Doble Grado en
Matemáticas y Estadística

TRABAJO FIN DE GRADO

Modelos estadísticos en Sports Analytics

Ana Solís García

Sevilla, Junio de 2022

Índice general

Prólogo . . . III
Resumen . . . V
Abstract . . . VI
Índice de Figuras . . . VII
Índice de Tablas . . . IX

1. Revisión de la literatura **1**
1.1. La estadística en el deporte . . . 1
1.2. Estado del arte . . . 1

2. Técnicas de aprendizaje estadístico y métodos algorítmicos **5**
2.1. Aprendizaje Estadístico . . . 5
2.1.1. Evaluación de la precisión los modelos . . . 6
2.1.2. Dilema sesgo-varianza . . . 7
2.1.3. Selección de modelos . . . 7
2.1.4. Medidas de rendimiento en problemas de clasificación . . . 9
2.1.5. Técnicas de remuestreo . . . 11
2.2. Redes Neuronales . . . 12
2.2.1. Modelo de neurona de McCulloch-Pitts . . . 12
2.2.2. Perceptrón simple o mononivel . . . 13
2.2.3. Algoritmo . . . 14
2.2.4. Perceptrón multinivel . . . 15
2.2.5. Sobreajuste . . . 17
2.3. Máquinas de vectores soporte . . . 18
2.3.1. Clasificador lineal de Margen Máximo . . . 18
2.3.2. Clases linealmente separables . . . 18
2.3.3. Clases no linealmente separables. Clasificador de vectores soporte . . . 20
2.3.4. Clasificadores no lineales. Máquinas de vectores soporte . . . 22

3. Aplicación al voleibol **25**
3.1. Creación de la base de datos . . . 25
3.1.1. Procedencia de los datos . . . 25
3.1.2. Modificaciones en la base de datos . . . 25
3.2. Descripción de los datos . . . 27
3.2.1. Breve descripción de las acciones técnicas . . . 27
3.2.2. Variables del estudio . . . 27
3.2.3. Análisis de los datos . . . 29
3.2.3.1. Valores atípicos (outliers) . . . 29
3.2.3.2. Correlaciones . . . 29
3.3. Modelos de predicción . . . 31
3.3.1. Redes Neuronales . . . 31
3.3.1.1. Descripción . . . 31
3.3.1.2. Evaluación (tabla de confusión) . . . 32
3.3.2. Máquinas de vectores soporte . . . 32
3.3.2.1. Descripción . . . 32
3.3.2.2. Evaluación . . . 33
3.3.2.3. Observaciones . . . 33
3.4. Conclusiones . . . 34

A. Código R **37**
A.1. Estudio descriptivo de los datos . . . 37
A.2. Estudio de las variables . . . 38
A.2.1. Gráficos y análisis de las variables . . . 38
A.2.2. Análisis de componentes principales . . . 39
A.3. Modelos estadísticos . . . 41
A.3.1. Regla simple de Bayes . . . 41
A.3.2. Análisis discriminante lineal . . . 43
A.3.3. Regresión Logística . . . 44
A.3.4. Redes Neuronales . . . 45
A.3.5. Vectores soporte . . . 46
A.3.6. Árbol de clasificación . . . 49
A.4. Resumen . . . 50
A.5. Predicción . . . 51

Bibliografía **53**

Prólogo

Desde que conocí algo más de cerca las aplicaciones de las matemáticas, y sobre todo de la estadística al mundo deportivo he intentado profundizar en el tema de forma frecuente. La información aportada por estas estadísticas resulta de gran utilidad para el rendimiento deportivo y para el manejo y toma de decisiones de los equipos tanto a nivel profesional como a nivel amateur. Es por eso que la motivación de este trabajo ha sido aplicar los conocimientos que me ha dado esta carrera a uno de mis deportes favoritos, el voleibol, con la intención de poder indagar de qué forma se han implementado las técnicas estadísticas en este deporte.

Me gustaría agradecer en primer lugar a mi tutor, Jose Luis Pino, por ayudarme y guiarme durante el desarrollo de este trabajo.

También, agradecer de forma especial a mi familia, a mis padres y a mi hermana por su paciencia y apoyo incondicional. Y por último, agradecer a todas las personas que me han ayudado y han acompañado en este intenso camino.

Resumen

El objetivo de este trabajo ha sido ajustar distintos modelos estadísticos a los datos de la liga española de voleibol femenino. La estadística es esencial en el mundo del deporte. Actualmente, el desarrollo de cualquier deporte o actividad física lleva consigo un estudio acerca de la realización del mismo. El fin de este estudio es analizar los datos recogidos y utilizarlos de la forma más eficiente posible de cara al éxito deportivo. Con respecto a la aplicación de modelos estadísticos, algunas de las principales funciones podrían ser describir el desarrollo de competiciones y/o partidos por parte de los equipos y poder predecir a partir de ellos.

En este Trabajo Fin de Grado (TFG), se han considerado los 132 partidos de la fase regular de la Liga Iberdrola de la temporada 2020/2021, con la finalidad de obtener aquellos modelos que mejor clasificaban el resultado final del partido a partir de las acciones que se desarrollaron durante los mismos. Es decir, se han buscado algoritmos estadísticos que den mayor fiabilidad para aplicarlos a nuevos datos y poder predecir con ellos.

Este trabajo se ha dividido en 3 capítulos. En el primero introducimos pequeñas nociones sobre la estadística deportiva y nos centramos en artículos recientes en los que la aplicación de métodos estadísticos ha sido el objetivo fundamental. Hablamos tanto de deportes populares, como lo es el fútbol, como del voleibol ya que es el objetivo principal de este trabajo.

En el Capítulo 2 hacemos una descripción de lo que es el Aprendizaje Estadístico, donde se engloban los modelos de regresión y clasificación y su correspondiente estudio. También se centra en modelos estadísticos con gran capacidad de aplicación a distintos ámbitos.

Se ha evaluado la mayoría de métodos estudiados a lo largo de la carrera sobre los datos disponibles. Además, a partir de las técnicas de selección y medición del rendimiento descritas en el Capítulo 2, finalmente se han seleccionado los modelos con mejor rendimiento y se ha realizado un estudio exhaustivo en el Capítulo 3. Se ha realizado una comparación empírica del modelo de Redes Neuronales y el de Máquinas de vectores soporte sobre el conjunto de datos. Estos modelos recogen lo relativo a las estadísticas de cada equipo en los encuentros disputados de la Liga Iberdrola. En la última parte de este capítulo se toman una serie de conclusiones finales acerca del estudio realizado.

Como anexo, se ha incluido el código de R donde se encuentran los cálculos descritos en el Capítulo 3 de la aplicación de los modelos. Y finalmente, se indican las referencias bibliográficas de los documentos consultados.

Abstract

The objective of this study has been to fit different statistical models to data of the Spanish women's volleyball league. Statistic is essential in sports world. Nowadays, the development of any sport or physical activity involves a study of how it is carried out. The purpose of this work is to analyse the data collected and use them in the most efficient way to lead to sports success. Regarding the application of statistical models, some of the main functions could be describe the development of competitions and/or matches and to be able to predict from them.

In this end-of-degree project we have been considered the 132 matches of the regular phase of "Iberdrola League" of the 2020/2021 season, with the aim of obtaining those models that best classified the final result of the match based on the actions that took place during the matches. In other words, statistical models have been sought to fit new data and to be able to predict from it.

This document has been divided into 3 chapters. In the first one, we introduce basic ideas about sports statistics and we focus on recent articles where the application of statistical methods is the main objective. We discuss popular sports, such as football, and others like volleyball, which is the goal of this work.

In Chapter 2 we describe what Statistical Learning is, where regression and classification models and their corresponding study are included. It also focuses on statistical models with high applicability to different areas.

Most of the methods studied during the degree have been evaluated on the available data. In addition, using selection and performance measurement techniques described in Chapter 2, the best performing models have been finally selected and an exhaustive study has been carried out in Chapter 3. An empirical comparison between Neural Networks model and the Support Vector Machines model on the dataset has been carried out. These models collect information related to the statistics of each team in the matches played in "Iberdrola League". In the last section of this chapter, a series of final conclusions are drawn about the study carried out.

As a Annexe, the R code has been included where the calculations described in Chapter 3 of the application of the models are found. Finally, all the bibliographical references of the documents consulted are indicated in the last section.

Índice de figuras

1.1. Zonas del campo 3

2.1. Curva ROC y AUC para dataset aSAH librería pROC para modelo con variable respuesta Y=s100b 10
2.2. Modelo matemático de una neurona de McCulloch-Pitts. 12
2.3. Perceptrón mononivel 14
2.4. Perceptrón multinivel con dos capas ocultas 16
2.5. Observaciones de entrenamiento por colores según la clase a la que pertenecen. Tres hiperplanos que separan las observaciones. James et al. [9] 19
2.6. Hiperplano de margen máximo representado por la recta de color negro junto con el margen, que es la distancia desde la recta a las líneas discontinuas. James et al. [9] 20
2.7. Conjunto de datos no separables por un hiperplano.James et al. [9] 21
2.8. Conjunto de datos separables de forma no lineal. James et al. [9] 22
2.9. SVM con funciones núcleo polinomial y radial aplicada a datos no separables. James et al. [9] 23

3.1. Tabla con las estadísticas del equipo FC Cartagena - Algar Surmenor . . . 26
3.2. Zonas del campo para la recepción.pla [1] 28
3.3. Boxplot variables 1 29
3.4. Boxplot variables 2 30
3.5. Tabla de los pares de variables con coeficientes de correlación mayores a 0.85 30
3.6. Red Neuronal con los datos de los partidos 31
3.7. Curva ROC modelo Perceptrón multicapas 33
3.8. Curva ROC modelo vectores soporte 34
3.9. Curva ROC modelo SVM con up-sampling 35
3.10. Tabla de resultados reales y de predicción 35
3.11. Tabla resumen de los resultados finales 36

Índice de tablas

2.1. Ejemplo de matriz de confusión para problema de clasificación de dos clases 9

3.1. Matriz de confusión Perceptrón multicapas 32
3.2. Matriz de confusión Vectores Soporte 33
3.3. Matriz confusión SVM con up-sampling 34

Capítulo 1

Revisión de la literatura

Revisión de la literatura científica específica del voleibol y genérica aplicada a otros deportes. Aplicabilidad por estructura de la información disponible para el voleibol.

1.1. La estadística en el deporte

La estadística es una disciplina que, a día de hoy, está presente en cualquier aspecto de nuestra vida. Todo a nuestro alrededor puede ser objeto de estudio. Estos estudios se realizan con el fin de obtener datos y conclusiones para poder conocer mejor nuestro entorno y hacernos la vida más fácil. Una de las aplicaciones de la estadística que en los últimos años ha tenido una gran relevancia es la estadística deportiva.

La aplicación de la estadística al deporte tiene como objetivo principal analizar el rendimiento de un jugador o de un equipo, con el fin de reducir los errores que se producen a lo largo de un partido. Así nació lo que se conoce actualmente como *Sports Analytics*, una serie de estadísticas que proporcionan una ventaja competitiva a un equipo o individuo. Para poder elaborarlas, se utilizan métodos estadísticos que, además de ayudar a la recogida y organización de los datos en el desarrollo de los entrenamientos permiten a los entrenadores y cuerpos técnicos conocer si se han cumplido o no los objetivos planteados y así modificar el rendimiento de sus equipos.

Todo esto es una forma de hacer que los aficionados estén al tanto de ese rendimiento por parte de los equipos. Con la cantidad de avances tecnológicos que se han desarrollado en los últimos años, se puede llegar a predecir todo tipo de eventos que podrían ocurrir en el mundo deportivo. Estos avances han ayudado a profundizar en otros ámbitos importantes relacionados con el deporte, tales como la salud, para la prevención de lesiones, marketing, para la elaboración de estrategias de comunicación más eficaces, e incluso en las casas de apuestas, para la construcción de modelos predictivos que utilizan para sus ofertas.

1.2. Estado del arte

Como hemos comentado antes, una forma de poder predecir si un equipo o un jugador va a ganar o perder un partido es aplicando modelos estadísticos a datos recogidos de experiencias anteriores, en este caso, de partidos ya finalizados. En el artículo Sanjurjo et

al. [19] se aplicó un modelo de regresión logística multivariable a una serie de partidos de fútbol, con el objetivo de saber si es posible predecir el resultado de transiciones ofensivas durante un partido. Se concluyó que, además de ser capaz de predecir ese resultado, también se pudo determinar qué variables influían más en estas predicciones.

Al igual que en el fútbol, se han estudiado modelos estadísticos en otros deportes como el voleibol. El voleibol es un deporte de equipo, en el que juegan 6 jugadores y cuyo objetivo es pasar el balón por encima de la red para conseguir que caiga en el suelo del campo contrario. Los partidos se juegan al mejor de 5 sets de 25 puntos cada uno, a excepción del último, que se juega a 15 puntos. Gana el set el equipo que llega a 25, o 15 en el quinto, habiendo una diferencia de 2 con respecto a los puntos de equipo contrario. De aquí podríamos sacar datos sobre jugadores, equipos completos e incluso sobre las ligas de cada país, para construir modelos que predigan el resultado de un punto o de un partido.

En voleibol hay muchos elementos técnicos tácticos que pueden ser objeto de estudio. De cada uno de ellos se puede realizar un análisis acerca de las posibles situaciones de fallo o acierto de ejecución, y con esos datos, construir variables y poder iniciar un estudio.

En el artículo Marelić et al. [16] se investigó la influencia de 5 elementos técnicos tácticos: bloqueo, defensa, saque, recepción del saque y remate, en la puntuación de un determinado partido. Se tomaron datos del equipo que jugó la fase final del campeonato nacional de Croacia y ganó la Copa de Europa en el mismo año. La muestra consistió en 149 sets jugados durante 43 partidos de esos campeonatos, se omitieron los quintos sets. En el modelo se tuvieron en cuenta 20 variables, de forma que de cada elemento táctico descrito se tomaron cuatro evaluaciones de la eficiencia de ejecución.

A partir de un modelo de regresión con el método *Forward*, se pudo ver qué elementos influían en mayor medida para que un equipo ganase un partido. El modelo creado terminó por explicar un 63 % de la variabilidad total de la variable "ganar el partido/perder el partido", la variable respuesta. Se concluyó que la variable predictora que más influía era la que se correspondía con el número de recepciones perfectas, es decir, recibir un saque de forma perfecta es importante para el resultado de un partido.

Los jugadores de voleibol se distinguen por su posición en el campo. Principalmente, podemos distinguir las siguientes: *colocador, rematador, bloqueador, líbero* y *opuesto*. En función de la acción en la que se encuentre el equipo (recepción o defensa) estas posiciones se vuelven algo más específicas. Puesto que uno de los objetivos principales de utilizar la estadística en este deporte es llegar a construir un modelo que mida la eficacia de los jugadores de un equipo, podemos construir las variables de estudio en función de la posición de cada uno. De esta forma, se han utilizado también ciertos modelos estadísticos para estudiar el resultado de un partido a partir de esas variables.

En la Liga Turca de 2017, en Akarçeşme [4], se aplicó un estudio de este tipo distinguiendo entre hombres y mujeres, en función de las posiciones de los jugadores en el campo. Todas las variables del estudio, excepto la eficacia del líbero, se calcularon como la diferencia entre los puntos conseguidos y los perdidos. La variable dependiente, sería también la variable binaria que indica si el equipo gana el partido (1) o si lo pierde (0). El resto de variables fueron:

- Eficacia del líbero, que evalúa el error cometido del jugador líbero en la recepción del saque. Solo tomaría valores negativos debido a que este jugador está especializado

en habilidades defensivas y no puede atacar el balón, luego este valor comienza en 0 y se vuelve más negativo a medida que aumentan los errores.

- Eficacia del colocador, diferencia entre puntos anotados y perdidos en las acciones en las que ha contribuido el colocador a lo largo del partido.
- Eficacia del bloqueador, se calcula de la misma forma que la anterior para los bloqueadores centrales.
- Eficacia del rematador, igual para los rematadores de zona "4" de la figura 1.1, la zona izquierda del campo más cercana a la red.
- Eficacia del opuesto, igual para los rematadores de zona "2" dela figura 1.1, zona derecha del campo más cercana a la red.

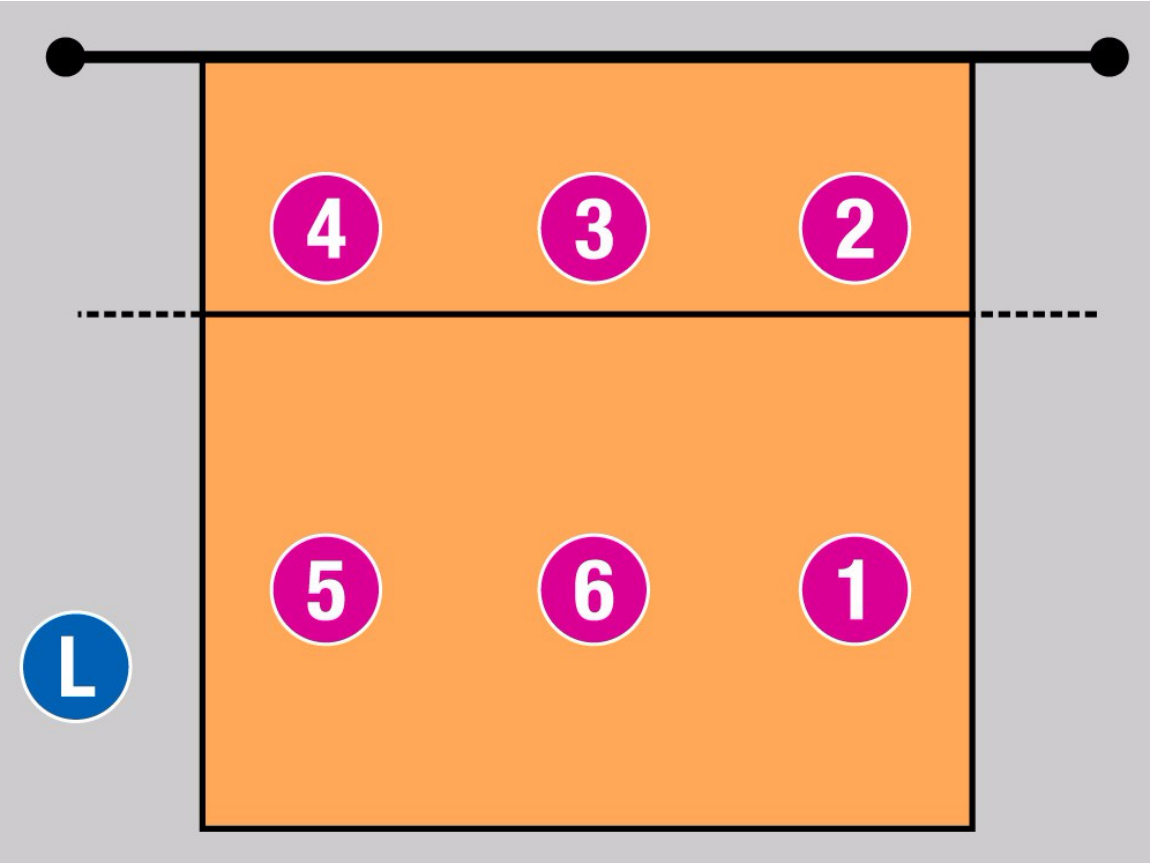

Figura 1.1: Zonas del campo

Se construyó un modelo de regresión logística como modelo de eficacia (ME) para cada liga, femenina y masculina. El ME clasificó con precisión el 83.45 % de los partidos ganados y perdidos de la liga femenina, con un valor de sensibilidad de 85.03 % y de especificidad de 81.88 % (sección 2.1.4). Para la liga masculina se obtuvo un 78.23 %, 78.77 % y 77.70 % para la precisión de clasificación (sección 2.1.3), sensibilidad y especificidad, respectivamente. Dos años más tarde se confirmó la fiabilidad y validez de este estudio.

Andrea Gabrio en su artículo *Bayesian hierarchical models for the prediction of volleyball results* Gabrio [7] propuso un modelo jerárquico bayesiano para la predicción de

las clasificaciones de los equipos femeninos de la liga italiana de voleibol, que también permitía estimar los resultados de cada partido de la liga. La muestra consistió en las estadísticas sobre los 12 equipos en 132 partidos de la fase regular de la liga 2017-2018. Para ello construyó variables relacionadas con la efectividad de los elementos técnicos de saque y ataque, desde el punto de vista ofensivo, y de bloqueo y defensa desde el punto de vista defensivo.

Se modelan 3 variables distintas: el número observado de puntos anotados por los dos equipos en un partido, la variable indicadora binaria que toma valor 1 si se jugaron 5 sets en un determinado partido y valor 0 si no; y la variable binaria para el ganador del partido. Se termina comparando la capacidad de predicción de dos de los modelos construidos referidos a la clasificación de los equipos de la liga y observa que hay pocas diferencias entre ellos y entre las clasificaciones observadas.

La revisión de la literatura científica muestra que es posible aplicar técnicas estadísticas al análisis de partidos, jugadores o tácticas del voleibol femenino en general y específicamente a las ligas nacionales de cada país.

Capítulo 2

Técnicas de aprendizaje estadístico y métodos algorítmicos

2.1. Aprendizaje Estadístico

El aprendizaje estadístico, Hastie et al. [8], juega un papel muy importante en muchos ámbitos de la ciencia, las finanzas y la industria. Desempeña un papel clave en los campos de la estadística, la minería de datos y la inteligencia artificial, y se cruza con áreas de la ingeniería y otras disciplinas.

El aprendizaje estadístico hace referencia a un amplio conjunto de herramientas para comprender unos datos determinados. Estas herramientas se clasifican en dos categorías: supervisadas y no supervisadas James et al. [9].

Aprendizaje Supervisado

Esta técnica trata de deducir una función con un conjunto de datos de entrenamiento $(x_1, y_1), \ldots, (x_n, y_n)$. Para cada dato observado x_i, contamos con una o más variables predictoras, también llamadas variables independientes, $x_i = (x_{i1}, \ldots, x_{ip})$, y además, una medida de respuesta asociada y_i, también conocida como variable dependiente. El objetivo es crear un modelo que relacione la respuesta con los predictores . A su vez, podemos considerar dos subobjetivos: poder predecir con exactitud la respuesta de observaciones futuras (*predicción*) o entender mejor la relación entre los predictores y la respuesta (*inferencia*). Ambas tareas tienen mucho en común, sobre todo como método de aproximación de funciones.

Hay dos tipos de predicción según la forma de la variable respuesta,

- *Regresión* cuando se trata de datos cuantitativos, como sería predecir el nivel de ingresos de un trabajador
- *Clasificación,* si estamos con datos cualitativos, por ejemplo, determinar si un correo es "spam". Las variables cualitativas también se conocen como variables categóricas, discretas o factores.

Los predictores, que son las variables de entrada, también varían en el tipo de medición. Esto ha dado lugar a distintos métodos utilizados en la predicción, algunos se definen de

forma más natural para las variables de entrada cuantitativas, otros para cualitativas y otros para ambos. Para el proceso de construcción del modelo se tienen en cuenta los errores realizados en las predicciones, con el fin de construir aquél que produzca mejores resultados

Aprendizaje no supervisado

Por el contrario, en el aprendizaje no supervisado nos encontramos con una situación más complicada. Para cada observación $i = 1, \ldots, n$ disponemos de un vector de mediciones x_i, pero no disponemos de una respuesta asociada y_i, carecemos de una respuesta que pueda supervisar nuestro análisis. El objetivo es inferir en las características (variables) con las que se describen esas observaciones con el fin de buscar alguna relación entre ellas para simplificar el análisis.

Una técnica muy común es el análisis de conglomerados o *clustering.* Se trata de agrupar las observaciones x_i sucesivamente en conglomerados de manera que, en cada nivel de la jerarquía, los conglomerados del mismo grupo sean más similares entre sí que los de grupos diferentes.

2.1.1. Evaluación de la precisión los modelos

Hay una amplia gama de métodos de aprendizaje estadístico. Por eso, es importante decidir para un conjunto de datos concreto cuál produce mejores resultados. Vamos a describir algunos de los conceptos que surgen al seleccionar el método más adecuado para un conjunto de datos específico.

Para evaluar el rendimiento de un método de aprendizaje estadístico en un conjunto de datos determinado, necesitamos cuantificar hasta qué punto el valor de respuesta predicho para una observación se acerca a la respuesta observada para esa observación.

Dado un conjunto de entrenamiento $(x_1, y_1), \ldots (x_N, y_N)$ y un modelo de la forma $Y = f(X) + \epsilon$ con $E[\epsilon] = 0$. Para modelos de regresión, la medida más utilizada es el *error cuadrático medio* (ECM), dado por

$$ECM = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}(x_i))^2 \tag{2.1}$$

donde $\hat{f}(x_i)$ es la predicción de la i-ésima observación. El ECM será menor si las respuestas predichas son cercanas a las respuestas observadas y será mayor si para algunas de las observaciones, la predicción y la verdadera respuesta difieren substancialmente.

Realmente, el interés no está en el ECM para los datos de entrenamiento, lo que queremos saber es si $\hat{f}(x_0)$ es aproximadamente igual a y_0, siendo (x_0, y_0) una observación test. Queremos elegir el método que dé menor ECM para la muestra test. Si tuviéramos una muestra test con suficientes observaciones podríamos calcular el *error cuadrático medio de predicción* de la misma forma que en 2.1.

Cuando un método dado produce un ECM de entrenamiento pequeño pero un ECM de prueba grande, se dice que estamos sobreajustando los datos. Esto ocurre porque nuestro procedimiento de aprendizaje estadístico se esfuerza demasiado por encontrar patrones en los datos que solo son causados por el azar. Se produzca o no sobreajuste, el ECM de

entrenamiento siempre va a ser menor que el test debido a que la mayoría de métodos de aprendizaje estadístico buscan minimizar el ECM de entrenamiento.

2.1.2. Dilema sesgo-varianza

Si suponemos que las observaciones de entrenamiento utilizadas para crear el modelo son independientes y que $E[\epsilon]$=0, $Var(\epsilon) = \sigma^2$, siendo ϵ los residuos del modelo, entonces el ECM de prueba esperado, para un valor dado x_0, siempre puede descomponerse en la suma de tres cantidades fundamentales

$$\begin{aligned} e(x_0) =& E[(Y - \hat{f}(x_0))^2 / X = x_0] \\ &= \sigma^2 + Sesgo(\hat{f}(x_0))^2 + Var(\hat{f}(x_0)) \end{aligned} \tag{2.2}$$

El primer término se denomina *error irreducible*, el segundo es el cuadrado del *sesgo* del modelo, que refleja la proximidad de la esperanza del valor de predicción con el valor real observado, y el tercer término indica la *varianza* del modelo, es decir, lo que variaría $\hat{f}$ si la estimáramos con un conjunto de entrenamiento diferente.

En general, los modelos más complejos tienen una varianza mayor, pero menor es el sesgo (al cuadrado) debido a que el valor medio de predicción se acerca más al valor real. Para modelos más simples ocurre al revés, el sesgo es mayor pero la varianza disminuye. Luego nos encontramos con el problema de no poder hacer pequeños los valores del sesgo y la varianza al mismo tiempo.

2.1.3. Selección de modelos

Suponemos que tenemos una respuesta Y de tipo cuantitativa, un vector de entradas X y un modelo de predicción $\hat{f}(X)$ que ha sido estimado a partir de una muestra de entrenamiento T. Denotamos la función de pérdida Hastie et al. [8], para medir los errores entre Y y $\hat{f}(X)$ como $L(Y, \hat{f}(X))$. Posibles funciones de pérdida son:

$$L(Y, \hat{f}(X)) = \begin{cases} (Y - \hat{f}(X))^2 & (error\ cuadrático) \\ |Y - \hat{f}(X)| & (error\ absoluto) \end{cases} \tag{2.3}$$

Anteriormente hemos hablado del ECM, que es un caso concreto del *error empírico* o *error de entrenamiento*, definido como el valor medio de la función de pérdida en la muestra de entrenamiento

$$\overline{err} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(x_i)) \tag{2.4}$$

Para el caso en que la variable respuesta Y es cualitativa o categórica con K clases, modelamos las probabilidades de cada clase $p_k(X) = P[Y = k|X]$ con la correspondiente regla de clasificación $\hat{f}(X) = arg\ max_k\ \hat{p}_k(X)$, podemos entonces definir la función de pérdida de las siguientes formas,

$$L(Y, \hat{f}(X)) = I(Y \ = \hat{f}(X)) \ \ (Función\ indicadora\ 0-1) \tag{2.5}$$

$$\begin{aligned} L(Y, \hat{p}_k(X)) &= -2 \sum_{k=1}^{K} I(Y = k) log\ \hat{p}_k(X) \\ &= -2 log\ \hat{p}_Y(X) \quad (-2\ x\ log - verosmilitud\ o\ desviación) \end{aligned} \tag{2.6}$$

Definición. Si tenemos un conjunto de entrenamiento T y un modelo de predicción $\hat{f}(X)$ ajustado sobre esos datos, definimos el *error de generalización* o *error test* como el error de predicción para una muestra de entrenamiento determinada,

$$Err_T = E_{(X,Y)}[L(y, \hat{f}(X))|T] \tag{2.7}$$

Definición. Dado un modelo de predicción $\hat{f}(X)$, definimos el *error esperado* de predicción como

$$Err = E[Err_T] = E[L(y, \hat{f}(X))] \tag{2.8}$$

El objetivo es conocer el error esperado de predicción de nuestro modelo $\hat{f}$. Por desgracia, el error de entrenamiento no es una buena estimación del error esperado. A medida que el modelo se hace más complejo, utiliza más los conjuntos de entrenamiento y se ajusta a estructuras cada vez más complejas. Como consecuencia, el sesgo disminuye pero la varianza se hace mayor y el error de entrenamiento disminuye de forma considerable. El problema es que si se hace nulo llegamos a la conclusión de que el modelo esta sobreajustando los datos de entrenamiento.

Partición de los datos

Si disponemos de un conjunto de datos de gran dimensión, lo mejor para la selección y la evaluación de los modelos es dividir el conjunto en tres:

- *Conjunto de entrenamiento*: se utiliza para ajustar los modelos,
- *Conjunto de validación*: permite estimar los parámetros de ajuste del modelo de cara a estimar el error de predicción para la selección del modelo,
- *Conjunto de prueba o test*: para calcular el error de generalización del modelo escogido.

Una partición típica podría ser 50 %-25 %-25 % para entrenamiento, validación y test, respectivamente.

Tabla 2.1: Ejemplo de matriz de confusión para problema de clasificación de dos clases

Observado	Predicción	
	no spam	spam
no spam	VN	FP
spam	FN	VP

2.1.4. Medidas de rendimiento en problemas de clasificación

Los modelos de clasificación usualmente generan dos tipos de predicciones, un valor continuo que suele ser en forma de probabilidad y una clase predicha que tiene la forma de una categoría discreta. A menudo, la atención se centra en la predicción discreta más que en la continua. Sin embargo, las estimaciones de probabilidad de cada clase son útiles para medir la confianza del modelo.

Kuhn et al. [11] describe que el método más común para evaluar el rendimiento en problemas de clasificación es la *matríz de confusión*. Se trata de una tabulación cruzada entre los valores observados y los valores predichos por el modelo. Los celdas diagonales indican los casos en los que las clases se clasifican correctamente, mientras que el resto de celdas ilustran el número de errores para cada posible caso.

Para el problema de clasificación binario de identificar correos como *"spam"*, considerando la clase *spam* como la *clase de interés*,la matriz de confusión será como se muestra en la tabla 2.1. Denotamos FN como la tasa de falsos negativos, FP la tasa de falsos positivos, VN la tasa de verdaderos negativos y VP la tasa de verdaderos positivos.

La métrica más utilizada es la conocida como tasa global de acierto o precisión, (o en el caso pesimista, la tasa de error) que refleja la concordancia entre las clases observadas y las predichas. En un problema de clasificación binario se calcularía de la siguiente forma,

$$T_a = \frac{VN + VP}{N} \tag{2.9}$$

donde N es el número total de casos de la muestra. La tasa de error sería $1 - Ta$. Si quisieramos verlo en porcentaje solo habría que multiplicar por 100 y tendríamos el *porcentaje de acierto* de la muestra, y de la misma forma, podríamos calcular la *tasa de acierto por clases*. Sin embargo, el uso de esta tasa puede traer algunas desventajas. En primer lugar, la tasa de acierto no distingue entre los tipos de errores cometidos, y en segundo lugar, no tiene en cuenta las frecuencias naturales de cada clase. En el filtrado de correo *spam*, el coste de eliminar erróneamente un correo electrónico importante es probablemente mayor que el de permitir que un correo electrónico de *spam* pase el filtro.

Para no centrarnos solo en la tasa global de acierto, para problemas de clasificación binarios, podemos tener en cuenta otras estadísticas adicionales . Introducimos el concepto de *sensibilidad* como la tasa de verdaderos positivos,

$$Sensibilidad(TVP) = \frac{VP}{N^{\circ}\ de\ casos\ positivos\ de\ la\ muestra} = \frac{VP}{VP + FN} \tag{2.10}$$

También, definimos la *especificidad* como la tasa de verdaderos negativos,

$$Especificidad(TVN) = \frac{VN}{N^{\circ}\ de\ casos\ negativos\ de\ la\ muestra} = \frac{VN}{VN + FP} \tag{2.11}$$

Suponiendo un nivel fijo de acierto para el modelo, normalmente hay que hacer una compensación entre la sensibilidad y la especificidad. Intuitivamente, al aumentar la sensibilidad de un modelo es probable que se produzca una pérdida de especificidad, ya que en nuestro caso habría más muestras predichas como "spam". Estas compensaciones son apropiadas cuando hay diferentes penalizaciones para cada tipo de error. El filtrado de correo spam, destaca la especificidad puesto que la gente está dispuesta a ver correos spam si eso implica no perder correos importantes (no-spam).

Curva ROC

La curva *ROC* es una técnica para evaluar este equilibrio entre sensiblidad y especificidad. En este caso se utilizan las probabilidades estimadas de cada clase a partir de un vector de predictores para comparar modelos.

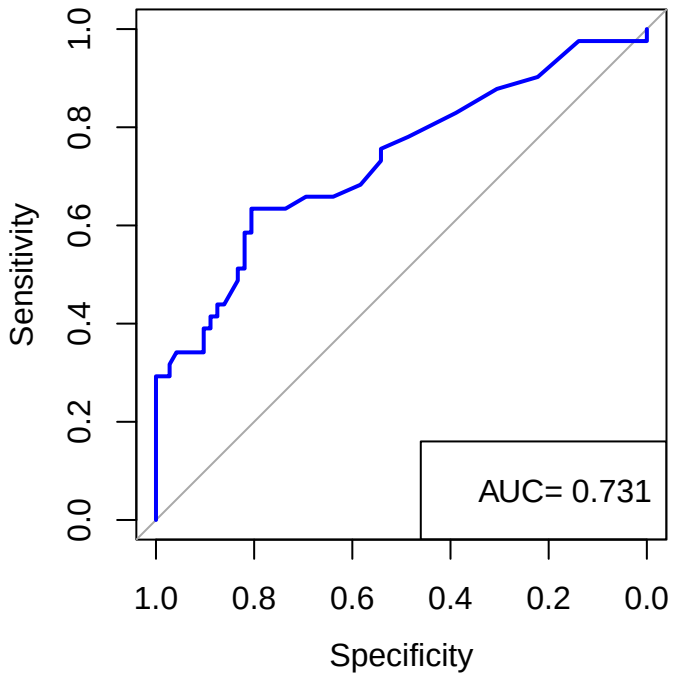

Figura 2.1: Curva ROC y AUC para dataset aSAH librería pROC para modelo con variable respuesta Y=s100b

Sean dos clases etiquetadas como "0" y "1" y dado un punto de corte p_c y un vector de predictores x, la regla de clasificación se puede expresar como

$$Y = \begin{cases} 1 & si \quad P[Y = 1/X = x] \geq p_c \\ \\ 0 & si \quad P[Y = 1/X = x] < p_c \end{cases} \tag{2.12}$$

La curva ROC es la herramienta que se encarga de elegir el umbral que maximiza el equilibrio entre la sensibilidad y la especificidad. Es la representación gráfica, figura 2.1, de la *sensibilidad* frente a 1-*especificidad* para cualquier valor del umbral.

El *AUC* (*area under the ROC curve*), mide el rendimiento de los modelos de forma que cuanto más cerca esté del valor 1 mejor será el modelo.

2.1.5. Técnicas de remuestreo

Hay situaciones en las que nos podemos encontrar un desequilibrio de clases, Kuhn et al. [11] dice que un método sencillo para reducir el impacto de este hecho en el entrenamiento de modelos es equilibrar las frecuencias de las clases seleccionando una muestra del conjunto de entrenamiento para que las tasas de cada clase se vuelvan aproximadamente iguales en la recogida inicial de datos.

Si no fuera posible un enfoque de muestreo a priori, existen enfoques a posteriori que ayudan a paliar los efectos del desequilibrio durante el entrenamiento del modelo. Dos de ellos son el muestreo ascendente o ***up-sampling*** y ***down-sampling*** o muestreo descendente.

En Ling y Li [12] describen el método up-sampling como un muestreo con reemplazamiento de las clases minoritarias hasta que cada clase tenga aproximadamente la misma frecuencia. Al hacer esto, algunas muestras de la clase minoritaria pueden aparecer en el conjunto de entrenamiento con una frecuencia bastante alta, mientras que las de la clase mayoritaria solo tienen una única realización en los datos.

El muestreo descendente trabaja de forma contraria, selecciona puntos de la clase mayoritaria para que esta tenga el mismo tamaño aproximadamente que las clases minoritarias. Hay dos formas preferibles de hacerlo. En primer lugar, tomar muestras aleatorias de la clase mayoritaria para que todas tenga aproximadamente el mismo tamaño. Y en segundo lugar, tomar una muestra conocida como *muestra boostrap* de todos los casos de forma que las clases estén equilibradas en el conjunto boostrap.

También cabe destacar el método *SMOTE*, una técnica de sobremuetreo de minorías sintéticas descrita por Chawla et al. [5]. Es un procedimiento de muestreo de datos que utiliza tanto el muestreo up-sampling con el muestreo down-sampling, dependiendo de la clase.

2.2. Redes Neuronales

Una red Neuronal es un modelo de regresión o clasificación en dos etapas, que suele representarse como un diagrama de red. El nombre de "redes neuronales" deriva del hecho de que que se desarrollaron por primera vez como modelos para el cerebro humano. Cada unidad representa una neurona, y las conexiones (enlaces en la figura 2.2) representan sinapsis.

La arquitectura de una red neuronal se representa mediante una matriz de pesos de conexión $\mathbf{W} = [w_{ij}]$, donde w_{ij} denota el peso de la conexión desde el nodo i al nodo j. Cuando w_{ij} =0, no hay conexión entre el nodo i y el nodo j . La existencia de pesos nulos hace que existan multitud de redes con topologías distintas.

Las redes neuronales que nosotros vamos a estudiar son las llamadas Redes Neuronales Alimentadas hacia delante. En estas, las conexiones entre las neuronas son en una dirección, de izquierda a derecha. Suele estar organizada en forma de capas y además, no hay conexión entre las neuronas de una misma capa y no hay retroalimentación entre las capas. De esta forma, si se trata de una red de capas totalmente conectada, cada nodo de cualquier capa se encuentra conectado a todas los nodos de la capa siguiente.

2.2.1. Modelo de neurona de McCulloch-Pitts

El primer y más simple modelo de red neuronal es el modelo matemático de neurona de McCulloch-Pitts que tiene la estructura que se muestra en la figura 2.2. Se usó como perceptrón de una capa para la clasificación de patrones linealmente separables.

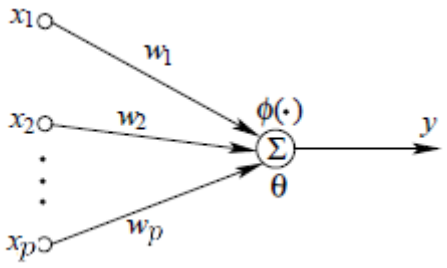

Figura 2.2: Modelo matemático de una neurona de McCulloch-Pitts.

La salida de la neurona es la siguiente:

$$s = \sum_{i=1}^{p} w_i x_i - \theta = \mathbf{w}^T \mathbf{x} - \theta \tag{2.13}$$

$$y = \phi(s) \tag{2.14}$$

donde s se refiere a la entrada a la red que viene dada por x_j que es la i-ésima entrada, w_i el peso asociado a la i-ésima entrada, $\mathbf{w} = (w_1, \ldots, w_p)^T$, $\mathbf{x} = (x_1, \ldots, x_p)^T$, θ es un umbral o sesgo que se fija y p es el número de variables de entrada. La función de activación $\phi(\cdot)$ suele ser una función continua o discontinua que transforma números

reales al intervalo (-1,1) o (0,1). Si la función de activación es la función (hard-limiter) paso nos será muy útil para la clasificación del vector $\mathbf{x}$ en dos clases. Las dos regiones de decisión están separadas por el hiperplano

$$\mathbf{w}^T\mathbf{x} - \theta = 0 \tag{2.15}$$

donde el umbral θ es un parámetro que se utiliza para alejar el límite de decisión del origen.

Las tres funciones de activación más populares son la función paso (hard-limiter)

$$\phi(x) = \begin{cases} 1 & si \quad x \geq 0 \\ \\ -1(o \;\; 0) & si \quad x < 0 \end{cases} \tag{2.16}$$

la función logística

$$\phi(x) = \frac{1}{(1 + e^{-\beta x})} \tag{2.17}$$

y la función tangente hiperbólica $\phi(x) = tanh(\beta x)$. Todas son funciones sigmoidales, es decir, monótonamente crecientes y que cumplen

$$-\infty < lim_{x \longrightarrow \, -\infty} \, \phi(x) < lim_{x \longrightarrow \, +\infty} \, \phi(x) < +\infty \tag{2.18}$$

En estas funciones, β normalmente es la unidad y se utiliza para controlar la inclinación de la función de activación.

2.2.2. Perceptrón simple o mononivel

Un perceptrón mononivel, también llamado perceptrón simple , es una Red de Neuronal Artificial alimentada hacia adelante formada por dos capas de nodos que verifican las siguientes dos condiciones:

- Los nodos de la primera capa no realizan ningún cálculo, limitándose a proporcionar los valores de entrada a la red. Esto nos quiere decir que los w_{ij}= 1 $\forall i, j$
- La segunda capa está formada por un solo nodo, que aplica la función de activación paso o signo (hard-limiter).

En Du y Swamy [6] se describe la idea principal del modelo: dado un vector de entrada de p variables $\mathbf{x} = (x_1, x_2, \ldots, x_p)$, extraer combinaciones lineales de forma que se obtengan nuevas características derivadas y mediante una función no lineal $\phi(\mathbf{x})$ predecir la respuesta y.

Para los modelos de regresión, se establece un único nodo de salida , es decir, la respuesta Y tiene dimensión 1. En cambio, para los modelos de clasificación con K clases, la capa de salida está formada por K nodos, donde el k-ésimo nodo modela la probabilidad de la clase k.

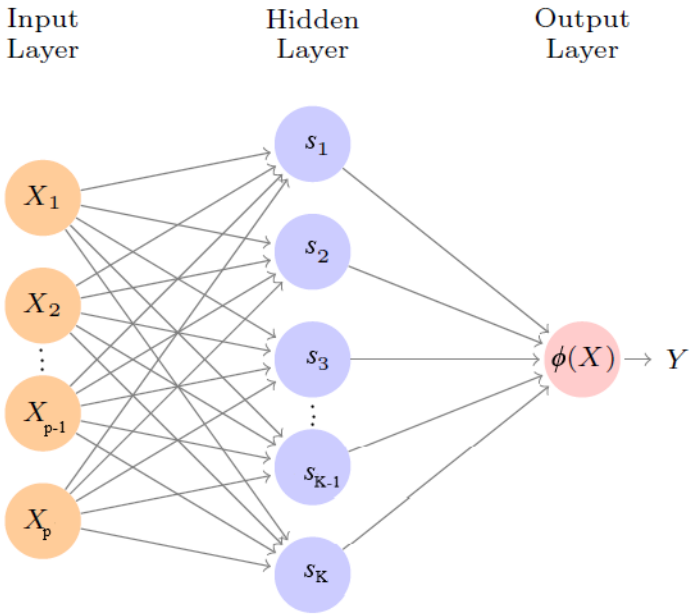

Figura 2.3: Perceptrón mononivel

El modelo de red Neuronal tiene la siguiente forma:

$$\mathbf{s} = W^T\mathbf{x} - \theta \tag{2.19}$$

$$\hat{Y} = \phi(\mathbf{s}) \tag{2.20}$$

donde $\mathbf{s} = (s_1, \ldots, s_K)^T$ siendo s_k las entradas en cada uno de los K nodos de la capa oculta , $\theta = (\theta, \ldots, \theta)^T$ los sesgos de la capa oculta , $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_K)$y $\phi(\mathbf{s}) = (\phi_1(s_1), \ldots, \phi_k(s_K))^T$

Como hemos dicho, es un modelo en dos etapas. En la primera se calculan los s_k, $k = 1, \ldots, K$, como función de las variables de entrada.

$$s_k = \sum_{i=1}^{p} w_i x_i - \theta_k \tag{2.21}$$

y después, los s_k de la capa oculta pasan a la capa de salida, donde se les aplican las funciones de activación correspondientes a cada neurona.

2.2.3. Algoritmo

Para ajustar una red neuronal se requiere la estimación a partir de los datos de los siguientes parámetros desconocidos, $w_i, \ldots, w_p$, conocidos como *pesos sinápticos* . Para realizar esta estimación podemos utilizar varios algoritmos. Suponemos que tenemos un conjunto de patrones de entrenamiento de n vectores p-dimensionales junto con sus valores de la variable respuesta, $y^{(r)}$, $r = 1, \ldots, n$. Definimos el error total de clasificación:

$$E(\mathbf{w}) = \sum_{x \in \widetilde{X}} (-\mathbf{w}^T x) \tag{2.22}$$

siendo $\widetilde{X}$ el conjunto de patrones clasificados incorrectamente por $\mathbf{w}$.

Antes de describir algunos de los algoritmos más utilizados citamos el siguiente teorema.

Teorema. *Bajo las condiciones del algoritmo y supuesto que el conjunto de patrones de entrenamiento verifica la condición de separabilidad lineal, la regla de aprendizaje del perceptrón converge y se obtiene un hiperplano de decisión entre las dos clases en tiempo finito.*

El teorema de convergencia del perceptrón se puede demostrar minimizando el error total de clasificación utilizando el método de descenso de gradiente. De esta forma, los pesos se modifican para que se reduzca el número de clasificaciones erróneas. Este teorema se puede aplicar a cualquier modelo con una o más capas ocultas.

El algoritmo de aprendizaje del perceptrón se conoce como regla delta y es el siguiente:

1. Fijamos una constante η
2. Hacer t=0
3. Generar aleatoriamente un vector inicial de pesos $w^{(0)} = (w_1^{(0)}, \ldots, w_p^{(0)})$
4. Mientras $E(w^{(0)})$ >0,

   Repetir

   { Hacer t = t +1

   Seleccionar uno de los n patrones de entrenamiento. Sea el patrón r_t el elegido y llamando $\hat{y}(x, w)$ a la respuesta obtenida a partir del vector x siendo w los pesos utilizados. Calcular un nuevo vector de pesos

   $w^{(t)} = w^{(t+1)} + \eta \ \cdot (y^{(r_k)} - \hat{y}(x^{(r_k)}, w^{(t-1)})) \cdot x^{(r_k)}$

Calcular $E(w^{(t)})$ }

Fin

La selección de η no afecta a la estabilidad del algoritmo de aprendizaje, sino a la velocidad de convergencia solo para vectores de pesos inicial distinto de cero. Normalmente suele tomarse como valor 0,5. El algoritmo acaba cuando los errores son suficientemente pequeños.

2.2.4. Perceptrón multinivel

El perceptrón multinivel es una red neuronal con dos o más capas entre la capa de entrada y de salida. Las capas situadas en medio reciben el nombre de capas ocultas o capas intermedias. Este tipo de red puede utilizarse para a clasificación de patrones no linealmente separables y para la aproximación de funciones. La arquitectura es como la representada en 2.4, un perceptrón con dos capas ocultas.

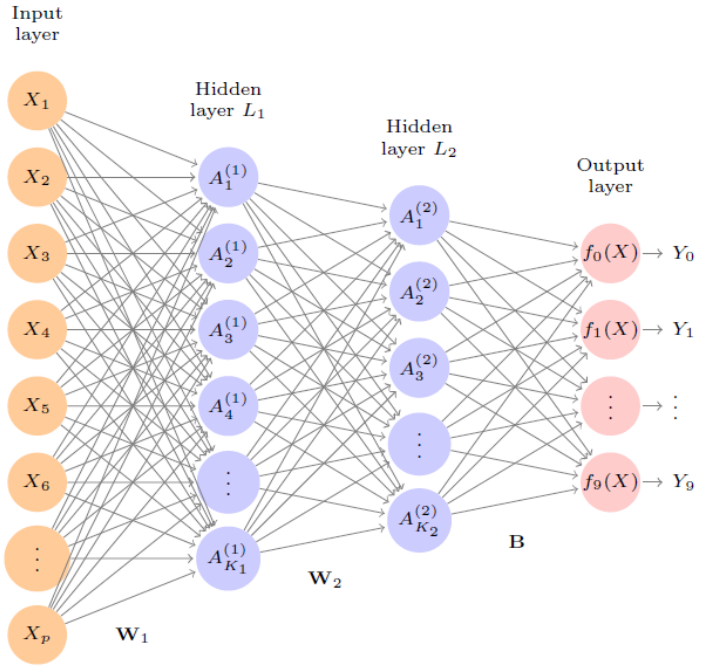

Figura 2.4: Perceptrón multinivel con dos capas ocultas

Supongamos ahora que hay M capas ocultas, cada una con K_m, $m = 1, \ldots, M$, nodos. Ahora, vamos a denotar la matriz de pesos de la capa $(m-1)$-ésima a la capa m-ésima como $\mathbf{W}^{(m-1)}$, el sesgo, la salida y la función de activación del nodo i-ésimo de la m-ésima capa como $\theta_i^{(m)}, o_i(m)$ y $\phi_i^{(m)}(\cdot)$. Para $m = 2, \ldots, M$ y el p-ésimo ejemplo

$$\hat{y}_p = o_p^{(M)}, \quad x_p = o_p^{(1)}, \tag{2.23}$$

$$s_p^{(m)} = [W^{(m-1)}]^T o_p^{(m-1)} + \theta^{(m)}, \tag{2.24}$$

$$o_p^{(m)} = \phi^{(m)}(s_p^{(m)}) \tag{2.25}$$

donde $s_p^{(m)} = (s_{p,1}^{(m)}, \ldots, s_{p,Km}^{(m)})^T$, $W^{(m-1)}$ es una matriz de dimensión K_{m-1} x K_m, el vector $o_{p,1}^{(m-1)} = (o_p^{(m-1)}, \ldots, o_{p,K_{m-1}}^{(m-1)})^T$, el vector de sesgos $\theta^{(m)} = (\theta_1^{(m)}, \ldots, \theta_{K_m}^{(m)})^T$, y $\phi^{(m)}(\cdot)$ aplica a la i-ésima componente, la función $\phi_i^{(m)}(\cdot)$. Normalmente se toman todas las $\phi_i^{(m)}(\cdot)$ como la misma función.

Como algoritmo para aplicar a las redes neuronales de múltiples capas el más común es la regla delta, pero de forma generalizada, extendiéndola a muchas capas. El algoritmo de *retropropagación* (BP) busca minimizar una función de coste, la siguiente función de error

$$E = \frac{1}{N} \sum_{p \in S} E_p = \frac{1}{2N} \sum_{p \in S} \|\hat{y}_p - y_p\|^2 \tag{2.26}$$

donde N es el tamaño de la muestra y S el conjunto de patrones de entrenamiento (x_p, y_p).

Sabemos que el algoritmo BP propaga hacia atrás el error entre la señal observada y la salida a través de la red. Después de proporcionar la entrada, la salida de la red se compara con un patrón dado y se calcula el error de cada unidad de salida. Esta señal de error se propaga hacia atrás y se establece así un bucle cerrado.

2.2.5. Sobreajuste

A menudo las redes neuronales tienen demasiados pesos, esto puede llevar a que se produzca un sobreajuste en el cálculo global del error. Para este tipo de situaciones es usual tener un conjunto de validación para la configuración de parámetros, pero además, también es útil un método de regularización conocido como *decaimiento de peso*. Consiste en introducir un término de penalización, $\lambda \geq 0$, en la función de error,

$$L(\mathbf{w}, \lambda) = E(\mathbf{W}) + \lambda \sum_{i=1}^{p} w_i^2 \tag{2.27}$$

λ influye en la repartición de pesos entre las capas; para valores más grandes de λ se tenderá a reducir los pesos a cero. Normalmente, para determinar λ se utiliza validación cruzada.

2.3. Máquinas de vectores soporte

El método de *Máquinas de Vectores Soporte* (SVM), James et al. [9], para clasificación comenzó a desarrollarse en la década de 1990 y ha crecido en popularidad desde entonces. Las SVM han demostrado tener un buen rendimiento en una variedad de entornos y se podrían considerar como uno de los mejores clasificadores "out of the box".

Para comenzar el desarrollo de este modelo de clasificación tenemos que introducir los conceptos de hiperplano y de clasificador lineal de margen máximo.

2.3.1. Clasificador lineal de Margen Máximo

Definimos el concepto de hiperplano en un espacio de dimensión p como un subespacio afín de dimensión $p-1$. De esta forma, un hiperplano en un espacio de dos dimensiones será un subespacio afín de una dimensión, es decir, una recta. Un hiperplano viene definido por la ecuación:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = X^T \beta + \beta_0 = 0 \tag{2.28}$$

Luego, si un punto del espacio p-dimensional, $X = (X_1, \ldots, X_p)^T$ satisface 2.28, entonces X se encuentra al hiperplano. Si se da el caso contrario, entonces,

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p > 0 \tag{2.29}$$

Que nos diría que X está a un lado del hiperplano. O también se puede dar que estuviese al otro lado del hiperplano,

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p < 0 \tag{2.30}$$

Así, podemos tomar un hiperplano como algo que divide a un espacio p-dimensional en dos mitades.

Supongamos ahora que tenemos una matriz $\mathbf{X}$ de dimensión n x p compuesta por n vectores de entrenamiento p-dimensionales, y que esas observaciones pertenecen a dos clases, $y_1, \ldots, y_n \in \{-1, 1\}$, donde -1 representa una clase y 1 la otra. También consideramos una observación como muestra test, $x^* = (x_1^*, \ldots, x_p^*)^T$. Nuestro objetivo es desarrollar un modelo a partir de la muestra de entrenamiento para poder clasificar correctamente la observación test x^*. Para ello primero vamos a diferenciar en caso separable y no separable.

2.3.2. Clases linealmente separables

En esta sección vamos a suponer que se puede construir un hiperplano que separe las observaciones del conjunto de entrenamiento en función de la clase a la que pertenecen. Un ejemplo de hiperplanos para el caso separable puede ser el de la imagen 2.5. Podemos considerar las observaciones de color azul como las que tienen y_i=1 y las moradas como y_i= -1. Así, el hiperplano de separación cumplirá

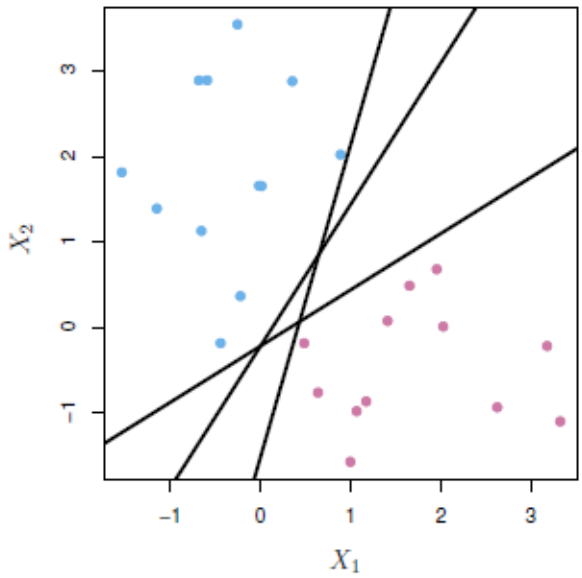

Figura 2.5: Observaciones de entrenamiento por colores según la clase a la que pertenecen. Tres hiperplanos que separan las observaciones. James et al. [9]

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} > 0 \quad si\ y_i = 1 \tag{2.31}$$

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} < 0 \quad si\ y_i = -1 \tag{2.32}$$

O lo que es lo mismo, el hiperplano de separación tiene la propiedad

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) > 0\ \forall i = 1, \ldots, n \tag{2.33}$$

Podemos tomar como regla de clasificación que se asigne a la observación test una clase u otra en función del lado del hiperplano en el que está; esto es, $G(x^*) = signo[f(x^*)]$ con $f(x^*) = \beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + \cdots + \beta_p x_p^*$. También, debemos tener en cuenta la magnitud de $f(x^*)$, si está lejos de cero, estará lejos del hiperplano y en consecuencia, la asignación de clase para x^* será más segura. En cambio, si $f(x^*)$ está cerca del cero, lo estará del hiperplano y podríamos preguntarnos si la asignación para x^* es correcta o no.

Como hemos visto antes en el caso anterior, podemos considerar más de un hiperplano de separación, luego también tendremos que tener en cuenta cuál escoger. Una opción es el hiperplano de margen máximo, es aquél que se encuentra más alejado de las observaciones de entrenamiento. Si calculamos la distancia de cada observación a los hiperplanos de separación, escogemos aquél en el que el margen sea mayor, es decir, la distancia mínima sea la más lejana a las observaciones de entrenamiento.

Para el conjunto de datos de la figura 2.5, tenemos en la figura 2.6 representado el que sería el hiperplano de margen máximo. A las dos observaciones en azul y la observación morada que están sobre las líneas discontinuas se les denomina *vectores soporte*. Curiosamente, este hiperplano solo depende de los vectores soporte y no del resto de observaciones. El movimiento de alguna de ellas no afectaría al hiperplano de separación, siempre que el movimiento de la observación no le haga cruzar la frontera establecida por el margen.

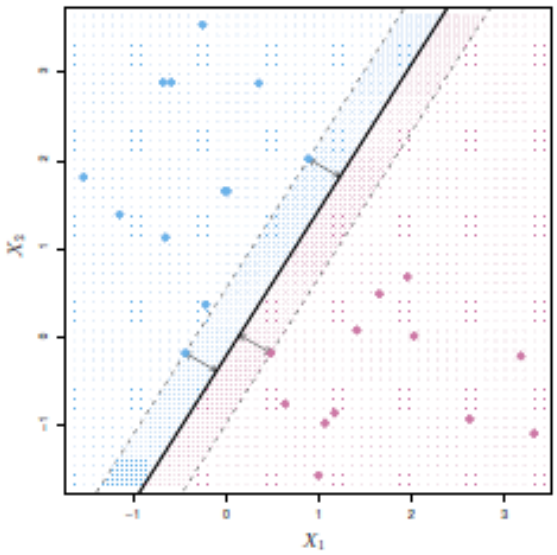

Figura 2.6: Hiperplano de margen máximo representado por la recta de color negro junto con el margen, que es la distancia desde la recta a las líneas discontinuas. James et al. [9]

Para construir el clasificador de margen máximo, considerando un conjunto de n observaciones con sus correspondientes indicadores de clase $y_1, \ldots, y_n \in \{-1, 1\}$, hay que resolver el siguiente problema de optimización

$$max_{\beta_0,\ldots,\beta_p,M} \quad M \tag{2.34}$$

$$s.a \sum_{j=1}^{p} \beta_j^2 = 1 \tag{2.35}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M \ \forall i = 1, \ldots, n \tag{2.36}$$

En este problema las restricciones 2.35 y 2.36 nos garantizan que cada observación está en el lado correcto del hiperplano y que además se encuentran al menos a una distancia M del mismo. Así, 2^*M representa el margen del hiperplano y el objetivo de este problema es encontrar $\beta_0, \ldots, \beta_p$ tales que maximicen M.

2.3.3. Clases no linealmente separables. Clasificador de vectores soporte

Podemos encontrarnos con la situación de 2.7, en la que las observaciones que pertenecen a dos clases no son separables por un hiperplano. En estos casos e incluso en aquellos en los que el margen es muy reducido, podríamos hacer uso de un hiperplano que no separe perfectamente las dos clases; con el fin de que haya una mayor robustez a las observaciones individuales y una mejor clasificación de la mayoría de las observaciones de entrenamiento.

El *clasificador de vectores soporte* se encarga de estas situaciones, intenta buscar un margen "suave", es decir, que pueda ser violado por algunas observaciones de cara a que

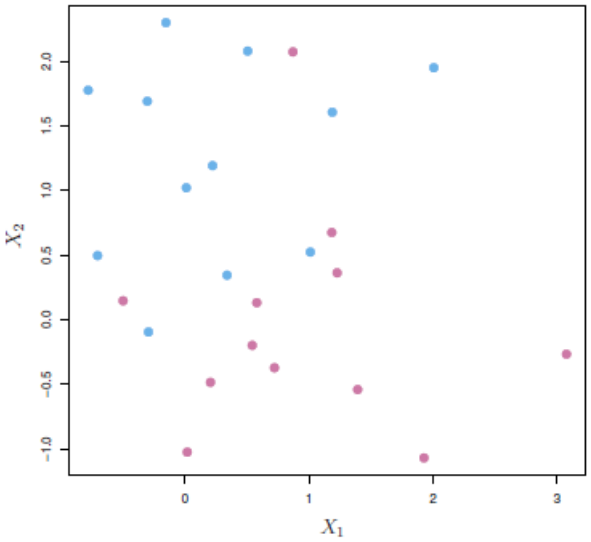

Figura 2.7: Conjunto de datos no separables por un hiperplano.James et al. [9]

podamos clasificar de forma segura el resto de observaciones de entrenamiento. El hiperplano que define al clasificador de vectores soporte viene dado por el siguiente problema de optimización

$$max_{\beta_0,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_p,M} \quad M \tag{2.37}$$

$$s.a \sum_{j=1}^{p} \beta_j^2 = 1 \tag{2.38}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i) \ \forall i = 1, \ldots, n \tag{2.39}$$

$$\epsilon_i \geq 0, \qquad \sum_{i=1}^{n} \epsilon_i \leq C \tag{2.40}$$

Cuando resolvemos el problema de optimización, la forma de proceder para clasificar x^* es la misma que hemos descrito anteriormente, basándonos en el signo de $f(x^*) = \beta_0 + \beta_1 x_1^* + \cdots + \beta_p x_p^*$.

Vamos a hacer algunas observaciones del problema 2.37-2.40. En primer lugar, los ϵ_i son variables de holgura que nos indican dónde se encuentra la observación i con respecto al margen y al hiperplano. Si ϵ_i =0, la i-ésima observación se encuentra en el lado correcto del margen, y si $\epsilon_i > 0$ diremos que la observación ha violado el margen, además, si $\epsilon_i > 1$, entonces se encuentran en el lado incorrecto del hiperplano.

Consideramos ahora el parámetro de ajuste C. Este delimita la suma de los ϵ_i y por tanto, determina el número y la gravedad de las violaciones al margen (y al hiperplano) que vamos a tolerar. Si C=0, entonces estaríamos en el problema 2.34-2.36 de clasificador de margen máximo. Para C >0, no debe haber más de C observaciones en el lado del hiperplano equivocado. A medida que C aumenta nos volvemos más tolerantes con las violaciones del margen.

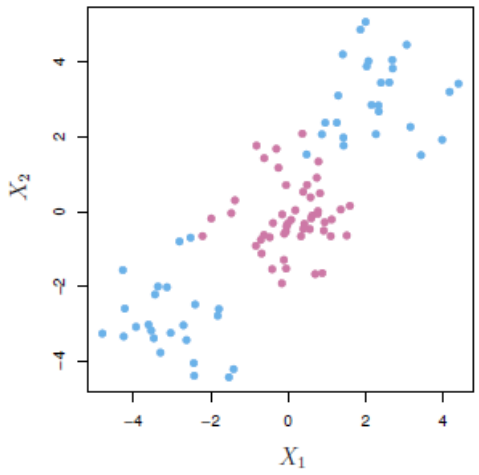

Figura 2.8: Conjunto de datos separables de forma no lineal. James et al. [9]

En la práctica C se suele escoger mediante validación cruzada y se encarga de controlar el sesgo y la varianza de la técnica de aprendizaje estadístico. El hecho de que solo los vectores soporte afecten al clasificador está directamente relacionado con el parámetro de ajuste C. Cuando es pequeño, buscamos márgenes más pequeños que rara vez se violan, entonces habrá menos vectores soporte, esto equivale a un clasificador que se ajusta mucho a los datos, lo que indica que puede tener un sesgo bajo pero una variabilidad alta. Y si C es mayor y permitimos más violaciones al mismo, el número de vectores soporte será mayor, esto indica que se ajustará menos a los datos y obtenemos un clasificador con mayor sesgo pero menor varianza.

2.3.4. Clasificadores no lineales. Máquinas de vectores soporte

En 2.8 no existe la posibilidad de construir un hiperplano de separación para este conjunto de datos. En estos casos, consideramos ampliar el espacio de variables utilizando funciones de los predictores, como términos cuadráticos, cúbicos o incluso de orden superior, para abordar la no linealidad.

Las *máquinas de vectores soporte* (SVM) son una extensión del clasificador de vectores soporte que resulta de ampliar el espacio de características, de variables predictoras, de forma específica utilizando *kernels* o funciones núcleo. En 2.37-2.40 la solución solo tiene en cuenta los productos internos de las observaciones, y no las observaciones en sí. El producto interno de dos vectores $x_i,\ x_{i'}$ viene dado por

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \tag{2.41}$$

El clasificador lineal de vectores soporte se puede representar por

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \tag{2.42}$$

con n parámetros α_i, uno para cada observación de la muestra. De esta forma, para

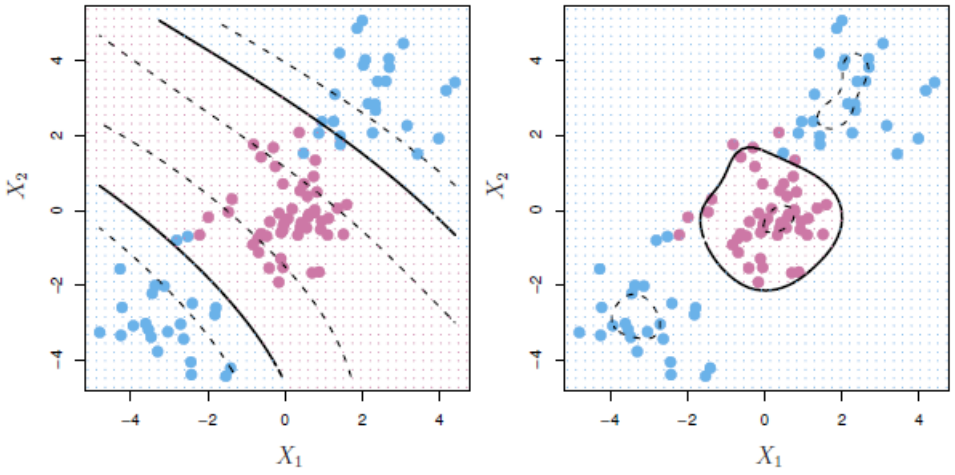

Figura 2.9: SVM con funciones núcleo polinomial y radial aplicada a datos no separables. James et al. [9]

estimar todos los parámetros, se necesita calcular los $\binom{n}{2}$ productos internos entre todos los pares de observaciones.

Al evaluar la función 2.42, nos damos cuenta de que los únicos α_i que son distintos de cero son aquellos que corresponden a los vectores soporte, luego, podemos reescribir la solución como $f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle$ siendo S la colección de índices que corresponden a los vectores soporte.

Podemos sustituir en 2.42 los productos internos por una generalización de los mismos $K(x_i, x_{i'})$, donde K se refiere a alguna función núcleo. En este caso, sería $K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}$, que es un núcleo lineal y que nos vuelve a dar el clasificador de vectores soporte. Para casos como el que hemos visto en 2.8, se podrían utilizar otras funciones núcleo, algunos de los más usuales son

$$N\acute{u}cleo\ polinomial\ de\ grado\ d: \ K(x, x') = (1 + \langle x, x' \rangle)^d \tag{2.43}$$

$$N\acute{u}cleo\ de\ base\ radial\ gaussiana: \ K(x, x') = exp(-\gamma \|x - x'\|^2) \tag{2.44}$$

La principal ventaja de utilizar este tipo de funciones es que hacemos que el método sea más sencillo computacionalmente, solo hay que calcular $K(x_i, x_{i'})$ para todos los $\binom{n}{2}$ distintos pares i, i'.

En la figura 2.9, podemos ver en la imagen de la izquierda que se ha aplicado una SVM con un núcleo polinómico de grado 3 a los datos de la figura 2.8. En la imagen de la derecha se aplicó una SVM con núcleo radial. En ambos casos podemos ver que este método da como resultado una regla de decisión más apropiada.

Capítulo 3

Aplicación al voleibol

3.1. Creación de la base de datos

3.1.1. Procedencia de los datos

Los datos han sido extraídos de la web oficial de la Real Federación Española de Voleibol, fed [2], en la que podemos encontrar estadísticas de todas las competiciones profesionales, tanto masculinas como femeninas, del voleibol español.

Se han recogido los datos de la Liga Iberdola de la temporada 2020/2021. Es conocida con ese nombre por motivos de patrocinio pero se refiere a la Superliga Femenina de Voleibol, es decir, la liga femenina de voleibol de máxima categoría en España. Nació en 1970, organizada por la Real Federación Española de Voleibol, Wikipedia [25]. El sistema de competición para determinar su posición en la clasificación consiste en dos vueltas en las que se enfrentan todos contra todos. Una vez terminada esa fase regular, los cuatro primeros equipos de la clasificación se vuelven a enfrentar en una eliminatoria al mejor de cinco partidos donde finalmente se proclama un ganador.

En la página correspondiente a la Liga Iberdola de la temporada 2020/2021, dentro del apartado *Estadísticas* se ha seleccionado *Detalles por equipos*. De cada equipo, se han utilizado solo los datos acerca de cada partido jugado (no se han utilizado los datos totales ni en porcentaje), de forma que por cada equipo tenemos 22 registros que corresponden a los 22 encuentros en los que han participado.

Los datos se han extraído mediante el portapapeles utilizando *Excel*, creando un archivo con formato *.xlsx* llamado *Partidos_20_21.xlsx*. Luego, hemos obtenido una base de datos con 264 registros y 26 variables.

3.1.2. Modificaciones en la base de datos

En esta misma hoja de cálculo, hemos realizado las siguientes modificaciones:

- La primera columna de nuestra base de datos se corresponde con los equipos que han participado en el encuentro (*ejemplo: "FC Cartagena - Algar Surmenor — Madrid Chamberí"*) . Como de cada partido nos interesan las estadísticas por separado

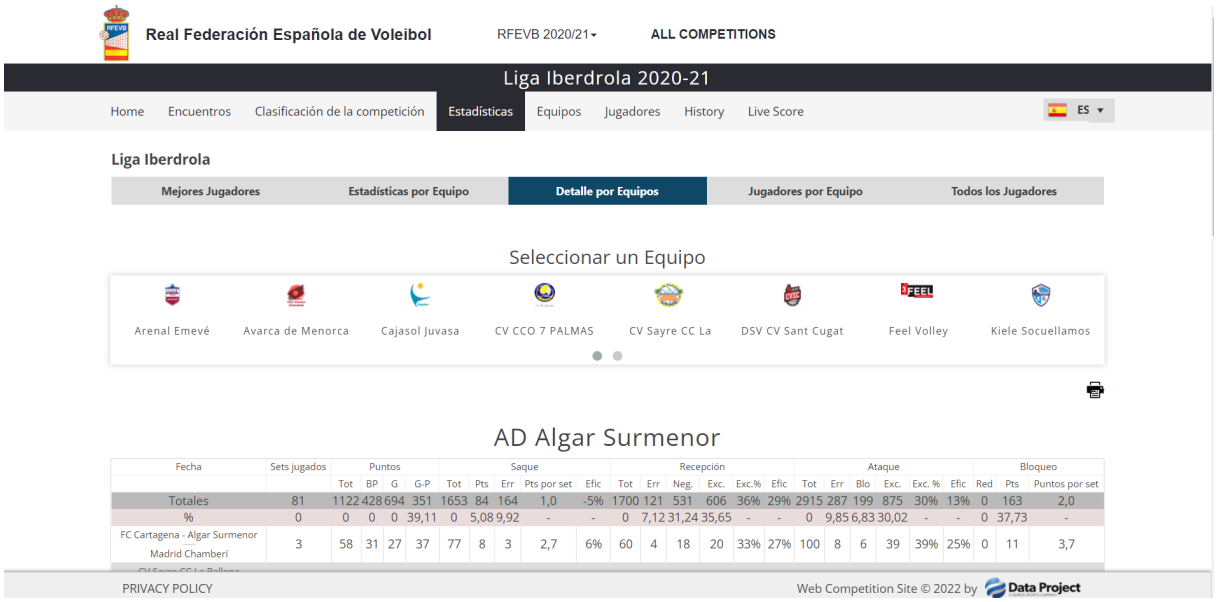

Figura 3.1: Tabla con las estadísticas del equipo FC Cartagena - Algar Surmenor

de cada equipo, hemos modificado la variable renombrándola como **Equipo**. En cada registro entonces se hace referencia al equipo del que hemos extraído las datos correspondientes.

- Se ha añadido una variable que nos será muy relevante en nuestro estudio. A esta variable la llamaremos **Ganado/Perdido** y nos indica con un 0 o un 1 si el equipo correspondiente a ese registro ha perdido o ganado ese partido, respectivamente.

- Hemos renombrado las variables correspondientes a cada acción técnica, con el fin de evitar confusión entre variables con nombres iguales o similares. De esta forma, se ha añadido el nombre de la acción a la que se refiere la variable al principio de cada una de ellas (*ejemplo: "Saque-Tot","Recep-Tot"*).

- Se ha encontrado un error en el registro número 245, en las estadísticas de "Sanaya Libby's La Laguna" correspondientes al partido contra "Osacc Haro Rioja Voley". La columna "Tot" es el resultado de sumar las columnas "Saque-Pts" , "Ataque-Exc" y "Bloqueo-Pts". El valor de "Tot" para ese registro es 51 que no coincide con la suma de los valores correspondientes (4 + 35 + 13). Finalmente, buscando en la pestaña *Encuentros* en la web fed [2] el partido "Sanaya Libby's La Laguna — Osacc Haro Rioja Voley" hemos encontrado la ficha de *DataVolley* en la que el valor de "Bloqueo-Pts" no coincide con el valor anterior. De esta forma, los valores para las columnas "Tot" y "Bloqueo-Pts" se han cambiado por `50` y `11` respectivamente.

Finalmente, queda una base de datos formada por 264 registros con 27 variables que describen las estadísticas correspondientes a todos los encuentros jugados durante esa temporada.

3.2. Descripción de los datos

3.2.1. Breve descripción de las acciones técnicas

Para poder introducir las variables necesitamos saber cuáles son las principales acciones técnicas del voleibol, que se recogen en Quintana [18]:

- Saque: acción de poner en juego el balón por el jugador zaguero derecho situado en la zona de saque, es decir, uno de los atacantes de detrás de la linea de 3 metros que divide la zona de ataque delantera.
- Recepción: interceptar y controlar un balón dirigiéndolo hacia otro compañero en buenas condiciones para poder jugarlo. Los balones bajos se reciben con los antebrazos unidos al frente a la altura de la cintura y los altos con los dedos, por encima de la cabeza.
- Ataque: toda acción de dirigir el balón al campo del adversario, excepto el saque y el bloqueo, se considera golpe de ataque.
- Bloqueo: acción de los jugadores cerca de la red encaminada a interceptar el balón que procede del campo contrario por encima del borde superior de la red. Sólo los delanteros pueden completar un bloqueo. Está prohibido bloquear el saque adversario.

3.2.2. Variables del estudio

En este estudio se ha trabajado con 18 variables, que se han seleccionado a través del **software R** con el interfaz gráfico *Rstudio*, al igual que el resto de modificaciones que se han llevado a cabo con el resto datos. Las variables estudiadas son las siguientes:

- Equipo: equipo al que corresponden las estadísticas extraídas de la base de datos para ese partido.
- Sets jugados: número de sets que se han jugado en el partido. Un partido de voleibol se juega al mejor de 5 sets y un equipo gana un set cuando llega a un total de 25 puntos con una ventaja de 2 puntos con respecto a los del equipo contrario. El quinto set se juega a 15 puntos.
- BP: (break-point) puntos que se consiguen cuando el equipo está sacando, es decir, manteniendo el saque.
- G: puntos conseguidos cuando el saque ha correspondido al equipo contrario.
- Saque-Tot: número de saques totales realizados durante el partido.
- Saque-Pts: puntos totales conseguidos con un saque directo, es decir, aquél saque en el que el balón cae directamente en el campo del equipo contrario.
- Saque-Err: número de saques fallados.

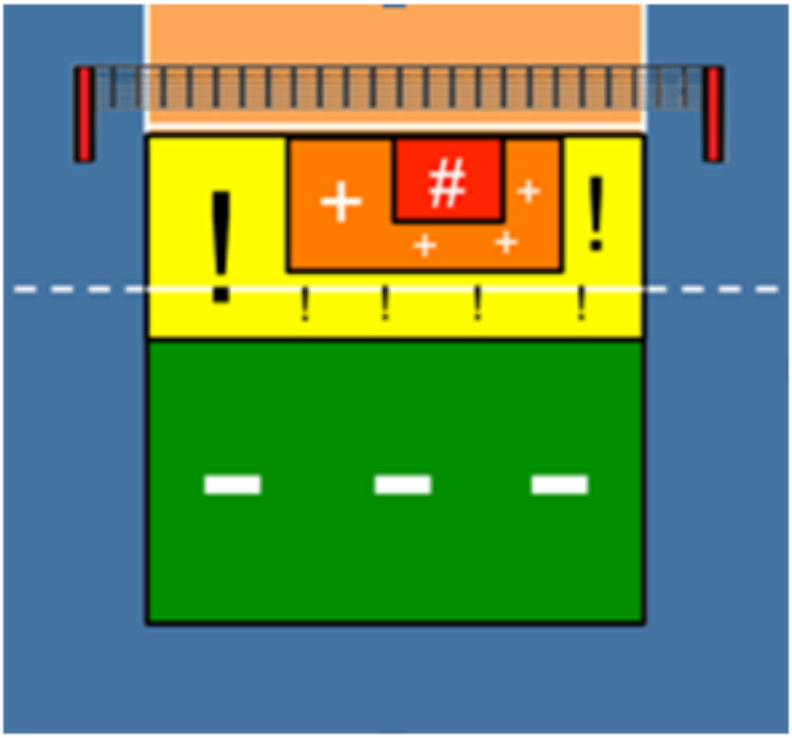

Figura 3.2: Zonas del campo para la recepción.pla [1]

- Recep-Tot: número de recepciones totales realizadas durante el partido.
- Recep-Err: número total de errores en las recepciones durante el partido.
- Recep-Neg: número de recepciones "negativas" entendiéndose por "negativas" aquellas en las que el balón va a la zona verde del campo en la figura 3.2.

- Recep-Exc: número de recepciones perfectas, aquellas en las que se dirige el balón a la zona roja de la figura 3.2 (zona donde se encuentra el colocador para realizar el segundo toque), de forma que sea más fácil construir una jugada.
- Ataque-Tot: número de acciones de ataque realizadas durante el partido.
- Ataque-Err: número total de errores en ataques (mandados fuera del campo o que no han pasado la red) durante el partido.
- Ataque-Blo: número total de ataques que han sido bloqueados durante el partido.
- Ataque-Exc: número de puntos que se han conseguido con un ataque durante el partido.
- Bloqueo-Red: número de acciones de bloqueo en las que se ha tocado la red.
- Bloqueo-Pts: puntos conseguidos por el bloqueo durante el partido.
- Ganado/Perdido: variable de tipo factor que toma valores 0 o 1 en función de si el equipo ha ganado o perdido el partido, respectivamente.

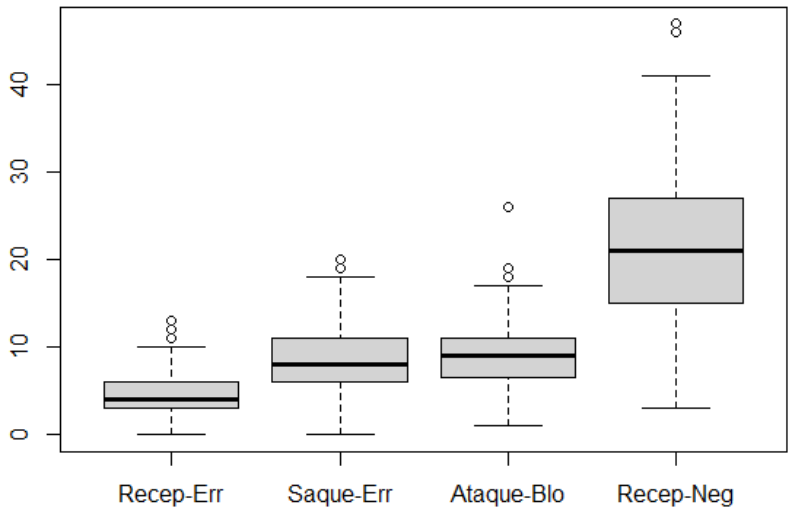

Figura 3.3: Boxplot variables 1

Como hemos indicado en los objetivos del trabajo, nuestro fin es buscar un modelo que sea capaz de predecir según las acciones que realiza un equipo durante un partido si lo gana o lo pierde. De esta forma, nuestra variable respuesta será la variable *Ganado/Perdido*.

3.2.3. Análisis de los datos

3.2.3.1. Valores atípicos (outliers)

Mediante gráficos de caja y bigotes, utilizando la función *boxplot()*, observamos que hay "valores atípicos" en algunas de las variables. En las variables *Recep-Err*, *Saque-Err*, *Ataque-Blo* o *Recep-Neg*, estos valores indican que ha habido muchos fallos en acciones técnicas durante el partido. Esto puede indicarnos que la probabilidad de que los equipos hayan perdido esos partidos es mayor.

En cambio, en variables como *Saque-Pts*, *Bloqueo-Pts* o *Recep-Exc*, que los valores sean mayores también pueden ayudarnos a distinguir si se ha ganado o no el partido.

3.2.3.2. Correlaciones

Analizamos las correlaciones entre las variables explicativas para ver si encontramos problemas de multicolinealidad, de forma que dos o más variables expresen información similar. El determinante de la matriz de correlaciones nos da un valor muy cercano a 0, lo que significa que hay variables con una alta correlación. Las variables con una correlación mayor a 0.85 son las que se muestran en la figura 3.5.

Para intentar solucionar este problema de multicolinealidad se ha realizardo un análisis de componentes principales que, finalmente, no nos ha resultado de utilidad puesto que

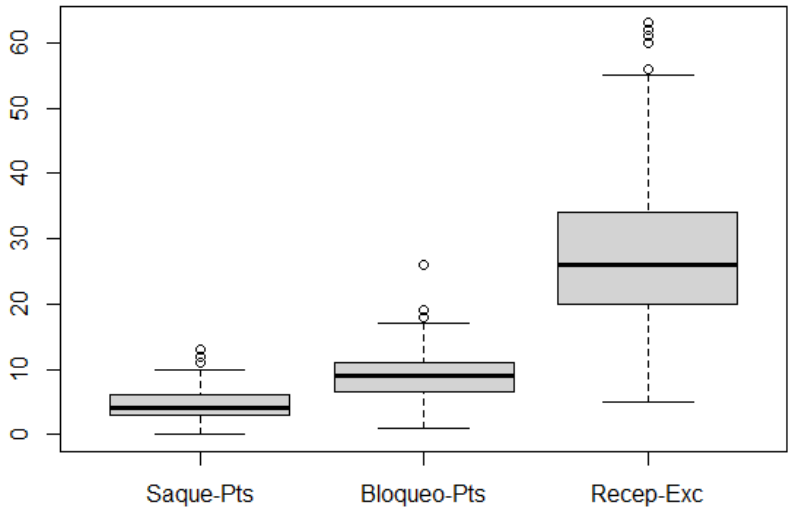

Figura 3.4: Boxplot variables 2

V1	V2	Correlación
Sets jugados	Saque-Tot	0.8729
G	Ataque-Exc	0.8984
Saque-Tot	Ataque-Exc	0.9052

Figura 3.5: Tabla de los pares de variables con coeficientes de correlación mayores a 0.85

las componentes principales seleccionadas no se pueden interpretar de una forma sencilla con respecto a las variables. Se ha continuado el estudio con las variables iniciales.

3.3. Modelos de predicción

En este apartado vamos a describir los modelos de predicción que mejor se han ajustado a los datos. Previamente, se ha realizado un estudio de los principales modelos estadísticos, mediante una partición de los mismos en muestra de entrenamiento y muestra test (70 %-30 %). Al analizar su rendimiento hemos obtenido dos modelos con resultados destacables con respecto al resto y los describimos a continuación.

3.3.1. Redes Neuronales

3.3.1.1. Descripción

Se ha ajustado un modelo utilizando la librería *caret*, con la función *train* mediante el método *nnet*. Para el tamaño de la capa oculta se ha probado con valores de 1 a 18 (que es el número de variables del estudio) y para valores del parámetro λ se ha evaluado en 0, 0.05 y 1.

La red final es *16-2-1 network with 37 weights, decay=0.1*, es decir, está formada por la capa de entrada con las variables predictoras, una capa oculta con dos neuronas y la capa de salida, de forma que en total hay 37 pesos. El parámetro λ (término de penalización en la función de error cuyo objetivo es reducir el sobreajuste, memoriza los datos de entrenamiento pero es incapaz de predecir correctamente nuevas observaciones) óptimo ha sido **0.1** . La forma de esta red neuronal se encunetra gráficamente en la figura 3.6.

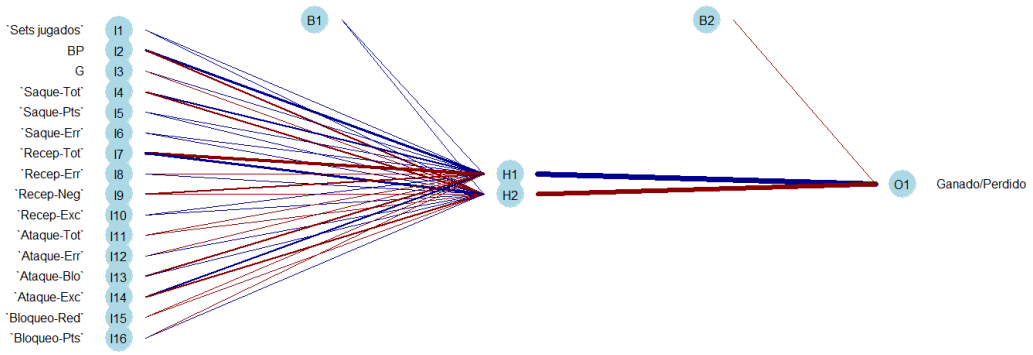

Figura 3.6: Red Neuronal con los datos de los partidos

Tabla 3.1: Matriz de confusión Perceptrón multicapas

Observado	Predicción	
	Perdido	Ganado
Perdido	38	3
Ganado	5	33

3.3.1.2. Evaluación (tabla de confusión)

Al evaluar en la muestra test obtenemos la siguiente matriz de confusión de la tabla 3.1.

De aquí se deduce lo siguiente:

- Hay 38 "verdaderos negativos", es decir, partidos que se han perdido y que el modelo los ha clasificado como perdidos.
- Hay 3 "falsos positivos", es decir, partidos que se han perdido y que el modelo ha clasificado como ganados.
- Hay 5 "falsos negativos", es decir, partidos que se han ganado y han sido clasificados como perdidos por el modelo.
- Hay 33 "verdaderos positivos", es decir, partidos ganados y que han sido clasificados como ganados por el modelo.

Luego la probabilidad de acierto para el modelo del Perceptrón multicapas es del 89.97 %, y para cada valor de la variable *Ganado/Perdido* las probabilidades de acierto son 92.68 % para la clase *Perdido* y 86.84 % para la clase *Ganado*.

Para terminar con la eficacia de este modelo, tenemos la representación gráfica de la curva ROC, en la figura 3.7 y el correspondiente valor del área bajo la curva, AUC, 0.9415918.

3.3.2. Máquinas de vectores soporte

3.3.2.1. Descripción

Al igual que en el modelo anterior, se ha utilizado la función *train* para el desarrollo del modelo pero esta vez con el método *svmRadial*. Este método utiliza la función núcleo de base radial gaussiana. El parámetro **C**, como ya hemos visto, indica cómo de severo y el número de violaciones del margen. Los valores que se han probado para este han sido 0.1, 1, 5, 10, 50 y para el parámetro γ han sido 0.025,0.035 y 0.5. En este caso γ está relacionado con la flexibilidad del modelo.

El modelo final que hemos obtenido tiene como parámetros $\mathbf{C = 1}$ y $\gamma = \mathbf{0.025}$

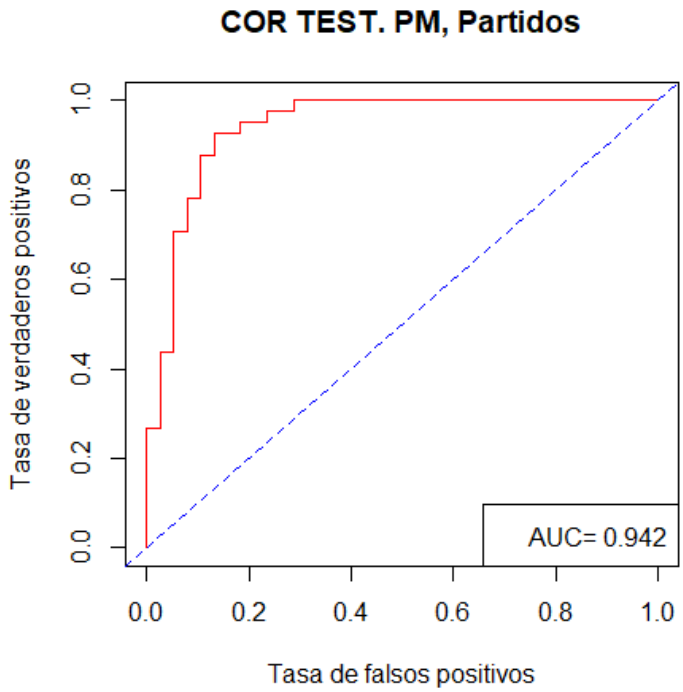

Figura 3.7: Curva ROC modelo Perceptrón multicapas

Tabla 3.2: Matriz de confusión Vectores Soporte

Observado	Predicción	
	Perdido	Ganado
Perdido	38	3
Ganado	7	31

3.3.2.2. Evaluación

Evaluando el modelo con la muestra test, obtenemos los siguientes resultado de la tabla 3.2.

En este caso, al tener 38 "verdaderos negativos" y 31 "verdaderos positivos" para este modelo la probabilidad de acierto es 87.34 %, algo menor que para el Perceptrón multicapas. Las probabilidades de acierto por clases son 92.68 % y 81.58 %, para la clase "Perdido" y para la clase "Ganado", respectivamente.

Con respecto a la curva ROC y el valor del área bajo la misma, AUC de 0.9557125, vemos en la figura 3.8 que es algo mayor que para el modelo anterior, aunque la probabilidad de acierto sea menor.

3.3.2.3. Observaciones

La dimensión de nuestra muestra de entrenamiento según las clases de la variable respuesta no es balanceada, tenemos 92 casos para la clase *Perdido* y 93 casos para la clase *Ganado*. Como hemos visto en 2.1.5, cuando nos encontramos con esta situación podemos

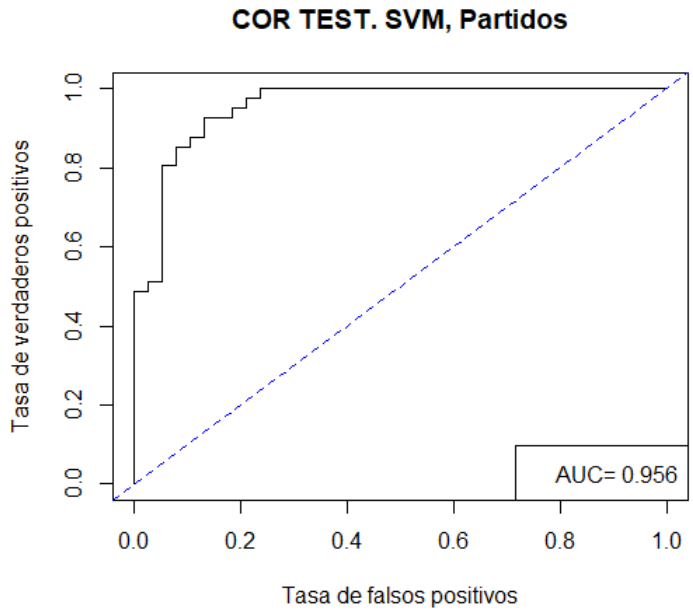

Figura 3.8: Curva ROC modelo vectores soporte

Tabla 3.3: Matriz confusión SVM con up-sampling

Observado	Predicción	
	Perdido	Ganado
Perdido	37	4
Ganado	4	34

utilizar la técnica Up-Sampling. La diferencia en los tamaños de las muestras es mínima luego, en principio, no sería necesario utilizar esta técnica. De todas formas, se ha querido comprobar si los resultados difieren con los obtenidos anteriormente.

Después de aplicar Up-Sampling a los datos y obtener una muestra de entrenamiento balanceada, se ha aplicado el modelo de Máquinas de Vectores Soporte. El modelo final obtenido (utilizando la misma función núcleo) tiene como parámetros $\mathbf{C = 5}$ y $\gamma = \mathbf{0.025}$.

Una vez realizada la evaluación del modelo, a partir de la matriz de confusión de la tabla 3.3, vemos que la probabilidad de acierto es mayor que en el modelo SVM, 89.87 %, debido a que difieren también las probabilidades de acierto por clases, 90.24 % para la clase *Perdido* y 89.47 % para la clase *Ganado*.

En cuanto a la curva ROC, el AUC es 0.9505777, muy parecido al del modelo anterior, representada en la figura 3.9.

3.4. Conclusiones

Hemos cogido como ejemplo para ilustrar esta idea algunos de los primeros partidos jugados en la temporada 2021/2022. En este caso,“Avarca de Menorca - Arenal Emevé” y “CV Kiele Socuéllamos - Feel Volley Alcobendas”.

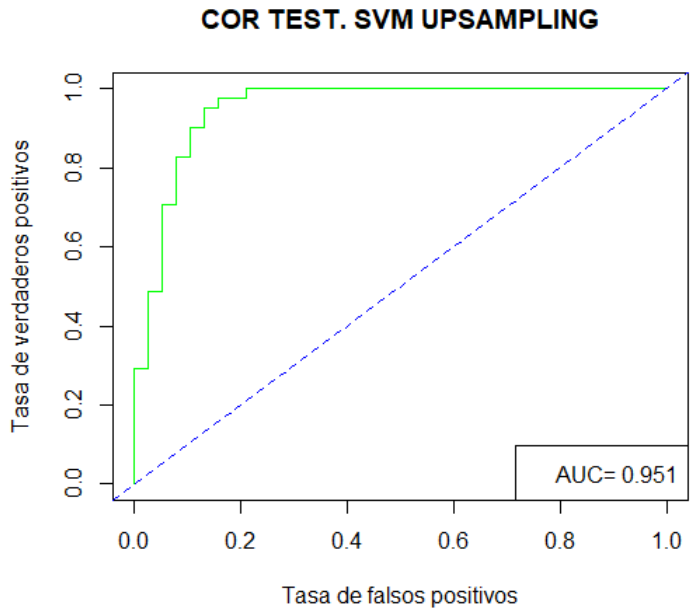

Figura 3.9: Curva ROC modelo SVM con up-sampling

Hemos extraído de la página de la federación para la Liga Iberdrola de 2021/2022, fed [3], las estadísticas correspondientes a estos equipos en esos partidos. Para predecir se ha utilizado la orden *predict*, con un objeto *tibble* que recoge esas estadísticas y se han organizado los resultados en la tabla descrita en la figura 3.10.

Equipos	Real	Prediccion
Avarca de Menorca	Ganado	Ganado
Avarca de Menorca	Ganado	Ganado
Avarca de Menorca	Ganado	Ganado
Arenal Emeve	Perdido	Ganado
Arenal Emeve	Perdido	Ganado
Arenal Emeve	Perdido	Perdido
CV Kiele Socuellamos	Ganado	Ganado
CV Kiele Socuellamos	Ganado	Ganado
CV Kiele Socuellamos	Ganado	Ganado
Feel Volley Alcobendas	Perdido	Perdido
Feel Volley Alcobendas	Perdido	Perdido
Feel Volley Alcobendas	Perdido	Perdido

Figura 3.10: Tabla de resultados reales y de predicción

En esa tabla encontramos los resultados reales y la predicción según los distintos modelos en el siguiente orden: perceptrón, vectores soporte y vectores soporte con up-sampling. Los únicos valores de predicción que no coinciden con la clasificación real son los del equipo "Arenal Emevé" al utilizarse los modelos del perceptrón y el de vectores soporte.

Estos modelos han clasificado bien, o el 100 % o un 75 % de los resultados reales. En

general, podríamos decir que son buenos ajustes pero no sería un resultado fiable puesto que lo recomendable sería evaluar los modelos para todo el conjunto de partidos de la temporada 2021/2022. Aun así , no podemos dejar de tener en cuenta que los modelos se han creado a partir de los resultados de otras temporadas. Esto puede dar lugar a grandes diferencias a la hora de construir modelos si se dan situaciones extremas, tales como lo ha sido la pandemia del COVID-19.

Con respecto a los modelos descritos en las secciones anteriores, podemos decir que interpretan bien los resultados de la Liga Iberdrola para la temporada 2020/2021. Las probabilidades de acierto han rondado el 90 %, que se puede considerar un nivel aceptable. Cabe destacar que el valor del AUC también se puede considerar bastante bueno, puesto que supera el 0.94, luego su capacidad de clasificación está más que aceptada.

Centrándonos en la comparación de los 3 modelos propuestos, podemos observar en la figura 3.11, que aquellos que tienen mayor probabilidad de acierto son el modelo del perceptrón y el de vectores soporte con la técnica de remuestreo up-sampling. Si tuviésemos que elegir alguno, nos interesaría quedarnos con este último puesto que, además el valor del área bajo la curva ROC es mayor que para la red neuronal.

Modelos	Descripción			
	Acierto	AUC	0	1
Perceptron	89.87	0.9415918	92.68	86.84
Vectores soporte	87.34	0.9557125	92.68	81.58
Vectores soporte con Upsampling	89.87	0.9505777	90.24	89.47

Figura 3.11: Tabla resumen de los resultados finales

Como línea de trabajo futura estaría el analizar los datos de otras temporadas para ver si se mantienen las tasas elevadas de acierto.

Apéndice A

Código R

Librerías utilizadas

```
library(readxl)
library(ggplot2)
library(dplyr)
library(knitr)
library(kableExtra)
library(tidyr)
```

Cargamos los datos

```
partidos2021 = read_excel("datos/Partidos_20_21.xlsx", sheet = 1,
                          range= "A2:AA266", col_names=T)
```

A.1. Estudio descriptivo de los datos

```
str(partidos2021)
```

Primero cambiamos la variable *Ganado/Perdido* a una variable dicotómica de tipo factor con valores 0 y 1 correspondientes a si el equipo ha perdido o ha ganado el partido.

```
partidos2021$`Ganado/Perdido` = as.factor(partidos2021$`Ganado/Perdido`)
str(partidos2021)
```

```
# Dimensión de los datos y resumen de las variables
dim(partidos2021)
summary(partidos2021)
```

A.2. Estudio de las variables

```
# Seleccionamos las variables para nuestro estudio
dat = partidos2021[,c(1:2,4:5,7:9,12:15,18:21,24:25,27)]
str(dat)
```

A.2.1. Gráficos y análisis de las variables

Boxplot

```
# Cambiamos el formato para hacer boxplot de las variables
partidos_boxplot1 = dat[,c(2:9)] %>%
  pivot_longer(names_to = "Variables",
               values_to = "Valores", cols=everything())
partidos_boxplot2 = dat[,c(10:17)] %>%
  pivot_longer(names_to = "Variables",
               values_to = "Valores", cols=everything())
```

```
# Representamos el gráfico boxplot para cada grupo de 8 variables
boxplot(dat[,2:9])
partidos_boxplot1 %>%
  ggplot(aes(x=`Variables` , y=Valores)) +
  geom_boxplot() +
  labs(
    title="Boxplot de 8 variables",
    x="Categorias",
    y="Valores")
```

```
boxplot(dat[,10:17])
partidos_boxplot2 %>%
  ggplot(aes(x=Variables , y=Valores)) +
  geom_boxplot() +
  labs(
    title="Boxplot de 8 variables siguientes",
    x="Categorias",
    y="Valores")
```

```
# Representación de las variables con valores 'outliers'.
boxplot(dat[c("Recep-Err","Saque-Err", "Ataque-Blo" ,"Recep-Neg")])
```

```
boxplot(dat[c("Saque-Pts","Bloqueo-Pts","Recep-Exc")])
```

Matriz varianzas y correlaciones

Analizamos ahora la matriz de varianzas/covarianzas y la matriz de correlaciones para ver qué variables pueden verse afectadas por los valores de otras.

```
var(dat[,c(2:17)])
cor = cor(dat[,2:17])
round(cor,3)
```

Para ver si hay variables explicativas que se encuentren muy correlacionadas realizamos el determinante de la matriz de correlaciones

```
det(cor)
heatmap(cor)
```

Tiene un valor muy próximo a cero luego eso significa que hay variables en las que existe una alta correlación entre ellas.

```
# Orden para obtener las variables mas correlacionadas
variables = colnames(dat[,2:17])
correlacionMax=0.85
corAltas = matrix (ncol = 3)
# Pasamos a una matriz triangular superior para que
# no nos salgan pares repetidos
cor[lower.tri(cor)] <- NA

# Bucle
for (i in 1:dim(cor)[1]){
    for (j in 1:dim(cor)[2]){
        if (is.na(cor[j,i]) && abs(cor[i,j])>correlacionMax && cor[i,j]<1){
          corAltas = rbind(corAltas,
                           c(variables[i],variables[j],round(cor[i,j],4)))
        }
    }
}
paresVariables = as.data.frame(corAltas[-1,])
colnames(paresVariables) =c("V1","V2","Correlación")

# Representación
kable(paresVariables, booktabs = T, format = "latex") %>%
kable_styling(latex_options = c( "scale_down"))
```

Esto puede indicar que existe un problema de multicolinealidad, en el que hay variables que me aportan información similar, luego esto puede dar lugar a interpretaciones erróneas. Para ello puede ser de gran ayuda un análisis de componentes principales.

A.2.2. Análisis de componentes principales

Su construcción no requiere supuesto de normalidad. No obstante, en poblaciones normales se pueden realizar tests de hipótesis y proporcionan interpretaciones útiles de los elipsoides de densidad constante.

```
cor =cor(dat[,2:17])
acp = princomp(dat[,2:17], cor=TRUE)
# Utilizamos cor = TRUE variables tipificadas ya que las escalas
# son muy distintas
summary(acp)
```

```
# Gráfico de sedimentacion
plot(acp, col="blue", main = "Componentes principales")
abline(h=mean(eigen(cor)$values), lwd=2,lty=2, col="red")
```

```
# Resumen de las componentes principales
resumen<- matrix(NA,nrow=length(acp$sdev),ncol=3)
resumen[,1]<-  acp$sdev^2 # eigen(cor)$values
resumen[,2]<- 100*resumen[,1]/sum(resumen[,1])
resumen[,3]<- cumsum(resumen[,2])
colnames(resumen)<- c("Autovalor","Porcentaje",
                      "Porcentaje acumulado")
```

Hasta la octava componente principal tenemos un 90 % de la variabilidad explicada.

Realizamos ahora el contraste de hipótesis para seleccionar el número de componentes principales (bajo hipótesis de normalidad multivariante).

```
apply(dat[,2:17],2 ,shapiro.test)
```

Se rechaza normalidad univariante para todas las variables a un nivel de signficación del 5 %. No tenemos normalidad multivariante

Coeficientes y correlaciones de las C.P

```
# Coeficientes que definen cada combinación lineal, si cogemos las 8 c.p
loadings(acp)[,1:8]
```

```
# Para calcular las correlaciones entre las variables y las componentes
cor_vc<-loadings(acp)%*%diag(acp$sdev)
cor_vc[,1:8] # Correlaciones para las 8 comp. principales
```

Intentamos interpretar las relaciones entre las componentes principales:

```
#Para ayudar a interpretar las CP 1 y 2:
plot(cor_vc[,1:2],type="n",
     main="Partidos 20/21",
     xlab="C.P. 1",ylab="C.P.2")
text(cor_vc[,1:2],labels=rownames(cor_vc),
     col="red",cex=0.6)
abline(h=0,v=0,lty=1,col="blue")
abline(v=0.5,lty=2)
abline(v=-0.5,lty=2)
abline(h=-0.5,lty=2)
```

```
#Para ayudar a interpretar las CP 4 y 5:
plot(cor_vc[,4:5],type="n",
     main="Partidos 20/21",
     xlab="C.P. 4",ylab="C.P.5")
text(cor_vc[,4:5],labels=rownames(cor_vc),
     col="red",cex=0.6)
abline(h=0,v=0,lty=1,col="blue")
abline(v=0.5,lty=2)
abline(v=-0.5,lty=2)
abline(h=-0.5,lty=2)
```

No se pueden interpretar bien las componentes principales, luego no nos son útiles para el estudio.

A.3. Modelos estadísticos

Partición entrenamiento/test

```
# Particion 70%-30%
n<- nrow(dat)
indin<- 1:n
nent<-ceiling(0.7*n)
ntest<- n-nent
# Semilla para reproducir el código
set.seed(2468)
indient<- sort(sample(indin,nent))
inditest<- setdiff(indin,indient)
datent<- dat[indient,]
dattest<- dat[inditest,]
```

A.3.1. Regla simple de Bayes

```
# Cargamos librerías
library(e1071)
modeloNB<- naiveBayes(`Ganado/Perdido` ~ ., data = datent[,2:18])

# Predicciones muestra test
preditestNB<- predict(modeloNB,dattest)

# Matriz de confusión
confutestNB<-table(dattest$`Ganado/Perdido`,preditestNB)

AciertoNB=round(100*mean(dattest$`Ganado/Perdido`==preditestNB),2)
SensEspecNB=round(100*diag(prop.table(confutestNB,1)),2)
```

```
# Cargamos librería para medir el rendimiento
library(ROCR)
probabi1<- predict(modeloNB,dattest,
                   type="raw")[,2] #Prob. ganar partido
prediobj<-prediction(probabi1,dattest$`Ganado/Perdido`)

# Gráfico curva ROC
plot(performance(prediobj, "tpr","fpr"),
     main="CoR TEST. Naive Bayes, SPAM",
     xlab="Tasa de falsos positivos", ylab="Tasa de verdaderos positivos")
abline(a=0,b=1,col="blue",lty=2)
aucNB<- as.numeric(performance(prediobj,"auc")@y.values)
legend("bottomright",legend=paste("AUC=",round(aucNB,3)))
```

Guardamos los resultados obtenidos

```
Resul=c(Acierto=AciertoNB,AUC=aucNB,SensEspecNB)
```

Otra libreria

```
# Eliminamos la librería del ejemplo anterior para que no interfiera con la
# que cargamos a continuación
detach("package:e1071")
library(naivebayes)
modeloNB2<- naive_bayes(`Ganado/Perdido` ~ ., data = datent[,2:18],
                        usekernel=TRUE,kernel ="epanechnikov",bw="nrd0",
                        usepoisson=T)

# Resumen del modelo
summary(modeloNB2)
```

```
# Predicciones muestra test
preditestNB2<- predict(modeloNB2,dattest[,2:17])

# Matriz de confusión
confutestNB2<-table(dattest$`Ganado/Perdido`,preditestNB2)

AciertoNB2=round(100*mean(dattest$`Ganado/Perdido`==preditestNB2),2)
SensEspecNB2=round(100*diag(prop.table(confutestNB2,1)),2)
```

```
probabi2<- predict(modeloNB2,dattest[,2:17],
                   type="prob")[,2] #Prob. ganado
prediobj2<-prediction(probabi2,dattest$`Ganado/Perdido`)

# Gráfico curva ROC
plot(performance(prediobj2, "tpr","fpr"),
```

```
     main="CoR TEST. Naive Bayes (2), Ganar partido",
     xlab="Tasa de falsos positivos", ylab="Tasa de verdaderos positivos")
abline(a=0,b=1,col="blue",lty=2)
aucNB2<- as.numeric(performance(prediobj2,"auc")@y.values)
legend("bottomright",legend=paste("AUC=",round(aucNB2,3)))
```

```
Resul=rbind(Resul,c(AciertoNB2,aucNB2,SensEspecNB2))
rownames(Resul)=c("Gauss","Kernel(Poisson)")
```

A.3.2. Análisis discriminante lineal

```
# Cargamos librería
library(MASS)
modeloLDA = lda(`Ganado/Perdido` ~. , datent[,2:18])
```

```
# Coeficientes del análisis discriminante lineal de Fisher en cada caso:
FLD=predict(modeloLDA)$x

#Representación
plot(FLD, col = datent[,18]$`Ganado/Perdido`)
abline(h=0,v=0,lty=3)
legend("bottomright",col=1:2,lty=1,
       legend=levels(datent$`Ganado/Perdido`))
```

```
# Predicción
preditestLDA=predict(modeloLDA,newdata=dattest[,2:18])$class

# Matriz de confusión
confutestLDA=table(Real=dat[inditest,18]$`Ganado/Perdido`,
                   Predic=preditestLDA)

AciertoLDA=round(100*mean(dattest$`Ganado/Perdido`==preditestLDA),2)
SensEspecLDA=round(100*diag(prop.table(confutestLDA,1)),2)
```

```
probabiLDA<- predict(modeloLDA,dattest[,2:17],
                   type="prob")$posterior[,2] #Prob. ganado
prediobjLDA<-prediction(probabiLDA,dattest$`Ganado/Perdido`)

# Gráfico curva ROC
plot(performance(prediobjLDA, "tpr","fpr"),
     main="CoR TEST. Analisis disc. Lineal, Ganar partido",
     xlab="Tasa de falsos positivos", ylab="Tasa de verdaderos positivos")
abline(a=0,b=1,col="blue",lty=2)
aucLDA<- as.numeric(performance(prediobjLDA,"auc")@y.values)
legend("bottomright",legend=paste("AUC=",round(aucLDA,3)))
```

```
Resul=rbind(Resul,c(AciertoLDA,aucLDA,SensEspecLDA))
rownames(Resul)=c("Gauss","Kernel(Poisson)","LDA")
```

A.3.3. Regresión Logística

```
modeloRL<- glm(`Ganado/Perdido`~.,family=binomial,data=datent[,2:18])

# Resumen del modelo
summary(modeloRL)
```

Vemos que todos los coeficientes asociados a las variables (y el termino independiente) no son significativos al 5 %, excepto las variables `Saque-Tot` (0.03686) y `Recep-Tot` (0.00955)

```
# Comprobamos si el modelo nos sirve para ajustar estos datos
# probabilidades estimadas por el modelo
prob=fitted(modeloRL)
generalhoslem::logitgof(datent$`Ganado/Perdido`, prob,g=10)
```

Nos queda un p-valor de 0.5358, luego podemos concluir que el modelo proporciona un buen ajuste.

Modelo con las variables significativas, `Saque-Tot` y `Recep-Tot`.

```
modeloRL1<- glm(`Ganado/Perdido`~.,family=binomial,
                data=datent[c("Saque-Tot","Recep-Tot","Ganado/Perdido")])
summary(modeloRL1)
```

El modelo obtenido es de la siguiente forma:

Ganado/Perdido = -3.5407 + Saque-Tot * 0.3841 + Recep-Tot * -0.3897

```
# Predicciones
preditestRL=as.numeric(predict(modeloRL1,dat[inditest,],
                               type="response")>0.5)

# Matriz de confusión
confutestRL<-table(Real=dat[inditest,18]$`Ganado/Perdido`,
                   Predic=preditestRL)

AciertoRL=round(
  100*mean(as.numeric(dattest$`Ganado/Perdido`)==(preditestRL+1)),2)
SensEspecRL=round(100*diag(prop.table(confutestRL,1)),2)
```

```
probabiRL<- predict(modeloRL1,dat[inditest,],type="response") #Prob. 1
prediobjRL<-prediction(probabiRL,dat[inditest,18])

# Gráfico curva ROC
plot(performance(prediobjRL, "tpr","fpr"),
     main="COR TEST. Regresion Logistica",
     xlab="Tasa de falsos positivos",
     ylab="Tasa de verdaderos positivos")
abline(a=0,b=1,col="blue",lty=2)
aucRL<- as.numeric(performance(prediobjRL,"auc")@y.values)
legend("bottomright",legend=paste("AUC=",round(aucRL,3)))
```

```
Resul=rbind(Resul,c(AciertoRL,aucRL,SensEspecRL))
rownames(Resul)=c("Gauss","Kernel(Poisson)","LDA","R.Logistica")
```

```
# Modificamos la codificación de los datos para poder aplicar los
# siguientes modelos
levels(dat$`Ganado/Perdido`)=c("Perdido","Ganado")
levels(dattest$`Ganado/Perdido`)=c("Perdido","Ganado")
levels(datent$`Ganado/Perdido`)=c("Perdido","Ganado")
```

A.3.4. Redes Neuronales

```
# Cargamos librerías
library(NeuralNetTools)
library(caret)

# Modelo
ctrlRD <- trainControl(method="cv",classProbs = T,
                       summaryFunction = twoClassSummary, verboseIter = F)

modeloPM <- train(`Ganado/Perdido`~ ., data = datent[,-1],
                  method = "nnet",
                  trControl = ctrlRD,
                  preProcess =c("center","scale"),
     tuneGrid=expand.grid(size=1:18,decay=c(0,0.05,0.1)))
```

```
# Resumen del modelo
modeloPM$finalModel
summary(modeloPM)
```

```
# Predicciones
preditestPM= predict(modeloPM,dattest[,-18])
```

```
# Matriz de confsuión
confutestPM=table(Real=dattest[,18]$`Ganado/Perdido`,
                  Predicción=preditestPM)
kableExtra::kable(confutestPM) %>%
 add_header_above(c("Real/Predicción"=1, " "=2))

AciertoPM=round(100*mean(dattest$`Ganado/Perdido`==preditestPM),2)
SensEspecPM=round(100*diag(prop.table(confutestPM,1)),2)
c(AciertoPM, SensEspecPM)
```

```
probabiPM= predict(modeloPM,newdata = dat[inditest,2:17] ,
                   type="prob")[,1] #Prob. ganar
prediobjPM=prediction(probabiPM,dat[inditest,18])

# Grafico curva ROC
plot(performance(prediobjPM, "tpr","fpr"),
     main="COR TEST. PM, Partidos",
     xlab="Tasa de falsos positivos",
     ylab="Tasa de verdaderos positivos")
abline(a=0,b=1,col="blue",lty=2)
aucPM= as.numeric(performance(prediobjPM,"auc")@y.values)
legend("bottomright",legend=paste("AUC=",round(aucPM,3)))
```

```
Resul=rbind(Resul,c(AciertoPM,aucPM,SensEspecPM))
rownames(Resul)=c("Gauss","Kernel(Poisson)","LDA","R.Logistica",
                  "Perceptron")
```

```
plotnet(modeloPM,pos_col='darkblue',neg_col='darkred')
neuralweights(modeloPM$finalModel) #coeficientes estimados
garson(modeloPM) #Importancia relativa de las variables de entrada
# en redes neuronales utilizando el algoritmo de Garson
olden(modeloPM) # Importancia relativa de las variables de entrada en las
# redes neuronales como la suma del producto de los pesos de conexion de
# entrada oculta, salida oculta, propuesta por Olden et al. 2004.
garson(modeloPM,bar_plot=FALSE)
sum(garson(modeloPM,bar_plot=FALSE))
```

A.3.5. Vectores soporte

```
table(datent$`Ganado/Perdido`) # datos no balanceados
```

Vamos a hacerlo con la librería caret.

```
#Definir opciones para train
ctrl <- trainControl(method="cv",classProbs=TRUE,
                     summaryFunction = twoClassSummary)

modeloSVM <- train(`Ganado/Perdido` ~ ., data = datent[,2:18],
                   method = "svmRadial",
                   trControl = ctrl,
                   preProcess = "range",
                   rangeBounds  =c(0,1),
                   tuneGrid = expand.grid(C=c(0.1,1,5,10,50),
                                          sigma=c(0.025,0.035,0.5)) )
#sigma = gamma (en nuestros apuntes)
```

```
# Predicciones
predictestSVM<- predict(modeloSVM,dattest[,2:17])

# Matrzi de confusión
confutestSVM<-table(Real=dattest$`Ganado/Perdido`,
                 Predicción=predictestSVM)
kableExtra::kable(confutestSVM) %>%
 add_header_above(c("Real/Predicción"=1, " "=2))

AciertoSVM=round(100*mean(dattest$`Ganado/Perdido`==predictestSVM),2)
SensEspecSVM=round(100*diag(prop.table(confutestSVM,1)),2)
```

```
probabiSVM= predict(modeloSVM,newdata = dat[inditest,2:17] ,
                    type="prob")[,1] #Prob. ganar
prediobjSVM=prediction(probabiSVM,dat[inditest,18])

# Grafico curva ROC
plot(performance(prediobjSVM, "tpr","fpr"),
     main="COR TEST. SVM, Partidos",
     xlab="Tasa de falsos positivos",
     ylab="Tasa de verdaderos positivos")
abline(a=0,b=1,col="blue",lty=2)
aucSVM= as.numeric(performance(prediobjSVM,"auc")@y.values)
legend("bottomright",legend=paste("AUC=",round(aucSVM,3)))
```

```
Resul=rbind(Resul,c(AciertoSVM,aucSVM,SensEspecSVM))
rownames(Resul)=c("Gauss","Kernel(Poisson)","LDA","R.Logistica",
                  "Perceptron ", "Vectores soporte")
```

Dibujar las clases

```
ggplot() +
  geom_point(data = dat, aes(x = `Ataque-Tot`,
                             y = G, color = `Ganado/Perdido`), size = 4)
```

Vamos a utilizar la técnica UPSAMPLE: se muestrea con reemplazamiento en la clase minoritaria para igualar el número de casos de la clase mayoritaria. Comparamos los dos modelos puesto que las muestras no son balanceadas por un registro.

```
# Balanceamos las clases
upSampled_train = upSample(datent[, 2:17],
                           datent$`Ganado/Perdido`)
dim(upSampled_train)
table(upSampled_train$Class)
names(upSampled_train)[17]= "Ganado/Perdido"
```

```
# Modelo
ctrl5 = trainControl(method = "cv",
                     number=5,
                    classProbs = TRUE,
                    summaryFunction = twoClassSummary)

SVMUp=train(`Ganado/Perdido` ~ .,
            data = upSampled_train,
            method = "svmRadial",
            preProcess = "range",
            rangeBounds =c(0,1),
            tuneLength=10,
            trControl = ctrl5,
            tuneGrid = expand.grid(C=c(0.1,1,5,10,50),
                                   sigma=c(0.025,0.035,0.05)),
            metric="Sens")
```

Evaluamos el modelo

```
# Predicciones
predictestUp = predict(SVMUp, dattest[,2:17])

# Matriz de confusión
confutestSVM_up<-table(Real=dattest$`Ganado/Perdido`,
                Pred=predictestUp)

kableExtra::kable(confutestSVM_up) %>%
 add_header_above(c("Real/Predicción"=1, " "=2))

AciertoSVM_up=round(100*mean(dattest$`Ganado/Perdido`==predictestUp),2)
SensEspecSVM_up=round(100*diag(prop.table(confutestSVM_up,1)),2)
```

```
probabiSVM_up= predict(SVMUp,newdata = dat[inditest,2:17] ,
                   type="prob")[,1] #Prob. ganar
prediobjSVM_up = prediction(probabiSVM_up,dat[inditest,18])
```

```
# Grafico curva ROC
plot(performance(prediobjSVM_up, "tpr","fpr"),
     main="COR TEST. SVM UPSAMPLING",
     xlab="Tasa de falsos positivos",
     ylab="Tasa de verdaderos positivos", col="green")
abline(a=0,b=1,col="blue",lty=2)
aucSVM_up = as.numeric(performance(prediobjSVM_up,"auc")@y.values)
legend("bottomright",legend=paste("AUC=",round(aucSVM_up,3)))
```

```
Resul=rbind(Resul,c(AciertoSVM_up,aucSVM_up,SensEspecSVM_up))
rownames(Resul)=c("Gauss","Kernel(Poisson)","LDA","R.Logistica",
                  "Perceptron ", "Vectores soporte",
                  "Vectores soporte con Upsampling")
```

A.3.6. Árbol de clasificación

```
# Cargamos librerías
library(rpart)
library(graphics)
modeloAB <- rpart(`Ganado/Perdido` ~ .,
                  data=datent[,2:18],method="class")
```

```
# Resumen del modelo
summary(modeloAB)
modeloAB$parms # Probabilidades a priori
```

```
# Representación del árbol
plot(modeloAB,main="Arbol de clasificacion",compress=TRUE)
text(modeloAB,col="blue")
```

```
plotcp(modeloAB) # tamaños
plotcp(modeloAB,upper = c("splits"),
       lty = 10,col=3) # numero de divisiones
```

```
# Tabla de resultados
printcp(modeloAB)
```

```
# Predicciones
predictestAB <- predict(modeloAB,type="class", dattest[,2:17])

# Matriz de confusión
confutestAB<-table(dattest$`Ganado/Perdido`,
                   predictestAB,deparse.level = 2)

AciertoAB=round(100*mean(dattest$`Ganado/Perdido`==predictestAB),2)
SensEspecAB=round(100*diag(prop.table(confutestAB,1)),2)
```

```
probabiAB= predict(modeloAB,newdata = dat[inditest,2:17] ,
                   type="prob")[,1]
prediobjAB = prediction(probabiAB,dat[inditest,18])

# Grafico curva ROC
plot(performance(prediobjAB, "tpr","fpr"),
     main="COR TEST. SVM UPSAMPLING",
     xlab="Tasa de falsos positivos",
     ylab="Tasa de verdaderos positivos")
abline(a=0,b=1,col="blue",lty=2)
aucAB = as.numeric(performance(prediobjAB,"auc")@y.values)
legend("bottomright",legend=paste("AUC=",round(aucAB,3)))
```

A.4. Resumen

```
Resul=rbind(Resul,c(AciertoAB,aucAB,SensEspecAB))
rownames(Resul)=c("Gauss","Kernel(Poisson)","LDA","R.Logistica",
                  "Perceptron ", "Vectores soporte",
                  "Vectores soporte con Upsampling",
                  "Arbol de clasificacion")
kableExtra::kable(Resul[5:7,], format ="latex") %>%
 add_header_above(c("Modelos"=1, " "=4)) %>%
  kable_styling(latex_options = c("striped", "scale_down"))
```

```
library(pROC)
ROCtestNB1 = roc(dattest$`Ganado/Perdido`, probabi1)
ROCtestNB2 = roc(dattest$`Ganado/Perdido`, probabi2)
ROCtestLDA = roc(dattest$`Ganado/Perdido`, probabiLDA)
ROCtestPM = roc(dattest$`Ganado/Perdido`, probabiPM)
ROCtestSVM = roc(dattest$`Ganado/Perdido`, probabiSVM)
ROCtestUp = roc(dattest$`Ganado/Perdido`, probabiSVM_up)
ROCtestAB = roc(dattest$`Ganado/Perdido`, probabiAB)
```

```
# Grafico curvas ROC de todos los modelos
plot(ROCtestNB1,col=1,lwd=2,main="ROC modelos")
lines(ROCtestNB2,col=2,lwd=2)
lines(ROCtestLDA,col=3,lwd=2)
lines(ROCtestPM,col=4,lwd=2)
lines(ROCtestSVM,col=5,lwd=2)
lines(ROCtestUp,col=6,lwd=2)
lines(ROCtestAB,col=7,lwd=2)
legend(x = "bottomright", legend = c("N.Bayes 1", "N.Bayes 2",
                                     "A. Discrim. Lineal",
                                     "Perceptron",
```

```
                                        "Vectores soporte", "V.sop upsampling",
                                        "Arbol clasific."), fill = 1:7, cex=0.7)
```

A.5. Predicción

Guardamos los datos de las estadísticas de cada eqipo en el partido correspondiente y realizamos las predicciones:

```
partido1AvarcaMenorca = tibble("Equipo"="Avarca Menorca", "Sets-jugados" = 5,
  "BP"=31,"G"=40,"Saque-Tot"=100, "Saque-Pts"=1,
                                "Saque-Err"=15, "Recep-Tot"=85, "Recep-Err"=6,
                                "Recep-Neg"=23, "Recep-Exc"=40,"Ataque-Tot"=145,
                                "Ataque-Err"=9, "Ataque-Blo"=12,"Ataque-Exc"=57,
                                   "Bloqueo-Red"=0,"Bloqueo-Pts"=13)
a1=predict(modeloPM, newdata = partido1AvarcaMenorca)
a2=predict(modeloSVM, newdata = partido1AvarcaMenorca)
a3=predict(SVMUp, newdata = partido1AvarcaMenorca)
```

```
partido1ArenalEmeve = tibble("Equipo"="Arenal Emeve", "Sets-jugados" = 5,
                               "BP"=34,"G"=39,"Saque-Tot"=102, "Saque-Pts"=6,
                               "Saque-Err"=17,"Recep-Tot"=85, "Recep-Err"=1,
                               "Recep-Neg"=35, "Recep-Exc"=34,"Ataque-Tot"=141,
                               "Ataque-Err"=6, "Ataque-Blo"=13,"Ataque-Exc"=46,
                                   "Bloqueo-Red"=0,"Bloqueo-Pts"=12)
b1=predict(modeloPM, newdata = partido1ArenalEmeve)
b2=predict(modeloSVM, newdata = partido1ArenalEmeve)
b3=predict(SVMUp, newdata = partido1ArenalEmeve)
```

```
partido1KS = tibble("Equipo"="CV Kiele Socuellamos", "Sets-jugados" = 3,
                    "BP"=24,"G"=33,"Saque-Tot"=73, "Saque-Pts"=4,
                    "Saque-Err"=4, "Recep-Tot"=55, "Recep-Err"=4,
                    "Recep-Neg"=16, "Recep-Exc"=15, "Ataque-Tot"=116,
                    "Ataque-Err"=5,  "Ataque-Blo"=7,"Ataque-Exc"=48,
                                   "Bloqueo-Red"=0,"Bloqueo-Pts"=5)
c1=predict(modeloPM, newdata = partido1KS)
c2=predict(modeloSVM, newdata = partido1KS)
c3=predict(SVMUp, newdata = partido1KS)
```

```
partido1FV = tibble("Equipo"="Feel Volley Alcobendas", "Sets-jugados" = 3,
                    "BP"=17,"G"=32,"Saque-Tot"=60, "Saque-Pts"=4,
                    "Saque-Err"=5,"Recep-Tot"=69, "Recep-Err"=4, "Recep-Neg"=8,
                    "Recep-Exc"=10,"Ataque-Tot"=119, "Ataque-Err"=12,
                    "Ataque-Blo"=5,"Ataque-Exc"=38,"Bloqueo-Red"=0,
                    "Bloqueo-Pts"=7)
d1=predict(modeloPM, newdata = partido1FV)
```

```
d2=predict(modeloSVM, newdata = partido1FV)
d3=predict(SVMUp, newdata = partido1FV)
```

```
# Tabla resumen con los resultados
Resultados = data.frame(Equipos = c(rep("Avarca de Menorca",3),
                                    rep("Arenal Emeve",3),
                          rep("CV Kiele Socuellamos",3),
                          rep("Feel Volley Alcobendas",3)),
                         Real = c(rep("Ganado",3),rep("Perdido",3),
                                  rep("Ganado",3),rep("Perdido",3)),
                         Prediccion = c(a1,a2,a3,b1,b2,b3,
                                        c1,c2,c3,d1,d2,d3))
kable(Resultados, format ="latex") %>%
  kable_styling(latex_options = c("striped", "scale_down")) %>%
  row_spec(4:5, color="red")
```

Bibliografía

[1] (2018). «El mister de voleibol. Lectura de plantilla de Datavolley. Parte I». https://elmisterdevoleibol.blogspot.com/2018/04/lectura-de-la-plantlla-de-datavolley.html.

[2] (2020). «Real Federación Española de voleibol. RFEVB 2020-21». https://rfevb-web.dataproject.com/CompetitionHome.aspx?ID=69.

[3] (2021). «Real Federación Española de voleibol. RFEVB 2021-22». https://rfevb-web.dataproject.com/CompetitionHome.aspx?ID=81.

[4] Akarçeşme, Cengiz (2017). «Is it possible to estimate match result in volleyball: a new prediction model». *Central European Journal of Sport Sciences and Medicine*, **19(3)**, pp. 5–17.

[5] Chawla, Nitesh V; Bowyer, Kevin W; Hall, Lawrence O y Kegelmeyer, W Philip (2002). «SMOTE: synthetic minority over-sampling technique». *Journal of artificial intelligence research*, **16**, pp. 321–357.

[6] Du, Ke-Lin y Swamy, Madisetti NS (2013). *Neural networks and statistical learning*. Springer Science & Business Media.

[7] Gabrio, Andrea (2021). «Bayesian hierarchical models for the prediction of volleyball results». *Journal of Applied Statistics*, **48(2)**, pp. 301–321.

[8] Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome H y Friedman, Jerome H (2009). *The elements of statistical learning: data mining, inference, and prediction*, tomo 2. Springer.

[9] James, Gareth; Witten, Daniela; Hastie, Trevor y Tibshirani, Robert (2021). *An introduction to statistical learning: with Applications in R*. Springer.

[10] Kuhn, M et al. (2019). «R Package Caret: Classification and Regression Training».

[11] Kuhn, Max; Johnson, Kjell et al. (2013). *Applied predictive modeling*, tomo 26. Springer.

[12] Ling, Charles X y Li, Chenghui (1998). «Data mining for direct marketing: Problems and solutions.» En: *Kdd*, tomo 98, pp. 73–79.

[13] Luque Calvo, Pedro L. (2017). *Escribir un Trabajo Fin de Estudios con R Markdown*. http://destio.us.es/calvo.

[14] —— (2019). *Cómo crear Tablas de información en R Markdown*. http://destio.us.es/calvo.

[15] Majka, Michal y Majka, Maintainer Michal (2020). «Package 'naivebayes'».

[16] Marelić, Nenad; Žufar, Goran y Omrčen, Darija (1998). «Influence of some situation-related parameters on the score in volleyball». *Kinesiology*, **30(2)**, pp. 55–65.

[17] Meyer, David; Dimitriadou, Evgenia; Hornik, Kurt; Weingessel, Andreas; Leisch, Friedrich; Chang, CC y Lin, CC (2021). «e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien [R Package Version 1.7-9]». *Comprehensive R Archive Network (CRAN)*.

[18] Quintana, Jorge (2010). «Principales técnicas de voleibol». https://www.fevochi.cl/2010/05/10/principales-tecnicas-de-voleibol/.

[19] Sanjurjo, Claudio Alberto Casal; López, José Luis Losada y Suárez, Toni Ardá (2015). «Análisis de los factores de rendimiento de las transiciones ofensivas en el fútbol de alto nivel». *Revista de Psicología del Deporte*, **24(1)**, pp. 103–110.

[20] Sing, T.; Sander, O.; Beerenwinkel, N. y Lengauer, T. (2005). «ROCR: visualizing classifier performance in R». *Bioinformatics*, **21(20)**, p. 7881. http://rocr.bioinf.mpi-sb.mpg.de.

[21] Therneau, Terry; Atkinson, Beth; Ripley, Brian y Ripley, Maintainer Brian (2015). «Package 'rpart'». *Available online: cran. ma. ic. ac. uk/web/packages/rpart/rpart. pdf (accessed on 20 April 2016)*.

[22] Wickham, Hadley y Bryan, Jennifer (2022). *readxl: Read Excel Files*. Https://readxl.tidyverse.org, https://github.com/tidyverse/readxl.

[23] Wickham, Hadley; Chang, Winston; Henry, Lionel; Pedersen, Thomas Lin; Takahashi, Kohske; Wilke, Claus; Woo, Kara; Yutani, Hiroaki y Dunnington, Dewey (2021). *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. https://CRAN.R-project.org/package=ggplot2. R package version 3.3.5.

[24] Wickham, Hadley; François, Romain; Henry, Lionel y Müller, Kirill (2021). *dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr. R package version 1.0.7.

[25] Wikipedia (2022). «Superliga Femenina de Voleibol — Wikipedia, La enciclopedia libre». https://es.wikipedia.org/w/index.php?title=Superliga_Femenina_de_Voleibol&oldid=144113124. [Internet; descargado 21-junio-2022].

[26] Xie, Yihui (2021). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. https://yihui.org/knitr/. R package version 1.36.

[27] Zhu, Hao (2021). *kableExtra: Construct Complex Table with kable and Pipe Syntax*. https://CRAN.R-project.org/package=kableExtra. R package version 1.3.4.