77,39,52,69,33,53,73,65,73
60,46,43,59,30,43,20,26,24,20,29,15,12,14,7,13,47,72,46,15,20,46,29,27,35,51,22,57,60,28,50
```

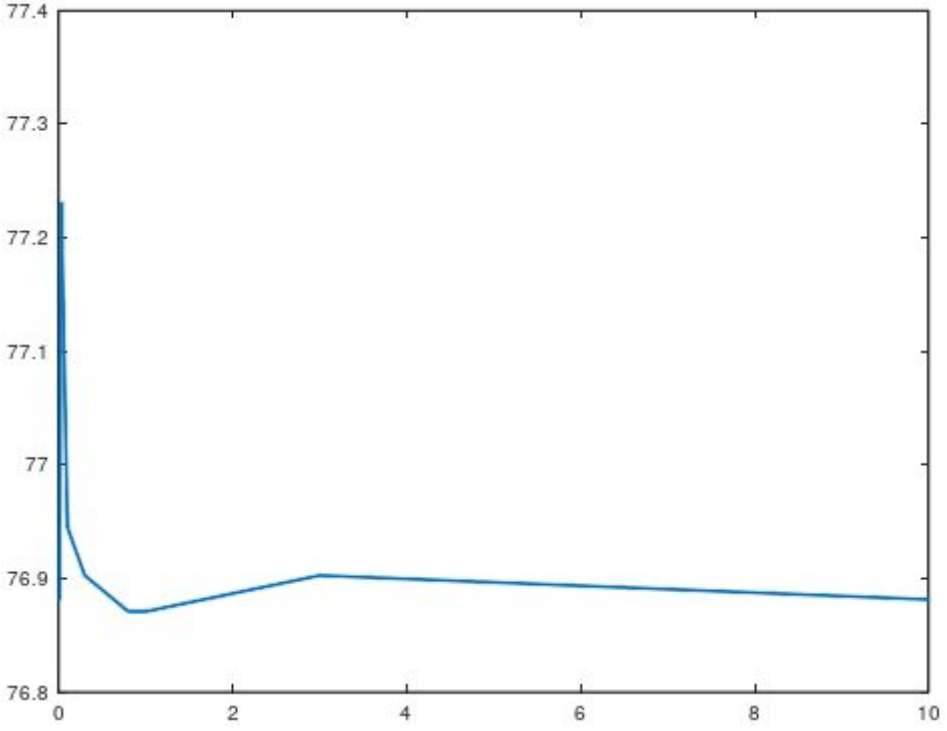
3- Regresión logística multiclase

En primer lugar, hemos utilizado los datasets resultantes para entrenar un sistema utilizando regresión logística multiclase.

3.1- Resultados obtenidos

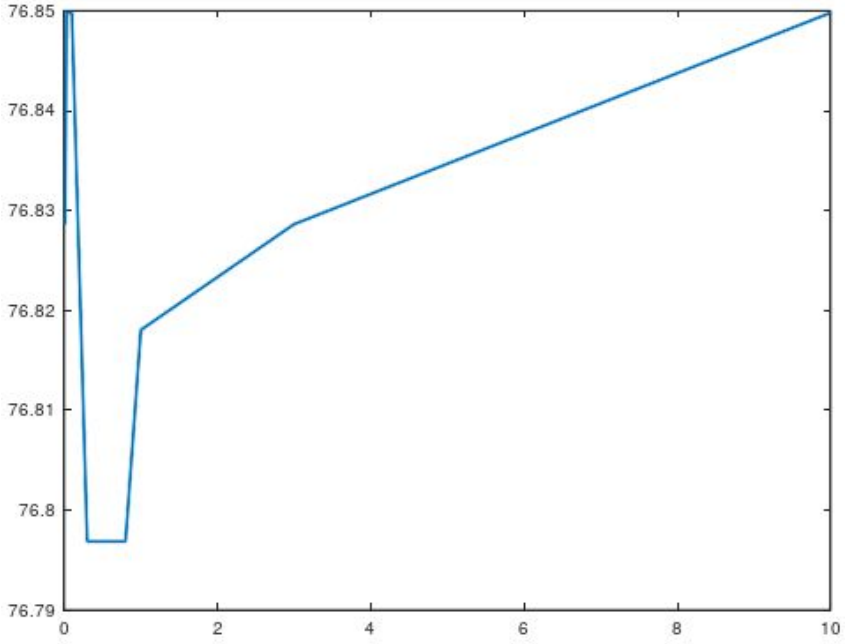
En este apartado, vamos a mostrar las precisiones obtenidas con distintos valores de lambda y distintos números de iteraciones.

3.1.1- Con 200 iteraciones



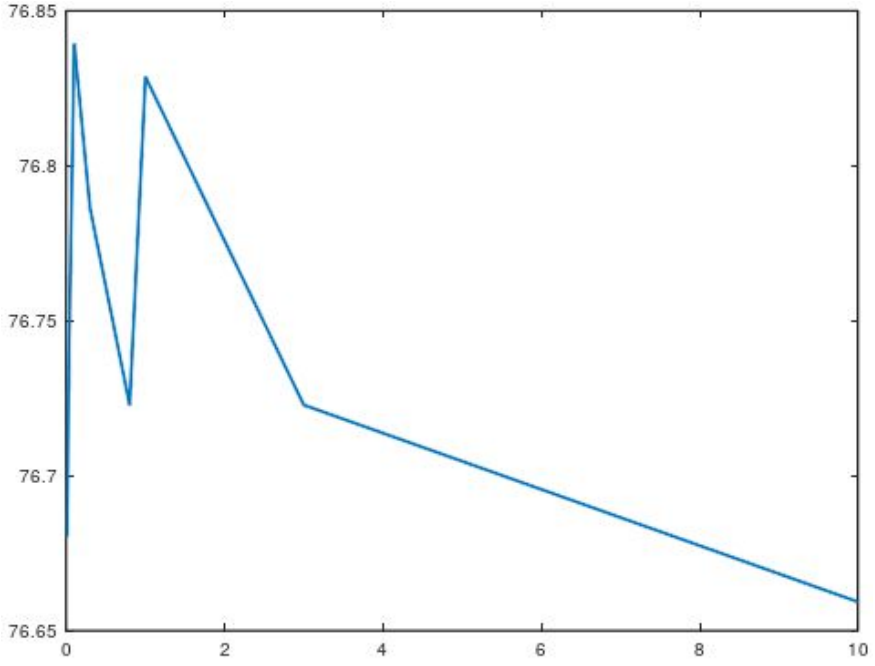
lambda	0.01	0.03	0.1	0.3	0.8	1	3	10
accuracy	76.882	77.231	76.945	76.903	76.871	76.871	76.903	76.882

3.1.2- Con 100 iteraciones



lambda	0.01	0.03	0.1	0.3	0.8	1	3	10
accuracy	76.829	76.850	76.850	76.797	76.797	76.818	76.829	76.850

3.1.3- Con 50 iteraciones



lambda	0.01	0.03	0.1	0.3	0.8	1	3	10
accuracy	76.680	76.733	76.839	76.786	76.723	76.829	76.723	76.659

3.2- Observaciones de los resultados

Viendo las gráficas anteriores, llegamos, como máximo, a obtener un **77.231%** de acierto de la clasificación de los datos de validación.

Este era el resultado esperable dada la naturaleza del problema y las capacidades de la regresión logística multicapa ya que:

1. El sistema evalúa distintos tipos de jugadores

Al evaluar jugadores que juegan en distintas posiciones y utilizar los mismos atributos para todos, añadimos una dificultad añadida al problema.

Cada jugador, en función de la posición, tiene más en cuenta un atributo u otro para inferir su nota global y, al tratar todos de la misma forma, estamos ofreciendo información al sistema, en algunas situaciones, información contradictoria.

Esto genera errores en la clasificación, lo que disminuye el porcentaje de acierto obtenido.

Un ejemplo de que este factor es determinante en el porcentaje de acierto de nuestro sistema es la eliminación de tipos de atributos y ejemplos de entrenamiento.

Cuando empezamos con el proyecto, eliminamos los ejemplos de entrenamiento de jugadores que fuesen porteros y todos los atributos del dataset referentes a este tipo de jugador.

Al eliminar un tipo de jugador, el porcentaje de acierto máximo aumentaba, aproximadamente, un 4%. Esto demuestra que el hecho de tratar distintos tipos de jugadores dificulta la clasificación.

2.Las diferencias entre jugadores de distintas clases, aunque a veces son notables, no siempre lo son

Por ejemplo, los jugadores de clase “Estrella” son aquellos con nota global mayor o igual que 85 mientras que los jugadores de clase “Normal” son los que tienen nota menor que 85 pero mayor que 65.

Entonces, los jugadores con nota global cercana a 85, los cuales suelen tener valores en sus atributos muy dispares en función de la posición o su función en el equipo, son mucho más difíciles de clasificar porque lo que les diferencia entre sí no está muy marcado en los valores de los atributos.

3.Dataset desbalanceado

Nuestro dataset no tiene, aproximadamente, el mismo número de ejemplos de entrenamiento para cada clase.

El número de ejemplos de entrenamiento de jugadores “Normales” es mucho mayor que la de jugadores “Estrella”, por ejemplo.

Esto dificulta considerablemente la tarea de clasificar los distintos tipos de jugadores.

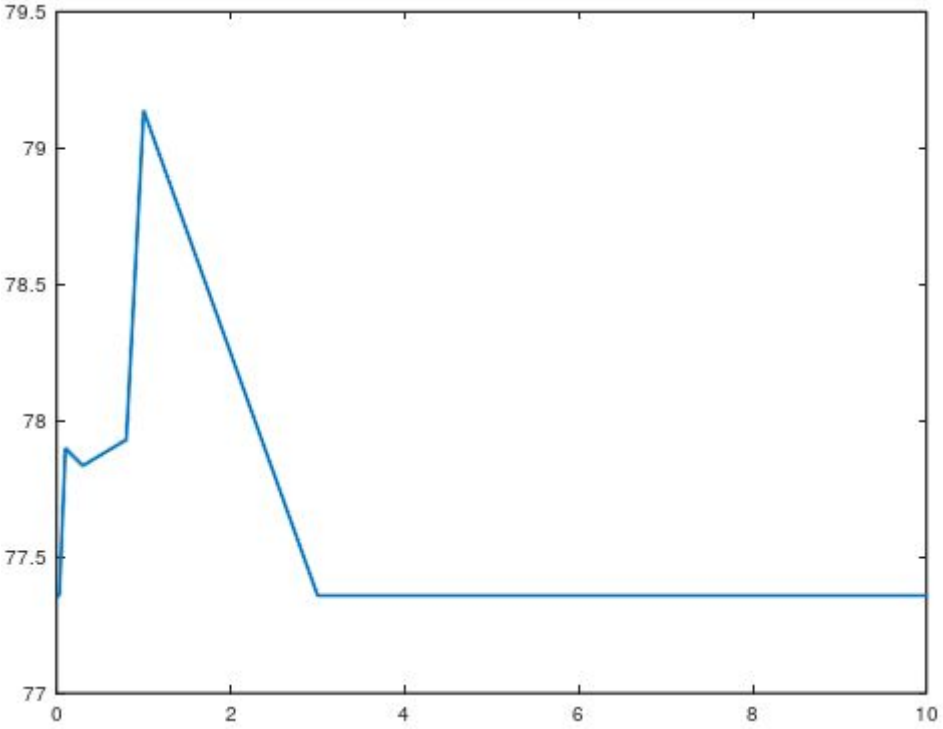
4- Redes Neuronales

Después, hemos utilizado estos datasets para entrenar una red neuronal con 30 entradas (tantas como atributos del dataset) y 3 salidas (tantas como clases a clasificar).

4.1- Resultados obtenidos

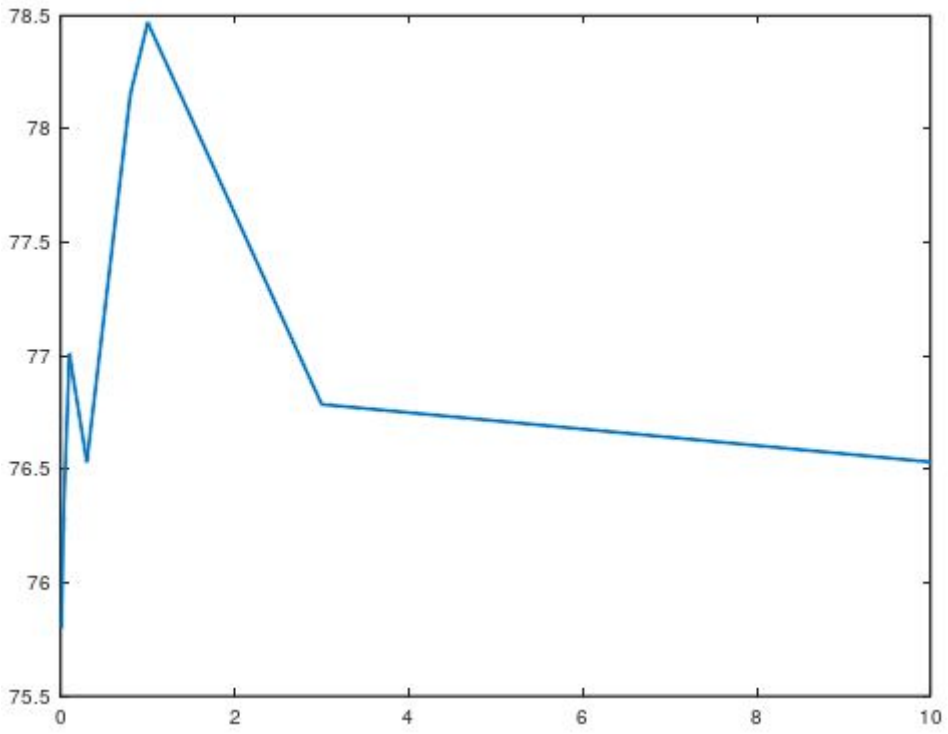
En este apartado, vamos a mostrar las precisiones obtenidas con distintos valores de lambda, distintos números de iteraciones y distinto número de capas ocultas .

4.1.1- Con 100 iteraciones y 100 capas ocultas



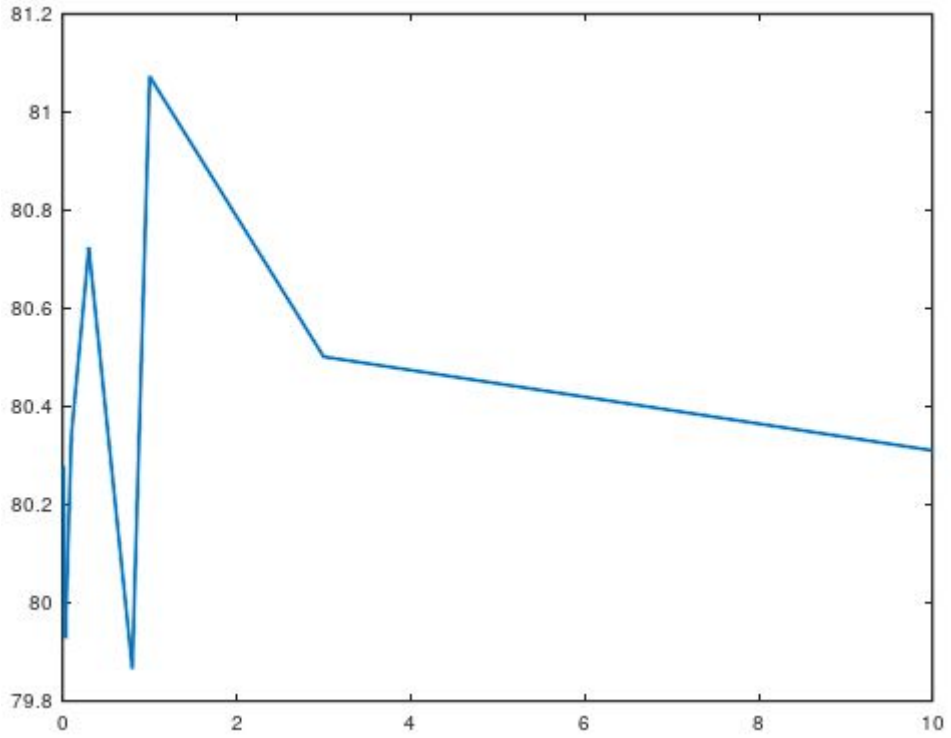
lambda	0.01	0.03	0.1	0.3	0.8	1	3	10
accuracy	77.358	77.358	77.898	77.834	77.930	79.136	77.358	77.358

4.1.2- Con 100 iteraciones y 50 capas ocultas



lambda	0.01	0.03	0.1	0.3	0.8	1	3	10
accuracy	75.802	76.278	77.009	76.532	78.152	78.469	76.786	76.532

4.1.3- Con 200 iteraciones y 200 capas ocultas



lambda	0.01	0.03	0.1	0.3	0.8	1	3	10
accuracy	80.279	79.930	80.343	80.724	79.867	81.073	80.502	80.311

4.2- Observaciones de los resultados

Viendo los resultados obtenidos, podemos observar que, usar redes neuronales, no mejora significativamente los resultado obtenidos.

Podemos ver que, por ejemplo, utilizando 200 capas ocultas en la red neuronal, solo obtenemos, aproximadamente, un 3% más de acierto frente a lo que obtuvimos utilizando regresión logística multicapa.

La mejora de debe al uso de redes neuronales, es decir, a aplicar, en el fondo, varias veces la técnica de regresión logística mientras que en apartado anterior sólo se aplicaba una vez.

Por otro lado, esta mejora es tan poco significativa debido a, como hemos explicado anteriormente, a la naturaleza del problema y a la dificultad de este.

El hecho de que clasifiquemos distintos tipos de jugadores, de que nuestro dataset esté desbalanceado y que las diferencias entre ejemplos de distintas clases no siempre son notables, ha hecho que la mejora sea ínfima.

Estas causas están explicadas en las observaciones de los resultados del apartado anterior.

En resumen, los resultados obtenidos utilizando redes neuronales, aunque presentan una precisión mayor que la obtenida utilizando regresión logística, no muestran una mejora lo suficientemente elevada para compensar la diferencia de coste computacional entre las dos técnicas. Esto se debe a la naturaleza del problema a resolver.

5- Support Vector Machine (SVM)

Por último, hemos utilizado los datasets resultantes para entrenar un sistema que utiliza support vector machines para resolver el problema propuesto.

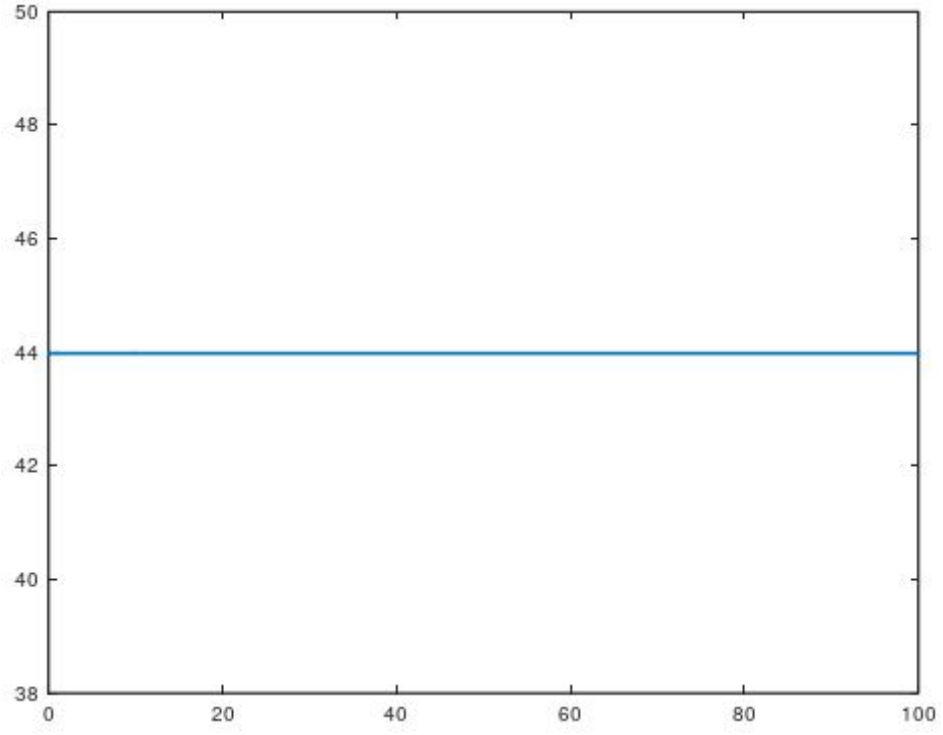
Para ello, hemos utilizado el esquema de One vs All, entrenando una SVM por cada posible clase o etiqueta.

5.1- Resultados obtenidos

Para analizar los resultados, hemos entrenado el sistema con distintos valores de C y hemos comparado las precisiones obtenidas.

El porcentaje de acierto obtenido, independientemente del kernel usado, del valor de C y de otros parámetros relevantes, siempre ha sido 43.982 %.

Utilizando TOL=0.001, max_passes=20 (linearKernel) y sigma=0.01(gaussianKernel)



Aparte de ser un porcentaje de acierto bajo, el hecho de que se repita el mismo porcentaje implica un mal uso de esta herramienta o algún fallo en la implementación.

En el siguiente apartado, se explica el porqué de este comportamiento.

5.2- Observaciones de los resultados

Al estudiar el porqué de este comportamiento, hemos observado que los distintos modelos entrenados (3 en total) infieren que todos los ejemplos de validación son de la clase que ellos mismos identifican.

Entonces, al seleccionar la predicción final, por el orden establecido, determina que todos los ejemplos son de la clase 1, de modo que solo acierta en la clasificación de los ejemplos de la clase 1 (que conforman el 43’982 de los datos de validación).

El hecho de que nuestro sistema clasifique mal era esperable por la naturaleza del problema y las características de las Support Vector Machines.

Por un lado, las SVM responden mal frente a problemas con datasets desbalanceados, es decir, datasets donde no hay, más o menos, el mismo número de ejemplos de entrenamiento de cada clase. En nuestro caso, nuestro dataset tiene muchos más ejemplos para jugadores de calidad “Normal” o “Buena” que ejemplos para jugadores “Estrella”.

Por otro lado, las SVM no son una buena opción en problemas con un elevado número de atributos. En nuestro caso, tenemos un total de 30 atributos, lo que dificulta la tarea de las SVM que componen nuestro sistema.