FACULTAD DE CIENCIAS SOCIALES Y JURÍDICAS DE ELCHE

Modelos Predictivos para Clasificaciones de La Liga: Un Enfoque Basado en Cuotas de Casas de Apuestas

Curso 2024/2025

Autor: Álvaro Martínez Ibáñez

Grado en Estadística Empresarial

Tutores: Juan Carlos Gonçalves Dosantos

Joaquín Sánchez Soriano

Elche

Junio 2025

Tabla de contenido

1. INTRODUCCIÓN

En el ámbito del aprendizaje automático, los métodos de clasificación desempeñan un papel clave al permitir asignar etiquetas o categorías a diferentes observaciones en función de sus características. Estos métodos se emplean en una amplia variedad de aplicaciones, que incluyen la clasificación de imágenes, el análisis de comportamiento de consumidores y, en este caso, la predicción de las clasificaciones de ligas deportivas. La principal finalidad en la predicción de ligas deportivas es anticipar el rendimiento de los equipos basándose en diversas características, como las cuotas de distintas casas de apuestas.

Es importante distinguir entre la clasificación supervisada, que es la base de este trabajo, y la clasificación no supervisada. En la clasificación supervisada se utilizan datos etiquetados, es decir, se conoce la categoría o clase a la que pertenece cada muestra, y se entrena un modelo con un conjunto de entrenamiento para luego validar su precisión con datos conocidos, mientras que la clasificación no supervisada no cuenta con etiquetas ni conocimiento previo, por lo que su objetivo es agrupar los datos según similitudes estadísticas o características compartidas, descubriendo patrones o categorías ocultas en el proceso [1]. Como hemos mencionado, nos centraremos en la clasificación supervisada.

A lo largo de este proyecto, se abordarán dos de las técnicas más destacadas y utilizadas en clasificación supervisada: las Máquinas de Vectores de Soporte (Support Vector Machine), a partir de ahora SVM, y los Bosques Aleatorios (Random Forest), a partir de ahora RF. Ambos métodos son conocidos por su capacidad para manejar datos complejos y, particularmente, por ser efectivos en la creación de modelos robustos. Las SVM se destacan especialmente por su habilidad para encontrar fronteras de decisión claras en problemas lineales, por su eficacia en entornos de alta dimensionalidad y por la eficiente gestión de memoria [2]. Por otro lado, los Bosques Aleatorios aportan flexibilidad y un rendimiento sobresaliente en tareas de clasificación mediante la combinación de múltiples árboles de decisión [3], aunque cabe mencionar que existen otros enfoques de clasificación que también pueden ser aplicables en casos similares, como Regresión Logística o Análisis Discriminante.

El objetivo de este trabajo es predecir la clasificación final de la temporada 2023/2024 de la liga española utilizando las cuotas proporcionadas por diferentes casas de apuestas. Los datos utilizados para el entrenamiento provienen de temporadas anteriores, lo que nos permitirá establecer patrones y relaciones que puedan reflejarse en los resultados futuros. Se originan de una página web especializada en estadísticas de fútbol, disponible en el siguiente enlace. Esta web recoge estadísticas detalladas desde la temporada 93/94 hasta la actualidad, así como las cuotas, de antes del inicio de cada encuentro, de diferentes casas de apuestas correspondientes a los tres posibles resultados: Ganador Local, Empate o Ganador Visitante. Entre las estadísticas proporcionadas se incluyen aspectos como los córneres de cada equipo, los disparos, los disparos a puerta… A lo largo del proyecto, se detallarán aspectos tanto de SVM como de RF y se evaluará cuál de estos métodos proporciona una mejor precisión en las predicciones de las clasificaciones de la liga y cuál se asemeja más a la realidad.

Este trabajo está estructurado de la siguiente manera. En el capítulo 1, introducción, a parte de lo ya mencionado arriba, incluye una explicación básica sobre qué es la clasificación supervisada y algunos de los métodos más utilizados. En el capítulo 2, dos técnicas de clasificación supervisada, se profundiza en las técnicas más relevantes (SVM y RF), se evalúa la precisión mediante la matriz de decisión, así como el estudio de algunas técnicas de validación. En el capítulo 3, aplicación, se presenta cómo se aplican estas técnicas y los resultados que se han obtenido. En el capítulo 4, conclusiones, se resumen los resultados que hemos obtenido en el estudio. En el capítulo 5, bibliografía, se incluyen las referencias utilizadas a lo largo del trabajo. Y por último, en el capítulo 6, apéndice, se presenta el código R que hemos desarrollado para el estudio realizado.

1.1 Qué es la clasificación supervisada

El aprendizaje supervisado se basa en la creación de un modelo estadístico para predecir la clase o el valor de una variable objetivo (output) empleando una o varias variables predictoras o explicativas (inputs). En función del tipo de variable objetivo, se pueden distinguir dos escenarios principales: los problemas de clasificación, cuando se predicen categorías, y los de regresión, si se trata de valores numéricos [4].

Este proceso parte de un conjunto de datos compuesto por observaciones o individuos cuyas características y valores de la variable objetivo ya se conocen. Para entrenar el modelo, se selecciona una muestra de estos datos, llamada "conjunto de entrenamiento". Una vez construido el modelo, se evalúa utilizando un conjunto de datos separado, conocido como "conjunto de validación", comparando las predicciones generadas con los valores reales para medir la precisión del modelo.

Es importante tener en cuenta que uno de los principales desafíos en el aprendizaje supervisado es evitar el overfitting (sobreajuste) y el underfitting (subajuste). El overfitting ocurre cuando el modelo se ajusta demasiado a los datos de entrenamiento, afectando a su capacidad de generar nuevos datos. El underfitting se da cuando el modelo no es lo suficientemente complejo para captar las relaciones subyacentes en los datos.

Cualquier método que ajuste un modelo para clasificar a un individuo en una clase, a partir de su conjunto de variables predictoras, se considera una herramienta de clasificación supervisada. Para comparar los diferentes modelos y seleccionar el que ofrezca un mejor rendimiento, se emplean técnicas de validación. Las técnicas de validación son métodos que permiten evaluar el rendimiento de un modelo dividiendo los datos en conjuntos de entrenamiento y prueba, asegurando su capacidad.

Finalmente, la evaluación del modelo elegido se realiza mediante un criterio que permite medir su precisión y determinar qué tan bien podría clasificar datos futuros.

1.2 Métodos de clasificación supervisada

Los métodos de clasificación supervisada varían principalmente según las suposiciones realizadas sobre los datos y el tipo de algoritmo desarrollado para aproximar el clasificador. No existe un único método de clasificación supervisada que garantice los mejores resultados en todos los escenarios. Por este motivo, en esta sección se ofrecerá una breve descripción de varios métodos de clasificación supervisada, junto con las condiciones necesarias para su aplicación. Más adelante, se profundizará en SVM y RF, que constituyen el tema central de este trabajo.

Algunos de los métodos de clasificación supervisada son: Regresión Logística, Análisis Discriminante, K-Nearest Neighbors, Árboles de Clasificación, SVM y RF.

Regresión logística:

La regresión logística se utiliza para modelar la probabilidad de que un individuo i pertenezca a una de dos posibles clases, asignándole la clase con mayor probabilidad. El cálculo exacto de esta probabilidad se conoce como Clasificador de Bayes, y la regresión logística es una aproximación a este clasificador.

Este método realiza la aproximación utilizando la función logística, lo que permite interpretarla como una probabilidad. El objetivo es encontrar la función logística que mejor se ajuste a la muestra, para la cual se conoce de antemano el valor de las variables predictoras y la clase correspondiente de cada individuo, como se mencionó previamente [5].

Análisis discriminante:

El Análisis Discriminante permite identificar las llamadas "funciones discriminantes", las cuales dividen a los individuos en t regiones según el conjunto de variables explicativas. Un individuo se asigna a la clase t si sus valores de x se encuentran dentro de la región correspondiente.

El número de funciones discriminantes se determina tomando el menor valor entre K - 1 (donde K es el número de categorías de la variable dependiente) y p (el número de variables independientes) [6].

K-Nearest Neighbors:

Este método clasifica a un individuo basándose en sus características observadas. Para hacerlo, busca los datos más similares a ese individuo dentro del conjunto de entrenamiento. Una vez identificados un número determinado de vecinos cercanos, se observa la clase a la que pertenece cada uno de ellos. La probabilidad de que el individuo

pertenezca a una determinada clase se calcula como la proporción de vecinos en el grupo que pertenecen a esa clase. Finalmente, se asigna al individuo la clase con la mayor proporción dentro de ese grupo [7].

Árboles de Clasificación:

Los árboles de clasificación son modelos visuales formados por nodos, ramas y hojas. Los nodos indican las pruebas o decisiones realizadas sobre los datos, las ramas reflejan las posibles respuestas a esas pruebas y las hojas corresponden a los resultados o predicciones finales. La creación del árbol comienza con un nodo raíz y avanza de manera recursiva dividiéndose en ramas hasta que todos los nodos sean terminales o puntos de decisión.

El objetivo principal en cada nodo es dividir el conjunto de datos de una forma que reduzca al máximo el error entre la categoría real y la categoría predicha. Para ello, se selecciona la variable que mejor permita separar los datos en subconjuntos homogéneos utilizando una medida de impureza como criterio de decisión. Entre otras medidas, encontramos el índice de Gini. Esta medida mide la probabilidad de clasificar de manera incorrecta una observación si se asigna de manera aleatoria. Su fórmula es:

$$1 - \sum_{i=1}^{k} p_i^2$$

El árbol selecciona en cada nodo la variable que consiga la menor impureza ponderada para dividir los datos. De este modo, el modelo busca optimizar la clasificación, y para evitar problemas de sobreajuste se establecen criterios de parada, como el número mínimo de observaciones por nodo o la poda del árbol para simplificarlo [8].

SVM

El objetivo de este método es encontrar el hiperplano que separa de manera óptima las clases, maximizando la distancia entre los puntos más cercanos a dicho plano. Dependiendo de la flexibilidad, se puede usar un margen rígido (donde todas las

observaciones están correctamente clasificadas) o un margen blando (donde algunas pueden estar dentro del margen o mal clasificadas). Cuando la separación no es lineal, se emplean kernels, como las polinómicas o gaussianas, para definir fronteras no lineales. Aunque originalmente es para problemas binarios, existen extensiones para manejar múltiples clases [9].

RF

Los RF son colecciones de árboles de clasificación que utilizan distintos subconjuntos del conjunto de entrenamiento original. Estos subconjuntos se crean mediante un proceso de muestreo. A cada individuo se le asigna la clase que se le haya asignado con mayor frecuencia entre los árboles. Normalmente, no se utilizan todas las variables para construir los árboles. Para decidir qué variables usar, se mide la importancia de cada variable explicativa en la predicción [10].

2. DOS TÉCNICAS DE CLASIFICACIÓN SUPERVISADA

En este apartado se desarrollarán dos de las principales técnicas más utilizadas en la clasificación supervisada: SVM y RF. Además, explicaremos su funcionamiento, así como su teoría matemática que hay detrás de cada una de ellas. No solo eso, sino que también abordaremos tanto la matriz de confusión como algunas técnicas de validación que permiten evaluar el rendimiento de los modelos.

Dado un conjunto de individuos Ω, cada uno con un vector de variables explicativas: $x_i = (x_{i1}, x_{i2},\ldots, x_{ik})$ y una variable clase: $y_i \in \{C_1, C_2,\ldots, C_z\}$ el objetivo es predecir la clase y de un nuevo individuo x_{new}.

Ω representa el conjunto de individuos en estudio, donde cada individuo está caracterizado por un conjunto de variables explicativas $X = (X_1, X_2, \ldots, X_k)$. Estas variables pueden ser cuantitativas o categóricas y describen distintas propiedades o características del individuo. Además, cada individuo tiene una variable de clase Y, que

es categórica y puede tomar valores en el conjunto $\{C_1, C_2 \dots C_z\}$, representando las diferentes categorías o clases a las que puede pertenecer. El objetivo es utilizar la información de las variables explicativas para predecir la clase Y de un nuevo individuo x_{new}.

2.1 SVM

SVM busca encontrar un hiperplano que separe las distintas clases de datos en un espacio de características, de manera que se maximice el margen entre las clases. El objetivo principal de SVM es encontrar ese hiperplano óptimo que maximiza la distancia entre las clases, lo que garantiza una buena generalización del modelo cuando se encuentra con nuevos datos.

HIPERPLANO:

En un espacio de características R^n, un hiperplano es un subespacio de dimensión n-1 que divide el espacio en dos regiones. En SVM, este hiperplano separa las diferentes clases de datos. El modelo SVM busca el hiperplano óptimo que maximice el margen entre las clases.

Matemáticamente, un hiperplano se puede representar como: $w^T x + b = 0$, donde w es el vector de pesos, x es el vector de características de un punto de datos y b, el sesgo.

El objetivo de SVM es encontrar los valores de w y b que permitan separar las clases de datos de forma óptima.

MARGEN DURO:

El margen duro se refiere a la situación en la que los datos son linealmente separables, es decir, podemos encontrar un hiperplano que separe perfectamente las clases sin ningún error. En este caso, el margen es la distancia entre el hiperplano y los puntos de datos más cercanos de cada clase, conocidos como vectores de soporte. Se define el margen como:

$$\frac{1}{|w|}$$

donde $|w|$ es la norma euclídea del vector de pesos w, que representa su magnitud y se calcula como:

$$|w| = \sqrt{\{w_1^2 + w_2^2 + \cdots + w_k^2\}}$$

El objetivo de SVM con margen duro es maximizar este margen, asegurando que todos los puntos estén correctamente clasificados sin errores. En términos matemáticos, el problema de optimización sería:

$$\min_{w,b} \frac{1}{2}|w|^2$$

Aunque la función objetivo depende únicamente del vector w, la optimización también se realiza respecto al sesgo b, ya que este interviene en las restricciones del modelo, que aseguran la correcta clasificación de los datos:

$$y_i(w^T x_i + b) \geq 1, \forall i,$$

donde y_i es la etiqueta de clase (+1 o -1) del punto x_i.

Esta restricción asegura que los puntos estén correctamente clasificados y separados por el hiperplano, es decir:

- Si $y_i = +1$, la restricción se convierte en $w^T x_i + b \geq 1$, lo que significa que el punto x_i debe estar en el lado correcto del hiperplano y fuera de la franja del margen.
- Si $y_i = -1$, la restricción se convierte en $w^T x_i + b \leq -1$, garantizando que el punto x_i esté en el otro lado del hiperplano y también fuera de la franja del margen.

Esta restricción garantiza que los puntos más cercanos al hiperplano estén justo en los bordes del margen, es decir, a una distancia de 1 por ambos lados. Ningún punto puede estar dentro del margen o en el lado incorrecto del hiperplano, ya que resultaría en errores de clasificación.

El margen duro, tal y como se puede observar en la Figura 1, solo es aplicable cuando los datos son perfectamente separables, lo cual rara vez es el caso en aplicaciones del mundo real.

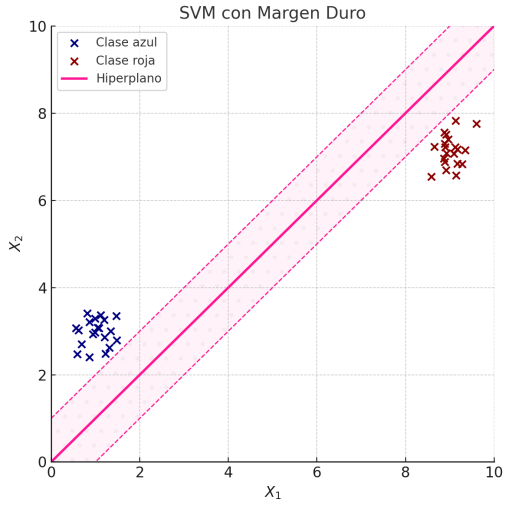

Figura 1: Margen duro

MARGEN BLANDO:

En la práctica, los datos raramente son perfectamente separables. En estos casos, se utiliza el margen blando, que permite algunos errores de clasificación para encontrar un equilibrio entre la complejidad del modelo y el ajuste de los datos.

El margen blando introduce variables de relajación ξ_i para permitir que algunos puntos se encuentren en el lado incorrecto del margen. Esto se traduce en un compromiso entre la maximización del margen y la minimización de los errores de clasificación.

El problema de optimización para el margen blando es:

$$\min_{w,b,\xi} \frac{1}{2}|w|^2 + T\sum_{i=1}^{N} \xi_i$$

sujeto a:

(1) $y_i(w^T x_i + b) \geq 1 - \xi_i, \qquad \forall i, i = 1 \ldots \mathrm{N}$

Esta restricción garantiza que para cada punto de datos x_i:

- Si $y_i = +1$, la restricción se convierte en $w^T x_i + b \geq 1 - \xi_i$, es decir, el punto debe estar en el lado correcto del margen.
- Si $y_i = -1$, la restricción se convierte en $w^T x_i + b \leq -1 + \xi_i$, lo que asegura que el punto esté en el lado correcto del margen.

Las variables de relajación ξ_i permiten que algunos puntos se encuentren en el margen o incluso en el lado equivocado del hiperplano. La variable ξ_i mide cuánto de lejos está el punto de estar mal clasificado; cuanto mayor sea ξ_i , más "permitido" está el error en ese punto.

(2) $\xi_i \geq 0, \quad \forall i, i = 1 \ldots N$

Esta restricción asegura que las variables de relajación ξ_i sean no negativas. No se puede permitir que una variable de relajación sea negativa, lo que implicaría que un error de clasificación es menor que cero.

La T es un hiperparámetro de regularización que controla el equilibrio entre maximizar el margen y minimizar los errores de clasificación. Afecta directamente al comportamiento del modelo y cómo se ajusta a los datos de entrenamiento:

- Si T es grande: El modelo penaliza fuertemente los errores de clasificación, lo que significa que intenta clasificar correctamente el mayor número de puntos posibles. Esto lleva a un ajuste más estricto de los datos de entrenamiento, reduciendo el margen y aumentando el riesgo de sobreajuste.
- Si T es pequeño: El modelo es más tolerante a los errores de clasificación, priorizando un margen más amplio. Esto puede ser beneficioso cuando se busca que el modelo generalice mejor a datos no vistos, permitiendo cierta flexibilidad en la clasificación.

Este enfoque permite que el modelo tenga mayor capacidad de generalización, tolerando algunos errores sin sobreajustarse a los datos de entrenamiento, tal y como se puede apreciar en la figura 2.

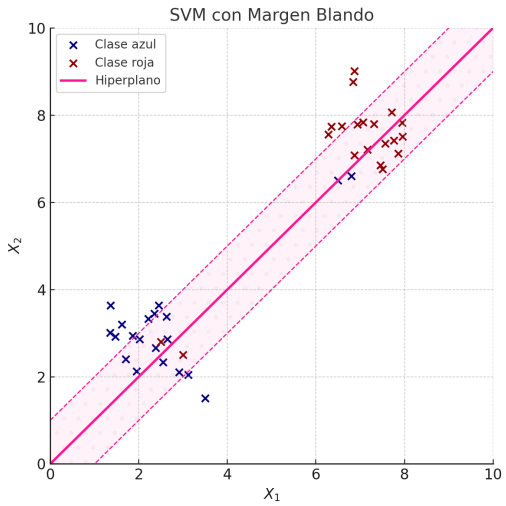

Figura 2: Margen blando

KERNELS:

Uno de los principales avances de SVM es el uso de kernels, que permiten aplicar SVM a problemas no lineales. El kernel es una función que toma los datos de entrada y los mapea a un espacio de características de mayor dimensión, donde el problema podría ser linealmente separable.

La función kernel toma dos puntos x_i y x_j y calcula el producto interior en el espacio transformado sin necesidad de calcular explícitamente las coordenadas en ese espacio. Esto se conoce como el "truco" del kernel. [11]

Algunos de los más comunes son:

- Lineal:

$$K(x_i, x_j) = x_i^T x_j$$

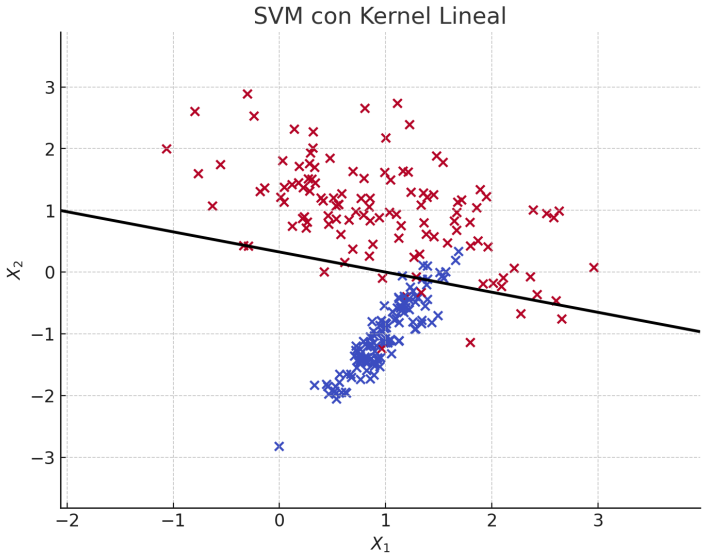

Figura 3: Kernel lineal

Este es simplemente el producto interior de los vectores originales, y se usa cuando los datos ya son linealmente separables.

- <u>Polinómico:</u>

$$K(x_i, x_j) = (x_i^T x_j + c)^d$$

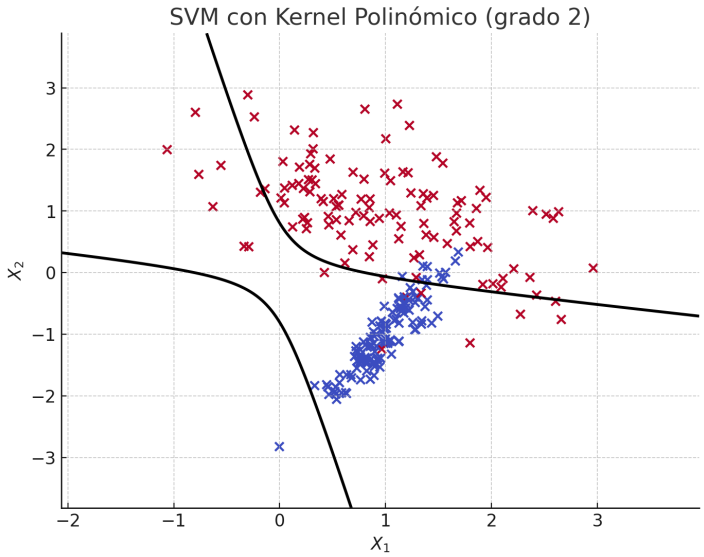

Figura 4: Kernel polinómico grado 2

donde c es una constante y d es el grado del polinomio. Este kernel mapea los datos a un espacio de mayor dimensión de forma no lineal.

- Gaussiano:

$$K(x_i, x_j) = exp\left(-\frac{2}{\sigma^2}|x_i - x_j|^2\right)$$

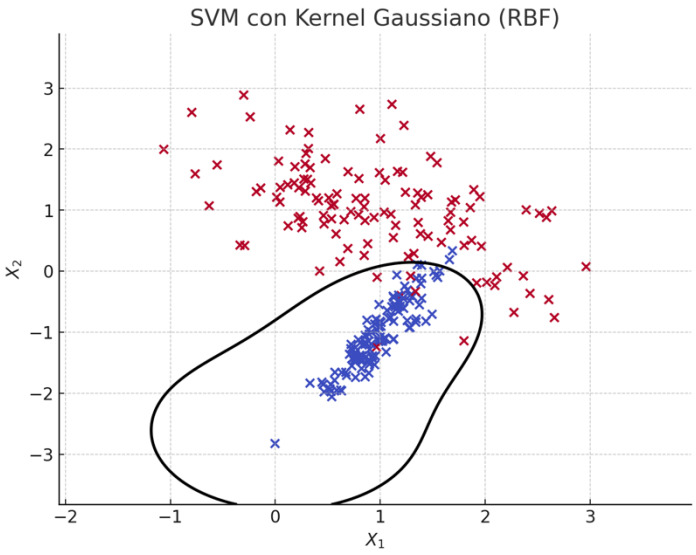

Figura 5: Kernel Gaussiano

Este kernel es muy popular y mapea los datos a un espacio infinito de características. Es especialmente útil cuando no se conoce la forma de la frontera de decisión, ya que puede capturar relaciones no lineales complejas.

- Sigmoide:

$$K(x_i, x_j) = tanh\left(\alpha x_i^T x_j + c\right)$$

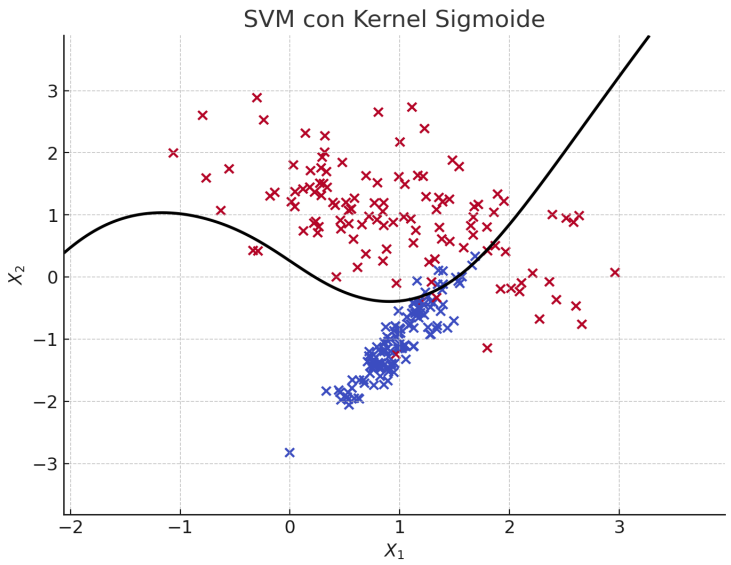

Figura 6: Kernel Sigmoide

Este kernel se inspira en las funciones de activación en redes neuronales y también permite modelar relaciones no lineales.

2.1.1.1 Clasificación con más de dos clases

Hasta el momento, nos hemos limitado al caso de clasificación binaria, es decir, clasificación en un entorno de dos clases. En la literatura, se han propuesto varias formas en el caso de Z clases, siendo las dos más populares: uno contra uno (OvO) y uno contra todos (OvA). Ambos enfoques son para cuando hay Z > 2.

UNO CONTRA UNO

Este enfoque consiste en entrenar $\left(\frac{Z(Z-1)}{2}\right)$ clasificadores, donde cada uno compara un par de clases de manera independiente. Por ejemplo, una de estas SVM podría comparar la z-ésima clase (+1) con la z'-ésima clase (-1). Para clasificar una nueva observación, se evalúa con cada uno de los $\left(\frac{Z(Z-1)}{2}\right)$ clasificadores y se cuenta cuántas veces se le asigna a cada una de las Z clases. La predicción final será la clase con mayor número de asignaciones en estas comparaciones por pares [9].

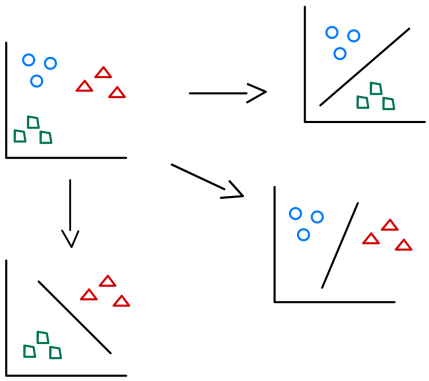

Figura 7: Uno contra uno

Ejemplo:

Supongamos que queremos clasificar una observación en tres clases: “Círculo”, “Cuadrado” y “Triángulo”. Al estar en el enfoque uno contra uno, se entrenarán tres clasificadores de SVM:

- Círculo vs Cuadrado
- Círculo vs Triángulo
- Cuadrado vs Triángulo

Dada una nueva observación, se evalúa en cada uno de los tres clasificadores SVM. Supongamos que se clasifican de la siguiente manera:

- Círculo vs Cuadrado → Círculo
- Círculo vs Triángulo → Círculo
- Cuadrado vs Triángulo → Triángulo

En la figura 7, vemos que se ha clasificado dos veces como “Círculo”, una vez como “Triángulo” y ninguna como “Cuadrado”. Dado que se ha clasificado como “Círculo” un mayor número de veces, la nueva observación se le asignaría la categoría “Círculo”.

En el caso de que el primero hubiese sido “´Círculo”, el segundo “Triángulo” y el tercero “Cuadrado”, habría un triple empate, siendo esta opción posible. Esta situación muestra una de las limitaciones del método, ya que no hay una clase claramente ganadora.

UNO CONTRA TODOS

Este enfoque entrena Z clasificadores SVM, uno por cada clase. Cada clasificador intenta distinguir una clase concreta del resto, etiquetando a la clase objetivo como +1 y a todas las demás como -1.

Matemáticamente, para cada clase z, se adjunta un clasificador con parámetros

$w^{(z)}$ y $b^{(z)}$, que resuelven un problema SVM clásico en el que:

- Las observaciones de la clase z son etiquetadas con $y_i = +1$
- Las observaciones del resto de clases tienen $y_i = -1$

Dada una nueva observación x, se calculan las puntuaciones:

$$s^{(z)}(x) = w^{(z)} \cdot x + b^{(z)}$$

La observación se asigna a la clase z con mayor valor de $s^{(z)}(x)$, es decir, aquella para la que el clasificador tiene mayor "confianza" en que la observación pertenece a dicha clase.

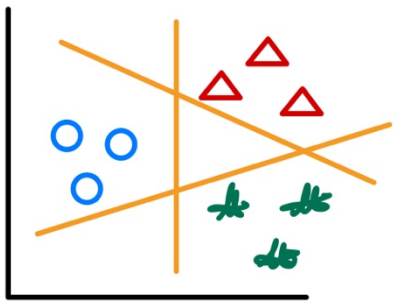

Figura 8: Uno contra todos

Ejemplo:

Supongamos que tenemos un problema de clasificación con 3 clases:

- Clase 1: Círculo
- Clase 2: Triángulo
- Clase 3: Estrella

Tal y como se aprecia en la *Figura 8,* vamos a construir 3 clasificadores SVM, uno para cada clase, con el objetivo de comparar esa clase frente a las demás

- SVM para Círculo: Compara la clase "Círculo" (+1) con las demás (-1)
- SVM para Triángulo: Compara la clase "Triángulo" (+1) con las demás (-1)
- SVM para Estrella: Compara la clase "Estrella" (+1) con las demás (-1)

Supongamos que, después de entrenar los SVM, los parámetros son:

- $w^{(1)} = (2, 3)$ $b^{(1)} = 1$
- $w^{(2)} = (0.5, 2)$ $b^{(2)} = -1$
- $w^{(3)} = (-1, 1)$ $b^{(3)} = 0$

Supongamos ahora que tenemos una observación de prueba donde x = (1,1):

- Círculo: $s^{(1)} = 2*1 + 3*1 + 1 = 6$
- Triángulo: $s^{(2)} = 0.5*1 + 2*1 -1 = 1.5$
- Estrella: $s^{(3)} = -1*1 + 1*1 + 0 = 0$

Comparamos los resultados y, al tratarse de la clase "Círculo" la que más puntuación ha obtenido, asignamos la observación prueba a la clase "Círculo".

2.2 RF

Antes de abordar el concepto de RF, se va a explicar matemáticamente los Árboles de clasificación, ya que los RF se componen de una colección de árboles.

ÁRBOLES DE CLASIFICACIÓN:

El espacio de predictores, es decir, el conjunto de todos los posibles valores que pueden tomar las variables explicativas $X = (X_1, X_2, \dots, X_k)$, se divide en J regiones disjuntas $R_1, R_2, \dots, R_J$

Cada una de estas regiones R_J agrupa individuos con características similares en función de las variables explicativas y se asocia a una clase específica dentro del conjunto $\{1, 2, \dots, Z\}$. De este modo, cualquier nueva observación x_{new} se clasifica según la región

a la que pertenezca. Si $x_{new} \in R_J$, entonces su clase predicha sería $\widehat{y_{new}} = t$, donde t es la clase asignada a esa región, tal y como se aprecia en la *Figura 9.*

La división del espacio de predictores en regiones se logra mediante un árbol de decisión, donde cada nodo aplica una regla basada en una variable explicativa. Estas reglas pueden ser de la forma:

- Para variables cuantitativas: condiciones como $X_i \leq cota \quad y \quad X_i > cota$
- Para variables categóricas: comparaciones como $X_i = t \quad o \quad X_i \neq t$, donde t representa un valor específico de la categoría

El árbol se construye de manera que en cada división se busca maximizar la pureza de las regiones resultantes, es decir, minimizar la mezcla de clases dentro de cada región. A medida que se realizan más divisiones, el espacio de predictores queda particionado en subconjuntos más homogéneos, facilitando la clasificación de nuevos individuos.

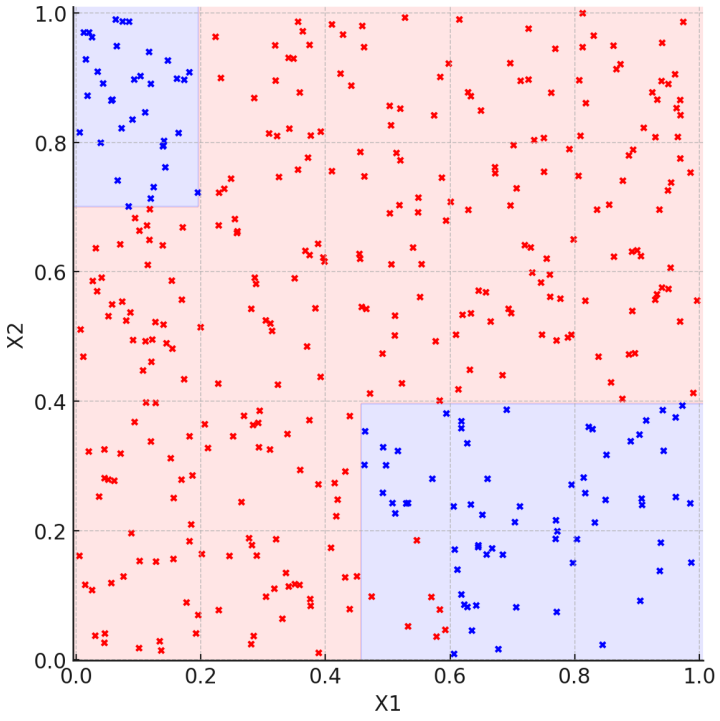

Figura 9: Árboles de clasificación

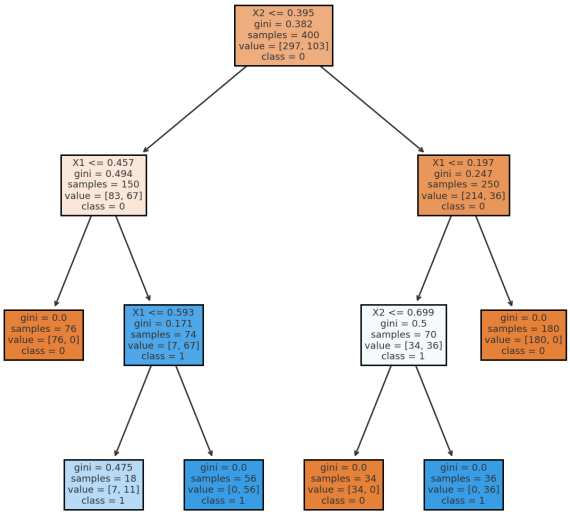

Figura 10: Predicción árbol de clasificación [12]

CONSTRUCCIÓN:

Regla de División: se trata de dividir un nodo en dos. El objetivo es disminuir la impureza del nodo padre, tal y como se aprecia en la *Figura 10*.

Sea p_{jt} la proporción de observaciones del conjunto de entrenamiento que pertenecen a la región j y son de la clase t. Las medidas de pureza más comunes son:

- Índice de Gini: un valor pequeño indica que un nodo contiene mayoritariamente observaciones de una única clase

$$G = \sum_{t=1}^{z} p_t(1 - p_t)$$

- Entropía cruzada: un valor pequeño indica que el nodo es puro

$$Entropía = -\sum_{t=1}^{z} p_t \log(p_t)$$

Elige la variable y la cota/clase tal que maximiza la ganancia de pureza.

Criterio de Parada: Ramificando mejoramos la pureza de los hijos, pero disminuimos el número de observaciones de estos nodos. El criterio de parada común es que el número de observaciones en cada nodo debe estar por encima de un mínimo.

Asignación de la Clase: Hay que asignar una clase t a cada región R_j. La clase de asignación se utiliza para clasificar futuras observaciones. La asignación usual de clase es que la t es la clase mayoritaria en el nodo hoja R_j.

PREDICCIONES:

Una vez el árbol está construido, es decir, las regiones R_j, j = 1,…,J están definidas y tienen una clase $t \in \{1, \dots, z\}$ asignada, una observación nueva x_{new} se clasifica: si $x_{new} \in R_j$, entonces $\widehat{y_{new}} = t$

Cada región R_j contiene un subconjunto del conjunto de entrenamiento (x_i, y_i), $i = 1, \dots, n$. La clase asignada t es la clase más común entre estos puntos.

Para cada clase $t \in \{1, \dots, z\}$, podemos estimar la probabilidad de que se dé la clase m dado que el vector está en la región R_j, esto es, $P(Y) = \left(X \in R_j\right)$, como:

$$\hat{p}_k(R_j) = \frac{1}{n_j} \sum_{x_i \in R_j} \mathrm{I}(y_i = t)$$

donde $\hat{p}_k(R_j)$ es la proporción estimada, n_j es el número total de puntos en R_j, y_i es la clase del punto x_i y $\mathrm{I}(y_i = t)$ vale 1 si $y_i = t$ y 0 en caso contrario.

La proporción de puntos en la región R_j que son de clase t.

Podemos expresar la clase asignada en la región R_j como:

$$t = \arg \max_{k=1,\dots,m} \hat{p}_k\,(R_j)$$

Una vez explicada la parte matemática de Árboles de clasificación, ya podemos explicar en qué consiste y profundizar más sobre RF.

RF:

Random Forest es una técnica de aprendizaje supervisado que genera múltiples árboles de decisión (*Figura 11)* sobre un conjunto de datos de entrenamiento. Los resultados de estos árboles se combinan para obtener un modelo más robusto que los árboles individuales. Cada árbol se construye utilizando un subconjunto aleatorio de observaciones del conjunto de datos, seleccionado mediante la técnica Bootstrap, un método de muestreo con reemplazo que estima el rendimiento de un modelo combinando el error de entrenamiento y el error "out-of-bag" (OOB), sin necesidad de dividir los datos en conjuntos separados.

El ensamblaje de los árboles se realiza combinando las salidas de todos los árboles, mediante el promedio en el caso de predicciones numéricas o el conteo de votos para predicciones categóricas. Esto mejora la precisión y generalización del modelo. Los árboles de decisión pueden ser inestables debido a que tienden a sobreajustarse a pequeñas variaciones en los datos, pero el enfoque de Random Forest, al combinar múltiples árboles, mitiga este problema, mejorando la estabilidad del modelo.

Cuando se tiene una nueva observación x_{new} se clasifica esta observación utilizando cada uno de los árboles de la colección, y se elige la clase mayoritaria como la predicción final. Este enfoque de colección de árboles permite mejorar el rendimiento del modelo, ya que la diversidad generada por los diferentes subconjuntos de entrenamiento aumenta la robustez y reduce el sobreajuste.

Cuando se construyen los árboles, generalmente cada uno solo considera un subconjunto de las variables disponibles. Normalmente, el número de variables consideradas en cada división de un árbol es la raíz cuadrada del número total de variables ($\sqrt{p}$), donde p es el número total de variables del conjunto de datos. Este enfoque mejora la diversidad entre los árboles y reduce la correlación entre ellos.

Si se utilizan todos los predictores disponibles para construir los árboles, el algoritmo RF se convierte en Bagging, una técnica que combina modelos entrenados con datos

muestreados con reemplazo. Este método sigue el principio de construir múltiples árboles usando subconjuntos de datos distintos para cada árbol y combinar sus predicciones, lo que mejora la estabilidad y precisión del modelo.

Además, RF realiza una selección implícita de características en el proceso de construcción de los árboles, lo que permite identificar las variables más relevantes para la tarea de clasificación. La importancia de cada variable se mide según su contribución al rendimiento del modelo, lo que facilita la interpretación del modelo y puede simplificarlo al enfocarse en los predictores más significativos.

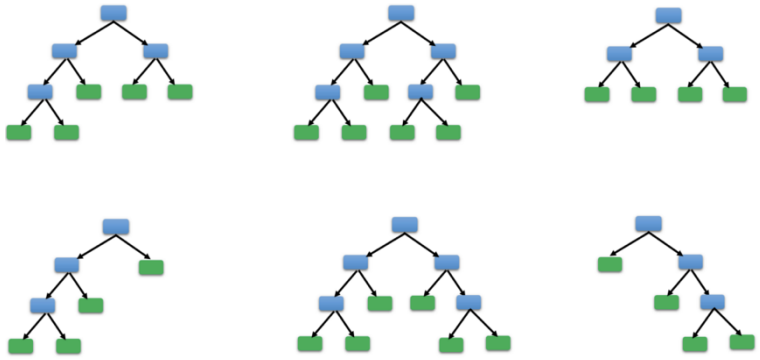

Figura 11: Ejemplo de un RF [13]

2.3 Matriz de Confusión

La forma de medir lo bien que ha funcionado un modelo más popular es la precisión, es decir, la proporción de individuos correctamente clasificados. Además de esta medida, también puede ser interesante conocer la proporción de observaciones cuya clase ha acertado el modelo en cada una de las distintas clases, ya que, algunas veces, el clasificador funciona mejor con una clase concreta.

Para poder saber esto, se mide la sensibilidad y la especificidad, las cuales se pueden deducir mediante una matriz de confusión: una tabla de doble entrada que compara el número de individuos predichos en cada clase con el número real de individuos en cada clase. Además, se puede conocer la tasa de falsos negativos (proporción de casos positivos que el modelo no detectó y clasificó erróneamente como negativos), así como la tasa de

falsos positivos (proporción de casos negativos que fueron incorrectamente clasificados como positivos).

Si la forma de una matriz de confusión es la que aparece en la siguiente tabla, entonces las fórmulas de las medidas que se desean obtener son las siguientes:

	Predicho Y = 0	Predicho Y = 0	Total observado
Observado Y = 0	$n_{0,0}$	$n_{0,1}$	n_0
Observado Y = 1	$n_{1,0}$	$n_{1,1}$	n_1
Total predicho	$\widehat{n_0}$	$\widehat{n_1}$	n

Donde:

- $n_{0,0}$ = número de casos correctamente clasificados como clase 0 (verdaderos negativos).
- $n_{0,1}$ = número de casos observados como clase 0 pero predichos como clase 1 (falsos positivos).
- $n_{1,0}$ = número de casos observados como clase 1 pero predichos como clase 0 (falsos negativos).
- $n_{1,1}$ = número de casos correctamente clasificados como clase 1 (verdaderos positivos).
- n_0 = total de casos observados como clase 0.
- n_1 = total de casos observados como clase 1.
- $\widehat{n_0}$ = total de casos predichos como clase 0.
- $\widehat{n_1}$ = total de casos predichos como clase 1.
- n = total de observaciones.

A partir de estos valores, se pueden calcular las siguientes métricas de evaluación del modelo:

- Precisión: % de predichos correctamente: $(n_{0,0} + n_{1,1}) / n$
- Sensibilidad: % de 1 observados que han sido correctamente identificados: $n_{1,1} / n_1$
- Especificidad: % de 0 observados que han sido correctamente identificados: $n_{0,0} / n_0$

- Tasa de falsos positivos: % de 0 observados predichos como 1: $n_{0,1} / n_0$
- Tasa de falsos negativos: % de 1 observados predichos como 0: $n_{1,0} / n_1$

2.4 Técnicas de Validación

Existen diversas técnicas para validar los métodos de clasificación, como las técnicas de validación cruzada. Algunas de ellos son:

Holdout:

Consiste en dividir un conjunto de datos en dos subconjuntos: entrenamiento y prueba. El conjunto de entrenamiento se utiliza para ajustar el modelo, mientras que el conjunto de prueba permite evaluar su rendimiento en datos nuevos mediante la matriz de confusión. Los datos se dividen en, por lo general, 70% para el entrenamiento, mientras que el 30% restante para la prueba. También es común que los datos se dividan en dos tercios y un tercio del total, pudiendo variar según la variedad y característica de los datos. La evaluación del clasificador depende de cómo se dividan los datos [7].

Validación cruzada Q-Fold:

Es una mejora del modelo anterior, ya que proporciona una estimación más estable del rendimiento del modelo. Los datos se dividen en q grupos (dependiendo del tamaño de la muestra). En cada iteración, el modelo se entrena utilizando $q-1$ de estos subconjuntos y se evalúa con el subconjunto restante. Este proceso se repite a veces, alternando el conjunto de prueba en cada iteración, de manera que cada subconjunto actúe como prueba una vez. Finalmente, se calcula el desempeño promedio del modelo a partir de todas las evaluaciones obtenidas [8].

Bootstrap:

Se utiliza para estimar el rendimiento de un modelo generando diversos subconjuntos de datos a partir de una muestra original. Para ello, se extraen muestras de tamaño n de manera aleatoria y con reemplazo, lo que significa que algunas observaciones pueden

repetirse mientras que otras pueden quedar excluidas. Cada una de estas muestras se emplea para entrenar el modelo, evaluando su desempeño con los datos no seleccionados, conocidos como "out-of-bag" (OOB). Este proceso se repite varias veces y se calcula el rendimiento final promediando las evaluaciones obtenidas. Un estimador comúnmente utilizado en este método es el estimador 0.632, que combina el error en los datos de entrenamiento y el error OOB de la siguiente manera:

$$Error_{Bootstrap} = 0.632 \times Error_{OOB} + 0.368 \times Error_{Entrenamiento}$$

El método Bootstrap resulta útil en conjuntos de datos pequeños, ya que permite generar múltiples estimaciones sin necesidad de dividir los datos en conjuntos de entrenamiento y prueba. Sin embargo, su efectividad depende de la representatividad de los datos originales [9].

3. APLICACIÓN

3.1 Presentación de los datos

El enfoque aplicado en este trabajo consiste en predecir los resultados de los partidos de la temporada 2023/2024 de LaLiga española utilizando como variables predictoras las cuotas ofrecidas por distintas casas de apuestas. Este procedimiento se ha replicado de forma paralela para las cuatro casas de apuestas analizadas: Bet365, Betway, William Hill y VC Bet. El objetivo es evaluar si existe alguna diferencia sustancial en la precisión de las predicciones según la casa de apuestas empleada como fuente de datos.

Los datos fueron extraídos de la siguiente web el día 6 de mayo de 2025. Esta plataforma recopila estadísticas históricas de partidos junto con cuotas de apuestas deportivas ofrecidas por diferentes casas como Bet365, Betway, William Hill y VC Bet.

Cada archivo contiene información sobre los equipos locales y visitantes de cada encuentro, junto con las cuotas previas al partido para los tres posibles resultados: victoria local, empate o victoria visitante. En particular, se han utilizado las siguientes variables

proporcionadas por cada casa de apuestas, correspondientes a las cuotas ofrecidas antes del partido para los tres resultados posibles:

- B365H, B365D, B365A: cuotas ofrecidas por Bet365 para los resultados de Victoria Local (H), Empate (D) y Victoria Visitante (A), respectivamente.
- BWH, BWD, BWA: cuotas de Betway, bajo la misma codificación: Victoria Local, Empate, Victoria Visitante.
- WHH, WHD, WHA: cuotas de William Hill para los tres posibles desenlaces del partido.
- VCH, VCD, VCA: cuotas de la casa VC Bet, también organizadas en el mismo orden (H, D, A).

Estas variables reflejan la percepción previa del mercado de apuestas sobre la probabilidad de que ocurra cada uno de los resultados, y se han utilizado como variables predictoras para estimar el resultado final de los partidos.

- Resultado: variable categórica con valores {-1, 0, 1}, que representa Victoria Visitante, Empate y Victoria Local, respectivamente.

Para el entrenamiento de los modelos se han combinado los datos de las tres primeras temporadas (2020/2021 a 2022/2023), mientras que la temporada 2023/2024 se ha reservado como conjunto de validación. Antes de entrenar los modelos, se ha llevado a cabo una conversión de todas las columnas de cuotas al tipo numérico y se ha asegurado que la variable Resultado sea tratada como factor para la clasificación.

Es importante destacar que, en aquellos partidos donde una de las casas de apuestas no ofrecía una determinada cuota, se ha imputado su valor calculando la media de las cuotas equivalentes de las otras tres casas para ese mismo partido.

El objetivo es predecir el resultado de los partidos de la temporada 2023/2024 a partir de las cuotas de cada casa de apuestas, utilizando diferentes algoritmos de clasificación como RF y SVM. Posteriormente, con las predicciones obtenidas, se simula una clasificación final para la temporada, asignando puntos a los equipos según las reglas estándar del fútbol (3 puntos por victoria, 1 por empate, 0 por derrota). La clasificación real de la temporada 2023/2024 es la que se muestra en la *Figura 12.*

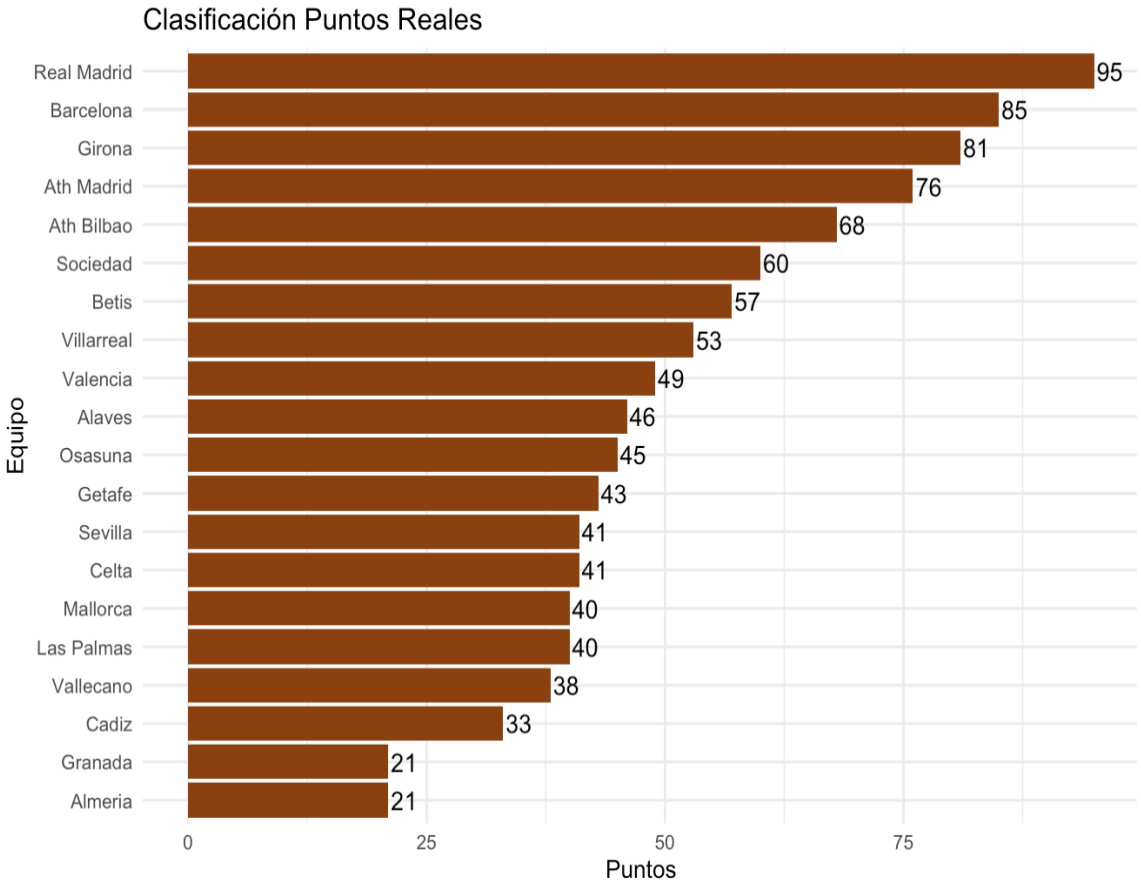

Figura 12: Clasificación real 2023/2024

3.2 RF

Cada modelo ha sido entrenado con 500 árboles (ntree = 500), considerando las 3 variables de entrada (mtry = 3) y fijando una semilla aleatoria mediante set.seed(100) para garantizar la reproducibilidad de los resultados. Una vez obtenidas las predicciones de los partidos de la temporada 2023/2024, se ha calculado la clasificación final predicha asignando: 3 puntos al equipo ganador, 1 punto a cada equipo en caso de empate y 0 puntos al equipo perdedor.

Para la casa de apuestas Bet365, se utilizaron como variables predictoras las cuotas B365H, B365D y B365A. La clasificación resultante generada por el modelo Random Forest ha sido:

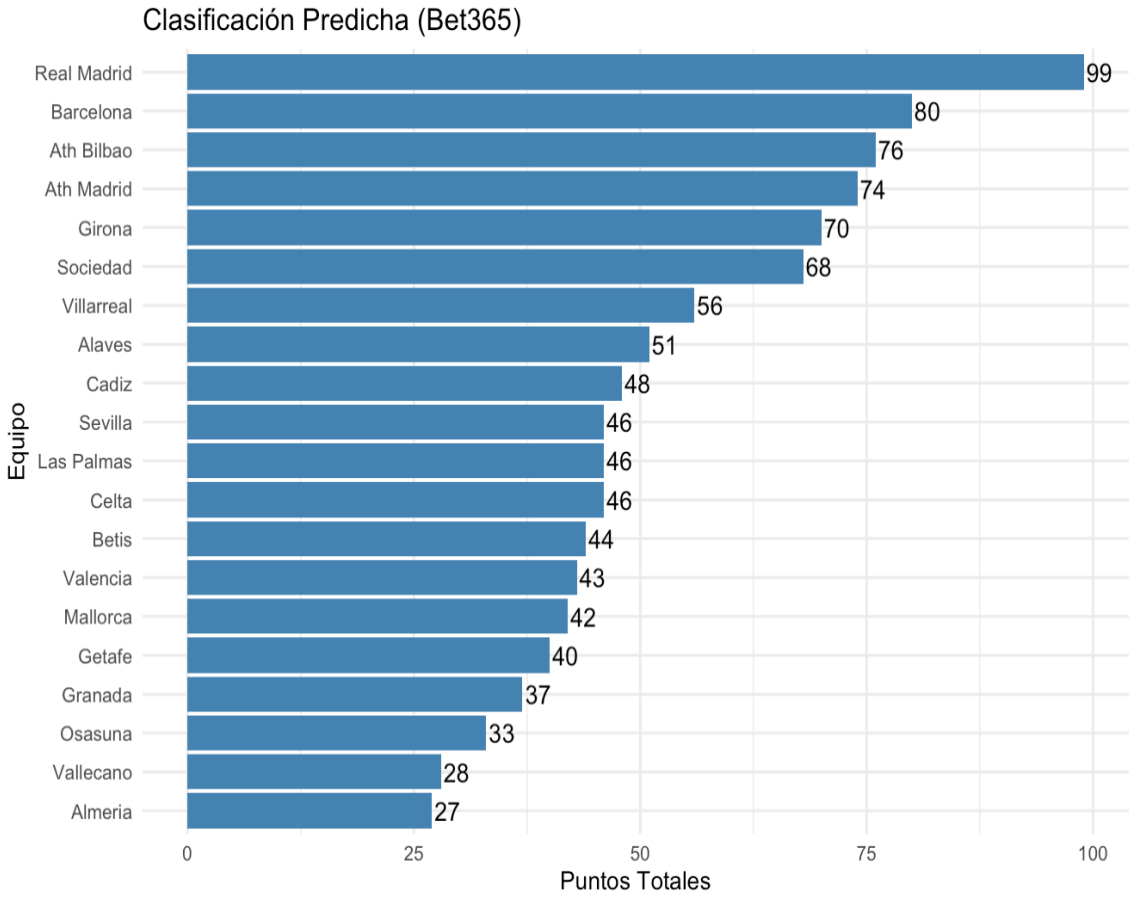

Figura 13: Clasificación Bet365 RF

En el caso de BetWay, se usaron las variables BWH, BWD y BWA. La clasificación predicha por el modelo ha sido:

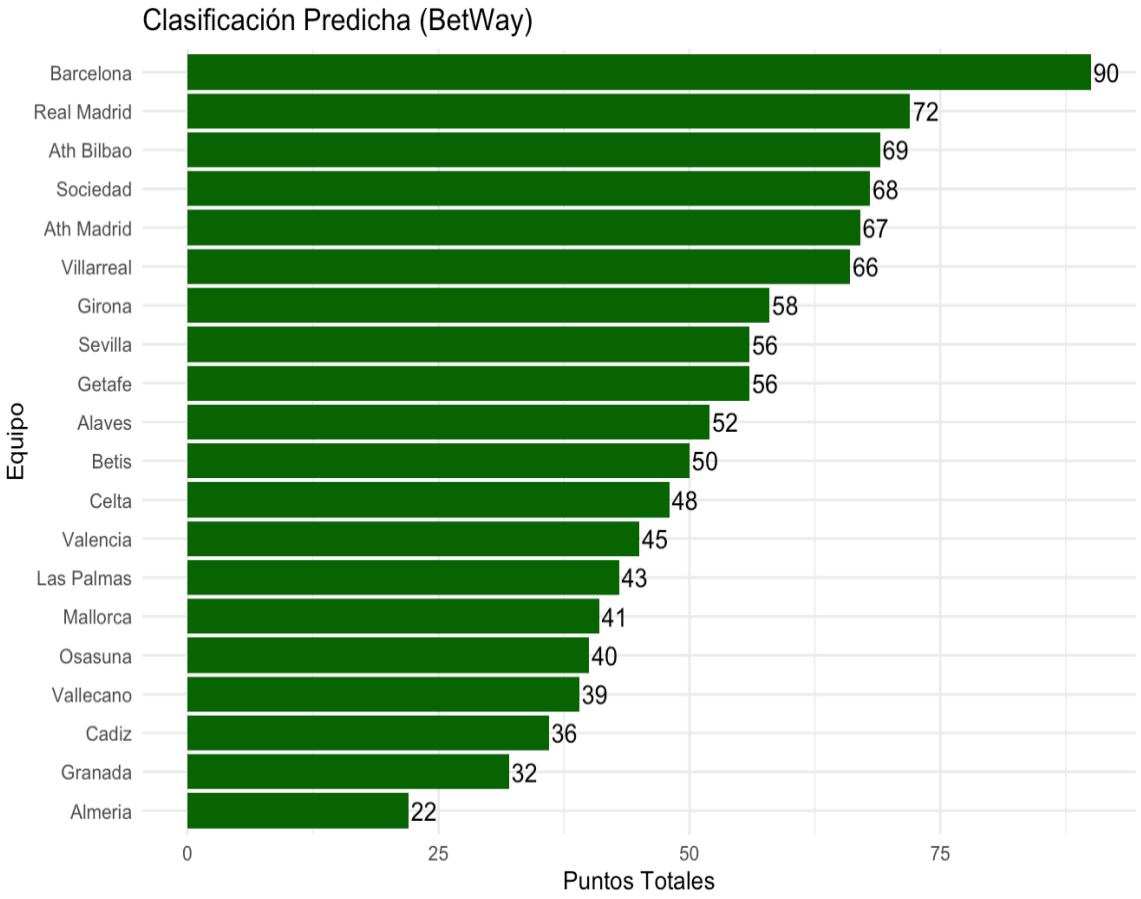

Figura 14: Clasificación BetWay RF

Para William Hill se emplearon las variables WHH, WHD y WHA. La clasificación obtenida mediante el modelo RF ha sido:

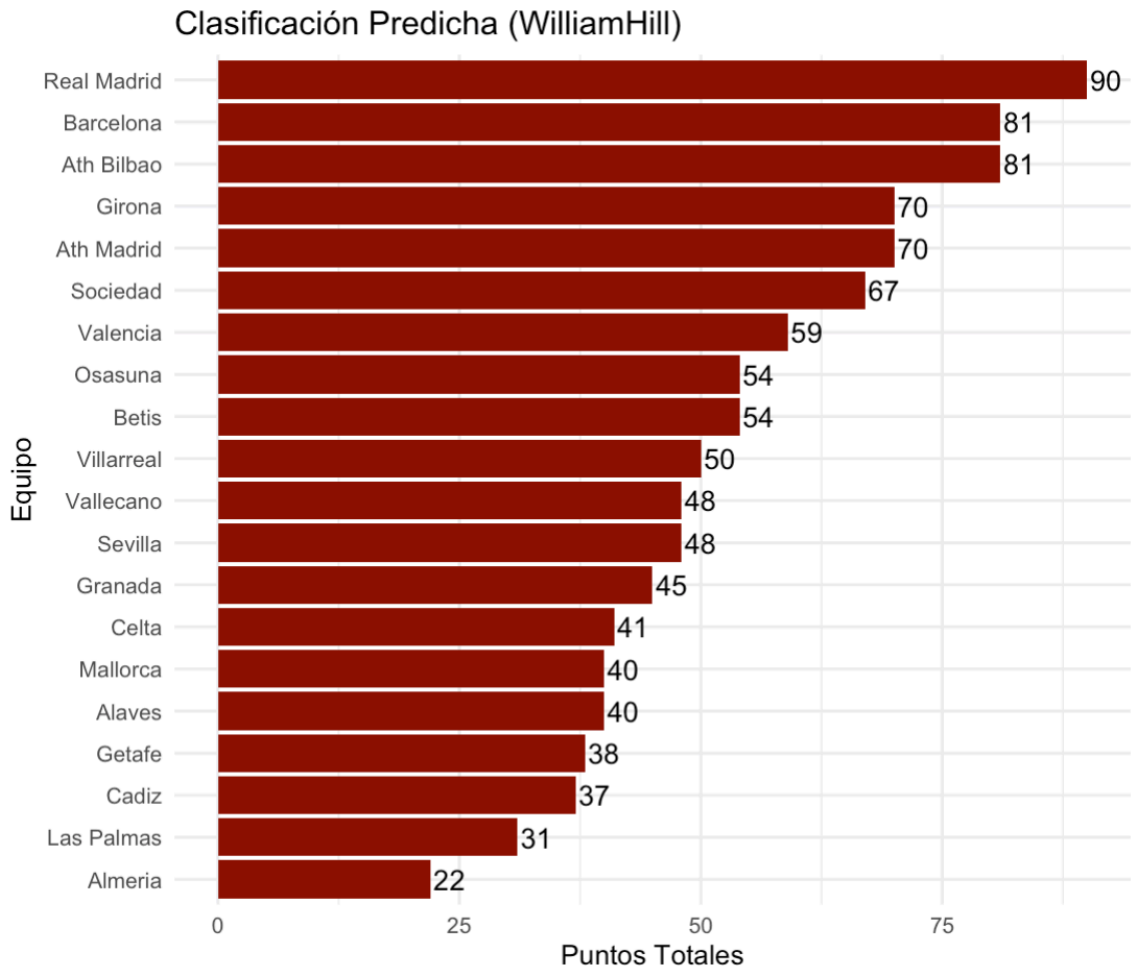

Figura 15: Clasificación WilliamHill RF

Por último, para la casa VC Bet se utilizaron VCH, VCD y VCA como variables predictoras. La clasificación estimada ha sido:

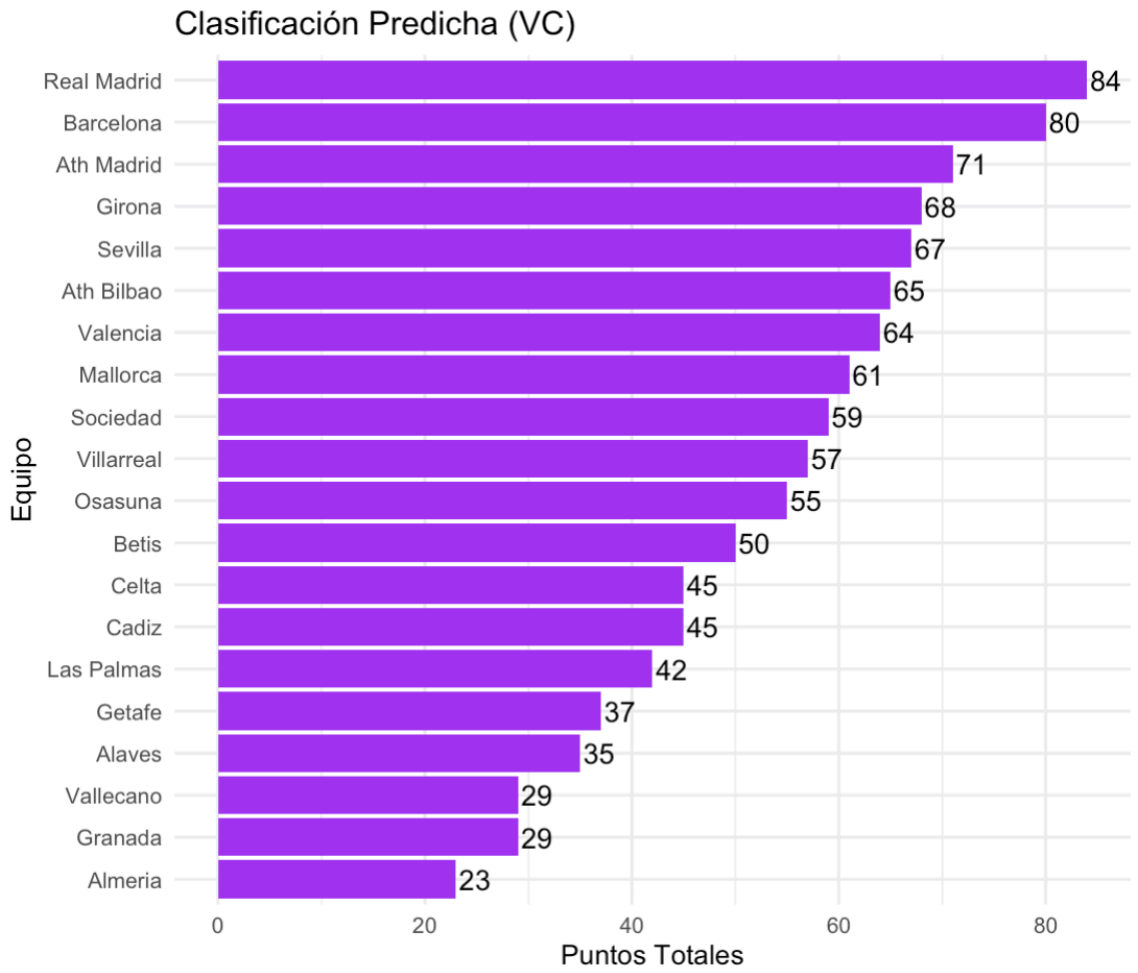

Figura 16: Clasificación VC Bet RF

La tabla resumen con los valores de MAE (Mean Absolute Error) tanto en puntos como en posiciones, obtenidos al comparar la clasificación predicha por el modelo RF con la clasificación real de la temporada 2023/2024 para cada casa de apuestas, ha sido:

Casa de Apuestas	MAE en Puntos	MAE en Posiciones
Bet365	7.25	2.70
BetWay	7.95	1.90
WilliamHill	6.85	2.10
VC	8.75	2.45

- Bet365: predice con notable precisión la clasificación final (MAE de solo 2.70 posiciones), aunque se desvió más en los puntos (MAE 7.25). Lo más destacable es que Girona superó todas las expectativas, acabando 3.º con 11 puntos más de lo esperado.

- BetWay: acierta bastante bien el orden final de los equipos (MAE en posiciones de 1.90), aunque sobreestimó a Barcelona y subestimó claramente a Girona, que terminó 3.º con 81 puntos frente a los 58 que se le asignaban. En puntos, su error medio fue de 7.95.
- WiliamHill: casa más precisa en puntos (MAE 6.85), aunque se desvió algo más en posiciones (MAE 2.1); acertó plenamente con el liderato del Real Madrid, pero sobrevaloró a Osasuna y Granada, y no anticipó el gran rendimiento de Girona (que acabó 3.º con 81 puntos frente a los 70 predichos).
- VC: casa menos precisa en puntos (MAE 8.75) y también se desvió notablemente en posiciones (MAE 2.45), especialmente al sobrevalorar a Sevilla y Mallorca y no anticipar el rendimiento de la Real Sociedad ni el buen desempeño del Girona, que acabó 3.º con 81 puntos frente a los 68 que le atribuían.

3.3 SVM

El segundo enfoque utilizado para la predicción de los resultados de la temporada 2023/2024 ha sido el algoritmo SVM, aplicado mediante dos estrategias para problemas multiclase: uno contra uno (OvO) y uno contra todos (OvA).
Al igual que con RF, los modelos se han aplicado de forma independiente para las casas de apuestas, utilizando las cuotas previas al partido como variables predictoras.

Para ambos enfoques, los modelos se han entrenado con los datos de las temporadas 2020/2021, 2021/2022 y 2022/2023. Las predicciones se han realizado sobre los partidos de la temporada 2023/2024 y, al igual que en el caso anterior, se ha asignado 3 puntos por victoria, 1 por empate y 0 por derrota para simular la clasificación final. Se ha fijado la semilla aleatoria con set.seed(100) para garantizar la reproducibilidad.

3.3.1 SVM OvO

En el enfoque OvO, se entrenan clasificadores binarios para cada par de clases posibles.
A continuación, se presentan las clasificaciones predichas para cada casa de apuestas mediante este enfoque:

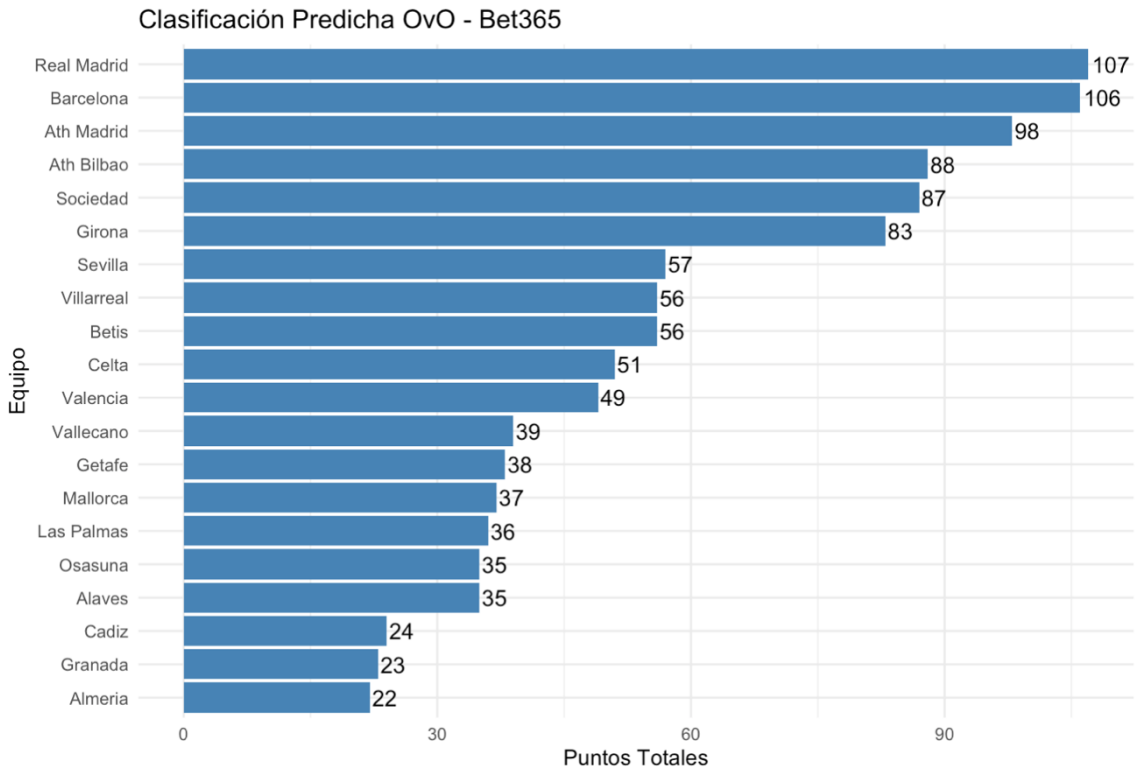

Figura 17: Clasificación Bet365 OvO

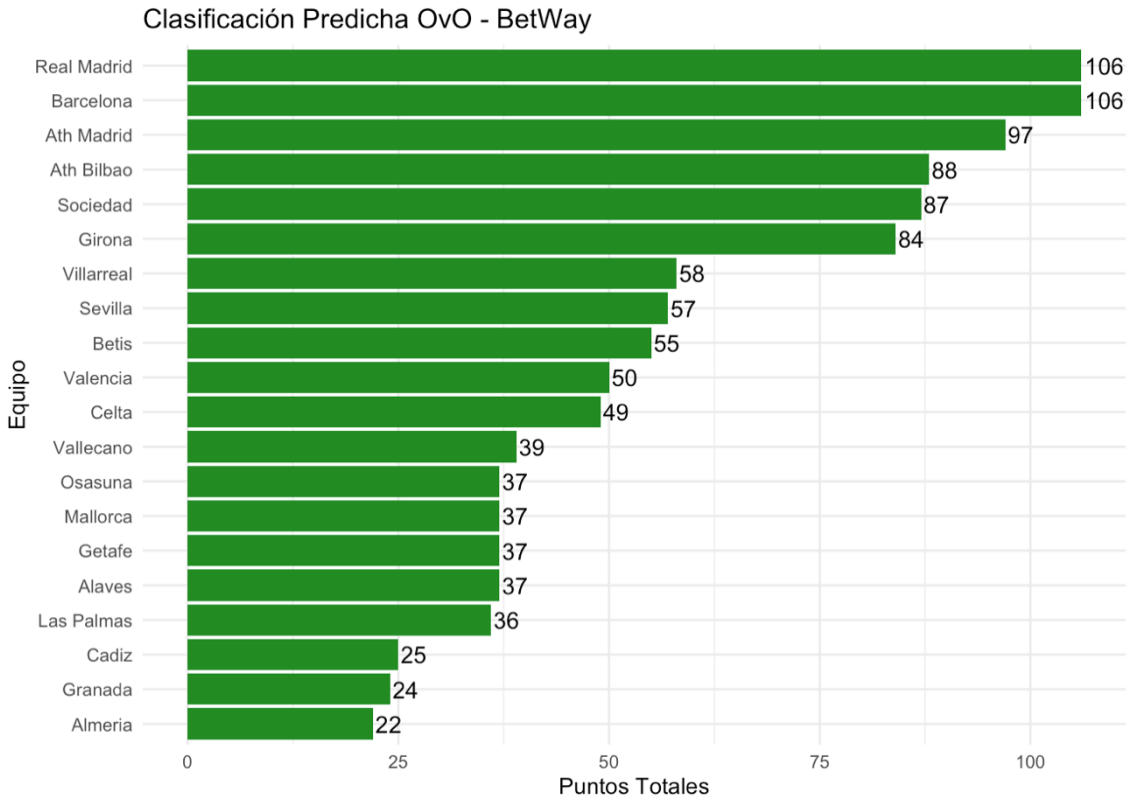

Figura 18: Clasificación BetWay OvO

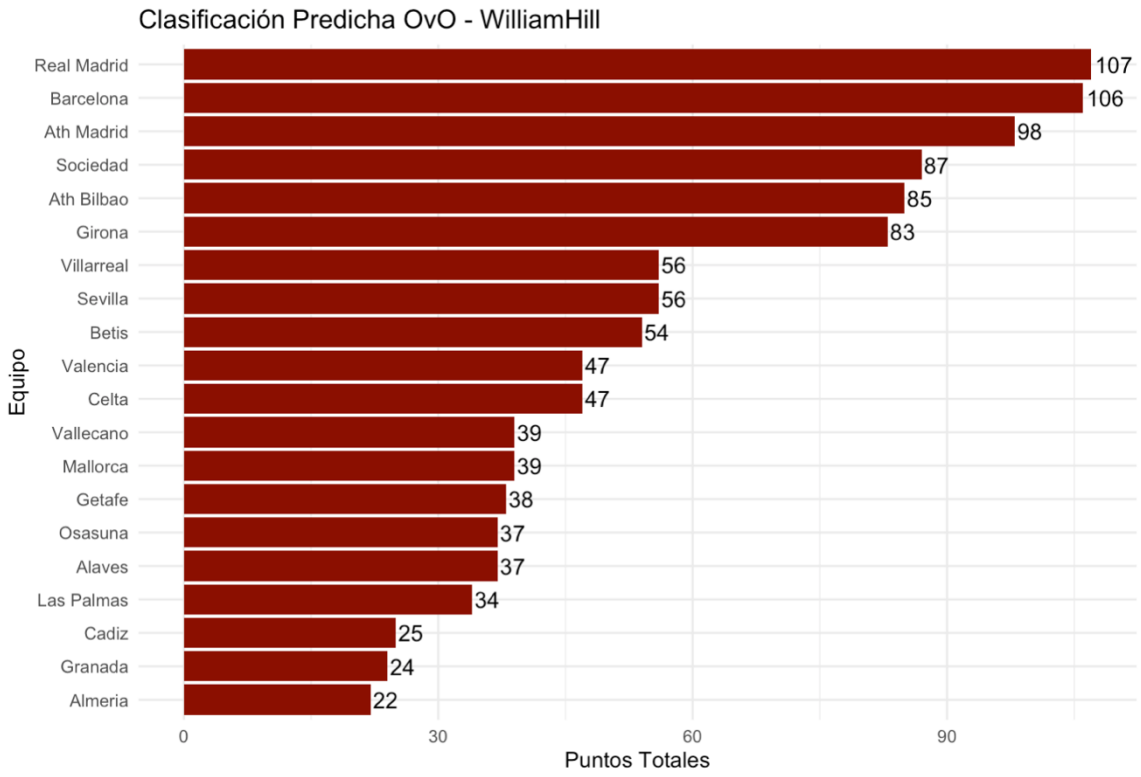

Figura 19: Clasificación WilliamHill OvO

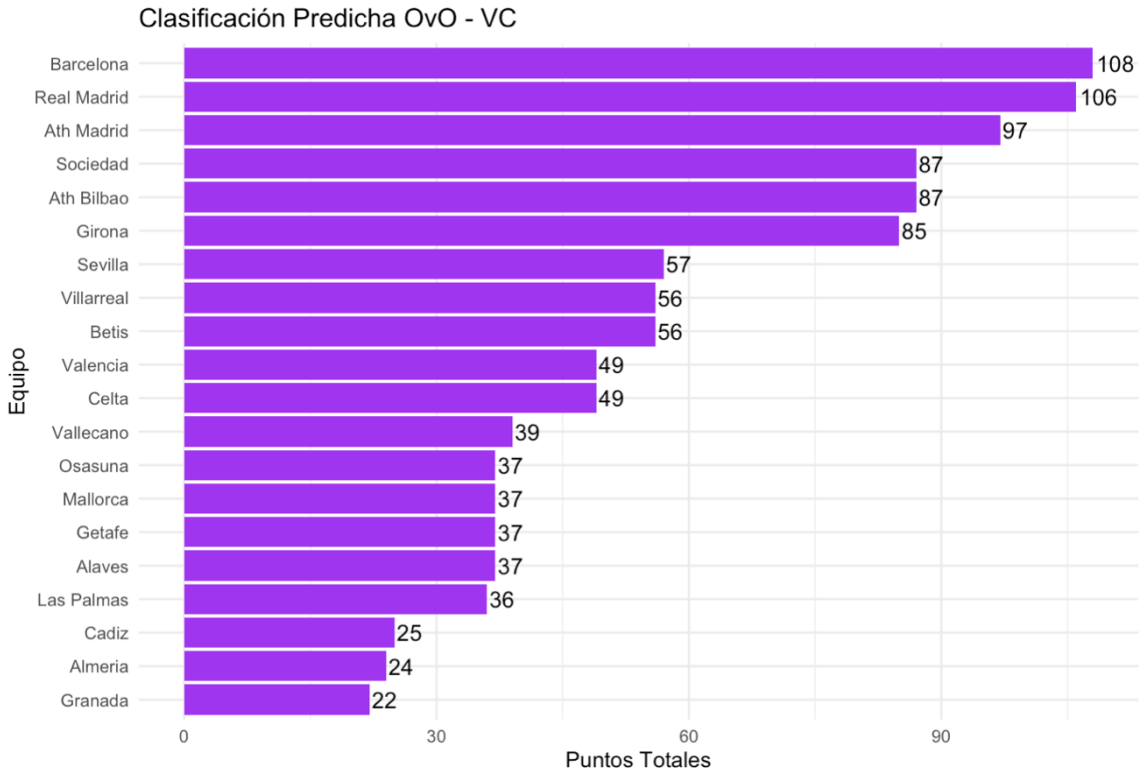

Figura 20: Clasificación VC Bet OvO

La tabla resumen con los valores de MAE (Mean Absolute Error) tanto en puntos como en posiciones, obtenidos al comparar la clasificación predicha por el modelo SVM OvO con la clasificación real de la temporada 2023/2024 para cada casa de apuestas, ha sido:

Casa de Apuestas	MAE en Puntos	MAE en Posiciones
Bet365	9	1.85
BetWay	8.9	1.7
WilliamHill	8.6	2.05
VC	8.8	1.8

- Bet365: Acierta el podio y el descenso, pero infló en exceso los puntos del top 2 (más de 100). Su MAE fue el peor en puntos (9), aunque bastante sólido en posiciones (1.85).
- BetWay: Fue la más precisa en posiciones (MAE 1.70), con un buen ajuste general del orden, aunque sobrevaloró a Madrid y Barça con 106 puntos cada uno.
- WiliamHill: mejor predicción en puntos (MAE 8.6), con buena aproximación en la parte alta y baja de la tabla, aunque falló más en el orden (MAE posiciones 2.05)
- VC: Se equivoca al colocar como campeón al Barcelona con 108 puntos, sobrevalorando su rendimiento real (85 pts y 2.º), y volvió a dejar a Girona fuera del podio. Aun así, ajustó bien la zona media-baja, con un MAE moderado en puntos (8.8) y bastante preciso en posiciones (1.8).

3.3.2 SVM OvA

En el enfoque OvA, se entrena un clasificador binario para cada clase, donde esta se enfrenta al resto. A continuación, se muestran las clasificaciones obtenidas con este modelo para cada casa de apuestas:

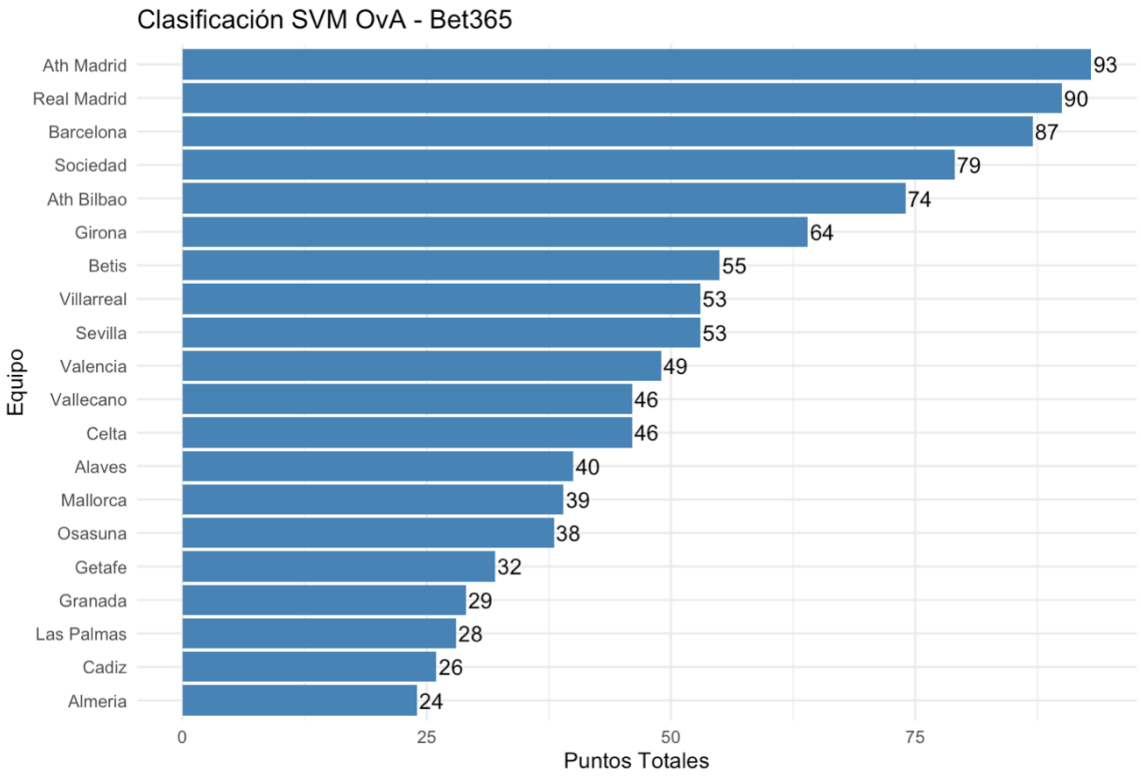

Figura 21: Clasificación Bet365 OvA

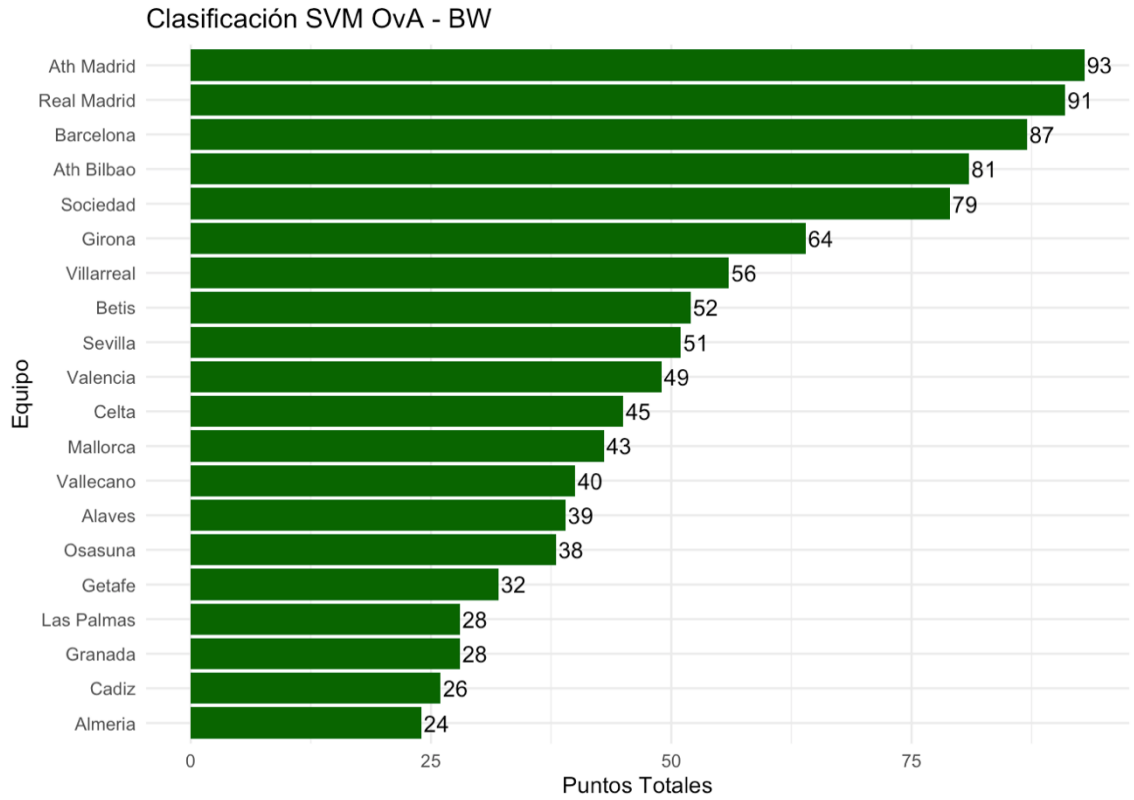

Figura 22: Clasificación BetWay OvA

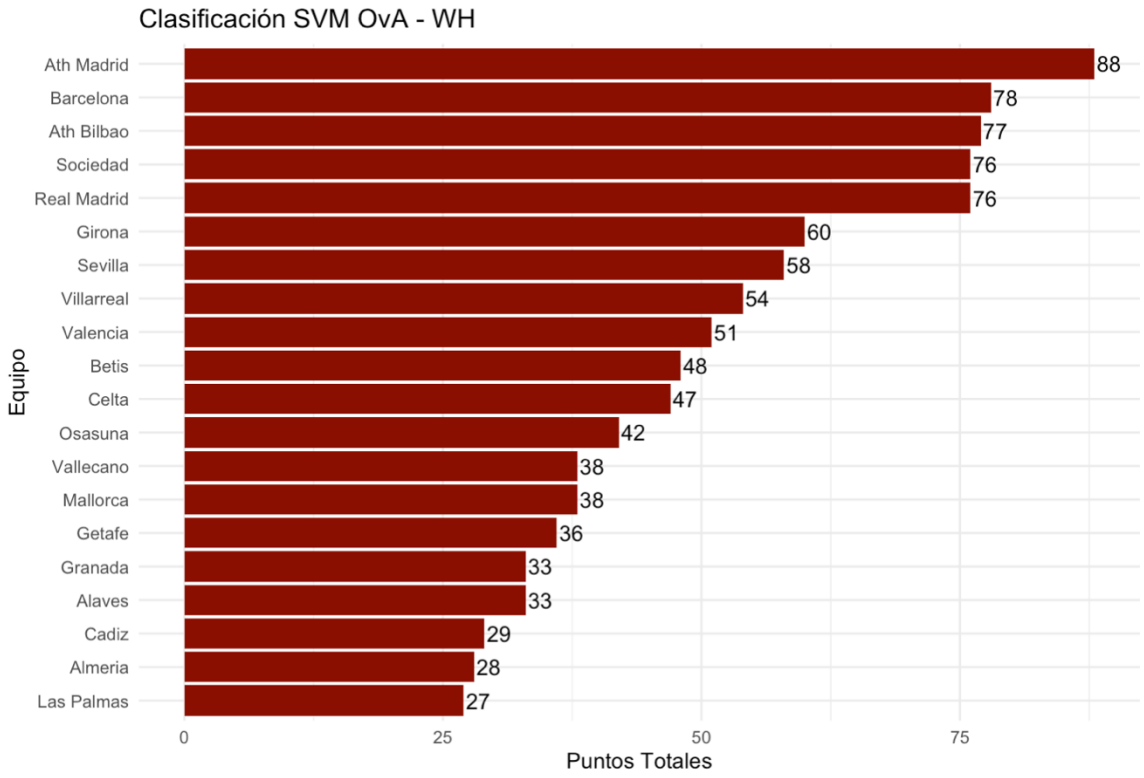

Figura 21: Clasificación WilliamHill OvA

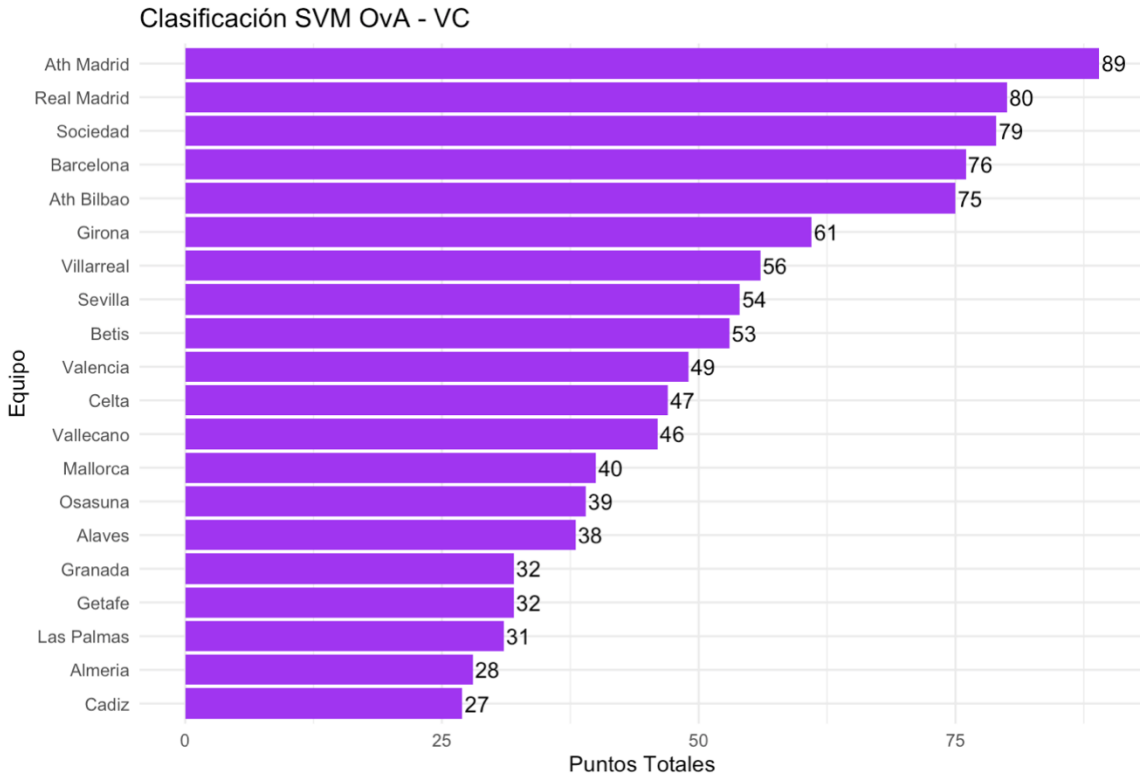

Figura 22: Clasificación VC Bet OvA

La tabla resumen con los valores de MAE (Mean Absolute Error) tanto en puntos como en posiciones, obtenidos al comparar la clasificación predicha por el modelo SVM OvO con la clasificación real de la temporada 2023/2024 para cada casa de apuestas, ha sido:

Casa de Apuestas	MAE en Puntos	MAE en Posiciones
Bet365	7.4	2.15
BetWay	7.65	2.2
WilliamHill	9	2.4
VC	8.75	2.5

- Bet365: Da como campeón al Atlético con 93 puntos, por delante del Real Madrid (90. Aun así, fue la casa más precisa tanto en puntos (MAE 7.4) como en posiciones (2.15).
- BetWay: También coloca líder al Atlético, aunque ajustó bien la zona media. Su MAE fue de 7.65 en puntos y 2.2 en posiciones, mostrando un rendimiento sólido, aunque sin grandes aciertos.
- WiliamHill: Relega al Real Madrid al 5.º puesto y da el liderato al Atlético, lo que dispara su MAE (9 en puntos y 2.4 en posiciones), siendo la menos precisa del grupo.
- VC: Otorga el título al Atlético y sobrevalora a la Real Sociedad (2.º con 80). Su predicción fue la más floja en orden (MAE 2.5), con errores de puntuación similares al resto (8.75).

4. CONCLUSIONES

Este trabajo evalúa la capacidad predictiva de distintos modelos de clasificación supervisada (RF y SVM, en sus versiones OvO y OvA) a partir de las cuotas de cuatro casas de apuestas —Bet365, BetWay, WilliamHill y VC— con el objetivo de estimar la clasificación final de LaLiga 2023/2024 y compararla con la real.
La evaluación se realiza mediante el Error Medio Absoluto (MAE) tanto en puntos como en posiciones, valorando no solo el rendimiento global, sino también la detección de aciertos clave como el campeón o el descenso, así como su comportamiento ante sorpresas imprevistas.

RF

- Bet365: Predice correctamente al Madrid como ganador, equivocándose en los equipos que descienden (Cádiz y Granada se salvan, descienden Osasuna y Rayo Vallecano).
- BetWay: Predice erróneamente al Barcelona como ganador, acertando en los tres equipos que descienden.
- WilliamHill: Predice correctamente al Madrid como ganador, acertando en dos de los tres equipos en descenso (Granada se salva y desciende Las Palmas).
- VC: Predice correctamente al Madrid como ganador, acertando en dos de los tres equipos en descenso (Cádiz se salva y desciende el Rayo Vallecano).

SVM OvO

- Bet365: Predice al Madrid como ganador y acierta en los tres equipos que descienden.
- BetWay: Predice al Madrid como ganador y acierta en los tres equipos que descienden.
- WilliamHill: Predice al Madrid como ganador y acierta en los tres equipos que descienden.
- VC: Erróneamente predice al Barcelona como líder, pero acierta en los tres equipos que descienden.

SVM OvA

En las cuatro casas de apuestas, el líder es el Atlético de Madrid, mientras que sí que aciertan en los tres equipos que descienden a la división de plata.

Tabla Resumen RF

	Equipo	Puntos_Reales	Puntos_Bet365	Puntos_BetWay	Puntos_WilliamHill	Puntos_VC	Pos_Real	Pos_Bet365	Pos_BetWay	Pos_WilliamHill	Pos_VC
1	Real Madrid	95	99	72	90	84	1	1	2	1	1
2	Barcelona	85	80	90	81	80	2	2	1	2	2
3	Girona	81	70	58	70	68	3	5	7	4	4
4	Ath Madrid	76	74	67	70	71	4	4	5	4	3
5	Ath Bilbao	68	76	69	81	65	5	3	3	2	6
6	Sociedad	60	68	68	67	59	6	6	4	6	9
7	Betis	57	44	50	54	50	7	13	11	8	12
8	Villarreal	53	56	66	50	57	8	7	6	10	10
9	Valencia	49	43	45	59	64	9	14	13	7	7
10	Alaves	46	51	52	40	35	10	8	10	15	17
11	Osasuna	45	33	40	54	55	11	18	16	8	11
12	Getafe	43	40	56	38	37	12	16	8	17	16
13	Celta	41	46	48	41	45	13	10	12	14	13
14	Sevilla	41	46	56	48	67	13	10	8	11	5
15	Mallorca	40	42	41	40	61	15	15	15	15	8
16	Las Palmas	40	46	43	31	42	15	10	14	19	15
17	Vallecano	38	28	39	48	29	17	19	17	11	18
18	Cadiz	33	48	36	37	45	18	9	18	18	13
19	Almeria	21	27	22	22	23	19	20	20	20	20
20	Granada	21	37	32	45	29	19	17	19	13	18

Tabla Resumen SVM OvO

	Equipo	Puntos_Reales	Puntos_Bet365	Puntos_BetWay	Puntos_WilliamHill	Puntos_VC	Pos_Real	Pos_Bet365	Pos_BetWay	Pos_WilliamHill	Pos_VC
1	Real Madrid	95	107	106	107	106	1	1	1	1	2
2	Barcelona	85	106	106	106	108	2	2	1	2	1
3	Girona	81	83	84	83	85	3	6	6	6	6
4	Ath Madrid	76	98	97	98	97	4	3	3	3	3
5	Ath Bilbao	68	88	88	85	87	5	4	4	5	4
6	Sociedad	60	87	87	87	87	6	5	5	4	4
7	Betis	57	56	55	54	56	7	8	9	9	8
8	Villarreal	53	56	58	56	56	8	8	7	7	8
9	Valencia	49	49	50	47	49	9	11	10	10	10
10	Alaves	46	35	37	37	37	10	16	13	15	13
11	Osasuna	45	35	37	37	37	11	16	13	15	13
12	Getafe	43	38	37	38	37	12	13	13	14	13
13	Celta	41	51	49	47	49	13	10	11	10	10
14	Sevilla	41	57	57	56	57	13	7	8	7	7
15	Mallorca	40	37	37	39	37	15	14	13	12	13
16	Las Palmas	40	36	36	34	36	15	15	17	17	17
17	Vallecano	38	39	39	39	39	17	12	12	12	12
18	Cadiz	33	24	25	25	25	18	18	18	18	18
19	Almeria	21	22	22	22	24	19	20	20	20	19
20	Granada	21	23	24	24	22	19	19	19	19	20

Tabla Resumen SVM OvA

	Equipo	Puntos_Reales	Puntos_Bet365	Puntos_BetWay	Puntos_WilliamHill	Puntos_VC	Pos_Real	Pos_Bet365	Pos_BetWay	Pos_WilliamHill	Pos_VC
1	Real Madrid	95	90	91	76	80	1	2	2	4	2
2	Barcelona	85	87	87	78	76	2	3	3	2	4
3	Girona	81	64	64	60	61	3	6	6	6	6
4	Ath Madrid	76	93	93	88	89	4	1	1	1	1
5	Ath Bilbao	68	74	81	77	75	5	5	4	3	5
6	Sociedad	60	79	79	76	79	6	4	5	4	3
7	Betis	57	55	52	48	53	7	7	8	10	9
8	Villarreal	53	53	56	54	56	8	8	7	8	7
9	Valencia	49	49	49	51	49	9	10	10	9	10
10	Alaves	46	40	39	33	38	10	13	14	16	15
11	Osasuna	45	38	38	42	39	11	15	15	12	14
12	Getafe	43	32	32	36	32	12	16	16	15	16
13	Celta	41	46	45	47	47	13	11	11	11	11
14	Sevilla	41	53	51	58	54	13	8	9	7	8
15	Mallorca	40	39	43	38	40	15	14	12	13	13
16	Las Palmas	40	28	28	27	31	15	18	17	20	18
17	Vallecano	38	46	40	38	46	17	11	13	13	12
18	Cadiz	33	26	26	29	27	18	19	19	18	20
19	Almeria	21	24	24	28	28	19	20	20	19	19
20	Granada	21	29	28	33	32	19	17	17	16	16

CASO GIRONA

Uno de los hallazgos más llamativos del estudio es que ningún modelo ni casa de apuestas logra anticipar el sorprendente rendimiento del Girona, que termina 3.º con 81 puntos.

Incluso las predicciones más precisas lo sitúan fuera del podio o con puntuaciones 10–20 puntos por debajo. Esto evidencia que los algoritmos basados en cuotas de mercado, aunque útiles para captar tendencias generales, no están diseñados para prever irrupciones excepcionales como la vivida por el equipo catalán esa temporada.

En resumen, se puede afirmar que entre todos los modelos y casas de apuestas analizadas a lo largo del presente proyecto, RF con WilliamHill se muestra como el modelo más preciso globalmente, con el MAE más bajo en puntos (6.85) y un buen rendimiento en posiciones (2.10), a pesar de que falla al predecir uno de los tres equipos descendidos.

Por otro lado, el modelo SVM OvO con BetWay se trata del más exacto en el orden de clasificación (MAE 1.7), acertando tanto en el liderato como en el descenso.

Como conclusión final, RF se muestra como el modelo más robusto en términos generales, mientras que los modelos SVM ofrecen ajustes competitivos en orden de clasificación, aunque con más irregularidades ante situaciones inesperadas.

Este proyecto muestra el potencial real de la Inteligencia Artificial en el deporte, pero también sus límites cuando la emoción, lo imprevisible y los factores humanos entran en juego.

En cuanto a su utilidad práctica, generar una clasificación predicha a partir de modelos y cuotas de apuestas resulta valioso para anticipar tendencias, favoritos o descensos. No obstante, estos modelos no reemplazan el análisis deportivo profundo un pueden generar exactitud en casos atípicos. Por tanto, se deben emplear como herramientas complementarias, útiles para análisis iniciales o de contraste, pero nunca como única fuente de decisión.

De cara a mejorar la capacidad predictiva, sería interesante incorporar variables contextuales o explorar modelos híbridos que combinen IA con análisis cualitativo. Esto permitiría que los modelos no solo aprendan de datos históricos, sino que también integren mejor la dinámica cambiante del deporte profesional.

5. BIBLIOGRAFÍA

[1] *Clasificación supervisada y no supervisada.* (2008c, abril 14). Advanced Tech Computing Group UTPL. https://advancedtech.wordpress.com/2008/04/14/clasificacion-supervisada-y-no-supervisada/

[2] *Ventajas y desventajas de SVM | Interactive Chaos.* (s. f.-c) https://interactivechaos.com/es/manual/tutorial-de-machine-learning/ventajas-y-desventajas-de-svm

[3] *Árboles de decisión, Random Forest, Gradient Boosting y C5.0.* (2017, febrero). Ciencia de Datos. https://cienciadedatos.net/documentos/33_arboles_de_prediccion_bagging_random_forest_boosting#Introducci%C3%B3n

[4] Hastie, T. et al (s.f.). The Elements of Statical Learning Data Mining, Inference, and Prediction [Archivo PDF] https://www.sas.upenn.edu/~fdiebold/NoHesitations/BookAdvanced.pdf

[5] Ibm. (2025b, febrero 5). Regresión logística. *IBM*. https://www.ibm.com/es-es/think/topics/logistic-regression

[6] *1.2. Análisis Discriminal Lineal y Cuadrático – documentación de scikit-learn – 0.24.1*. (s.f.) https://qu4nt.github.io/sklearn-doc-es/modules/lda_qda.html

[7] Plain Concepts. (2024, 25 noviembre). *KNN – Plain concepts*. https://www.plainconcepts.com/es/glosario/knn/

[8] Montero, G.F. y J. (s.f.) *Capítulo 24 Árboles de clasificación y regresión | Fundamentos de ciencia de datos en R*. https://cdr-book.github.io/cap-arboles.html?utm_source=chatgpt.com

[9] Hastie, T. et al (2023). *An Introduction to Statistical Learning with Applications in R* [Archivo PDF] https://hastie.su.domains/ISLR2/ISLRv2_corrected_June_2023.pdf.download.html

[10] Barajas, D. (2024). *Bosque Aleatorio en la detención temprana del suicidio* [Archivo PDF] https://www.terc.mx/index.php/terc/article/view/404/298

[11] *Kernels | Interactive Chaos.* (s.f.) https://interactivechaos.com/es/manual/tutorial-de-machine-learning/kernels

[12] Palazón, M. (s.f.). *Técnicas Estadísticas en Análisis de Mercad* [Archivo PDF]

[13] Hernández, F. (2024, 6 diciembre). *4 Random Forests | Modelos predictivos.* https://fhernanb.github.io/libro_mod_pred/rand-forests.html

6. APÉNDICE

```r
# Cargar librerías necesarias
library(readxl)
library(randomForest)
library(dplyr)
library(ggplot2)
library(e1071)

# Leer los datos
data2021 <- read_excel("2021datos.xlsx")
data2122 <- read_excel("2122datos.xlsx")
data2223 <- read_excel("2223datos.xlsx")
data2324 <- read_excel("2324datos.xlsx")

# Unir los datos de entrenamiento
entrenamiento <- rbind(data2021, data2122, data2223)

# Asegurar que 'Resultado' sea factor
entrenamiento$Resultado <- as.character(entrenamiento$Resultado)
entrenamiento$Resultado <- factor(entrenamiento$Resultado, levels = c("-1", "0", "1"))

set.seed(100)
```

En esta parte, se cargan las principales librerías. Posteriormente, se importan los datasets correspondientes a las temporadas 2020/21, 2021/22, 2022/23 y 2023/24. Los tres primeros combinan un único conjunto de entrenamiento, mientras que el último se utiliza como conjunto para predecir. Finalmente, se transforma la variable “Resultado” en un factor categórico con tres clases posibles: victoria local (1), empate (0) y victoria visitante (-1), y se fija una semilla para la reproducibilidad.

```r
# ================ MODELO BET365 ================

# Convertir variables a numérico
vars_Bet365 <- c("B365H", "B365D", "B365A")
entrenamiento[vars_Bet365] <- lapply(entrenamiento[vars_Bet365], as.numeric)
data2324[vars_Bet365] <- lapply(data2324[vars_Bet365], as.numeric)

# Entrenar modelo
modelo_rf_Bet365 <- randomForest(Resultado ~ B365H + B365D + B365A,
                                 data = entrenamiento,
                                 ntree = 500,
                                 mtry = 3,
                                 importance = TRUE)

# Predecir resultados
predicciones_Bet365 <- predict(modelo_rf_Bet365, newdata = data2324)
data2324$Prediccion_Bet365 <- as.integer(as.character(predicciones_Bet365))

# Crear matriz de predicciones
matriz_final_Bet365 <- data2324[, c("HomeTeam", "AwayTeam", "B365H", "B365D", "B365A", "Prediccion_Bet365")]

# Calcular puntos
puntos_Bet365 <- data.frame(Equipo = character(), Puntos = numeric(), stringsAsFactors = FALSE)
for (i in 1:nrow(matriz_final_Bet365)) {
  local <- matriz_final_Bet365$HomeTeam[i]
  visitante <- matriz_final_Bet365$AwayTeam[i]
  resultado <- matriz_final_Bet365$`Prediccion_Bet365`[i]

  if (resultado == 1) {
    puntos_Bet365 <- rbind(puntos_Bet365, data.frame(Equipo = local, Puntos = 3))
  } else if (resultado == 0) {
    puntos_Bet365 <- rbind(puntos_Bet365, data.frame(Equipo = local, Puntos = 1))
    puntos_Bet365 <- rbind(puntos_Bet365, data.frame(Equipo = visitante, Puntos = 1))
  } else {
    puntos_Bet365 <- rbind(puntos_Bet365, data.frame(Equipo = visitante, Puntos = 3))
  }
}

# Tabla final
tabla_final_Bet365 <- puntos_Bet365 %>%
  group_by(Equipo) %>%
  summarise(Puntos_Totales = sum(Puntos)) %>%
  arrange(desc(Puntos_Totales))

# Gráfico
tabla_final_Bet365 <- tabla_final_Bet365 %>% arrange(Puntos_Totales)
ggplot(tabla_final_Bet365, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_text(aes(label = Puntos_Totales), hjust = -0.1, size = 4) +
  coord_flip() +
  labs(title = "Clasificación Predicha (Bet365)", x = "Equipo", y = "Puntos Totales") +
  theme_minimal()
```

Se entrena un modelo de RF usando como variables predictoras las cuotas de victoria, empate y derrota ofrecidas por Bet365. Tras predecir los resultados de cada partido de la temporada 2023/24, se asignan puntos en función del resultado estimado y se genera una clasificación final. Esta clasificación se representa visualmente en un gráfico de barras, permitiendo comparar el rendimiento previsto por el modelo con la clasificación real.

```r
# ================= MODELO BETWAY =================

# Convertir variables a numérico
vars_BetWay <- c("BWH", "BWD", "BWA")
entrenamiento[vars_BetWay] <- lapply(entrenamiento[vars_BetWay], as.numeric)
data2324[vars_BetWay] <- lapply(data2324[vars_BetWay], as.numeric)

# Entrenar modelo
modelo_rf_BetWay <- randomForest(Resultado ~ BWH + BWD + BWA,
                                 data = entrenamiento,
                                 ntree = 500,
                                 mtry = 3,
                                 importance = TRUE)

# Predecir resultados
predicciones_BetWay <- predict(modelo_rf_BetWay, newdata = data2324)
data2324$Prediccion_BetWay <- as.integer(as.character(predicciones_BetWay))

# Crear matriz de predicciones
matriz_final_BetWay <- data2324[, c("HomeTeam", "AwayTeam", "BWH", "BWD", "BWA", "Prediccion_BetWay")]

# Calcular puntos
puntos_BetWay <- data.frame(Equipo = character(), Puntos = numeric(), stringsAsFactors = FALSE)
for (i in 1:nrow(matriz_final_BetWay)) {
  local <- matriz_final_BetWay$HomeTeam[i]
  visitante <- matriz_final_BetWay$AwayTeam[i]
  resultado <- matriz_final_BetWay$`Prediccion_BetWay`[i]

  if (resultado == 1) {
    puntos_BetWay <- rbind(puntos_BetWay, data.frame(Equipo = local, Puntos = 3))
  } else if (resultado == 0) {
    puntos_BetWay <- rbind(puntos_BetWay, data.frame(Equipo = local, Puntos = 1))
    puntos_BetWay <- rbind(puntos_BetWay, data.frame(Equipo = visitante, Puntos = 1))
  } else {
    puntos_BetWay <- rbind(puntos_BetWay, data.frame(Equipo = visitante, Puntos = 3))
  }
}

# Tabla final
tabla_final_BetWay <- puntos_BetWay %>%
  group_by(Equipo) %>%
  summarise(Puntos_Totales = sum(Puntos)) %>%
  arrange(desc(Puntos_Totales))

# Gráfico
tabla_final_BetWay <- tabla_final_BetWay %>% arrange(Puntos_Totales)
ggplot(tabla_final_BetWay, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
  geom_bar(stat = "identity", fill = "darkgreen") +
  geom_text(aes(label = Puntos_Totales), hjust = -0.1, size = 4) +
  coord_flip() +
  labs(title = "Clasificación Predicha (BetWay)", x = "Equipo", y = "Puntos Totales") +
  theme_minimal()
```

El modelo RF utiliza las cuotas de BetWay como variables de entrada para predecir los resultados de la temporada. A partir de esas predicciones se construye una tabla de puntuaciones y una clasificación final. Se representa gráficamente para comparar la predicción con la realidad, facilitando el análisis del rendimiento de este operador de apuestas como fuente de información predictiva.

```r
# ================ MODELO WILLIAMHILL ================

# Convertir variables a numérico
vars_WilliamHill <- c("WHH", "WHD", "WHA")
entrenamiento[vars_WilliamHill] <- lapply(entrenamiento[vars_WilliamHill], as.numeric)
data2324[vars_WilliamHill] <- lapply(data2324[vars_WilliamHill], as.numeric)

# Entrenar modelo
modelo_rf_WilliamHill <- randomForest(Resultado ~ WHH + WHD + WHA,
                                      data = entrenamiento,
                                      ntree = 500,
                                      mtry = 3,
                                      importance = TRUE)

# Predecir resultados
predicciones_WilliamHill <- predict(modelo_rf_WilliamHill, newdata = data2324)
data2324$Prediccion_WilliamHill <- as.integer(as.character(predicciones_WilliamHill))

# Crear matriz de predicciones
matriz_final_WilliamHill <- data2324[, c("HomeTeam", "AwayTeam", "WHH", "WHD", "WHA", "Prediccion_WilliamHill")]

# Calcular puntos
puntos_WilliamHill <- data.frame(Equipo = character(), Puntos = numeric(), stringsAsFactors = FALSE)
for (i in 1:nrow(matriz_final_WilliamHill)) {
  local <- matriz_final_WilliamHill$HomeTeam[i]
  visitante <- matriz_final_WilliamHill$AwayTeam[i]
  resultado <- matriz_final_WilliamHill$`Prediccion_WilliamHill`[i]

  if (resultado == 1) {
    puntos_WilliamHill <- rbind(puntos_WilliamHill, data.frame(Equipo = local, Puntos = 3))
  } else if (resultado == 0) {
    puntos_WilliamHill <- rbind(puntos_WilliamHill, data.frame(Equipo = local, Puntos = 1))
    puntos_WilliamHill <- rbind(puntos_WilliamHill, data.frame(Equipo = visitante, Puntos = 1))
  } else {
    puntos_WilliamHill <- rbind(puntos_WilliamHill, data.frame(Equipo = visitante, Puntos = 3))
  }
}

# Tabla final
tabla_final_WilliamHill <- puntos_WilliamHill %>%
  group_by(Equipo) %>%
  summarise(Puntos_Totales = sum(Puntos)) %>%
  arrange(desc(Puntos_Totales))

# Gráfico
tabla_final_WilliamHill <- tabla_final_WilliamHill %>% arrange(Puntos_Totales)
ggplot(tabla_final_WilliamHill, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
  geom_bar(stat = "identity", fill = "darkred") +
  geom_text(aes(label = Puntos_Totales), hjust = -0.1, size = 4) +
  coord_flip() +
  labs(title = "Clasificación Predicha (WilliamHill)", x = "Equipo", y = "Puntos Totales") +
  theme_minimal()
```

En este modelo se emplean las cuotas de William Hill para predecir los resultados de cada encuentro mediante RF. Con las predicciones se construye una clasificación final basada en los puntos obtenidos por cada equipo, que luego se representa en una gráfica. Esto permite evaluar hasta qué punto esta casa de apuestas se ajusta a la realidad competitiva de la temporada.

```r
# ================ MODELO VC ================

# Convertir variables a numérico
vars_VC <- c("VCH", "VCD", "VCA")
entrenamiento[vars_VC] <- lapply(entrenamiento[vars_VC], as.numeric)
data2324[vars_VC] <- lapply(data2324[vars_VC], as.numeric)

# Entrenar modelo
modelo_rf_VC <- randomForest(Resultado ~ VCH + VCD + VCA,
                             data = entrenamiento,
                             ntree = 500,
                             mtry = 3,
                             importance = TRUE)

# Predecir resultados
predicciones_VC <- predict(modelo_rf_VC, newdata = data2324)
data2324$Prediccion_VC <- as.integer(as.character(predicciones_VC))

# Crear matriz de predicciones
matriz_final_VC <- data2324[, c("HomeTeam", "AwayTeam", "VCH", "VCD", "VCA", "Prediccion_VC")]

# Calcular puntos
puntos_VC <- data.frame(Equipo = character(), Puntos = numeric(), stringsAsFactors = FALSE)
for (i in 1:nrow(matriz_final_VC)) {
  local <- matriz_final_VC$HomeTeam[i]
  visitante <- matriz_final_VC$AwayTeam[i]
  resultado <- matriz_final_VC$`Prediccion_VC`[i]

  if (resultado == 1) {
    puntos_VC <- rbind(puntos_VC, data.frame(Equipo = local, Puntos = 3))
  } else if (resultado == 0) {
    puntos_VC <- rbind(puntos_VC, data.frame(Equipo = local, Puntos = 1))
    puntos_VC <- rbind(puntos_VC, data.frame(Equipo = visitante, Puntos = 1))
  } else {
    puntos_VC <- rbind(puntos_VC, data.frame(Equipo = visitante, Puntos = 3))
  }
}

# Tabla final
tabla_final_VC <- puntos_VC %>%
  group_by(Equipo) %>%
  summarise(Puntos_Totales = sum(Puntos)) %>%
  arrange(desc(Puntos_Totales))

# Gráfico
tabla_final_VC <- tabla_final_VC %>% arrange(Puntos_Totales)
ggplot(tabla_final_VC, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
  geom_bar(stat = "identity", fill = "purple") +
  geom_text(aes(label = Puntos_Totales), hjust = -0.1, size = 4) +
  coord_flip() +
  labs(title = "Clasificación Predicha (VC)", x = "Equipo", y = "Puntos Totales") +
  theme_minimal()
```

Este modelo utiliza las cuotas ofrecidas por la casa de apuestas VC como variables predictoras. A través de un RF se estiman los resultados de cada partido y se calcula la puntuación final para cada equipo. El gráfico resultante permite visualizar cómo habría sido la clasificación según VC y valorar su precisión frente a la realidad.

```r
# ================ COMPARACIÓN FINAL ================
clasificacion_real <- data.frame(
  Equipo = c("Real Madrid", "Barcelona", "Girona", "Ath Madrid", "Ath Bilbao",
             "Sociedad", "Betis", "Villarreal", "Valencia", "Alaves",
             "Osasuna", "Getafe", "Celta", "Sevilla", "Mallorca",
             "Las Palmas", "Vallecano", "Cadiz", "Almeria", "Granada"),
  Puntos_Reales = c(95, 85, 81, 76, 68, 60, 57, 53, 49, 46,
                    45, 43, 41, 41, 40, 40, 38, 33, 21, 21)
)


clasificacion_real <- clasificacion_real %>%
  mutate(Equipo = reorder(Equipo, Puntos_Reales))

ggplot(clasificacion_real, aes(x = Equipo, y = Puntos_Reales)) +
  geom_col(fill = "saddlebrown") +
  geom_text(aes(label = Puntos_Reales), hjust = -0.1, size = 4) +
  coord_flip() +  # Para que los equipos aparezcan en eje y
  labs(title = "Clasificación Puntos Reales",
       x = "Equipo",
       y = "Puntos") +
  theme_minimal()


calcular_error <- function(tabla_modelo, nombre_modelo) {
  comp <- merge(clasificacion_real, tabla_modelo, by = "Equipo", all.x = TRUE)
  comp$Error_Abs <- abs(comp$Puntos_Reales - comp$Puntos_Totales)
  comp$Modelo <- nombre_modelo
  return(comp)
}

calcular_errores_completos <- function(tabla_modelo, nombre_modelo) {
  comp <- merge(clasificacion_real, tabla_modelo, by = "Equipo", all.x = TRUE)
  comp$Error_Abs <- abs(comp$Puntos_Reales - comp$Puntos_Totales)
  comp$Pos_Real <- rank(-comp$Puntos_Reales, ties.method = "min")
  comp$Pos_Pred <- rank(-comp$Puntos_Totales, ties.method = "min")
  comp$Error_Pos <- abs(comp$Pos_Real - comp$Pos_Pred)
  comp$Modelo <- nombre_modelo
  return(comp)
}

# Comparaciones
comparaciones <- rbind(
  calcular_error(tabla_final_Bet365, "Bet365"),
  calcular_error(tabla_final_BetWay, "BetWay"),
  calcular_error(tabla_final_WilliamHill, "WilliamHill"),
  calcular_error(tabla_final_VC, "VC")
)

# MAE por modelo
mae_modelos <- comparaciones %>%
  group_by(Modelo) %>%
  summarise(MAE = mean(Error_Abs, na.rm = TRUE))
print(mae_modelos)

# Errores de posición
errores_todos <- rbind(
  calcular_errores_completos(tabla_final_Bet365, "Bet365"),
  calcular_errores_completos(tabla_final_BetWay, "BetWay"),
  calcular_errores_completos(tabla_final_WilliamHill, "WilliamHill"),
  calcular_errores_completos(tabla_final_VC, "VC")
)

resumen_error_modelos <- errores_todos %>%
  group_by(Modelo) %>%
  summarise(MAE_Puntos = mean(Error_Abs, na.rm = TRUE),
            MAE_Posicion = mean(Error_Pos, na.rm = TRUE))
print(resumen_error_modelos)
```

Este bloque compara el rendimiento de los cuatro modelos RF (uno por casa de apuestas) frente a la clasificación real de LaLiga 2023/2024. Se calculan dos tipos de errores:

- MAE en puntos: mide la diferencia promedio en puntos entre la clasificación real y la predicha

- MAE en posiciones: mide el desajuste promedio en el puesto de cada equipo

Esta comparación permite identificar qué casa de apuestas proporciona cuotas más alineadas con los resultados reales cuando se usan como entrada en modelos de predicción. Los resultados son clave para la evaluación global del rendimiento de cada modelo.

```r
# ================ SVM OvO ================

entrenar_y_predecir_SVM_OvO <- function(nombre_modelo, columnas_cuotas, df_entrenamiento, df_prediccion) {
  colnames_pred <- paste0("Prediccion_SVM_OvO_", nombre_modelo)

  # Entrenamientos para 1 vs 0
  ent_1vs0 <- df_entrenamiento[df_entrenamiento$Resultado %in% c("1", "0"), ]
  ent_1vs0$Y <- factor(ifelse(ent_1vs0$Resultado == "1", 1, 0))
  mod_1vs0 <- svm(Y ~ ., data = ent_1vs0[, c(columnas_cuotas, "Y")], kernel = "radial", cost = 1, scale = FALSE)

  # Entrenamientos para 1 vs -1
  ent_1vsm1 <- df_entrenamiento[df_entrenamiento$Resultado %in% c("1", "-1"), ]
  ent_1vsm1$Y <- factor(ifelse(ent_1vsm1$Resultado == "1", 1, 0))
  mod_1vsm1 <- svm(Y ~ ., data = ent_1vsm1[, c(columnas_cuotas, "Y")], kernel = "radial", cost = 1, scale = FALSE)

  # Entrenamientos para 0 vs -1
  ent_0vsm1 <- df_entrenamiento[df_entrenamiento$Resultado %in% c("0", "-1"), ]
  ent_0vsm1$Y <- factor(ifelse(ent_0vsm1$Resultado == "0", 1, 0))
  mod_0vsm1 <- svm(Y ~ ., data = ent_0vsm1[, c(columnas_cuotas, "Y")], kernel = "radial", cost = 1, scale = FALSE)

  # Predicciones
  p1 <- predict(mod_1vs0, newdata = df_prediccion[, columnas_cuotas])
  p2 <- predict(mod_1vsm1, newdata = df_prediccion[, columnas_cuotas])
  p3 <- predict(mod_0vsm1, newdata = df_prediccion[, columnas_cuotas])

  # Votación mayoritaria
  pred_final <- mapply(function(a, b, c) {
    votos <- c(ifelse(a == 1, 1, 0),
               ifelse(b == 1, 1, -1),
               ifelse(c == 1, 0, -1))
    as.integer(names(sort(table(votos), decreasing = TRUE)[1]))
  }, p1, p2, p3)

  df_prediccion[[colnames_pred]] <- pred_final
  return(df_prediccion)
}

# ================ APLICACIÓN DE LA FUNCIÓN A TODAS LAS CASAS ================

# Asegurar que todas las columnas de cuotas estén en formato numérico
cuotas_bet365 <- c("B365H", "B365D", "B365A")
cuotas_betway <- c("BWH", "BWD", "BWA")
cuotas_wh <- c("WHH", "WHD", "WHA")
cuotas_vc <- c("VCH", "VCD", "VCA")

entrenamiento[cuotas_bet365] <- lapply(entrenamiento[cuotas_bet365], as.numeric)
data2324[cuotas_bet365] <- lapply(data2324[cuotas_bet365], as.numeric)

entrenamiento[cuotas_betway] <- lapply(entrenamiento[cuotas_betway], as.numeric)
data2324[cuotas_betway] <- lapply(data2324[cuotas_betway], as.numeric)

entrenamiento[cuotas_wh] <- lapply(entrenamiento[cuotas_wh], as.numeric)
data2324[cuotas_wh] <- lapply(data2324[cuotas_wh], as.numeric)

entrenamiento[cuotas_vc] <- lapply(entrenamiento[cuotas_vc], as.numeric)
data2324[cuotas_vc] <- lapply(data2324[cuotas_vc], as.numeric)

# Aplicar SVM OvO para las 4 casas
data2324 <- entrenar_y_predecir_SVM_OvO("Bet365", cuotas_bet365, entrenamiento, data2324)
data2324 <- entrenar_y_predecir_SVM_OvO("BetWay", cuotas_betway, entrenamiento, data2324)
data2324 <- entrenar_y_predecir_SVM_OvO("WilliamHill", cuotas_wh, entrenamiento, data2324)
data2324 <- entrenar_y_predecir_SVM_OvO("VC", cuotas_vc, entrenamiento, data2324)
```

En este bloque se implementa el modelo SVM One-vs-One (OvO), que divide el problema multiclase en submodelos binarios. Cada submodelo enfrenta dos posibles resultados (victoria, empate o derrota), y la predicción final se basa en votación mayoritaria.

Este enfoque permite un análisis fino en problemas donde existen tres posibles clases, como en los partidos de fútbol. La función desarrollada se aplica luego a las cuotas de las cuatro casas de apuestas, generando una predicción de cada jornada de LaLiga.

Estas predicciones se transformarán en puntos para clasificar a los equipos y posteriormente compararse con la clasificación real.

```r
# ================ FUNCIÓN PARA TABLA DE PUNTOS Y GRÁFICO ================
calcular_tabla_ovo <- function(df, col_pred, color_barra, titulo) {
  puntos <- data.frame(Equipo = character(), Puntos = numeric())
  for (i in 1:nrow(df)) {
    local <- df$HomeTeam[i]
    visitante <- df$AwayTeam[i]
    resultado <- df[[col_pred]][i]
    if (resultado == 1) {
      puntos <- rbind(puntos, data.frame(Equipo = local, Puntos = 3))
    } else if (resultado == 0) {
      puntos <- rbind(puntos, data.frame(Equipo = local, Puntos = 1))
      puntos <- rbind(puntos, data.frame(Equipo = visitante, Puntos = 1))
    } else {
      puntos <- rbind(puntos, data.frame(Equipo = visitante, Puntos = 3))
    }
  }
  tabla <- puntos %>%
    group_by(Equipo) %>%
    summarise(Puntos_Totales = sum(Puntos)) %>%
    arrange(desc(Puntos_Totales))

  # Gráfico
  ggplot(tabla, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
    geom_bar(stat = "identity", fill = color_barra) +
    geom_text(aes(label = Puntos_Totales), hjust = -0.1, size = 4) +
    coord_flip() +
    labs(title = titulo, x = "Equipo", y = "Puntos Totales") +
    theme_minimal()

  return(tabla)
}

# ================ GENERACIÓN DE TABLAS Y GRÁFICOS PARA CADA CASA ================
tabla_ovo_Bet365 <- calcular_tabla_ovo(data2324, "Prediccion_SVM_OvO_Bet365", "orange", "Clasificación Predicha OvO - Bet365")
tabla_ovo_BetWay <- calcular_tabla_ovo(data2324, "Prediccion_SVM_OvO_BetWay", "forestgreen", "Clasificación Predicha OvO - BetWay")
tabla_ovo_WilliamHill <- calcular_tabla_ovo(data2324, "Prediccion_SVM_OvO_WilliamHill", "darkred", "Clasificación Predicha OvO - WilliamHill")
tabla_ovo_VC <- calcular_tabla_ovo(data2324, "Prediccion_SVM_OvO_VC", "purple", "Clasificación Predicha OvO - VC")

# ================ GRÁFICOS EXPLÍCITOS PARA OvO ================
ggplot(tabla_ovo_Bet365, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_text(aes(label = Puntos_Totales), hjust = -0.1, size = 4) +
  coord_flip() +
  labs(title = "Clasificación Predicha OvO - Bet365", x = "Equipo", y = "Puntos Totales") +
  theme_minimal()

ggplot(tabla_ovo_BetWay, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
  geom_bar(stat = "identity", fill = "forestgreen") +
  geom_text(aes(label = Puntos_Totales), hjust = -0.1, size = 4) +
  coord_flip() +
  labs(title = "Clasificación Predicha OvO - BetWay", x = "Equipo", y = "Puntos Totales") +
  theme_minimal()

ggplot(tabla_ovo_WilliamHill, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
  geom_bar(stat = "identity", fill = "darkred") +
  geom_text(aes(label = Puntos_Totales), hjust = -0.1, size = 4) +
  coord_flip() +
  labs(title = "Clasificación Predicha OvO - WilliamHill", x = "Equipo", y = "Puntos Totales") +
  theme_minimal()

ggplot(tabla_ovo_VC, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
  geom_bar(stat = "identity", fill = "purple") +
  geom_text(aes(label = Puntos_Totales), hjust = -0.1, size = 4) +
  coord_flip() +
  labs(title = "Clasificación Predicha OvO - VC", x = "Equipo", y = "Puntos Totales") +
  theme_minimal()
```

Tras aplicar el modelo SVM OvO a los datos de cada casa de apuestas, se calculan los puntos por equipo a partir de los resultados predichos, generando así una clasificación simulada. Posteriormente, se visualizan mediante gráficos de barras los puntos obtenidos por cada equipo, lo que permite comparar fácilmente las predicciones de cada casa con la clasificación real de la temporada.

```r
# ================ COMPARACIÓN FINAL ================

clasificacion_real <- data.frame(
  Equipo = c("Real Madrid", "Barcelona", "Girona", "Ath Madrid", "Ath Bilbao",
             "Sociedad", "Betis", "Villarreal", "Valencia", "Alaves",
             "Osasuna", "Getafe", "Celta", "Sevilla", "Mallorca",
             "Las Palmas", "Vallecano", "Cadiz", "Almeria", "Granada"),
  Puntos_Reales = c(95, 85, 81, 76, 68, 60, 57, 53, 49, 46,
                    45, 43, 41, 41, 40, 40, 38, 33, 21, 21)
)


calcular_error <- function(tabla_modelo, nombre_modelo) {
  comp <- merge(clasificacion_real, tabla_modelo, by = "Equipo", all.x = TRUE)
  comp$Error_Abs <- abs(comp$Puntos_Reales - comp$Puntos_Totales)
  comp$Modelo <- nombre_modelo
  return(comp)
}

calcular_errores_completos <- function(tabla_modelo, nombre_modelo) {
  comp <- merge(clasificacion_real, tabla_modelo, by = "Equipo", all.x = TRUE)
  comp$Error_Abs <- abs(comp$Puntos_Reales - comp$Puntos_Totales)
  comp$Pos_Real <- rank(-comp$Puntos_Reales, ties.method = "min")
  comp$Pos_Pred <- rank(-comp$Puntos_Totales, ties.method = "min")
  comp$Error_Pos <- abs(comp$Pos_Real - comp$Pos_Pred)
  comp$Modelo <- nombre_modelo
  return(comp)
}

# Comparaciones
comparaciones <- rbind(
  calcular_error(tabla_ovo_Bet365, "Bet365"),
  calcular_error(tabla_ovo_BetWay, "BetWay"),
  calcular_error(tabla_ovo_WilliamHill, "WilliamHill"),
  calcular_error(tabla_ovo_VC, "VC")
)

# MAE por modelo
mae_modelos <- comparaciones %>%
  group_by(Modelo) %>%
  summarise(MAE = mean(Error_Abs, na.rm = TRUE))
print(mae_modelos)

# Errores de posición
errores_todos <- rbind(
  calcular_errores_completos(tabla_ovo_Bet365, "Bet365"),
  calcular_errores_completos(tabla_ovo_BetWay, "BetWay"),
  calcular_errores_completos(tabla_ovo_WilliamHill, "WilliamHill"),
  calcular_errores_completos(tabla_ovo_VC, "VC")
)

resumen_error_modelos <- errores_todos %>%
  group_by(Modelo) %>%
  summarise(MAE_Puntos = mean(Error_Abs, na.rm = TRUE),
            MAE_Posicion = mean(Error_Pos, na.rm = TRUE))
print(resumen_error_modelos)
```

Se calcula el error absoluto medio (MAE) en puntos y en posiciones para cada casa de apuestas, comparando las clasificaciones predichas con la real. Este análisis permite

evaluar cuál casa y modelo predicen con mayor precisión el rendimiento final de los equipos, considerando tanto la exactitud en puntos como en puestos en la tabla.

```r
# ==================== SVM OVA  ====================
data2021 <- read_excel("2021datos.xlsx")
data2122 <- read_excel("2122datos.xlsx")
data2223 <- read_excel("2223datos.xlsx")
data2324 <- read_excel("2324datos.xlsx")
entrenamiento <- rbind(data2021, data2122, data2223)
entrenamiento$Resultado <- as.character(entrenamiento$Resultado)

svm_ova_no_std <- function(df_train, df_test, columnas) {

  for (col in columnas) {
    df_train[[col]] <- as.numeric(as.character(df_train[[col]]))
    df_test[[col]] <- as.numeric(as.character(df_test[[col]]))
  }

  df_train$Y1 <- ifelse(df_train$Resultado == "1", 1, -1)
  df_train$Y0 <- ifelse(df_train$Resultado == "0", 1, -1)
  df_train$YM1 <- ifelse(df_train$Resultado == "-1", 1, -1)

  svm1 <- svm(Y1 ~ ., data = df_train[, c(columnas, "Y1")], kernel = "linear", scale = FALSE, decision.values = TRUE)
  svm0 <- svm(Y0 ~ ., data = df_train[, c(columnas, "Y0")], kernel = "linear", scale = FALSE, decision.values = TRUE)
  svmm1 <- svm(YM1 ~ ., data = df_train[, c(columnas, "YM1")], kernel = "linear", scale = FALSE, decision.values = TRUE)

  f1 <- attr(predict(svm1, df_test[, columnas], decision.values = TRUE), "decision.values")
  f0 <- attr(predict(svm0, df_test[, columnas], decision.values = TRUE), "decision.values")
  fm1 <- attr(predict(svmm1, df_test[, columnas], decision.values = TRUE), "decision.values")

  pred <- mapply(function(a, b, c) {
    vals <- c("1" = abs(a), "0" = abs(b), "-1" = abs(c))
    as.integer(names(which.max(vals)))
  }, f1, f0, fm1)

  df_test$Prediccion_SVM_OvA <- pred
  return(df_test)
}

# ==================== CLASIFICACIÓN ====================
clasificacion_teams <- function(df, col_pred, titulo) {
  puntos <- data.frame(Equipo = character(), Puntos = numeric())
  for (i in 1:nrow(df)) {
    local <- df$HomeTeam[i]
    visitante <- df$AwayTeam[i]
    resultado <- df[[col_pred]][i]
    if (resultado == 1) {
      puntos <- rbind(puntos, data.frame(Equipo = visitante, Puntos = 3))
    } else if (resultado == 0) {
      puntos <- rbind(puntos, data.frame(Equipo = local, Puntos = 1))
      puntos <- rbind(puntos, data.frame(Equipo = visitante, Puntos = 1))
    } else if (resultado == -1) {
      puntos <- rbind(puntos, data.frame(Equipo = local, Puntos = 3))
    }
  }
  tabla <- puntos %>%
    group_by(Equipo) %>%
    summarise(Puntos_Totales = sum(Puntos)) %>%
    arrange(desc(Puntos_Totales))

  return(tabla)
}
```

Se construye un modelo SVM One-vs-All (OvA) para clasificar los resultados de los partidos según las probabilidades de cada casa. Para ello, se entrena un modelo independiente para cada clase (victoria local, empate, victoria visitante), y se asigna la categoría con mayor valor absoluto de función de decisión. Luego, se genera una tabla de

puntos para cada equipo a partir de las predicciones obtenidas, permitiendo visualizar la clasificación final según este enfoque.

```r
# ==================== EJECUCIÓN Y GRAFICOS ====================

columnas_bet365 <- c("B365H", "B365D", "B365A")
resultado_bet365 <- svm_ova_no_std(entrenamiento, data2324, columnas_bet365)
tabla_bet365 <- clasificacion_teams(resultado_bet365, "Prediccion_SVM_OvA", "Clasificación SVM OvA - Bet365")

columnas_bw <- c("BWH", "BWD", "BWA")
resultado_bw <- svm_ova_no_std(entrenamiento, data2324, columnas_bw)
tabla_bw <- clasificacion_teams(resultado_bw, "Prediccion_SVM_OvA", "Clasificación SVM OvA  - BW")

columnas_wh <- c("WHH", "WHD", "WHA")
resultado_wh <- svm_ova_no_std(entrenamiento, data2324, columnas_wh)
tabla_wh <- clasificacion_teams(resultado_wh, "Prediccion_SVM_OvA", "Clasificación SVM OvA  - WH")

columnas_vc <- c("VCH", "VCD", "VCA")
resultado_vc <- svm_ova_no_std(entrenamiento, data2324, columnas_vc)
tabla_vc <- clasificacion_teams(resultado_vc, "Prediccion_SVM_OvA", "Clasificación SVM OvA SIN Estandarización - VC")

ggplot(tabla_bet365, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  geom_text(aes(label = Puntos_Totales), hjust = -0.1) +
  coord_flip() +
  labs(title = "Clasificación SVM OvA - Bet365", x = "Equipo", y = "Puntos Totales") +
  theme_minimal()

ggplot(tabla_bw, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
  geom_bar(stat = "identity", fill = "darkgreen") +
  geom_text(aes(label = Puntos_Totales), hjust = -0.1) +
  coord_flip() +
  labs(title = "Clasificación SVM OvA - BW", x = "Equipo", y = "Puntos Totales") +
  theme_minimal()

ggplot(tabla_wh, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
  geom_bar(stat = "identity", fill = "darkred") +
  geom_text(aes(label = Puntos_Totales), hjust = -0.1) +
  coord_flip() +
  labs(title = "Clasificación SVM OvA - WH", x = "Equipo", y = "Puntos Totales") +
  theme_minimal()

ggplot(tabla_vc, aes(x = reorder(Equipo, Puntos_Totales), y = Puntos_Totales)) +
  geom_bar(stat = "identity", fill = "purple") +
  geom_text(aes(label = Puntos_Totales), hjust = -0.1) +
  coord_flip() +
  labs(title = "Clasificación SVM OvA - VC", x = "Equipo", y = "Puntos Totales") +
  theme_minimal()
```

Se aplica el modelo SVM OvA a los datos de la temporada 23/24 para las cuatro casas de apuestas (Bet365, BetWay, WilliamHill y VC). A partir de las predicciones, se calculan los puntos totales por equipo y se representan en gráficos de barras, mostrando visualmente la clasificación predicha por cada modelo según cada casa.

```r
# ================ COMPARACIÓN FINAL ================

clasificacion_real <- data.frame(
  Equipo = c("Real Madrid", "Barcelona", "Girona", "Ath Madrid", "Ath Bilbao",
             "Sociedad", "Betis", "Villarreal", "Valencia", "Alaves",
             "Osasuna", "Getafe", "Celta", "Sevilla", "Mallorca",
             "Las Palmas", "Vallecano", "Cadiz", "Almeria", "Granada"),
  Puntos_Reales = c(95, 85, 81, 76, 68, 60, 57, 53, 49, 46,
                    45, 43, 41, 41, 40, 40, 38, 33, 21, 21)
)

calcular_error <- function(tabla_modelo, nombre_modelo) {
  comp <- merge(clasificacion_real, tabla_modelo, by = "Equipo", all.x = TRUE)
  comp$Error_Abs <- abs(comp$Puntos_Reales - comp$Puntos_Totales)
  comp$Modelo <- nombre_modelo
  return(comp)
}

calcular_errores_completos <- function(tabla_modelo, nombre_modelo) {
  comp <- merge(clasificacion_real, tabla_modelo, by = "Equipo", all.x = TRUE)
  comp$Error_Abs <- abs(comp$Puntos_Reales - comp$Puntos_Totales)
  comp$Pos_Real <- rank(-comp$Puntos_Reales, ties.method = "min")
  comp$Pos_Pred <- rank(-comp$Puntos_Totales, ties.method = "min")
  comp$Error_Pos <- abs(comp$Pos_Real - comp$Pos_Pred)
  comp$Modelo <- nombre_modelo
  return(comp)
}

# Comparaciones
comparaciones <- rbind(
  calcular_error(tabla_bet365, "Bet365"),
  calcular_error(tabla_bw, "BetWay"),
  calcular_error(tabla_wh, "WilliamHill"),
  calcular_error(tabla_vc, "VC")
)

# MAE por modelo
mae_modelos <- comparaciones %>%
  group_by(Modelo) %>%
  summarise(MAE = mean(Error_Abs, na.rm = TRUE))
print(mae_modelos)

# Errores de posición
errores_todos <- rbind(
  calcular_errores_completos(tabla_bet365, "Bet365"),
  calcular_errores_completos(tabla_bw, "BetWay"),
  calcular_errores_completos(tabla_wh, "WilliamHill"),
  calcular_errores_completos(tabla_vc, "VC")
)

resumen_error_modelos <- errores_todos %>%
  group_by(Modelo) %>%
  summarise(MAE_Puntos = mean(Error_Abs, na.rm = TRUE),
            MAE_Posicion = mean(Error_Pos, na.rm = TRUE))
print(resumen_error_modelos)
```

Se calcula el error medio absoluto (MAE) en puntos y en posiciones para cada casa de apuestas bajo el modelo SVM OvA, comparando la clasificación predicha con la real. Este análisis cuantifica la precisión del modelo, permitiendo identificar qué casa ha ofrecido una predicción más ajustada a la realidad.

```r
# ================ TABLA RESUMEN FINAL ================
# RESUMEN TABLA RF
resumen_tablaRF <- clasificacion_real %>%
  mutate(
    Puntos_Bet365 = tabla_final_Bet365$Puntos_Totales[match(Equipo, tabla_final_Bet365$Equipo)],
    Puntos_BetWay = tabla_final_BetWay$Puntos_Totales[match(Equipo, tabla_final_BetWay$Equipo)],
    Puntos_WilliamHill = tabla_final_WilliamHill$Puntos_Totales[match(Equipo, tabla_final_WilliamHill$Equipo)],
    Puntos_VC = tabla_final_VC$Puntos_Totales[match(Equipo, tabla_final_VC$Equipo)],
    Pos_Real = rank(-Puntos_Reales, ties.method = "min"),
    Pos_Bet365 = rank(-Puntos_Bet365, ties.method = "min"),
    Pos_BetWay = rank(-Puntos_BetWay, ties.method = "min"),
    Pos_WilliamHill = rank(-Puntos_WilliamHill, ties.method = "min"),
    Pos_VC = rank(-Puntos_VC, ties.method = "min"),
  ) %>%
  select(Equipo, Puntos_Reales,
         Puntos_Bet365, Puntos_BetWay, Puntos_WilliamHill, Puntos_VC,
         Pos_Real, Pos_Bet365, Pos_BetWay, Pos_WilliamHill, Pos_VC,
         )
print(resumen_tablaRF)
View(resumen_tablaRF)


# RESUMEN TABLA OvO
resumen_tablaOvO <- clasificacion_real %>%
  mutate(
    Puntos_Bet365 = tabla_ovo_Bet365$Puntos_Totales[match(Equipo, tabla_ovo_Bet365$Equipo)],
    Puntos_BetWay = tabla_ovo_BetWay$Puntos_Totales[match(Equipo, tabla_ovo_BetWay$Equipo)],
    Puntos_WilliamHill = tabla_ovo_WilliamHill$Puntos_Totales[match(Equipo, tabla_ovo_WilliamHill$Equipo)],
    Puntos_VC = tabla_ovo_VC$Puntos_Totales[match(Equipo, tabla_ovo_VC$Equipo)],
    Pos_Real = rank(-Puntos_Reales, ties.method = "min"),
    Pos_Bet365 = rank(-Puntos_Bet365, ties.method = "min"),
    Pos_BetWay = rank(-Puntos_BetWay, ties.method = "min"),
    Pos_WilliamHill = rank(-Puntos_WilliamHill, ties.method = "min"),
    Pos_VC = rank(-Puntos_VC, ties.method = "min"),
  ) %>%
  select(Equipo, Puntos_Reales,
         Puntos_Bet365, Puntos_BetWay, Puntos_WilliamHill, Puntos_VC,
         Pos_Real, Pos_Bet365, Pos_BetWay, Pos_WilliamHill, Pos_VC,
  )
print(resumen_tablaOvO)
View(resumen_tablaOvO)


# RESUMEN TABLA OvA
resumen_tablaOvA <- clasificacion_real %>%
  mutate(
    Puntos_Bet365 = tabla_bet365$Puntos_Totales[match(Equipo, tabla_bet365$Equipo)],
    Puntos_BetWay = tabla_bw$Puntos_Totales[match(Equipo, tabla_bw$Equipo)],
    Puntos_WilliamHill = tabla_wh$Puntos_Totales[match(Equipo, tabla_wh$Equipo)],
    Puntos_VC = tabla_vc$Puntos_Totales[match(Equipo, tabla_vc$Equipo)],
    Pos_Real = rank(-Puntos_Reales, ties.method = "min"),
    Pos_Bet365 = rank(-Puntos_Bet365, ties.method = "min"),
    Pos_BetWay = rank(-Puntos_BetWay, ties.method = "min"),
    Pos_WilliamHill = rank(-Puntos_WilliamHill, ties.method = "min"),
    Pos_VC = rank(-Puntos_VC, ties.method = "min"),
  ) %>%
  select(Equipo, Puntos_Reales,
         Puntos_Bet365, Puntos_BetWay, Puntos_WilliamHill, Puntos_VC,
         Pos_Real, Pos_Bet365, Pos_BetWay, Pos_WilliamHill, Pos_VC,
  )
print(resumen_tablaOvA)
View(resumen_tablaOvA)
```

Por último, se construyen tres tablas comparativas que integran los puntos y posiciones reales junto a los estimados por cada modelo (RF, SVM OvO y SVM OvA) para las cuatro casas de apuestas. Estas tablas permiten visualizar de forma clara cuál combinación de modelo y casa se aproxima más a la clasificación real, tanto en puntos como en posiciones. Sirven como base sólida para identificar el enfoque predictivo más preciso.