Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Estadística

Análisis y predicción de resultados en partidos de tenis

Autor: Pablo Peláez Marín
Tutor: Luis Ángel García Escudero
Curso 2023-2024

Agradecimientos

En primer lugar, a mi tutor de TFG, que aceptó la propuesta de dirigir un proyecto que tenía pensado desde hace unos años. Le agradezco el tiempo, esfuerzo y correcciones que ha realizado, ayudándome siempre que fue necesario.

También en el ámbito académico a todos los profesores, que impartieron sus asignaturas con entusiasmo y me permitieron adquirir conocimientos que he puesto en práctica en este trabajo y seguro que me serán esenciales en el futuro.

Un gran reconocimiento a los compañeros de la universidad, que muchos de ellos se convirtieron en grandes amigos.

Por último, un agradecimiento especial a mi familia, amigos y personas que han pasado por mi vida en estos últimos años. Su apoyo incondicional me ha permitido trabajar día tras día.

Resumen

En este trabajo se desarrolla una herramienta que permite el análisis de los jugadores de tenis profesionales a partir de los datos disponibles de cada uno de sus partidos. Concretamente, se obtiene información sobre la evolución de los torneos, sus características, el patrón de juego de los tenistas y las relaciones que se establecen entre las distintas variables. Esto permitiría descubrir a un jugador y a su cuerpo técnico cuál puede ser la mejor estrategia para ganar un partido concreto.

Adicionalmente, se aplican técnicas de análisis de datos para calcular la probabilidad de victoria de cada jugador/a en un partido oficial. Estas técnicas serán Support Vector Machines, Regresión logística, Random Forest, Gradient Boosting y modelo Bradley Terry. En este sentido, se evalúa y compara el rendimiento de todas estas técnicas para los datos pertenecientes al ámbito del tenis.

Estas herramientas y procedimientos son aplicables tanto al circuito masculino (ATP) como al femenino (WTA).

Abstract

In this project, a tool that allows the analysis of professional tennis players is developed throughout the data available from each of their matches. Specifically, this enables us to obtain the evolution of the tournaments, characteristics, game patterns of the players and the relationships established between the different variables. This could allow a player and his coaching team to discover the best strategy to win a specific match.

In addition to that, data analysis tools are used to calculate the probability of win of each player on a certain official match. These techniques will be: Support Vector Machines, Logistic Regression, Random Forest, Gradient Boosting and the Bradley Terry model. In this regard, the previously mentioned techniques are evaluated and the performance of each of them is compared.

This analysis will be developed both for the men's (ATP) and women's (WTA) tours.

Índice general

Capítulo 1

Introducción

1.1. Contexto del problema y objetivos

El trabajo se sitúa dentro del ámbito del análisis de datos y aprendizaje automático para datos sobre tenistas profesionales. Hay diversas fuentes de datos que se combinarán para extraer la información deseada.

El objetivo será explicar el funcionamiento de los torneos de tenis, las características de los jugadores/as profesionales y realizar predicciones acertadas sobre los resultados de partidos concretos. Para ello, se explorarán técnicas estadísticas estudiadas en las asignaturas de la titulación, mas alguna ampliación.

1.2. Estructura de la memoria

El documento presenta una introducción, para posteriormente en la Sección 2 mostrar una descripción de la metodología empleada, basada en el aprendizaje supervisado, aprendizaje no supervisado y la inclusión del modelo de Bradley Terry. En la Sección 3 se presentan unas nociones sobre el tenis, ya que tiene reglamento que puede ser útil para comprender las demás partes del trabajo. En la Sección 4 se realiza una descripción de la base de datos y se detallan las transformaciones de los datos. En la Sección 5 se pone en práctica toda la metodología y aparecen los resultados obtenidos, además de comparaciones de las mejores técnicas. Finalmente, se llega a las conclusiones y líneas futuras en la Sección 6. Las imágenes y tablas tienen su propio índice al final, y por último está añadido el código desarrollado.

1.3. Herramientas

Para la elaboración de este trabajo se utilizará software adecuado. Para su elección se ha tenido en cuenta el manejo del alumno con cada una de esas herramientas y en las ventajas que pueden ofrecer respecto a otras opciones que ofrecen el mismo servicio.

1.3.1. R

El lenguaje de programación que se utiliza para desarrollar el trabajo es R, un lenguaje de código abierto e interpretado. En principio está diseñado para implementar técnicas estadísticas, pero también cuenta con otras funcionalidades.

Permite crear programas, aplicaciones, mediciones estadísticas, gran variedad de gráficos, modelos lineales, no lineales, series temporales, problemas de clasificación y muchas más cosas [1].

Aunque R tiene su propio equipo de desarrollo, permite que los usuarios puedan crear librerías que formen parte del proyecto R y puedan ser utilizadas por otros usuarios.

Presenta una documentación completa en formato LaTeX y R puede integrarse e interpretarse con otros lenguajes de programación [1].

Todas estas ventajas se suman a que está disponible para todos los sistemas operativos. La mayor desventaja quizá sea que no es de los lenguajes de programación más rápidos, pero como los datos que se manejan en este trabajo no son excesivamente grandes, se opta por R como herramienta principal.

Este lenguaje de programación contiene numerosos paquetes y se han utilizado algunos de ellos. Por ejemplo, el paquete **randomForest**, **e1071** que permite implementar SVM y **xgboost** usado para Gradient Boosting. También se ha puesto en práctica **BradleyTerry2**, que permite ajustar modelos de regresión logística para datos con muestras pareadas. Este paquete resulta muy útil para el ámbito de este trabajo ya que está pensado para modelar la probabilidad de victoria en enfrentamientos entre dos individuos, además de otras aplicaciones.

1.3.2. RStudio

El entorno de desarrollo que se usa en este proyecto es RStudio, ya que está diseñado para trabajar con el lenguaje de programación R. En concreto se usa la última versión estable disponible, correspondiente a la de diciembre del año 2022.

Algunas de sus prestaciones son el editor de código fuente, que permite ejecutarlo desde ahí. También posee una herramienta de depuración interactiva, consola, visor de datos y distintos espacios de trabajo, entre otras funcionalidades [2].

1.3.3. Overleaf

Se trata de la herramienta que se ha utilizado para la redacción de la memoria. Está pensada para la creación colaborativa de texto principalmente científico y de investigación. En este caso permite la edición en código LaTeX de manera online con guardado automático, pudiendo añadir todas las fórmulas, imágenes y referencias de manera rápida y sencilla [3].

Capítulo 2

Metodología

2.1. Técnicas descriptivas y aprendizaje no supervisado

2.1.1. Gráficos de variables y correlaciones

Se ha trabajado con distintas visualizaciones gráficas de las variables en nuestros conjuntos de datos. Se pretende mostrar la información que aportan los datos de la manera visual más clara posible. Para ello se emplean gráficos de dispersión, gráficos de barras, gráficos de líneas y gráficos de cajas. También, a nivel descriptivo, se ha ampliado la información con tablas explicativas, cuando ha sido conveniente.

En los conjuntos de datos con un gran número de variables existirán ciertas correlaciones, que miden de qué manera están relacionadas las variables entre sí y que en este caso resultan de gran interés. Cuando se comparan las variables dos a dos, la correlación, en caso de que exista, puede ser positiva o negativa. La correlación positiva indica que a medida que el valor de una variable aumenta, el valor de la otra también aumenta. Por otro lado, si el valor disminuye, el valor para la otra variable también disminuye. La correlación negativa indica que a medida que el valor de una variable aumenta, el valor de la otra disminuye. Una medida habitual es el coeficiente de correlación de Pearson, cuyo valor está comprendido en el intervalo $[-1, 1]$. Valores próximos a -1 indican una correlación negativa fuerte, mientras que valores próximos a 1 indican correlación positiva fuerte. Valores próximos a 0 quieren decir que no hay correlación entre las variables.

2.1.2. Análisis de Componentes Principales

Es un método estadístico no supervisado utilizado para reducir la dimensionalidad en el conjunto de datos, pero conservando al máximo la información que aportan las variables. Así, el conjunto de variables se reducirá a una serie de "componentes principales".

Para implementar este método se utilizan como fundamento algebraico los autovalores y autovectores de la matriz de varianzas y covarianzas. Los autovectores de una matriz cuadrada son los que son múltiplo de ese vector al multiplicarlos

por esa misma matriz. El orden de las componentes sigue el de los autovalores por orden decreciente y cada componente principal se corresponde con un autovector. Por lo tanto, la primera componente es la correspondiente con el autovector asociado al autovalor más alto [4].

2.1.3. Análisis Cluster

Al tratarse de una técnica de análisis no supervisado, tampoco se tiene información a priori sobre la clase a la que pertenece cada elemento. El análisis cluster se utiliza para identificar grupos denominados "clusters". La caracteística fundamental es que los "clusters" agrupan elementos en función de su similitud. Los grupos deben ser homogéneos, con un número de elementos razonable para cada grupo.

Uno de los tipos de clustering más habitual es k-medias, el cual se utilizará en este trabajo. Se debe fijar un número de "clusters" de antemano y cada "cluster" está asociado a un centro, que se obtienen minimizando una función objetivo. El algoritmo de k-medias lo que hace es mover estos centros iterativamente para minimizar la varianza total dentro de los "clusters". Dado un conjunto inicial de centros, el algoritmo de k-medias alterna dos pasos hasta alcanzar la convergencia [5]:

- Para cada centro, se identifica el subconjunto de puntos que está más cerca de un centro que de cualquier otro centro.

- Se calculan las medias de cada variable para los puntos de cada "cluster". Cada vector medio se convierte en el nuevo centro para ese "cluster".

2.2. Técnicas predictivas

En esta seccíón se describen las técnicas utilizadas en el proyecto para la predicción de los resultados de partidos de tenis. Todas ellas forman parte del aprendizaje supervisado correspondiente a la clasificación, ya que se pretende clasificar cada partido según el ganador. El aprendizaje supervisado parte de un conjunto de datos de entrenamiento, que sirve para entrenar o "aprender" de los datos. Por otra parte, se suele utilizar un conjunto de datos de prueba, que permite evaluar los modelos predictivos ajustados.

Al tratarse de un problema de clasificación con una clase respuesta binaria, resultarán de utilidad las curvas ROC, que representan en el eje de abscisas la tasa de falsos positivos y en el eje de ordenadas la tasa de verdaderos positivos [6].

2.2.1. Regresión Logística

Con este método se pretende modelar el logaritmo de las razones de probabilidades a posteriori de las clases a través de funciones lineales, asegurando que la suma de estas probabilidades sea 1 y que todas ellas están comprendidas en el intervalo $(0, 1)$. Los parámetros de estas relaciones lineales se ajustan por máxima verosimilitud, suponiendo que [5]:

$$\log\left(\frac{Pr(G=1|X=x)}{Pr(G=K|X=x)}\right) = \beta_{10} + \beta_1^T x$$

$$\log\left(\frac{Pr(G=2|X=x)}{Pr(G=K|X=x)}\right) = \beta_{20} + \beta_2^T x$$

$$y$$

$$\log\left(\frac{Pr(G=K-1|X=x)}{Pr(G=K|X=x)}\right) = \beta_{(K-1)0} + \beta_{K-1}^T x$$

En este tipo de modelos entra en juego el concepto de AIC (Criterio de Información de Akaike), que se trata de una medida que indica el equilibrio que hay entre la calidad de ajuste y la simplicidad del modelo. Resulta preferible una medida de AIC baja. Su cálculo viene determinado por la siguiente fórmula:

$$AIC = 2k - 2\ln(L)$$

siendo k el número de parámetros estimados en el modelo y L el valor máximo de la función de verosimilitud obtenido en el ajuste del modelo.

En el momento de seleccionar un modelo, también se pueden utilizar otros métodos de selección:

- Forward Selection (Selección hacia adelante): empieza con un modelo vacío y va añadiendo variables una por una hasta llegar a un modelo óptimo.
- Backward Selection (Selección hacia atrás): empieza con un modelo que contiene todas las variables y va eliminando algunas hasta llegar a un modelo óptimo
- Both (Hacia adelante y hacia atrás): va alternando la inclusión y eliminación de variables.

2.2.2. Random Forest

Traducido como bosque aleatorio, es una técnica de aprendizaje supervisado aplicada en problemas de regresión y clasificación. Tiene bastante utilidad ya que ofrece resultados con una precisión razonable y no es prácticamente dependiente

de parámetros de ajuste a elegir correctamente. En esencia es una combinación de árboles de decisión que mediante técnicas remuestreo y aleatorización en las variables evita el sobreajuste [7].

El método Random Forest es un caso particular del método de bagging, generándose muestras patrón bootstrap y desarrollándose al máximo los árboles asociados para unas pocas variables elegidas aleatoriamente. En cada nodo se utilizan $\sqrt{p}$ variables de las p variables originales. Se guardan los árboles resultantes y las nuevas observaciones se asignan a clases usando "votación de la mayoría" [5].

Otra característica de los Random Forest es que permite estimar el error de generalización usando un procedimiento conocido como "out-of-bag".

Su principal desventaja es la difícil interpretación del modelo resultante, aunque se compensa con una medida de la contribución de cada variable a la precisión final.

2.2.3. Support Vector Machines

Este método de aprendizaje supervisado trata de centrarse en las observaciones fronterizas entre grupos para realizar la clasificación [5]. En el caso separable, esa frontera es un hiperplano separante que maximiza la distancia entre las observaciones más cercanas a dicho hiperplano. Cuando las observaciones no son separables linealmente, se pueden usar kernels (núcleos) para llevar los datos a un espacio de dimensión mayor donde sí sean separables y permitir ciertas violaciones, pero añadiendo una penalización a esas violaciones. Los dos tipos de núcleos que más se usan son el polinómico y el de base radial, cuyos parámetros se determinan por validación cruzada.

2.2.4. Gradient Boosting

Esta técnica de aprendizaje supervisado es también válida para problemas de clasificación. Se construyen modelos de árboles secuencialmente que van reduciendo el error teniendo en cuenta en mayor medida las observaciones más complejas de clasificar pero sin caer en sobreajuste.

Para lograr el objetivo de encontrar una función que se aproxime de manera adecuada a la variable respuesta, se utiliza una función de pérdida, que se optimiza minimizando el gradiente. También intervienen los términos de regularización, que ayudan a disminuir el sobrajuste. Por ejemplo, el número de iteraciones y la profundidad de los árboles son términos de regularización comunmente utilizados.

2.3. Modelo de Bradley Terry

El modelo Bradley-Terry [8] en un enfrentamiento entre un jugador i y un jugador j supone que la probabilidad de que i gane a j es:

$$P(i > j) = p_i/(p_i + p_j).$$

Si suponemos que $p_i = e^{\lambda_i}$, entonces el modelo también se puede expresar de la forma:

$$logit(P(i > j)) = \log\left(\frac{P(i > j)}{1 - P(i > j)}\right) = \log\left(\frac{P(i > j)}{P(j > i)}\right) =$$
$$= \log(P(i > j)) - \log(P(j > i)) = \lambda_i - \lambda_j.$$

Suponiendo independencia entre las observaciones, los parámetros p_i pueden estimarse por máxima verosimilitud. Usaremos con este fin el paquete **Bradley-Terry2** de R, que realiza llamadas internas a la función `glm()` de modelos lineales generalizados. A continuación se detalla la manera de maximizar la verosimilitud para el modelo específico de Bradley Terry [9]:

La verosimilitud se obtiene desde:

$$\prod_{ij}[P(i > j)]^{w_{ij}}$$

donde w_{ij} es el número de veces que el jugador i gana al jugador j y $w_{ij} = 0$.

Nuestro objetivo es maximizar esta verosimilitud en los parámetros

$$p = [p_1, \ldots, p_n].$$

Partimos de la log-verosimilitud $L(p)$ definida como

$$L(p) = \ln \sum_{i=1}^{n}\sum_{j=1}^{n}[P(i > j)]^{w_{ij}} = \sum_{i=1}^{n}\sum_{j=1}^{n}\ln\left(\frac{p_i}{p_i + p_j}\right)^{w_{ij}} =$$
$$= \sum_{i=1}^{n}\sum_{j=1}^{n}[w_{ij}\ln(p_i) - w_{ij}\ln(p_i + p_j)]$$

El máximo se obtiene diferenciando $L(p)$ respecto a cada p_i, e igualando a 0 para obtener que los p_i en p deben verificar:

$$p_i = \frac{\sum_{j\neq i} w_{ij}}{\sum_{j\neq i}\left(\frac{w_{ij}+w_{ji}}{p_i+p_j}\right)}, \quad \text{para todo } i.$$

No hay una forma correcta de obtener estos p_i, pero sí un procedimiento iterativo para su obtención.

El paquete **BradleyTerry2** permite incluir covariables en la obtención de los parámetros λ_i, de tal forma que sean dependientes de valores $x_{i1}, ..., x_{ip}$ que el

tenista i-esimo tome en p variables explicativas, a través del siguiente predictor lineal:

$$\lambda_i = \sum_{r=1}^{p} \beta_r x_{ir} + \varepsilon_i,$$

donde ϵ_i es un error aleatorio de media 0 [8].

Capítulo 3

El deporte del tenis

3.1. Reglas

El tenis es un deporte con muchos años de antigüedad, cuyos inicios constan del siglo XVIII en el continente europeo. Ha pasado de ser un deporte practicado exclusivamente por las clases altas a ser uno de los deportes más populares, con millones de aficionados por todo el mundo.

Para practicarlo se necesita esencialmente una raqueta con cuerdas y una pelota. El terreno de juego es una pista de forma rectangular delimitada por una serie de líneas que indican los límites en los que puede botar dicha pelota. En el medio hay una red que separa la pista en dos mitades de dimensiones exactamente iguales. La pista puede ser de cualquier material, pero profesionalmente se reconocen tres tipos: cemento, hierba y tierra batida [10].

El tenis se puede jugar de manera individual (1 contra 1) o en parejas (2 contra 2). Esta última se denomina la modalidad de dobles, que a su vez puede ser masculina, femenina o mixta. El objetivo esencial es golpear la pelota con la raqueta de manera que caiga en el campo contrario y lograr que el rival no la devuelva. Solamente se puede golpear una vez la pelota cada vez que entra a nuestro campo y no se puede tocar la red ni golpear en el campo de adversario. Además, la pelota puede botar como máximo una vez en nuestro campo antes de cada golpeo. La excepción es la modalidad de tenis en silla de ruedas, en la cual se permite que la pelota bote dos veces.

Una característica que diferencia al tenis de la mayoría de deportes es que el tiempo de un partido no está fijado de antemano, añadiendo emoción e incertidumbre a los encuentros. La manera de determinar quién gana un partido es la siguiente: cuando un jugador no consigue devolver la bola dentro de los límites, el rival se anota un punto. Con un mínimo de cuatro puntos se gana un juego, con seis juegos se gana un set y el primero que gane dos o tres sets gana el partido, dependiendo del tipo de torneo.

3.2. Vocabulario específico

El deporte del tenis está repleto de terminología específica y, aunque alguna nos suene de verla en la televisión, conviene repasar algunas palabras ya que formarán

parte del análisis posterior. Cabe señalar que cada uno de estos términos es válido tanto para modalidad invividual como dobles.

- **Servicio**: también denominado saque, es el golpe que permite poner la bola en juego. Es el propio jugador quien se lanza la bola con la mano para posteriormente golpearla con la raqueta. No está permitido pasar la línea de fondo y la pelota debe botar en el cuadro de saque opuesto al lado del que servimos. Hay dos oportunidades para realizar este golpeo.
- **Resto**: también denominado devolución, es el golpe inmediatamente posterior al servicio y lo realiza el jugador rival, esta vez con una sola oportunidad pero con toda la pista disponible para que bote la pelota.
- **Juego**: es el sistema de puntuación básico en el tenis. Cuando un jugador gana un punto se anota 15 en el marcador, si gana dos puntos se anota 30, con tres se anota 40 y con cuatro se anota un juego, con la particularidad de que debe haber mínimo dos puntos de diferencia respecto al jugador rival. Es decir, si el marcador es de 40-40 y un jugador gana el punto, este se anotará ventaja en el marcador y con otro punto más ganaría el juego. Si por contra es el otro jugador quien gana el punto, se vuelve al marcador 40-40. En la modalidad de dobles no existe esta diferencia de dos, por lo que se juega un "punto decisivo" cuando el marcador es 40-40.
- **Set**: determina que un jugador ha ganado 6 juegos antes que el rival. Cuando alguien consigue ganar un set se reinicia la cuenta de juegos para disputar un nuevo set. La particularidad es que en caso de empate a 5 juegos habrá que llegar a 7 juegos, pero en caso de empate a 6 juegos ya no hay que ganar por diferencia de dos juegos y se disputa en su lugar un "tie break", cuyo funcionamiento se detalla a continuación.
- **Tie Break**: se trata de la manera de desempatar un set cuando el marcador es 6-6. Uno de los jugadores comienza sacando el primer punto, y a partir de ahí los jugadores se turnan para sacar cada dos puntos. Ahora la puntuación no sigue el sistema de los juegos tradicionales, sino que se cuenta de manera natural desde el 0 hasta el 7. El jugador que antes llegue a siete puntos con una diferencia de al menos dos puntos gana el tie break. El encuentro se pospone hasta que alguien gana por una diferencia de mínimo dos puntos, pudiendo haber resultados de 8-6, 9-7, 10-8, etc. No obstante, el marcador del set cuando se dispute un tie break será siempre 7-6, añadiendo entre paréntesis la puntuación menor del tie break. Por ejemplo: 7-6(5).
- **Break Point**: traducido a punto de rotura/quiebre, es el momento en el que el jugador que realiza el resto se encuentra a tan solo un punto de ganar un juego. Obtiene relevancia ya que lo habitual es que el jugador que saca gane el juego, por lo que estos "break point" serán más bien escasos [11].

- **Ace**: se trata de un saque en el cual el jugador que lo devuelve no es capaz de tocar la pelota. Para conseguirlo se requiere de una combinación entre potencia y colocación [11].

- **Doble Falta**: ocurre cuando el jugador que saca no consigue poner en juego la pelota con las dos oportunidades que se le otorgan.

3.3. Competiciones y sistema de clasificación

Para comprender esta sección conviene diferenciar el circuito masculino del femenino, ya que están regidos por asociaciones diferentes que aplican sus propias normas. La Asociación de Tenistas Profesionales (ATP) es la encargada de la organización del tenis masculino, mientras que la Asociación de Tenis Femenino (WTA) es la encargada del tenis femenino. La modalidad de dobles está incluida también en estas asociaciones.

En el circuito masculino hay competiciones prácticamente todas las semanas desde enero hasta noviembre. En orden de relevancia decreciente son Grand Slam, Masters 1000, ATP 500, ATP 250, Challenger y torneos ITF. Cada torneo tiene sus propias fechas, organización y pistas.

La clasificación o ranking se calcula sumando la puntuación obtenida en cada torneo, con la particularidad de que únicamente cuentan los torneos disputados en las últimas 52 semanas. No hay límite de participación en cantidad de torneos, pero solamente puntúan los 19 mejores resultados. La excepción es el torneo ATP Finals, que es el último torneo torneo de la temporada y cuenta para el ranking de manera adicional, pero solo pueden acceder a él los 8 mejores tenistas de esa misma temporada [12].

De manera particular, cada torneo otorga unos puntos según la ronda alcanzada. Dependiendo de la categoría del torneo se otorgan más o menos puntos. Los Grand Slam adjudican 2000 puntos al ganador, 1200 al finalista, y así se va descendiendo hasta los 10 puntos que otorga participar en la primera ronda. Los Masters 1000 otorgan 1000 puntos al ganador, los torneos ATP otorgan 500 o 250, los Challenger y Futures tienen el rango entre 175 y 10 puntos máximo.

La utilidad de esta clasificación entre jugadores reside en que el número de participantes en cada torneo es limitado, por lo que el ranking determina qué jugadores pueden participar en el torneo, aunque siempre hay plazas reservadas para invitaciones o jugadores de fases previas. Además, dentro de cada torneo hay un sistema de cabezas de serie ordenados por ranking, permitiendo que los mejores jugadores no se enfrenten entre sí en las primeras rondas.

En cuanto al tenis femenino, muchas características son iguales pero conviene destacar algún cambio. Por ejemplo, para el ranking tan solo cuentan los 16 mejores torneos en cuanto a puntuación obtenida. Los Grand Slam y torneos WTA importantes varían un poco la puntuación otorgada a las finalistas, semifinalistas,

etc.

Otra particularidad importante es que en el circuito femenino todos los partidos son al mejor de 3 sets, mientras que en el masculino los Grand Slam son al mejor de 5 y todos los demás al mejor de 3.

3.4. Cadenas de Markov

Para entender mejor el funcionamiento de un partido de tenis desde el ámbito matemático, se puede explicar una aplicación de las cadenas de Markov al juego del tenis. Cabe resaltar que no es el enfoque que se seguirá a lo largo del trabajo.

En una cadena de Markov, visto como un proceso en tiempo discreto $\{X_n\}_{n=0}^{\infty}$, el conjunto de estados futuros posibles están limitados exclusivamente por el estado actual, que es la puntuación en cualquier momento del juego [13] y verifican la hipótesis con un espacio de estados discreto E y la propiedad de Markov:

$P(X_{n+1} = j|X_0 = i_0, \ldots, X_{n-1} = i_{n-1}, X_n = i) = P(X_{n+1} = j|X_n = i)$ para cualquier elección de $i_0, \ldots, i_{n-1}, i, j \in E$ tal que $P(X_0 = i_0, \ldots, X_{n-1} = i_{n-1}, X_n = i) > 0$.

Se consideran dos jugadores A y B que disputan un partido de tenis. En este caso se trata de modelizar la disputa de un juego, considerando que el jugador A saca y gana cada punto con una probabilidad p, mientras que el jugador B gana cada punto con probabilidad q [14] y que el resultado de los distintos puntos se puede considerar independiente. El estado inicial es el de arriba, con marcador 0-0. Los dos estados finales son "GameA" y "GameB", diferenciando si gana el jugador A o el B.

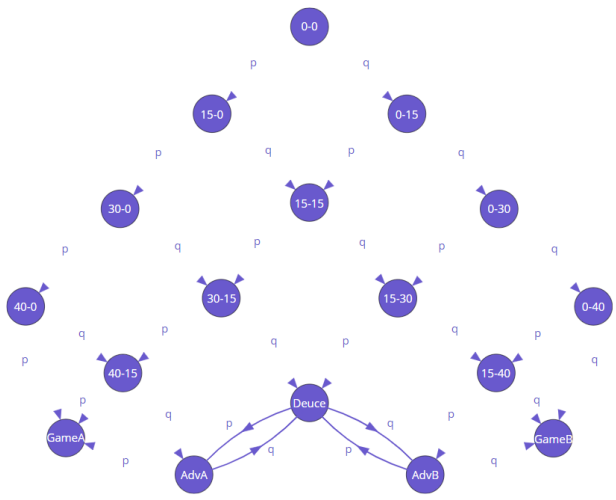

Figura 3.1: Grafo juego de tenis [15]

El gráfico de la Figura 3.2 muestra en el eje de abscisas la probabilidad de que el jugador A gane un punto y en el eje de ordenadas la probabilidad de que el jugador gane un juego en el que está comprendido ese punto.

Siendo p la probabilidad de que el jugador A gane el punto, la fórmula para esa función que representa la probabilidad de ganar un juego es se puede ver [4] que coincide con

$$f(p) = \frac{p^2}{p^2 + (1-p)^2}$$

Por ejemplo, si el jugador gana un punto con probabilidad 0.6, ganará el juego con una probabilidad próxima a 0.75.

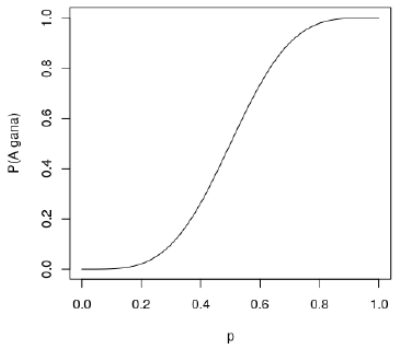

Figura 3.2: Probabilidad de ganar un juego en función de la probabilidad de ganar un punto

Capítulo 4

Base de datos

4.1. Origen y características

Todos los datos usados en este TFG provienen del repositorio GitHub creado por Jeff Sackmann. Se encuentran disponibles para descargarlos sin ningún tipo de permiso ya que es un repositorio público. La única limitación es hacer un uso no comercial de los datos, lo cual se va a cumplir.

Se utilizan dos ramas concretas de este repositorio:

- Tennis_atp [16]: aquí se encuentra almacenada información sobre los jugadores masculinos, con el ranking, resultados y estadísticas variadas.

  Todos los conjuntos de datos vienen en formato "csv", es decir, archivos con los valores separados por comas.

  En cuanto al ranking, el primer dataset es de la década de los 70, pero se encuentra bastante incompleto ya que comienza en el año 1973 y no hay datos referentes a los puntos. Los primeros años de la década de los 80 también están incompletos, concretamente no hay datos correspondientes a los años 1981, 1982 y 1983. Los puntos de clasificación empiezan a almacenarse en la década de los 90 (atp_rankings_90s), por lo que a partir de este momento sí que están razonablemente completos como para poder trabajar con ellos. Las variables contenidas son la fecha de medición (`ranking_date`), ranking (`rank`), id del jugador (`player`) y puntos en la clasificación (`points`).

  También hay un dataset llamado atp_rankings_current y otro con el nombre wta_rankings_current que contienen información sobre la fecha del ranking a 1 de enero de 2024, ranking, identificador del jugador o la jugadora y puntos. Los nombres de estas variables son los mismos que para los demás conjuntos de datos referentes al ranking.

  Los resultados de los partidos con su respectiva información se encuentran agrupados por temporada y por nivel. Así, encontramos el circuito de torneos Future, circuito de torneos Challenger junto a las previas ATP y finalmente el circuito ATP. Este último será el más interesante ya que es donde participan los mejores jugadores y por lo tanto hay más datos. Por su parte, las estadísticas de cada partido se encuentran disponibles desde 1991 para

el circuito ATP. Por ejemplo, conjunto de datos de partidos disputados en el año 1991 se llama `datos_partidos_1991`. A continuación, se describe cada una de las variables de los dataset de partidos [16]:

- `tourney_id`: es el identificador único para cada torneo. Está representado por el año y un número de tres cifras separados por un guión.
- `tourney_name`: el nombre del torneo.
- `surface`: superficie de las pistas en las que se juega el torneo en cuestión.
- `draw_size`: tamaño del cuadro, que no tiene por qué coincidir con el número de jugadores participantes.
- `tourney_level`: Para el circuito masculino están "G" = Grand Slams, "M" = Masters 1000s, "A" = otros eventos, "C" = Challengers, ´´S" = Satellites/ITFs, "F" = ATP Tour finals, "D" = Copa Davis.

  También aparece la denominación "E" = exhibición, "J" = juniors y "T" = tenis en equipo.
- `tourney_date`: fecha del torneo.
- `match_num`: identificador único del partido dentro del torneo, generalmente desde el 300 hasta el 1.
- `winner_id`: identificador numérico del jugador que gana el partido.
- `winner_seed`: número de cabeza de serie en el torneo.
- `winner_entry`: "WC" = invitado, "Q" = procedente del previa, "LL" = lucky loser (sustituto de un jugador que causa baja), "PR" = ranking protegido, "ITF" = entrada por ITF.
- `winner_name`: nombre del ganador del partido.
- `winner_hand`: mano hábil del ganador, zurdo o diestro.
- `winner_ht`: altura del ganador.
- `winner_ioc`: nacionalidad del ganador.
- `winner_age`: edad del ganador.
- `score`: resultado del partido.
- `best_of`: número de sets que se pueden llegar a disputar en el partido.
- `round`: ronda del torneo.
- `minutes`: tiempo que dura el partido.
- `w_ace`: número de saques directos del ganador.
- `w_df`: número de dobles faltas del ganador.
- `w_svpt`: número de puntos disputados al saque por parte del ganador.

- `w_1stIn`: número de primeros servicios que el gandor del partido realiza con éxito.
- `w_1stWon`: número de puntos que gana con primeros servicios el ganador.
- `w_2ndWon`: número de puntos que gana con segundos servicios el ganador.
- `w_SvGms`: número de juegos al servicio por parte del ganador.
- `w_bpSaved`: número de bolas de break salvadas por parte del ganador.
- `w_bpFaced`: número de bolas de break disputadas por parte del ganador con su propio servicio.
- `winner_rank`: ranking del ganador.
- `winner_rank_points`: puntos en la clasificación del ganador.

Las variables que comienzan por "winner" o por "w" tienen su equivalente con las palabras "loser" y "l" para represntar la misma característica pero para el jugador que pierde el partido.

Esta rama también contiene un dataset llamado atp_players con el identificador de jugador, nombre, primer apellido, mano hábil, fecha de nacimiento, nacionalidad, altura e identificador en Wikidata. Aquí se ha cambiado el nombre de la variable `player_id` por `player` para hacer un "join" con el dataset referente al ranking y así poder trabajar con estos datos a la vez.

Los datos sobre torneos de dobles vienen recogidos en el periodo 2000-2020. Al no estar actualizados hasta la actualidad no va tener utilidad para el trabajo, por lo que nos centraremos en los torneos individuales.

Para descargar los datos de esta rama tennis_atp se puede utilizar el siguiente código R (equivalente para datos femeninos cambiando "atp" por "wta"):

```
zip <- 'TennisData_ATP.zip'
if (!file.exists(zip)){
  download.file('https://github.com/JeffSackmann/
  tennis_atp/archive/master.zip', destfile = zip)
}
carpeta <- 'tennis_atp-master/'
if (!dir.exists(carpeta)) unzip(zip)
```

- Tennis_wta [16]: aquí se encuentra almacenada información sobre las jugadoras femeninas, con el ranking, resultados y estadísticas variadas.

  Los datos vienen en el formato "csv" igual que para el circuito masculino, pero hay alguna diferencia. En concreto, el primer dataset para el ranking es de los años 80 y no de los 70. Lo que en masculino era Challenger, previas ATP y Future, en femenino se agrupa en previas WTA y torneos ITF.

Hay códigos alternativos para la variable `tourney_level` como son "P" = Premier, "PM" = Premier Mandatory, "I" = International. El nivel de los torneos ITFs viene determinado por el premio en miles de dólares, por ejemplo "15" = ITF. "T1", "T2", etc representan los torneos femeninos antiguos, mientras que "D" es para torneos por equipos. Además, no hay datos sobre los torneos en modalidad de dobles.

4.2. Transformaciones de variables

Se trabaja con los `datos_partidos_atp` y `datos_partidos_wta` en programas R diferentes ya que se han visto características diferentes y nos ayudará para realizar comparaciones entre el tenis femenino y masculino. Todas las modificaciones de este apartado están realizadas para los dos conjuntos de datos, a no ser que se mencione lo contrario. Se eliminan los espacios en blanco en algunos valores y se modifica el nombre de "Us Open" en los torneos para que sea siempre "US Open" [17].

En cuanto a la Copa Davis, se decide excluirla del estudio ya que hay demasiados datos perdidos y además es una competición que no puntúa para el ranking. Se trata de una competición que ha cambiado su formato en los últimos años ya que antes se jugaba al mejor de 5 sets, por lo que este factor influiría demasiado en algunos estudios como por ejemplo sobre la duración de los partidos. En el circuito femenino las competiciones equivalentes son la Fed Cup y BJK Cup. Para ser homogéneo en el criterio, se decide eliminar estas competiciones también.

Las ATP Tour Finals y Las NextGen Finals del circuito masculino no aparecen codificadas siempre de la misma manera en la variable `tourney_level`, por lo que se codifican todas con la letra "F".

En ambos conjuntos de datos se añaden las variables `date_new` y `year` que determinan la fecha en un formato correcto (YYYY-MM-DD) y el año en el que se disputa el partido. También se añade la variable `id` por si se llega a necesitar el identificador de cada partido empezando desde el 1. La variable `surface` se transforma de la siguiente manera: se crean las variables `surface_clay` y `surface_grass`, correspondientes a variables binarias que contienen el valor "1" si el partido se juega en tierra batida o hierba respectivamente, mientas que contienen el valor "0" si no se juega en esa superficie.

Todos los valores perdidos de las variables `winner_seed` y `loser_seed` pasan a procesarse como ceros. Después de este momento aparecen algo menos de 6000 filas con valores NA en los datos ATP y algo más de 20000 para los datos WTA, que son eliminadas de los dataset ya que aportarían problemas a la hora de entrenar los modelos. En los datos WTA pueden parecer demasiadas filas eliminadas, pero la mayor parte de los datos faltantes estaban en los años menos recientes, que justamente son los que menos valor tienen a la hora de realizar predicciones.

Posteriormente, se incorporan las variables `player1_id` y `player2_id`, que son

respectivamente el jugador con un valor de id menor y mayor para dicho partido. De esta manera se almacenan los jugadores sin distinguir quién es el ganador y así no hay pistas a la hora de realizar predicciones. Con estos dos jugadores diferenciados se crean las variables que contienen datos sobre los jugadores, como por ejemplo la `edad`, `ranking`, `altura`, `aces`, `dobles faltas`, etc. Otra varible útil que se añade es la cuenta de victorias de cada jugador a lo largo de su carrera, `win_count`. A partir de ella surge la necesidad de crear la variable de proporción de victorias llamada `prop_vict`. Para medir el estado de forma reciente, se añade la variable `prop_last10matches`, que mide la proporción de victorias en los últimos 10 partidos para los jugadores. Esto nos permitirá predecir partidos a partir del estado de forma reciente de cada jugador. De esta manera se recogen los mismos partidos recientes para cada jugador, ya que recogiendo periodos recientes de 1 mes, 2 meses, etc, los partidos jugados por cada jugador son muy distintos. Además, probando este tipo de variable mediante meses resulta ser no significativa en los modelos.

Todas las variables del párrafo anterior se restan entre sí para tener almacenada la diferencia entre el jugador 1 y el jugador 2. La última variable añadida es `player1_win`, que es binaria y contiene el valor 1 si gana el jugador 1 (valor 0 en caso contrario). Esta última será la variable respuesta en los modelos del trabajo.

Todos estos datos precisan de una transformación. Se consideran diferentes tipos de normalización. En primer lugar, se descarta la normalización min-max, en la cual todos los valores estarían comprendidos entre 0 y 1, siendo próximos a 0 los valores más bajos del conjunto de datos original. El inconveniente principal es que hay algún valor atípico correspondiente a partidos que han durado bastante más tiempo de lo habitual, por lo que esta normalización obtiene muchos valores cercanos a 0 y pocos valores cercanos a 1.

Se opta por una normalización estándar de los datos. Así, se centran los datos en cero (restando la media) y se escalan con una desviación típica de uno (dividiendo por la desviación típica). Para ello se ha utilizado la función `scale()` de R con sus parámetros *center* y *scaler* establecidos en el valor `TRUE`. En alguna ocasión se han usado transformaciones logarítmicas porque pensamos que proporcionan escalas más naturales para los datos por las fuertes asimetrías encontradas.

Para trabajar con el paquete **BradleyTerry2** hay que tener en cuenta una transformación adicional. Esta es que los jugadores sean factores con los mismos niveles. Es decir, `player1_id` y `player2_id` deben contener el nombre de los mismos jugadores, apareciendo mínimo una vez en `player1_id` y `player2_id`. Para lograrlo, se hace una intersección de los niveles de `player1_id` y `player2_id`. Posteriormente se crea un factor que representa los jugadores 1 y 2 pero filtrando por los niveles seleccionados antes. Con este tipo de modelos no se aplica una normalización inicialmente.

Capítulo 5

Resultados

5.1. Estudio descriptivo y análisis no supervisado

A continuación, se presentan resúmenes estadísticos con gráficos útiles y un análisis de las variables que participan en el estudio. La sección comprende un estudio de valores faltantes, correlaciones, variables cualitativas y variables cuantitativas. La útima parte corresponde con un análisis de componentes principales y clustering, pertenecientes al aprendizaje no supervisado.

5.1.1. Datos perdidos

En la Tabla 5.1 y Tabla 5.2 se observa la cantidad de valores perdidos en los datos. Aparece un gran número en `winner_seed` y `loser_seed`, que no es preocupante ya que tan solo suele haber 8, 16 o 32 cabezas de serie en un torneo, por lo que los demás jugadores no son cabezas de serie y siempre son más de la mitad. Todos estos valores se han procesado de la manera mencionada en la Sección 4.

tourney_id	tourney_name	surface	draw_size	tourney_level	tourney_date	match_num
0	0	0	0	0	0	0
winner_id	winner_seed	winner_entry	winner_name	winner_hand	winner_ht	winner_ioc
0	22623	0	0	0	769	0
winner_age	loser_id	loser_seed	loser_entry	loser_name	loser_hand	loser_ht
5	0	30010	0	0	0	1569
loser_ioc	loser_age	score	best_of	round	minutes	w_ace
0	12	0	0	0	4145	2682
w_df	w_svpt	w_1stIn	w_1stWon	w_2ndWon	w_SvGms	w_bpSaved
2682	2682	2682	2682	2682	2681	2682
w_bpFaced	l_ace	l_df	l_svpt	l_1stIn	l_1stWon	l_2ndWon
2682	2682	2682	2682	2682	2682	2682
l_SvGms	l_bpSaved	l_bpFaced	winner_rank	winner_rank_points	loser_rank	loser_rank_points
2681	2682	2682	248	248	633	633

Tabla 5.1: Conteo de NAs por variable ATP

tourney_id	tourney_name	surface	draw_size	tourney_level	tourney_date	match_num
0	0	0	0	0	0	0
winner_id	winner_seed	winner_entry	winner_name	winner_hand	winner_ht	winner_ioc
0	17772	0	0	0	2454	0
winner_age	loser_id	loser_seed	loser_entry	loser_name	loser_hand	loser_ht
8	0	25540	0	0	0	3984
loser_ioc	loser_age	score	best_of	round	minutes	w_ace
0	15	0	0	0	17420	4440
w_df	w_svpt	w_1stIn	w_1stWon	w_2ndWon	w_SvGms	w_bpSaved
4462	4440	4440	4440	4440	15611	4440
w_bpFaced	l_ace	l_df	l_svpt	l_1stIn	l_1stWon	l_2ndWon
4440	4443	4462	4440	4440	4440	4440
l_SvGms	l_bpSaved	l_bpFaced	winner_rank	winner_rank_points	loser_rank	loser_rank_points
15611	4440	4440	625	625	1203	1203

Tabla 5.2: Conteo de NAs por variable WTA

5.1.2. Gráficos de dispersión y Correlaciones

En los gráficos de dispersión que se muestran en la Figura 5.1 y la 5.2, se muestran algunos de los diagramas de dispersión "dos a dos" más relevantes para los datos de tenistas masculinos. Se divide según los partidos que se juegan al mejor de 3 sets y al mejor de 5, ya que la duración del partido y las demás variables se pueden ver afectadas. Las variables seleccionadas son las correspondientes al ganador de un encuentro, pero expresan las mismas conclusiones que las variables del perdedor. Por lo general, la relación entre las variables es lineal y creciente. Esto se nota menos en la variable `w_2ndWon`. A medida que aumenta la duración de un partido aumenta también el número de puntos al servicio, primeros saques dentro, primeros saques ganados y número de juegos al saque. Con la variable `minutes` se aprecian los dos outliers correspondientes a partidos muy largos, uno de ellos ostenta el record de duración y se disputó en el torneo de Wimbledon 2010 entre John Isner y Nicolas Mahut, resultando vencedor el primero de ellos.

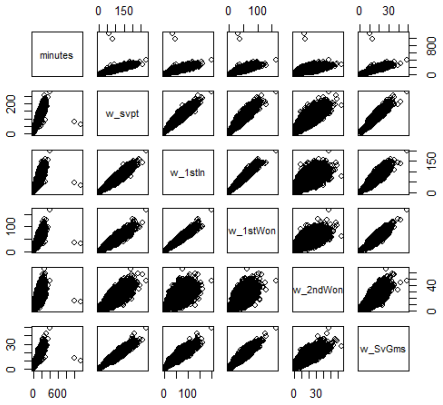

Figura 5.1: Al mejor de 3 sets

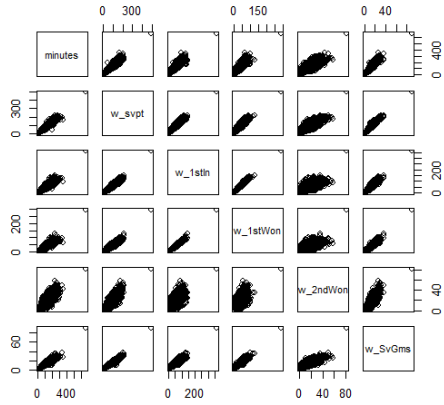

Figura 5.2: Al mejor de 5 sets

En la siguiente Tabla 5.3 aparece el valor de las correlaciones de Pearson para algunas de las variables de interés. Se trata de una matriz simétrica en la cual las variables `w_1stIn` y `w_1stWon` son las más correladas (0.96).

	minutes	w_svpt	w_1stIn	w_1stWon	w_2ndWon	w_SvGms
minutes	1.00	0.92	0.87	0.84	0.73	0.90
w_svpt	0.92	1.00	0.95	0.91	0.78	0.94
w_1stIn	0.87	0.95	1.00	0.96	0.59	0.90
w_1stWon	0.84	0.91	0.96	1.00	0.57	0.91
w_2ndWon	0.73	0.78	0.59	0.57	1.00	0.77
w_SvGms	0.90	0.94	0.90	0.91	0.77	1.00

Tabla 5.3: Correlaciones ATP

5.1.3. Variables cualitativas

Ahora se observan unos diagramas de barras en las Figuras 5.3 y 5.4. En cuanto a la superficie de juego, el circuito masculino y femenino comparten el mismo patrón, ya que hay muchos torneos en pista dura, pocos en hierba y un número intermedio en tierra.

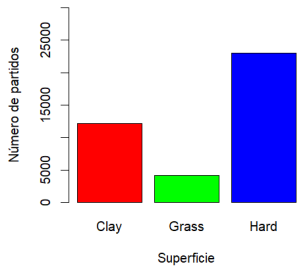

Figura 5.3: Partidos ATP superficies

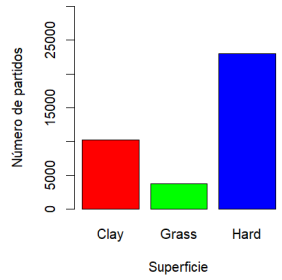

Figura 5.4: Partidos WTA superficies

En la Figura 5.5 hay un gráfico de mosaico en el cual se aprecian mejor las diferencias en superficie entre el circuito masculino y femenino. La mayor diferencia es que en el circuito masculino se disputan más torneos en proporción en tierra batida que en el femenino. Además, mirando el eje de abscisas se puede apreciar que el número de torneos masculinos es bastante mayor que el de femeninos, pero esto es debido a los valores perdidos en partidos femeninos.

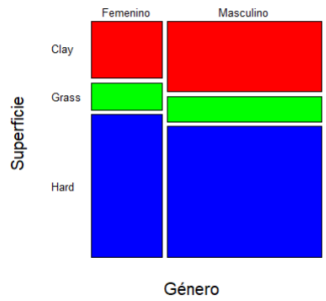

Figura 5.5: Gráfico de mosaico para las superficies

En las Tablas 5.4 y 5.5 aparece el número de partidos en los que se ve involucrada cada nacionalidad de jugadores y jugadoras en el periodo 2010-2023. Se han seleccionado las 10 más frecuentes para cada género, siendo comunes España, Francia, Estados Unidos, Alemania, Italia, Rusia y Australia. Se distinguen tres grandes potencias en el tenis femenino: Estados Unidos, Rusia y República Checa, mientras que en el tenis masculino las tres potencias son España, Francia y Estados Unidos.

USA	RUS	CZE	GER	FRA	ROU	ITA	ESP	CHN	AUS
7668	6288	5022	3760	3177	2995	2968	2847	2772	2394

Tabla 5.4: Nacionalidades WTA

ESP	FRA	USA	GER	ARG	ITA	RUS	AUS	SRB	CRO
8206	7345	6876	4505	4056	3750	3363	3140	3005	2088

Tabla 5.5: Nacionalidades ATP

En las Tablas 5.6 y 5.7 se puede ver el tipo de torneos jugados cada año, apreciándose la suspensión de las competiciones durante varios meses del año 2020 a causa de la pandemia de Covid19, con el Roland Garros incluido. En el circuito masculino aparecen las NextGen Finals en el Año 2017 y los torneos que más abundan son los ATP250 y ATP500, que llegan a solaparse algunas semanas. En cuanto al femenino, los torneos que más abundan son Premier e International. En el año 2023 hay un Grand Slam femenino que no se incluye porque hay datos faltantes. En este caso no hay WTA Finals de la NextGen, aparecen 2 en algunos años ya que se divide por fase de grupos y fase final.

tourney_level	2016	2017	2018	2019	2020	2021	2022	2023
ATP500/250	53	53	53	49	26	49	56	52
ATP FINALS	1	2	2	2	1	2	2	2
GRAND SLAM	4	4	4	4	3	4	4	4
MASTER 1000	9	9	9	9	3	8	8	9

Tabla 5.6: Tipos de torneo jugados cada año ATP

tourney_level	2016	2017	2018	2019	2020	2021	2022	2023
WTA FINALS	2	2	2	2	0	1	1	2
GRAND SLAM	4	4	4	4	3	4	3	4
INTERNATIONAL	35	37	31	32	13	21	29	31
PREMIER	20	14	16	14	8	19	20	16

Tabla 5.7: Tipos de torneo jugados cada año WTA

5.1.4. Variables cuantitativas

Hay una carácterística que nos da una pista sobre un aspecto que introduce variabilidad en los datos. Esta es el porcentaje de veces que vence un jugador con ranking inferior, que en el circuito ATP es del 34.6236 %, mientras que en la WTA es del 37.1693 %. Esto indica que es más probable que gane el jugador o jugadora con mejor ranking. En el circuito femenino se dan lugar más "sorpresas", es decir, que el partido lo gane la jugadora de peor ranking. Esto se tendrá en cuenta a la hora de seleccionar las variables más influyentes, ya que previsiblemente el ranking tendrá un gran peso a la hora de determinar el ganador de un partido.

En las Figuras 5.6 y 5.7 se observa la evolución de los partidos jugados desde 2020 hasta 2023 por los jugadores/as top 10 al terminar el año 2023. El año 2020 es una anomalía ya que se cancelaron numerosos torneos. Por lo demás, el número de torneos jugados es más o menos constante salvo para los jugadores que experimentaron lesiones como por ejemplo en el año 2022 Zverev, Hurkacz y Vondrousova.

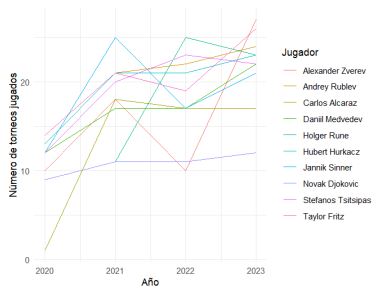

Figura 5.6: Evolución de partidos jugados top10 ATP

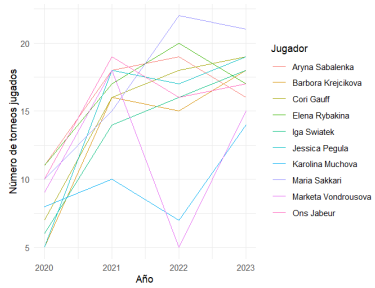

Figura 5.7: Evolución de partidos jugados top10 WTA

En los siguientes gráficos de barras de las Figuras 5.8 y 5.9 se puede apreciar el número de torneos que se disputan cada año. Cabe resaltar que en el tenis femenino los resultados son fiables a partir de 2016, ya que previamente se dispone de muchos datos perdidos. El número de torneos permanece constante salvo por la excepción del año 2020, en el que tuvieron que cancelarse numerosos torneos. En el año 2015, también aparecen menos torneos en el circuito masculino debido a valores perdidos. 2018 es el año en que más torneos masculinos se registraron, mientras que para el circuito femenino el mejor año en este aspecto fue 2021.

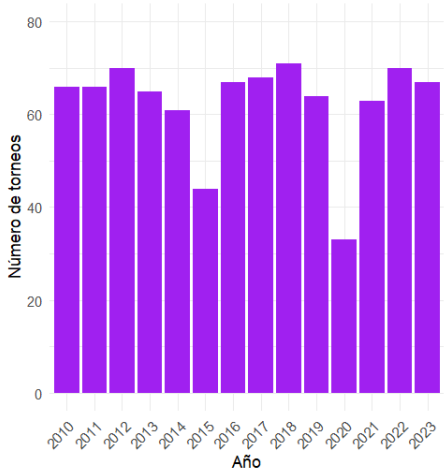

Figura 5.8: Torneos al año ATP

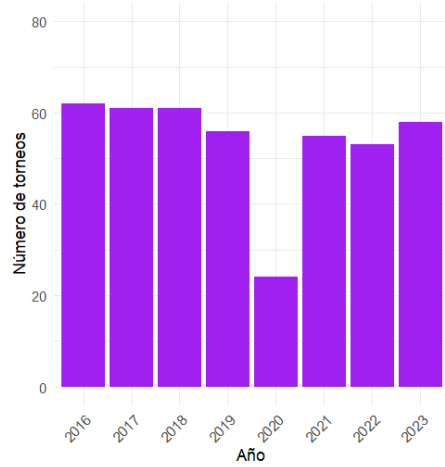

Figura 5.9: Torneos al año WTA

En la Figura 5.10 se aprecia un gráfico de dispersión con una tendencia, la cual indica que los jugadores más altos de la ATP realizan más saques directos en sus partidos, así que en principio la altura da ventaja a la hora de sacar. Para paliar la diferencia entre torneos que disputan al mejor de 5 sets y al mejor de 3 sets, se han calculado los aces por juego, dividiendo la variable `w_aces` entre la variable `w_SvGms`.

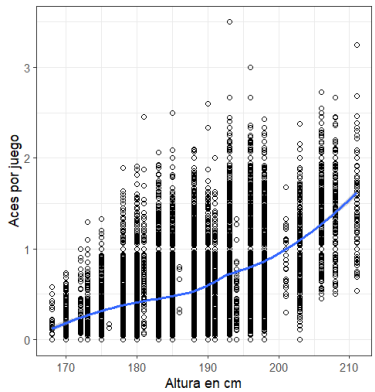

Figura 5.10: Estatura vs aces por juego ATP

En la Figura 5.11 se puede observar un gráfico de dispersión que relaciona la edad del jugador con el ranking ATP. La mayor parte de los jugadores tienen una edad comprendida entre 20 y 35 años. Tan solo hay 8 partidos en los cuales participan jugadores mayores de 40 años, por lo que la edad para retirarse casi siempre está antes de cumplir los 40 años. La tendencia muestra que los jugadores ATP alcanzan su ranking óptimo entre los 23 y los 33 años aproximadamente.

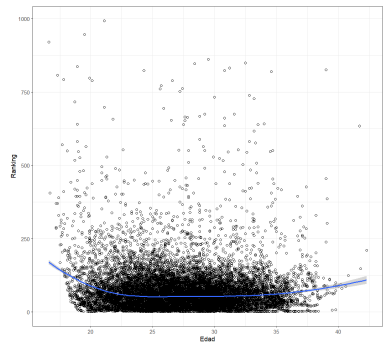

Figura 5.11: Edad vs Ranking

Por brevedad, solamente se expone el siguiente gráfico de líneas de la Figura 5.12 para el circuito femenino. Se ha creado previamente un vector con las jugadoras que terminaron el año 2023 en las diez posiciones más altas de la clasificación. Se puede apreciar su evolución en cuanto al porcentaje de victorias desde 2020, siendo Iga Swiatek la que domina este aspecto reflejándose así en el ranking, ya que es la número 1. En el año 2021 las marcas de estas jugadoras fueron bastante parecidas, pero algunas de ellas bajaron su rendimiento en 2022 aunque al año

siguiente retomaron el buen porcentaje de victorias. Las excepciones son Aryna Sabalenka y Maria Sakkari, que han bajado su rendimiento en 2023.

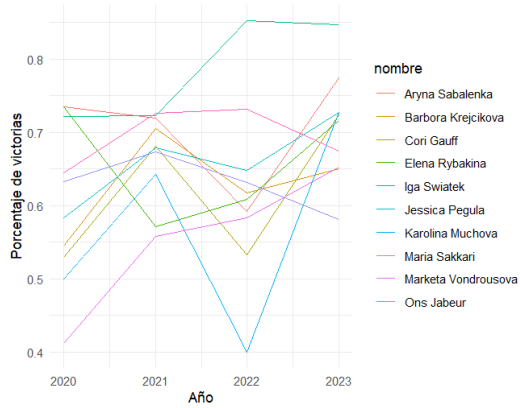

Figura 5.12: Porcentaje de victorias por jugadora del Top10 al terminar 2023 [18]

En el siguiente gráfico de líneas de la Figura 5.13 se puede comprobar la evolución del número de tenistas masculinos en el top 200 de la clasificación según su nacionalidad. Para ello se han incluido años previos (desde la década de los 90). Los datos empleados son los que provienen de la unión entre los datos atp_players y atp_matches ya que se necesita el ranking actualizado a una misma fecha para todos jugadores. A principios de los 90, el dominio era estadounidense, pero a finales de la década llegó un dominio español, para después alternarse con dominio francés y de nuevo estadounidense. Australia alcanzó su auge en la década de los 90 mintras que Argentina hizo lo propio en la década del 2000. En estos últimos años se aprecia un aumento notable de tenistas italianos y franceses en el top200, mientras que España se sitúa en el mínimo con tan solo diez jugadores.

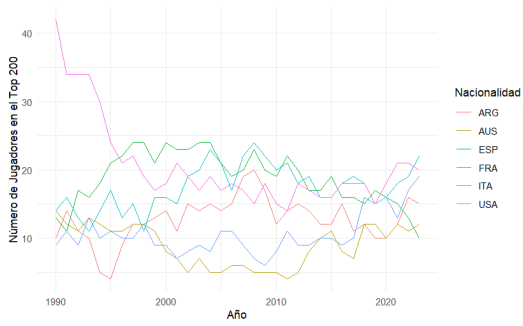

Figura 5.13: Evolución del número de jugadores en el Top200 según nacionalidad

Las siguientes Tablas 5.8, 5.9 muestran los 10 jugadores y jugadoras con mayor porcentaje de victorias en 2023. Se ha filtrado para que solamente aparezcan jugadores que hayan disputado al menos 20 partidos en ese año. Muchos de ellos coinciden con los mejores jugadores en cuanto a ranking, pero otros no. Esto se debe a que no todas las victorias aportan la misma puntuación. Por ejemplo los tres útimos jugadores de la tabla no acabaron dentro de las diez primeras posiciones del ranking en 2023. Con las tenistas pasa algo parecido ya que la quinta, octava y novena en la tabla no acabaron entre las diez primeras del ranking. En este caso también se debe a que no disputaron tantos partidos como sus rivales debido a lesiones y otras causas.

En el balance de aces con dobles faltas los mejores son Taylor Fritz y Elena Rybakina, mientras que dos de los mejores del mundo como son Carlos Alcaraz e Iga Swiatek son los que realizan menos saques directos por partido. A pesar de ello, Swiatek es la que menos bolas de break afronta con su saque. La variable `jug` almacena el número de partidos jugados, siendo Daniil Medvedev quien jugó más partidos. Los jugadores que más tiempo pasaron el pista fueron Alexander Zverev e Iga Swiatek, con 159 y 114 horas respectivamente. Los partidos de Grand Slam en el circuito masculino son al mejor de 5 y eso explica que los hombres pasen más tiempo en pista de media.

nombre	vict	ace	df	svpt	1stIn	SvGms	bpS	bpF	jug	time
Novak Djokovic	88.89	7.11	3.02	84.94	53.93	13.93	3.24	4.78	54	128
Carlos Alcaraz	83.78	4.14	2.16	75.82	49.96	12.20	3.31	5.12	74	151
Jannik Sinner	80.82	6.15	1.86	76.47	45.22	12.45	3.44	5.07	73	149
Daniil Medvedev	77.78	7.06	4.14	75.91	48.81	11.90	3.64	5.47	81	154
Taylor Fritz	70.42	9.46	2.28	77.79	46.70	12.66	3.37	5.14	71	132
Andrey Rublev	69.23	7.64	1.99	82.01	50.24	13.09	3.40	5.49	78	151
Alexander Zverev	67.53	7.71	2.81	77.08	54.47	12.38	3.32	5.23	77	159
Grigor Dimitrov	66.67	7.16	3.44	77.82	47.56	12.54	3.35	5.40	57	112
Frances Tiafoe	65.38	7.88	2.29	77.46	46.21	12.29	3.27	5.04	52	96
Karen Khachanov	65.31	6.76	2.00	78.02	49.61	12.37	3.45	5.47	49	98

Tabla 5.8: Jugadores con más % victorias en 2023

nombre	vict	ace	df	svpt	1stIn	SvGms	bpS	bpF	jug	time
Iga Swiatek	84.72	1.58	1.54	57.12	37.54	9.43	2.17	4.00	72	114
Aryna Sabalenka	77.42	5.85	4.24	67.82	41.56	10.42	3.52	5.63	62	102
Jessica Pegula	72.73	2.30	2.45	66.36	39.23	10.21	3.68	6.62	66	105
Karolina Muchova	72.73	3.52	2.45	76.45	47.95	11.39	4.61	7.41	44	90
Belinda Bencic	72.50	3.25	3.92	69.50	42.48	10.75	3.62	5.97	40	74
Cori Gauff	72.41	3.98	3.34	67.00	40.40	10.26	3.91	6.78	58	92
Elena Rybakina	71.70	7.72	3.11	73.58	42.04	11.08	3.85	6.25	53	93
Elina Svitolina	71.43	3.71	3.38	76.90	47.52	11.38	4.52	8.29	21	38
Petra Kvitova	67.74	5.19	5.23	69.97	43.65	10.71	3.26	5.68	31	52
Ons Jabeur	67.44	3.33	2.95	70.19	38.53	10.67	4.14	7.67	43	72

Tabla 5.9: Jugadoras con más % victorias en 2023

En los siguientes diagramas de cajas de las Figuras 5.14 y 5.15, se puede apreciar la duración de los partidos de Grand Slam en el periodo 2010-2023. Destaca el partido más largo de la historia en Wimbledon (masculino) y también un partido largo en el US Open (femenino). Se vuelve a demostrar que los partidos en Grand Slam masculinos son más largos de media ya que se disputan al mejor de 5 sets. Parece que en el US Open los partidos son algo más largos de media y los más cortos se dan en Wimbledon.

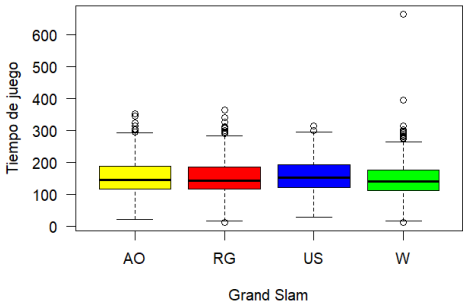

Figura 5.14: Duración GS ATP

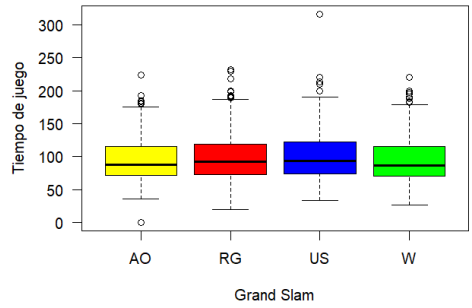

Figura 5.15: Duración GS WTA

5.1.5. Análisis de Componentes Principales

Se procede a realizar el estudio de las componentes principales para los 300 jugadores/as mejor clasificados al finalizar 2023. Para ello se ha utilizado la función `prcomp()` de R con la opción de *scale* seleccionada en TRUE.

La función `summary()` ofrece el resumen que aparece en las Tablas 5.10 y

5.11. Las 7 componentes principales explican toda la variabilidad de los datos, pero analizando la varianza de cada componente deberíamos quedarnos con un número menor. Para el circuito ATP, las tres primeras componentes principales son capaces de explicar bastante varianza ya que es prácticamente un 80 %, pero con dos componentes sería fácilmente interpretable de manera geométrica ya que se representarían en dos dimensiones. Entre las dos explican algo más del 66 % de la variabilidad de los datos. Para el circuito femenino, se explica algo más del 76 % de la variabilidad de los datos con tres componentes, mientras que con dos componentes se explica algo más de un 63 %.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Standard deviation	1.8543	1.1081	0.9625	0.70144	0.58571	0.55700	0.51176
Proportion of Variance	0.4912	0.1754	0.1323	0.07029	0.04901	0.04432	0.03741
Cumulative Proportion	0.4912	0.6666	0.7990	0.86926	0.91827	0.96259	1.00000

Tabla 5.10: Resumen numérico del PCA jugadores top300

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Standard deviation	1.8140	1.0733	0.9701	0.80932	0.64693	0.58883	0.44263
Proportion of Variance	0.4701	0.1646	0.1344	0.09357	0.05979	0.04953	0.02799
Cumulative Proportion	0.4701	0.6347	0.7691	0.86269	0.92248	0.97201	1.00000

Tabla 5.11: Resumen numérico del PCA jugadoras top300

Las Figuras 5.16 y 5.17 muestran información gráfica sobre las componentes principales para el circuito ATP. En la primera se puede observar que el "codo" de la gráfica está en la segunda componente principal, por lo que es una buena decisión tomar dos componentes principales. La segunda gráfica representa las variables en un gráfico de dos dimensiones, siendo representada la componente 1 en el eje de abscisas y la componente 2 en el eje de ordenadas. Como era de esperar, los jugadores mejor clasificados en el ranking están situados en la dirección de las flechas de la izquierda. Esto sucede ya que los mejores jugadores tienen un elevado porcentaje de victorias. El número 42 es Andy Murray, que destaca en este gráfico debido a que obtuvo muchas victorias en años anteriores, llegando a ser número 1 del mundo.

Las Figuras 5.18 y 5.19 muestran la misma información pero referente al circuito WTA. Aquí el "codo" de la gráfica también está marcado en la segunda componente. En el segundo gráfico destaca el dominio de las tres mejores jugadoras del mundo y la cantidad de victorias de las jugadoras Kvitova (número 17) y Pliskova (número 39). Tanto en el masculino como el femenino, la flecha del ranking apunta en el sentido que cabía esperar ya que el ranking bajo lo interpretamos como algo bueno.

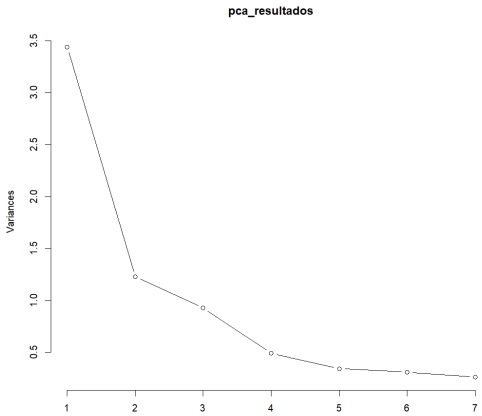

Figura 5.16: Varianza explicada ATP

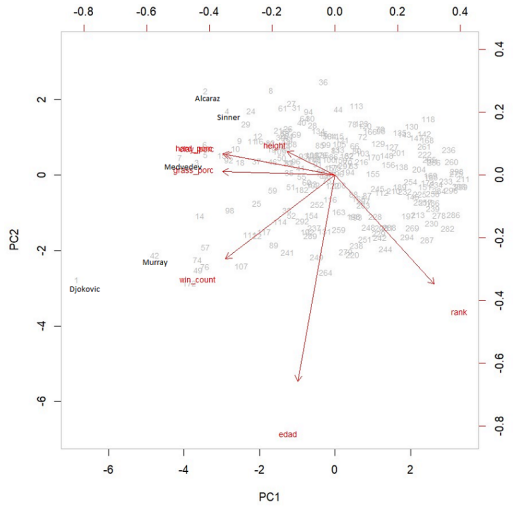

Figura 5.17: Componentes 1 y 2 ATP

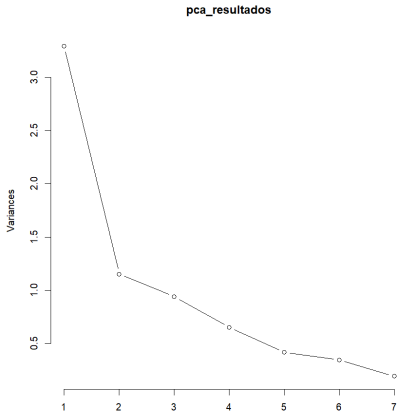

Figura 5.18: Varianza explicada WTA

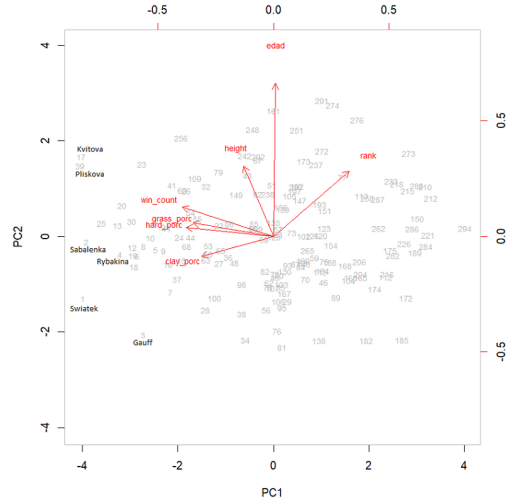

Figura 5.19: Componentes 1 y 2 WTA

5.1.6. Análisis Cluster

Se utiliza la función de R `kmeans()` para realizar un clustering k-medias por jugadores. El número de clusters es elegido de antemano con valor 3.

En el siguiente gráfico de dispersión de la Figura 5.20 se representan las 2 primeras componentes principales para el circuito ATP y se marcan los distintos clusters usando colores distintos. Hay un grupo pequeño representado en color

verde que corresponde a los jugadores más destacados. El cluster representado en rojo agrupa jugadores con valor alto en la segunda componente y valor intermedio en la primera, mientras que el cluster azul muestra los jugadores con menor valor en la componente 1. Por su parte, el cluster representado en color verde muestra los jugadores con mayor valor en la componente 1.

En el mismo gráfico pero para el circuito femenino de la Figura 5.21, se aprecian tres clusters que también asocian a las jugadoras según las dos primeras componentes. La separación en grupos es diferente, ya que es la componente 1 la que realiza gran parte de la separación por grupos. El cluster representado en color rojo agrupa las jugadoras con un valor bajo en la componente 1. En cuanto al cluster de color azul, agrupa a las jugadoras con valor alto en la componente 1. El otro cluster está representado en color verde y agrupa a las jugadoras con valor intermedio en la componente 1. También es cierto que la componente 2 realiza una ligera separación entre el cluster azul y el verde, tendiendo a situar el cluster azul en valores más altos de la componente 2 y el cluster verde en valores más bajos.

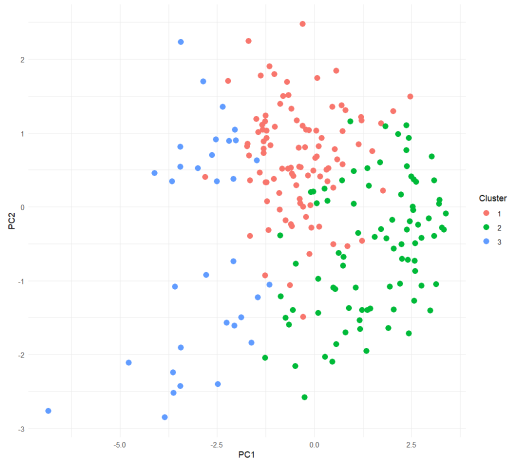

Figura 5.20: Cluster top 300 ATP

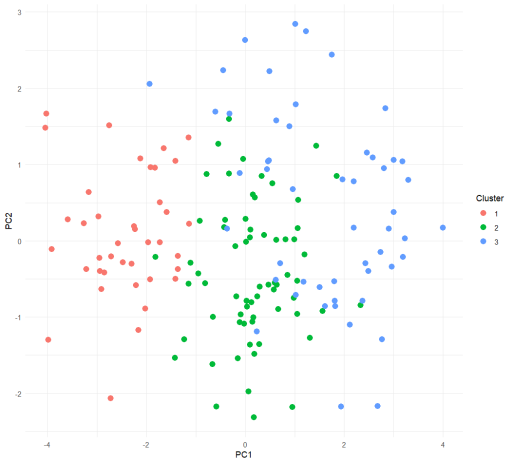

Figura 5.21: Cluster top 300 WTA

5.2. Métodos predictivos

En total son 4 métodos de aprendizaje supervisado clásicos los que se van a emplear, siguiendo un orden de trabajo similar: Regresión Logística, Random Forest, Support Vector Machines y Gradient Boosting.

Las variables que se incluirán inicialmente en los modelos serán `surface_clay`, `surface_grass`, `minutes`, `diff_age`, `diff_rank`, `diff_win_count`, `diff_height`, `diff_prop`, `diff_prop_last10matches` y `player1_win`. Todas ellas se han expli-

cado en el apartado "Transformaciones de variables" de la Sección 4. La variable respuesta será siempre `player1_win` y todas las demás variables son explicativas. La precisión del modelo inicial con estas variables es la que se denomina "P inic". Como medida de precisión, se usa la proporción de partidos acertados sobre el total.

Se cree necesario transformar la variable referente al ranking, ya que en cuanto al nivel de juego no es lo mismo la diferencia entre un jugador con ranking 1 y otro con ranking 30 que la diferencia entre un jugador clasificado como 501 y otro como 530. Aunque matemáticamente es igual, en la práctica la primera diferencia es mucho mayor. Para paliar esto, se prueban las transformaciones logarítmica, inversa y raíz cuadrada. La primera transformación se realiza según la fórmula $log(rank1) - log(rank2)$, la siguiente es según la fórmula $1/rank1 - 1/rank2$ y la última es según la fórmula $\sqrt{rank1}$ - $\sqrt{rank2}$. La manera de referirnos a sus precisiones será "P log", "P frac", "P sqrt" respectivamente. Con la transformación logarítmica se obtendrán unos resultados ligeramente mejores y se seleccionará para realizar los siguientes métodos de predicción.

A continuación, al pensar en una posible "interacción" en la variable ranking, se opta por trabajar con las variables `rank1`, `diff_rank` y `rank_inter` en cuanto al apartado de ranking, por lo tanto se añaden 2 variables que antes no estaban. Cada una de ellas representa respectivamente el ranking del jugador 1, la diferencia de ranking del jugador 1 respecto al jugador 2 y la última es el producto entre las dos variables anteriores, es decir, el producto entre el ranking del jugador 1 y la diferencia de ranking del jugador 1 respecto al jugador 2. Estas variables que surgen de la "interacción" en la variable ranking se aplican en los modelos que utilizan las 3 transformaciones del ranking: logarítmica, inversa y raíz cuadrada.

También se implementarán modelos utilizando datos para superficies concretas y jugadores concretos [19] en la Sección 5. Para ello, se utilizan todas las variables presentadas anteriormente, pero esta vez ya se ha seleccionado la transformación del ranking que mejor funciona: la logarítmica. Al haber muchos jugadores en este aspecto, tan solo se seleccionan algunos destacados como ejemplo. Para denominar la precisión por superficies se utiliza la notación "P clay", "P grass" y "P hard". En cuanto a los jugadores, se realiza utilizando la letra "P" seguida del nombre del jugador/a, por ejemplo: "P Djokovic".

Al final de cada método de aprendizaje supervisado se detallarán los modelos que han ofrecido una precisión mayor para cada categoría de interés.

Para estas implementaciones se calcula la precisión de la siguiente manera: una vez usada la función `predict()` de R, se cuenta el número de aciertos comparando este resultado con la variable `player1_win`. Los aciertos se dividen por el total y así se obtiene la precisión del modelo para poder compararlo con todos los demás. La función de R `mean()` permite realizarlo.

5.2.1. Conjuntos de entrenamiento y test

Una vez se han estandarizado los datos con la función `scale()` de R para tratar de poner las variables en pie de igualdad, pasan a dividirse en dos conjuntos distintos: el de entrenamiento y el de test. Se dividirán de dos maneras diferentes. La forma inicial será mediante el tradicional método de división "2/3 1/3", quedando aproximadamente un 66,66 % de los datos totales en el entrenamiento y un 33,33 % en el conjunto test. A partir de ahora nos referiremos a esta forma de dividir los datos como "2/3 1/3". La segunda forma será seleccionando en el conjunto de entrenamiento los datos desde 2010 hasta 2022, ambos incluidos, y en el conjunto de prueba el año 2023. En las siguientes páginas del documento, para referirnos a esta última manera utilizaremos la notación "Test 2023".

5.2.2. Regresión Logística

Para implementar este modelo se utiliza la función `glm()` de R, seleccionando la familia binomial.

En este caso, la función `predict()` no devuelve las predicciones como ceros o unos, por lo que se realiza un condicional en el que los valores predichos mayores que 0.5 se interpretan como 1, y los menores o iguales que 0.5 se interpretan como 0. También se ha probado con los umbrales óptimos obtenidos a partir de las curvas ROC, pero obtienen unos resultados igual de precisos. Las predicciones procesadas se comparan con la variable `player1_win` para obtener la precisión.

Una opción importante es la de utilizar métodos de selección de variables, que permitan seleccionar las variables significativas en el modelo, además de optimizar el criterio de información de Akaike (AIC). Con este fin, se utiliza la función de R `stepAIC()` perteneciente al paquete **MASS**. Se añade el parámetro *direction = "both"* para utilizar el método de selección de variables hacia adelante y atrás, ya que es el que selecciona las variables de una manera más acertada para este caso.

La Tabla 5.12 resume la precisión de los métodos de Regresión Logística de la manera inicial y para cada una de las transformaciones del ranking con la interacción añadida, según se explicó en la sección anteriormente. Se comprueba que, efectivamente, la transformación logarítmica para el ranking ofrece una precisión ligeramente mayor, y que cualquiera de las transformaciones es útil para ser más precisos en la predición.

Datos	Train	P inic	P log	P frac	P sqrt
ATP	2/3 1/3	0.665	0.685	0.682	0.684
ATP	Test 2023	0.658	0.679	0.676	0.674
WTA	2/3 1/3	0.653	0.670	0.668	0.674
WTA	Test 2023	0.655	0.674	0.671	0.674

Tabla 5.12: Precisión Regresión Logística métodos generales

En la Tabla 5.13 se pueden apreciar las precisiones del método Regresión Logística utilizando datos de las superficies concretas en cada caso. De esta manera se mejora la precisión, aunque en pista dura se nota menos ya que es la superficie más habitual y en la cual todos los jugadores están acostumbrados a jugar. La tierra batida y hierba son superficies en las que hay jugadores especialistas, por lo que sus predicciones son algo más acertadas, llegando a unas precisiones del 80 %.

Datos	Train	P clay	P grass	P hard
ATP	2/3 1/3	0.816	0.810	0.708
ATP	Test 2023	0.816	0.786	0.700
WTA	2/3 1/3	0.713	0.786	0.699
WTA	Test 2023	0.707	0.800	0.701

Tabla 5.13: Precisión Regresión Logística por superficie

A continuación, en la Tabla 5.14, se puede apreciar la precisión del método Regresión Logística para los datos de algunos jugadores ATP. Llama la atención la baja precisión que obtiene este método para el jugador Djokovic dividiendo los datos en entrenamiento y test de la forma 2/3 1/3.

Train	P Djokovic	P Alcaraz	P Hurkacz
2/3 1/3	0.601	0.833	0.816
Test 2023	0.907	0.838	0.828

Tabla 5.14: Precisión Regresión Logística por jugadores

A continuación, en la Tabla 5.15, se puede apreciar la precisión del método Regresión Logística para los datos de algunas jugadoras WTA. En el caso de Gauff la precisión no es tan elevada como se podía esperar.

Train	P Swiatek	P Gauff	P Siegemund
2/3 1/3	0.930	0.847	0.881
Test 2023	0.917	0.845	0.810

Tabla 5.15: Precisión Regresión Logística por jugadoras

A continuación, aparece un modelo que ha obtenido una precisión mayor en Regresión Logística:

Mejor modelo para pista dura en datos femeninos: Regresión Logística con transformación del ranking mediante logaritmo y división de los datos en entrenamiento desde 2010 hasta 2022 y en test el año 2023. Se obtuvo una precisión de 0.701.

En la Figura 5.22 se observa la información del modelo. Las variables significativas a nivel 0.05 son la diferencia de ranking, diferencia de altura, diferencia

de victorias, diferencia de proporción de victorias y diferencia de proporción de victorias en los últimos 10 partidos.

```
                          Estimate Std. Error z value Pr(>|z|)
(Intercept)               0.008960   0.026510   0.338  0.73536
minutes                   0.041385   0.027559   1.502  0.13317
diff_avg_age              0.005403   0.027160   0.199  0.84232
log_rank1                 0.001918   0.037254   0.051  0.95895
diff_rank_log            -0.199308   0.077647  -2.567  0.01026 *
inter_rank               -0.100354   0.069470  -1.445  0.14858
diff_avg_height           0.062237   0.026581   2.341  0.01921 *
diff_win_count           -0.135936   0.035340  -3.847  0.00012 ***
diff_prop                 1.680687   0.048983  34.312  < 2e-16 ***
diff_prop_last10matches -0.219029   0.032171  -6.808 9.87e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

    Null deviance: 11582  on 8354  degrees of freedom
Residual deviance:  8612  on 8345  degrees of freedom
AIC: 8632
```

Figura 5.22: Modelo Regresión Logística Pista Dura Test 2023 WTA

Sin embargo, aplicando el método de selección de variables hacia adelante y hacia atrás combinado, se obtienen los resultados de la Figura 5.23. Este método añade, además de las variables que se han visto que eran significativas a nivel 0.05, las variables `minutes` e `inter_rank`.

```
Coefficients:
(Intercept)     minutes      diff_rank_log    inter_rank     diff_avg_height
0.008926        0.041343     -0.197092        -0.100808      0.062283

diff_win_count        diff_prop         diff_prop_last10matches
-0.134357             1.679214          -0.218369
```

Figura 5.23: Aplicación del método de selección de variables

5.2.3. Random Forest

La función de R que se utiliza es `randomForest()`, perteneciente al paquete **randomForest**. La Tabla 5.16 resume la precisión de los métodos Random Forest de la manera inicial y para cada una de las transformaciones del ranking. Los resultados son bastante parecidos a los obtenidos mediante Regresión Logística.

Datos	Train	P inic	P log	P frac	P sqrt
ATP	2/3 1/3	0.660	0.683	0.686	0.684
ATP	Test 2023	0.648	0.673	0.666	0.673
WTA	2/3 1/3	0.639	0.661	0.653	0.662
WTA	Test 2023	0.625	0.658	0.660	0.659

Tabla 5.16: Precisión Random Forest métodos generales

La Tabla 5.17 resume la precisión de los métodos de Random Forest para cada superficie concreta. En este caso se mejora mucho la precisión en partidos de tierra batida del circuito ATP.

Datos	Train	P clay	P grass	P hard
ATP	2/3 1/3	0.851	0.811	0.702
ATP	Test 2023	0.851	0.779	0.687
WTA	2/3 1/3	0.701	0.764	0.687
WTA	Test 2023	0.710	0.780	0.676

Tabla 5.17: Precisión Random Forest por superficie

A continuación, en la Tabla 5.18, se puede apreciar la precisión del método Random Forest para los datos de algunos jugadores ATP. En todos los casos superan una precisión del 80 %.

Train	P Djokovic	P Alcaraz	P Hurkacz
2/3 1/3	0.855	0.850	0.851
Test 2023	0.870	0.865	0.813

Tabla 5.18: Precisión Random Forest por jugadores

A continuación, en la Tabla 5.19 se puede apreciar la precisión del método Random Forest para los datos de algunas jugadoras WTA. En algún caso se llega a una precisión del 90 %.

Train	P Swiatek	P Gauff	P Siegemund
2/3 1/3	0.915	0.830	0.915
Test 2023	0.903	0.793	0.905

Tabla 5.19: Precisión Random Forest por jugadoras

A continuación, aparece un modelo que ha obtenido una precisión mayor en Random Forest:

Mejor modelo general para datos masculinos: Random Forest con transformación del ranking mediante la función inversa. La división del conjunto de

datos es 2/3 para el entrenamiento y 1/3 para el test. Se obtuvo una precisión de 0.686.

En la Figura 5.24 se representa la disminución del error a medida que aumenta el número de árboles utilizados en el entrenamiento. La disminución más significativa se da en los primeros 30 árboles.

En el gráfico de barras de la Figura 5.25, se muestra la importancia de las variables a la hora de entrenar el Random Forest para los datos que tenemos en este caso. Destaca por bastante la proporción de victorias.

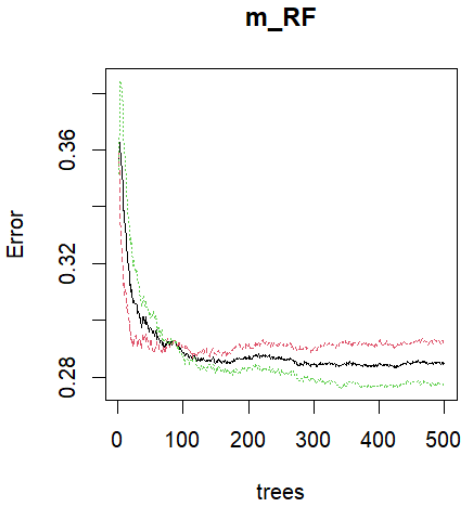

Figura 5.24: Tasa de error RF ATP

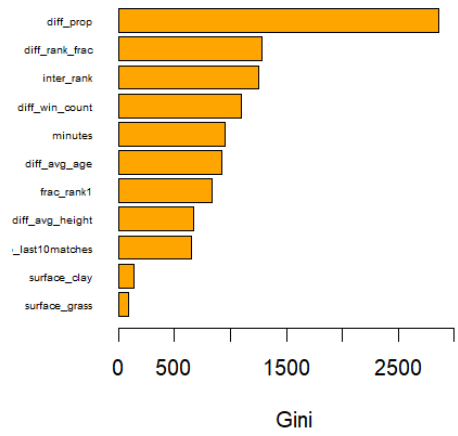

Figura 5.25: Importancia de variables

5.2.4. Support Vector Machines

La precisión se calcula de la misma manera que en Random Forest. El kernel utilizado es el polinómico ya que tras experimentar y realizar validación cruzada con otros kernels como el radial y lineal, el kernel polinómico fue el que mejor se ajustaba a los datos. La función que se utiliza para SVM es `svm()` del paquete **e1071** de R.

La Tabla 5.20 resume la precisión de los métodos de Support Vector Machines. Para los datos por género y las distintas divisiones de entrenamiento se obtienen unas prediciones relativamente parecidas.

La Tabla 5.21 resume la precisión de los métodos de Support Vector Machines para cada superficie.

Datos	Train	P inic	P log	P frac	P sqrt
ATP	2/3 1/3	0.661	0.683	0.682	0.680
ATP	Test 2023	0.650	0.675	0.668	0.674
WTA	2/3 1/3	0.646	0.664	0.665	0.666
WTA	Test 2023	0.651	0.672	0.668	0.666

Tabla 5.20: Precisión SVM métodos generales

Datos	Train	P clay	P grass	P hard
ATP	2/3 1/3	0.816	0.787	0.704
ATP	Test 2023	0.816	0.783	0.699
WTA	2/3 1/3	0.710	0.747	0.690
WTA	Test 2023	0.702	0.750	0.683

Tabla 5.21: Precisión Support Vector Machines por superficie

A continuación, en la Tabla 5.22, se puede apreciar la precisión del método Support Vector Machines para los datos de algunos jugadores ATP.

Train	P Djokovic	P Alcaraz	P Hurkacz
2/3 1/3	0.877	0.817	0.816
Test 2023	0.815	0.851	0.797

Tabla 5.22: Precisión Support Vector Machines por jugadores

A continuación, en la Tabla 5.23, se puede apreciar la precisión del método Support Vector Machines para los datos de algunas jugadoras WTA.

Train	P Swiatek	P Gauff	P Siegemund
2/3 1/3	0.845	0.847	0.898
Test 2023	0.833	0.828	0.905

Tabla 5.23: Precisión Support Vector Machines por jugadoras

A continuación, aparece un modelo que ha obtenido una precisión mayor en Support Vector Machines:

Mejor modelo general para datos femeninos: Support Vector Machines con transformación del ranking mediante logaritmo y división de los datos en entrenamiento desde 2010 hasta 2022 y en test el año 2023. Se obtuvo una precisión de 0.672.

En la Figura 5.26 se observa la información del modelo. Se trata de un SVM con núcleo polinómico y el número de vectores soporte es 9127.

```
SVM-Type:  C-classification
SVM-Kernel:  polynomial
    cost:  1
    degree:  3
    coef.0:  0
Number of Support Vectors:  9179
( 4590 4589 )
Number of Classes:  2
Levels: 0 1
```

Figura 5.26: Modelo Support Vector Machines Test 2023 WTA

5.2.5. Gradient Boosting

Empleando la función de R `xgboost()` del paquete **xgboost** con el parámetro `objective = ''binary:logistic''` se obtienen las siguientes tasas de error.

La Tabla 5.24 resume la precisión de los métodos de Gradient Boosting. Aparentemente da como resultado unas precisiones bajas respecto a los demás métodos.

Datos	Train	P inic	P log	P frac	P sqrt
ATP	2/3 1/3	0.651	0.664	0.672	0.672
ATP	Test 2023	0.646	0.662	0.660	0.666
WTA	2/3 1/3	0.629	0.644	0.637	0.649
WTA	Test 2023	0.625	0.631	0.638	0.640

Tabla 5.24: Precisión Gradient Boosting métodos generales

La Tabla 5.25 resume la precisión de los métodos de Gradient Boosting para cada superficie. Se mantiene la tendencia de precisión más baja.

Datos	Train	P clay	P grass	P hard
ATP	2/3 1/3	0.644	0.793	0.696
ATP	Test 2023	0.839	0.793	0.686
WTA	2/3 1/3	0.685	0.769	0.667
WTA	Test 2023	0.685	0.765	0.671

Tabla 5.25: Precisión Gradient Boosting por superficie

A continuación, en la Tabla 5.26, se puede apreciar la precisión del método Gradient Boosting para los datos de algunos jugadores ATP.

A continuación, en la Tabla 5.27, se puede apreciar la precisión del método Gradient Boosting para los datos de algunas jugadoras WTA.

Train	P Djokovic	P Alcaraz	P Hurkacz
2/3 1/3	0.822	0.883	0.839
Test 2023	0.870	0.905	0.813

Tabla 5.26: Precisión Gradient Boosting por jugadores

Train	P Swiatek	P Gauff	P Siegemund
2/3 1/3	0.845	0.763	0.898
Test 2023	0.903	0.793	0.905

Tabla 5.27: Precisión Gradient Boosting por jugadoras

A continuación, aparece un modelo que ha obtenido una precisión mayor en Gradient Boosting:

Mejor modelo para datos de Carlos Alcaraz: Gradient Boosting con transformación del ranking mediante logaritmo y división de los datos en entrenamiento desde 2010 hasta 2022 y en test el año 2023. Se obtuvo una precisión de 0.905.

En la Figura 5.27 se observa la información del modelo.

```
                Length Class              Mode
handle               1 xgb.Booster.handle externalptr
raw              82707 -none-             raw
niter                1 -none-             numeric
evaluation_log       2 data.table         list
call                14 -none-             call
params               2 -none-             list
callbacks            2 -none-             list
feature_names       11 -none-             character
nfeatures            1 -none-             numeric
```

Figura 5.27: Modelo Gradient Boosting Test 2023 Alcaraz

5.3. Modelos Bradley Terry

La función `BTm()` del paquete **BradleyTerry2** de R permite entrenar el modelo. Hay varias maneras de pasar los datos a esta función, pero la que se elige es la siguiente, según los argumentos de dicha función:

- outcome = `player1_win`: variable binaria que indica 1 si el jugador 1 ha ganado y 0 en caso contrario.
- player1 = `player1_id`: identificador del primer jugador, es un factor.
- player2 = `player2_id`: identificador del segundo jugador, también es un factor.

Los datos utilizados son los de entrenamiento, correspondietes a la selección de años 2010-2022 o la división 2/3 1/3. Un requisito para aplicar este modelo es que ambos factores deben tener los mismos niveles, por lo que se crea un conjunto de datos nuevo que filtra según los jugadores que aparecen al menos una vez tanto en la variable `player1_id` como en `player2_id`.

Para este primer modelo que denominamos M1, los coeficientes que salen de resultado se corresponden con la habilidad de cada uno de los jugadores, es decir, si el valor es alto y positivo, el jugador tendrá probabilidades más altas de victoria. Dicho de otra manera, si un jugador tiene coeficiente positivo significa que tiene una habilidad relativa mayor que la media del grupo de jugadores en el modelo, y si un jugador tiene un coeficiente negativo significa que tiene una habilidad relativa menor que la media del grupo de jugadores.

A continuación, se va a estudiar el efecto de la habilidad que producen estos modelos de Bradley Terry. Se aplicará concretamente para los datos masculinos y femeninos utilizando como conjunto de entrenamiento el periodo 2010-2022 y como test el 2023. Para ello, se ha tenido que realizar un ligero preprocesamiento de los datos, ya que era necesario juntar el dataset que incluye las habilidades con el dataset `atp_rankings_current`, que contiene el ranking a fecha 1 de enero de 2024 y `atp_players`, que contiene el nombre de los jugadores asociados a un identificador común en todos los conjuntos de datos, a partir del cual se unen los tres datasets con la función `merge()` de R. También se ordena según el ranking del jugador/a.

En la Figura 5.28 aparece un gráfico de dispersión, en el cual el eje de abscisas representa la habilidad de cada jugador y el eje de ordenadas representa el ranking nada más empezar el año 2024. Se aprecia como las habilidades se han agrupado en 3 bloques, pero el mayoritario es el que abarca el rango $(-2, 4)$ y tiene más interés de interpretación. Se puede observar como la mayor habilidad corresponde al jugador Novak Djokovic, número 1 del ranking. La otra habilidad que supera el valor 3 es la de Rafael Nadal, que aunque ocupa un ranking superior al 600, obtiene una buena puntuación en el otro aspecto debido a su buen balance de partidos ganados en el pasado.

En la Figura 5.29 se puede observar un gráfico de barras que representa la habilidad para los 20 primeros jugadores del ranking al comenzar 2024. Hay dos jugadores que obtienen valor perdido en la habilidad ya que por lesión no han disputado suficientes partidos en 2023 como para que el modelo los clasifique de manera adecuada. Para los jugadores top del ranking parece que la habilidad sí es más precisa, ya que disminuye según el ranking es peor. Destaca el jugador situado en la posición 8 (Holger Rune) con una habilidad demasiado baja ya que sus mejores resultados han llegado en 2023, año que no se utiliza para entrenar los datos.

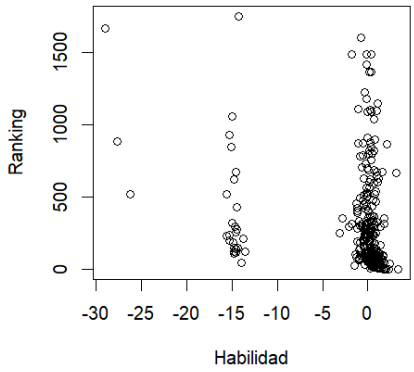

Figura 5.28: Habilidad jugadores ATP

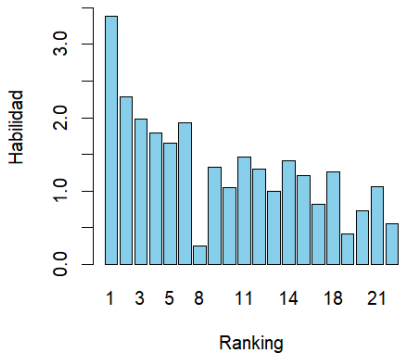

Figura 5.29: Habilidad jugadores Top20 ATP

En la Figura 5.30 aparece el mismo gráfico de puntos que antes pero esta vez para el circuito femenino. Se aprecia como las habilidades se han agrupado en 2 bloques, pero el mayoritario es el que abarca el rango $(-5, 0)$ y tiene más interés de interpretación. La relación entre el ranking y la habilidad es más apreciable en este caso.

En la Figura 5.31 aparece el mismo gráfico de barras que para el circuito masculino. Se ha tenido que transformar la habilidad ya que en todos los casos se obtienen valores negativos, por lo se ha representado la inversa de la habilidad.

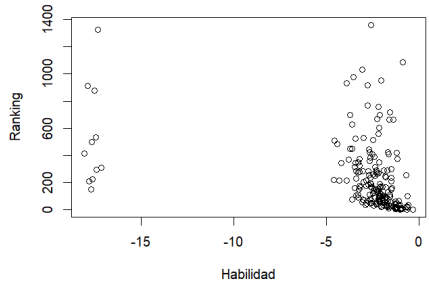

Figura 5.30: Habilidad jugadoras WTA

Destaca la habilidad de la número 1 como en el caso masculino, pero también la jugadora número 17 del ranking (Petra Kvitova), que obtuvo muchas victorias en el pasado.

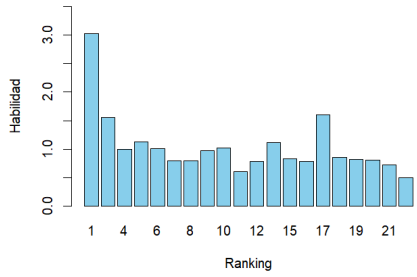

Figura 5.31: Habilidad jugadoras Top20 WTA

A continuación, en la Tabla 5.28, se detalla la precisión del modelo Bradley-Terry que hemos denominado M1 y contiene la variable respuesta y las dos variables de jugadores. Para predecir, se usan las probabilidades p_i y p_j que aporta el modelo, resultando ganador el jugador con una probabilidad mayor. La división de los datos "Test 2023" da una mejor precisión.

En el modelo M2 se han añadido una serie de factores fijos como son el ranking, el tiempo de juego y la altura de cada jugador. La función de enlace es "probit" de los modelos lineales generalizados. En la Tabla 5.29 se pueden ver las precisiones.

Datos	Train	Precision
ATP	2/3 1/3	0.572
ATP	2023	0.611
WTA	2/3 1/3	0.598
WTA	2023	0.631

Tabla 5.28: Comparación de los métodos Bradley Terry M1

Datos	Train	Precision
ATP	2/3 1/3	0.569
ATP	2023	0.603
WTA	2/3 1/3	0.595
WTA	2023	0.632

Tabla 5.29: Comparación de los métodos Bradley Terry M2

A continuación, se realiza una prueba realizando las mismas estimaciones anteriores pero con reducción del sesgo, es decir, en lugar de estimar los parámetros mediante máxima verosimilitud estándar, implementar la alternativa de entrenar por máxima verosimilitud con sesgo reducido. Para ello hay que cargar el paquete **brglm** de R y cambiar el parámetro `br` a TRUE. Esta reducción de sesgo no llega a notarse en los resultados ya que algunos enfrentamientos entre jugadores se repiten muchas veces y sobretodo hay un elevado número de jugadores, lo cual hace que el tiempo de ejecución sea de varias horas. Por eso no se muestran los resultados en este documento.

En las Tablas 5.30 y 5.31 se añaden los modelos M3 y M4. El primero es similar a M2, con la misma función de enlace. Lo que cambia son los factores fijos, ya que en este caso tenemos la diferencia de logaritmos del ranking, la diferencia de victorias, diferencia de proporción de victorias, tiempo de juego, superficie, ronda, diferencia de altura y diferencia de edad. Así, todos los factores fijos que podrían ser de interés están incluidos. En cuanto al M4, es igual al M3 salvo por la eliminación de los factores fijos referentes a la altura, tiempo de juego y ronda.

Datos	Train	Precision
ATP	2/3 1/3	0.537
ATP	2023	0.554
WTA	2/3 1/3	0.551
WTA	2023	0.525

Tabla 5.30: Comparación de los métodos Bradley Terry M3

Datos	Train	Precision
ATP	2/3 1/3	0.547
ATP	2023	0.558
WTA	2/3 1/3	0.553
WTA	2023	0.534

Tabla 5.31: Comparación de los métodos Bradley Terry M4

5.4. Comparación de los métodos

En esta sección se comparan los distintos métodos de aprendizaje supervisado empleados en el trabajo seleccionando únicamente los mejores en el apartado general, para cada superficie y para cada jugador/a. También aparecen clasificados según el sexo (ATP, WTA) y la manera de hacer la separación de datos de entrenamiento y test. Los métodos generales aparecen ordenados de mayor a menor precisión en cada tabla. En los métodos por superficie y jugadores solamente se selecciona el mejor método para cada uno.

En la Tabla 5.32, se comparan los distintos métodos de aprendizaje supervisado empleados en el trabajo para el circuito ATP. El método que ofrece una mayor precisión para el circuito masculino es el de Random Forest, seleccionando en el conjunto de prueba el útimo tercio de los datos y la transformación del ranking mediante la inversa. Esta es aproximadamente de 0.686.

Método	Trans rank	Train	Precisión
Random Forest	Inversa	2/3 1/3	0.686
Regresión Logística	Logarítmica	2/3 1/3	0.685
Regresión Logística	Raíz cuadrada	2/3 1/3	0.684
Random Forest	Raíz cuadrada	2/3 1/3	0.684

Tabla 5.32: Comparación métodos generales ATP

En la Tabla 5.33 se comparan los distintos métodos de aprendizaje supervisado empleados en el trabajo para el circuito WTA. Hay varios métodos empatados, siendo más precisos los métodos de Regresión Logística con las transformaciones del ranking logarítmica o raíz cuadrada y seleccionando el año 2023 en el conjunto de prueba. Esta precisión es de 0.674.

En la Tabla 5.34 se compara la precisión de los diferentes métodos en partidos ATP pero separando las superficies previamente, por lo que las variables `surface_clay` y `surface_grass` se han eliminado del modelo. También se han diferenciado las dos maneras de entrenar los datos que veníamos practicando anteriormente. Aparece la mejor precisión para cada superficie.

En la Tabla 5.35 se compara la precisión de los diferentes métodos en partidos WTA separando por superficies.

Método	Trans rank	Train	Precisión
Regresión Logística	Logarítmica	Test 2023	0.674
Regresión Logística	Raíz cuadrada	Test 2023	0.674
Regresión Logística	Raíz cuadrada	2/3 1/3	0.674
Support Vector Machines	Logarítmica	Test 2023	0.672

Tabla 5.33: Comparación métodos generales WTA

Superficie	Método	Entrenamiento	Precisión
Tierra batida	Random Forest	Test 2023	0.851
Hierba	Random Forest	2/3 1/3	0.811
Pista dura	Regresión Logística	2/3 1/3	0.708

Tabla 5.34: Comparación de los métodos por superficies ATP

Superficie	Método	Entrenamiento	Precisión
Tierra batida	Regresión Logística	2/3 1/3	0.713
Hierba	Regresión Logística	Test 2023	0.800
Pista dura	Regresión Logística	Test 2023	0.701

Tabla 5.35: Comparación de los métodos por superficies WTA

A continuación, se realizan las comparaciones mediante el otro enfoque del problema, consistente en analizar los partidos de un jugador en concreto. De esta manera se podrá predecir el desempeño de cada tenista y comparar lo predicho con lo que realmente ocurrió. En la Tabla 5.36 se comparan los resultados para algunos jugadores del ranking ATP.

Jugador	Método	Entrenamiento	Precisión
Djokovic	Regresión Logística	Test 2023	0.907
Alcaraz	Gradient Boosting	Test 2023	0.905
Hurkacz	Random Forest	2/3 1/3	0.851

Tabla 5.36: Comparación de los métodos por jugadores ATP

En la siguiente tabla 5.37 se comparan los resultados para algunas jugadoras del ranking WTA.

Jugadora	Método	Entrenamiento	Precisión
Swiatek	Random Forest	2/3 1/3	0.930
Siegemund	Random Forest	2/3 1/3	0.915
Gauff	Regresión Logística	2/3 1/3	0.847

Tabla 5.37: Comparación de los métodos por jugadoras WTA

Vamos a comprobar que este método de predecir por jugadoras ofrece unos

resultados razonables con el caso de Swiatek, una jugadora que en el año 2023 obtuvo un porcentaje de victorias de aproximadamente el 85 %. Si dijéramos que gana siempre, tendríamos una precisión de 0.85, lo cual no estaría mal aparentemente, pero en realidad no seríamos capaces de predecir cuando pierde esta jugadora. Si nos fijamos en la precisiones de los modelos en los que interviene Swiatek, cualquiera de los métodos ofrece una precisión superior a 0.85, siendo Random Forest el que mayor precisión obtiene (0.930) para la división de datos de entrenamiento y test "2/3 1/3".

En este caso interesa medir también la tasa de falsos positivos, que aparece en la Tabla 5.38, es decir, partidos en los que se ha predicho una victoria pero el resultado real fue derrota para la jugadora en cuestión. Para calcularlo, se extraen las predicciones con valor 1 (victoria) que difieren con el valor 0 de `player1_win` (derrota). El número de filas que se extraen se dividen por el total de predicciones de victoria.

La tasa de falsos positivos para Swiatek es razonable salvo con el método SVM, pero se comprueba también para otros jugadores masculinos con tasas de victoria en el año 2023 superiores al 80 % como son Djokovic y Alcaraz.

Para el jugador Djokovic observamos que además de tener una tasa de acierto elevada, somos capaces de predecir de manera efectiva qué partidos concretos va a perder ese jugador. En cambio, con Alcaraz varía bastante respecto a los algoritmos empleados. Random Forest y Gradient Boosting obtienen una tasa de menos de 0.2, mientras que la tasa falsos positivos de Regresión Logística y Support Vector Machines es algo elevada. Una explicación podría ser que Djokovic lleva muchos años jugando torneos y por lo tanto tenemos más datos para entrenar el modelo y ser más precisos en la predicción de sus partidos.

Método	Swiatek	Djokovic	Alcaraz
Regresión Logística	0.083	0.065	0.286
Random Forest	0.154	0.087	0.143
Support Vector Machines	0.333	0.111	0.300
Gradient Boosting	0.200	0.087	0.167

Tabla 5.38: Tasa de falsos positivos

Se añade también la tasa de falsos negativos en la Tabla 5.39, es decir, partidos en los que se ha predicho una derrota pero el resultado real fue victoria para la jugadora o jugador en cuestión. Para calcularlo se extraen las predicciones con valor 0 (derrota) que difieren con el valor 1 de `player1_win` (victoria). El número de filas que se extraen se dividen por el total de predicciones de derrota.

Nuestra hipótesis era que la tasa de falsos negativos fuera menor que la de falsos positivos ya que en estos jugadores top la dificultad estaba en predecir cuándo van a perder. Esto se cumple salvo para el caso de Djokovic, el cual sorprende con tasas de falsos negativos elevadas. Esto desmiente la conclusión previa sobre él,

Método	Swiatek	Djokovic	Alcaraz
Regresión Logística	0.083	0.250	0.149
Random Forest	0.085	0.375	0.134
Support Vector Machines	0.133	0.444	0.125
Gradient Boosting	0.070	0.375	0.081

Tabla 5.39: Tasa de falsos negativos

ya que el modelo infravalora su capacidad para ganar partidos. Una explicación podría ser que durante el año 2017 estuvo lesionado y en su regreso perdió muchos encuentros, lo cual entrena el modelo en su contra. También la pandemia de 2020 le afectó, ya que no participó en numerosos torneos por no tener certificado de vacunación, mermando así su ranking.

Por último, se representa en la Figura 5.32 una comparación de las distintas curvas ROC para los cuatro tipos de modelos de clasificación utilizados con la jugadora Iga Swiatek. Esta curva ROC compara la tasa de falsos positivos frente a la tasa de verdaderos positivos para distintos valores del "umbral" con la probabilidad necesaria para predecir que el jugador va a vencer. Para este caso concreto, compara los partidos que Swiatek gana pero son predichos como derrota frente a los partidos que Swiatek gana y se predicen como victoria para distintos valores de este umbral. El área bajo la curva ROC (AUC) es 0.900 para Gradient Boosting, 0.920 para random Forest, 0.934 para Support vector Machines y 0.944 para Regresión Logística, por lo que este último método sería el preferible en este aspecto para predecir partidos de la jugadora Swiatek. Vemos que hay métodos que parecen destacar ligeramente en relación a tasas de verdaderos positivos o en reducir las tasas de falsos negativos, lo que podría ser de utilidad para elegir el método si estamos interesados particularmente en alguna de estas características.

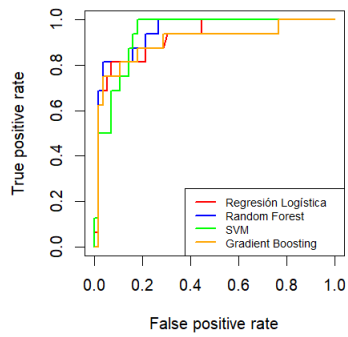

Figura 5.32: Comparación curvas ROC jugadora Swiatek

Capítulo 6

Conclusiones

6.1. Resultados destacados

Los resultados del trabajo se dividen en dos partes:

- Análisis descriptivo y técnicas de aprendizaje no supervisado para adquirir un cierto conocimiento inicial del funcionamiento de los torneos de tenis y de sus jugadores. Se han apreciado diferencias entre el circuito femenino y masculino y se han estudiado en más detalle los jugadores/as mejor clasificados en el ranking.

- Métodos predictivos con precisiones de aproximadamente 0.68 en modelos que utilizan todos los partidos, entre 0.71 y 0.85 si se predice según las superficies de juego y por último entre 0.82 y 0.92 si se predicen los partidos de jugadores concretos.

6.2. Líneas futuras

Según se iba desarrollando el proyecto, iban surgiendo nuevas ideas y mejoras. En el ámbito de la implementación de métodos predictivos se han introducido todas las que han funcionado, pero siempre puede caber alguna mejora en la precisión de los modelos ya sea transformando las variables de otra manera o con algoritmos diferentes.

Por ejemplo, las redes neuronales pueden servir como ampliación en el ámbito de nuestro problema ya que suelen obtener unos resultados acertados. En particular, el uso de redes neuronales recurrentes (RNN) podría ser investigado ya que hay un carácter secuencial claro (la evolución temporal de los jugadores es muy importante). Por contra, si se tomaran datos de muchos años, el esfuerzo computacional sería elevado. Además, las redes neuronales tienen una interpretación más difícil.

En el TFG se han realizado predicciones antes de jugar los partidos, pero también cabe la posibilidad de hacer predicciones dinámicas que evolucionen según avance el marcador del partido. Para ello se necesitarían datos más precisos sobre el comportamiento de cada jugador en las distintas fases de un partido de tenis.

Si se pudieran obtener datos más concretos sobre los partidos, como por ejemplo velocidad de los golpes, dirección, duración de cada punto, errores no forzados, derechas y reveses ganadores, etc, sería interesante realizar un análisis de los patrones de juego de cada jugador. Esto podría ayudar a mejorar las debilidades de cada jugador y a conocer mejor la manera de jugar de sus rivales.

Los elementos desarrollados podrían incluirse en algún tipo de aplicación para visualizarse de manera más interactiva. Esto debe hacerse de manera no comercial, ya que los datos utilizados así lo requieren.

Índice de figuras

Índice de tablas

Bibliografía

[1] Thulin, M. (2021). *Modern Statistics with R, from wrangling and exploring data to inference and predictive modelling*. Chapman and Hall/CRC.

[2] Posit, *RStudio IDE*. URL: `https://posit.co/downloads/` Fecha de acceso: 11-07-2024.

[3] Overleaf, *"About Us"*. URL: `https://es.overleaf.com/about` Fecha de acceso: 17-03-2024.

[4] Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

[5] Hastie, T., Tibshirani, R., Friedman, J. (2009) *The Elements of Statistical Learning. Data Mining, Inference, and Prediction. Second Edition*. Springer.

[6] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

[7] Kuhn, M., Johnson, K. (2013). *Applied Predictive Modeling*. Springer.

[8] Turner, H., Firth, D. *"Bradley-Terry Models in R: The BradleyTerry2 Package"*. URL: `https://cran.r-project.org/web/packages/BradleyTerry2/vignettes/BradleyTerry.pdf` Fecha de acceso: 13-03-2024.

[9] Wikipedia, *"Bradley-Terry model"*. URL:`https://en.wikipedia.org/wiki/Bradley%E2%80%93Terry_model` Fecha de acceso: 19-04-2024.

[10] USTA, *"Sistema de Puntuación: Juegos y Sets"*. URL: `https://www.usta.com/es/home/improve/tips-and-instruction/national/tennis-scoring-rules.html` Fecha de acceso: 19-02-2024.

[11] ATP Tour, *"Tenis Explicado: Aprende El Juego"*. URL: `https://www.atptour.com/es/news/tennis-explained-learn-the-game` Fecha de acceso: 19-02-2024.

[12] Nag, U. Olympics. *Cómo funciona el ranking mundial de tenis y en qué se diferencian los sistemas ATP y WTA*. URL: `https://olympics.com/es/noticias/como-funciona-ranking-mundial-tenis-diferencias-sistemas-atp-wta` Fecha de acceso: 21-02-2024.

[13] Tennisviz, *"Tennis Visuals"*. URL: `https://tennisviz.blogspot.com/2015/10/game-tree.html` Fecha de acceso: 19-02-2024.

[14] Durrett, R. *"Essentials of Stochastic Processes"*. Chapter 1: Markov Chains. URL: `https://services.math.duke.edu/~rtd/EOSP/EOSP2E.pdf` Fecha de acceso: 19-02-2024.

[15] Grebensjikova, A. *"Transition graph of a Markov chain modeling a tennis game"*. URL: `https://chart-studio.plotly.com/~a.n.grebensjikova/0.embed` Fecha de acceso: 11-07-2024.

[16] Sackmann, J. *"GitHub"*. URL: `https://github.com/JeffSackmann` Fecha de acceso: 17-02-2024.

[17] González de la Plaza, A. *"TennisBI : Aplicación Shiny para la visualización de datos de tenis". Trabajo de Fin de Grado en Universidad de Valladolid, año 2023*. URL: `https://uvadoc.uva.es/handle/10324/63235` Fecha de acceso: 19-02-2024.

[18] R-blogguers. *"Using R to study tennis players"*. URL: `https://www.r-bloggers.com/2017/02/using-r-to-study-tennis-players/` Fecha de acceso: 21-02-2024.

[19] Roel, T. *"RPubs - Regresión Logística - Aplicación en el Tenis"*. URL:`https://rpubs.com/tomroel/regresion-logistica-rafael-nadal` Fecha de acceso: 05-04-2024.

Apéndice A

Código R

A.1. Procesamiento de los datos

En esta sección se adjunta el código R empleado en este trabajo que procesa los datos, añade variables y las transforma.

```
#Librerias
library(readr)
library(data.table)
library(dplyr)
library(ggplot2)
library(stringr)
library(tidyr)
library(caret)
library(randomForest)
library(e1071)
library(xgboost)

setwd("C:/Users/Usuario/Documents/TFG/Estadistica/Datos/tennis_
    atp-master/")

#Agrupar los csv de partidos atp en uno solo
#Desde 2010 hasta 2023
lista_atp <- list()

for (ano in 2010:2023) {
  nombre_csv <- paste("atp_matches_", ano, ".csv", sep="")
  datos_atp_separados <- read.csv(nombre_csv)
  lista_atp[[ano]] <- datos_atp_separados
}

datos_partidos_atp <- do.call(rbind, lista_atp)
#write.csv(datos_partidos_atp, "datos_partidos_atp.csv")

attach(datos_partidos_atp)

#Recodificar
#Variable tourney_name
#Eliminar espacios en blanco
datos_partidos_atp$tourney_name <- trimws(datos_partidos_atp$
    tourney_name)
#US Open
```

```
datos_partidos_atp$tourney_name <- gsub("Us Open", "US Open",
    datos_partidos_atp$tourney_name)

#Eliminar Davis Cup por su complejidad y datos perdidos
#!grepl para quedarme con las filas que no contengan Davis Cup
datos_partidos_atp <- subset(datos_partidos_atp, !grepl("Davis
    Cup", tourney_name))

#Datos sin NAs en variables de ranking
datos_partidos_atp <- datos_partidos_atp[complete.cases(winner_
    rank, loser_rank), ]

#Seed poner 0 en los NAs
datos_partidos_atp$winner_seed <- replace(datos_partidos_atp$
    winner_seed, is.na(datos_partidos_atp$winner_seed), 0)
datos_partidos_atp$loser_seed <- replace(datos_partidos_atp$
    loser_seed, is.na(datos_partidos_atp$loser_seed), 0)

#Codificar bien las NextGen Finals y Tour FInals
datos_partidos_atp <- datos_partidos_atp %>% mutate(tourney_
    level = ifelse(tourney_name == "NextGen Finals", "F", tourney
    _level))
datos_partidos_atp <- datos_partidos_atp %>% mutate(tourney_
    level = ifelse(tourney_name == "Tour Finals", "F", tourney_
    level))

#Extraer las filas que quedan con NAs
datos_partidos_atp[rowSums(is.na(datos_partidos_atp)) > 0, ]
#Son menos de 6000, las elimino
datos_partidos_atp <- na.omit(datos_partidos_atp)

#Transformar categoricas a numericas
datos_partidos_atp$surface <- as.numeric(factor(datos_partidos_
    atp$surface))
datos_partidos_atp$round <- as.numeric(factor(datos_partidos_atp
    $round))

#Anadir algunas varibles utiles
datos_partidos_atp$date_new <- as.Date(as.character(datos_
    partidos_atp$tourney_date),'%Y%m%d')
datos_partidos_atp$year <- format(datos_partidos_atp$date_new,'%
    Y')
datos_partidos_atp$id <- 1:nrow(datos_partidos_atp)
#datos_partidos_atp$mix_id <- paste(datos_partidos_atp$winner_id
    , datos_partidos_atp$loser_id, sep = "_")
#datos_partidos_atp$mix_id_order <- paste(pmin(datos_partidos_
    atp$winner_id, datos_partidos_atp$loser_id),
#                                         pmax(datos_partidos_
    atp$winner_id, datos_partidos_atp$loser_id),
#                                         sep = "_")
datos_partidos_atp$player1_id <- pmin(datos_partidos_atp$winner_
    id, datos_partidos_atp$loser_id)
```

```
datos_partidos_atp$player2_id <- pmax(datos_partidos_atp$winner_
    id, datos_partidos_atp$loser_id)

#Unir las variables nuevas al dataframe
datos_partidos_atp <- datos_partidos_atp %>%
  mutate(
    avg_age1 = ifelse(player1_id == winner_id, winner_age, loser
        _age),
    avg_rank1 = ifelse(player1_id == winner_id, winner_rank,
        loser_rank),
    avg_height1 = ifelse(player1_id == winner_id, winner_ht,
        loser_ht),
    avg_ace1 = ifelse(player1_id == winner_id, w_ace, l_ace),
    avg_df1 = ifelse(player1_id == winner_id, w_df, l_df),
    avg_svpt1 = ifelse(player1_id == winner_id, w_svpt, l_svpt),
    avg_1stIn1 = ifelse(player1_id == winner_id, w_1stIn, l_1
        stIn),
    avg_1stWon1 = ifelse(player1_id == winner_id, w_1stWon, l_1
        stWon),
    avg_2ndWon1 = ifelse(player1_id == winner_id, w_2ndWon, l_2
        ndWon),
    avg_SvGms1 = ifelse(player1_id == winner_id, w_SvGms, l_
        SvGms),
    avg_bpSaved1 = ifelse(player1_id == winner_id, w_bpSaved, l_
        bpSaved),
    avg_bpFaced1 = ifelse(player1_id == winner_id, w_bpFaced, l_
        bpFaced),
    avg_age2 = ifelse(player2_id == winner_id, winner_age, loser
        _age),
    avg_rank2 = ifelse(player2_id == winner_id, winner_rank,
        loser_rank),
    avg_height2 = ifelse(player2_id == winner_id, winner_ht,
        loser_ht),
    avg_ace2 = ifelse(player2_id == loser_id, l_ace, w_ace),
    avg_df2 = ifelse(player2_id == loser_id, l_df, w_df),
    avg_svpt2 = ifelse(player2_id == loser_id, l_svpt, w_svpt),
    avg_1stIn2 = ifelse(player2_id == loser_id, l_1stIn, w_1stIn
        ),
    avg_1stWon2 = ifelse(player2_id == loser_id, l_1stWon, w_1
        stWon),
    avg_2ndWon2 = ifelse(player2_id == loser_id, l_2ndWon, w_2
        ndWon),
    avg_SvGms2 = ifelse(player2_id == loser_id, l_SvGms, w_SvGms
        ),
    avg_bpSaved2 = ifelse(player2_id == loser_id, l_bpSaved, w_
        bpSaved),
    avg_bpFaced2 = ifelse(player2_id == loser_id, l_bpFaced, w_
        bpFaced),
    diff_avg_age = avg_age1 - avg_age2,
    diff_avg_rank = avg_rank1 - avg_rank2,
    diff_avg_height = avg_height1 - avg_height2,
    diff_avg_ace = avg_ace1 - avg_ace2,
```

```
    diff_avg_df = avg_df1 - avg_df2,
    diff_avg_svpt = avg_svpt1 - avg_svpt2,
    diff_avg_1stIn = avg_1stIn1 - avg_1stIn2,
    diff_avg_1stWon = avg_1stWon1 - avg_1stWon2,
    diff_avg_2ndWon = avg_2ndWon1 - avg_2ndWon2,
    diff_avg_SvGms = avg_SvGms1 - avg_SvGms2,
    diff_avg_bpSaved = avg_bpSaved1 - avg_bpSaved2,
    diff_avg_bpFaced = avg_bpFaced1 - avg_bpFaced2,
    player1_win_count = ave(ifelse(player1_id == winner_id, 1,
        0), player1_id, FUN = cumsum),
    player2_win_count = ave(ifelse(player2_id == winner_id, 1,
        0), player2_id, FUN = cumsum),
    player1_play_count = ave(ifelse(player1_id == winner_id |
        player1_id == loser_id, 1, 0), player1_id, FUN = cumsum),
    player2_play_count = ave(ifelse(player2_id == winner_id |
        player2_id == loser_id, 1, 0), player2_id, FUN = cumsum),
    diff_win_count= player1_win_count - player2_win_count,
    diff_rank_frac = 1 / avg_rank1 - 1 / avg_rank2,
    diff_rank_log = log(avg_rank1) - log(avg_rank2),
    player1_win = ifelse(player1_id == winner_id, 1, 0),
    player1_prop = player1_win_count / player1_play_count,
    player2_prop = player2_win_count / player2_play_count,
    diff_prop = player1_prop - player2_prop,
    #Interaccion
    log_rank1 = log(avg_rank1),
    inter_rank = log_rank1*diff_rank_log
  )

#Nuevas variables de forma reciente
#Ultimos 10 partidos, prop victorias
#Ultimos 3 meses, prop victorias
datos_partidos_atp$prop_last10matches_p1 <- NA
datos_partidos_atp$prop_last10matches_p2 <- NA

for (i in 1:nrow(datos_partidos_atp)) {
  player1 <- datos_partidos_atp$player1_id[i]
  player2 <- datos_partidos_atp$player2_id[i]
  fecha <- datos_partidos_atp$date_new[i]
  #ultimos 10 partidos de cada jugador
  ultimos_10_partidos_player1 <- subset(datos_partidos_atp, (
      player1_id == player1 | player2_id == player1) & date_new <
       fecha)[1:10, ]
  ultimos_10_partidos_player2 <- subset(datos_partidos_atp, (
      player1_id == player2 | player2_id == player2) & date_new <
       fecha)[1:10, ]
  #Proporcion
  prop_last10matches_player1 <- if (nrow(ultimos_10_partidos_
      player1) > 0) {
    mean(ifelse(ultimos_10_partidos_player1$winner_id == player1
        , 1, 0))
  } else {
    0
```

```
  }
  prop_last10matches_player2 <- if (nrow(ultimos_10_partidos_
      player2) > 0) {
    mean(ifelse(ultimos_10_partidos_player2$winner_id == player2
        , 1, 0))
  } else {
    0
  }

  datos_partidos_atp[i, "prop_last10matches_p1"] <- prop_
      last10matches_player1
  datos_partidos_atp[i, "prop_last10matches_p2"] <- prop_
      last10matches_player2
}

#Reemplazar Nas por 0
datos_partidos_atp[is.na(datos_partidos_atp)] <- 0
datos_partidos_atp

#Diferencias
datos_partidos_atp$diff_prop_last10matches <- datos_partidos_atp
    $prop_last10matches_p1 - datos_partidos_atp$prop_
    last10matches_p2
datos_partidos_atp

#Prueba eliminar primeros partidos ya que no hay casi partidos
    previos
#2%
eliminar <- round(0.02 * nrow(datos_partidos_atp))
datos_partidos_atp <- datos_partidos_atp[-seq_len(eliminar), ]

#Extraer las variables
datos_partidos <- datos_partidos_atp[, c("surface", "round", "
    minutes", "diff_avg_age", "log_rank1", "diff_rank_log", "
    inter_rank", "diff_avg_height", "diff_win_count", "diff_prop"
    , "diff_prop_last10matches", "player1_win")]

#Min-max normalization
min_max_normalization <- function(x) {
  (x - min(x)) / (max(x) - min(x))
}
datos_partidos_minmax <- as.data.frame(lapply(datos_partidos,
    min_max_normalization))

#Normalizacion con scale(). center y scaler true
datos_partidos_normalizados <- scale(datos_partidos[,c
    (-1,-2,-12)], center = TRUE, scale = TRUE)
datos_partidos_normalizados <- as.data.frame(datos_partidos_
    normalizados)
datos_partidos_normalizados <- cbind(surface=datos_partidos$
    surface, round=datos_partidos$round, datos_partidos_
    normalizados, player1_win=datos_partidos$player1_win)
```

A.2. Estudio descriptivo y analisis no supervisado

En esta sección se adjunta el código R empleado en este trabajo para realizar un estudio descriptivo y análisis no supervisado.

```
#Resumen variables numericas
datos_partidos_atp_compl_numeric<-datos_partidos_atp[, -c
    (1,2,3,4,5,6,7,8,9,10,11,12,14,16,17,18,19,20,22,24,25,26)]
summary(datos_partidos_atp_compl_numeric)

#Deteccion de outliers variables numericas
boxplot(winner_ht, main="Boxplot winner_ht")
boxplot(winner_age, main="Boxplot winner_age")
boxplot(loser_ht, main="Boxplot loser_ht")
boxplot(loser_age, main="Boxplot loser_age")
boxplot(w_ace, main="Boxplot w_ace")
boxplot(w_df, main="Boxplot w_df")
boxplot(w_svpt, main="Boxplot w_svpt")
boxplot(w_1stIn, main="Boxplot w_1stIn")
boxplot(w_1stWon, main="Boxplot w_1stWon")
boxplot(w_2ndWon, main="Boxplot w_2ndWon")
boxplot(w_SvGms, main="Boxplot w_SvGms")
boxplot(w_bpSaved, main="Boxplot w_bpSaved")
boxplot(l_bpFaced, main="Boxplot l_bpFaced")
boxplot(l_df, main="Boxplot l_df")
boxplot(l_1stIn, main="Boxplot l_1stIn")
boxplot(l_1stWon, main="Boxplot l_1stWon")
boxplot(l_2ndWon, main="Boxplot l_2ndWon")
boxplot(l_SvGms, main="Boxplot l_SvGms")
boxplot(l_bpSaved, main="Boxplot l_bpSaved")
boxplot(l_bpFaced, main="Boxplot l_bpFaced")
#Lo que se ve como outlier es por el partido de Wimbledon entre
    Isner y Mahut

#Anadir algunas varibles utiles
datos_partidos_atp$date_new <- as.Date(as.character(datos_
    partidos_atp$tourney_date),'%Y%m%d')
datos_partidos_atp$year <- format(datos_partidos_atp$date_new,'%
    Y')
datos_partidos_atp$id <- 1:nrow(datos_partidos_atp)
#datos_partidos_atp$mix_id <- paste(datos_partidos_atp$winner_id
    , datos_partidos_atp$loser_id, sep = "_")
#datos_partidos_atp$mix_id_order <- paste(pmin(datos_partidos_
    atp$winner_id, datos_partidos_atp$loser_id),
#                                         pmax(datos_partidos_
    atp$winner_id, datos_partidos_atp$loser_id),
#                                         sep = "_")
datos_partidos_atp$player1_id <- pmin(datos_partidos_atp$winner_
    id, datos_partidos_atp$loser_id)
datos_partidos_atp$player2_id <- pmax(datos_partidos_atp$winner_
    id, datos_partidos_atp$loser_id)
```

```
#Crear un array que almacene el nombre de todos los jugadores
    sin que se repitan
lista_jugadores <- unique(c(winner_name, loser_name))
lista_jugadores

#Unir las variables nuevas al dataframe
datos_partidos_atp <- datos_partidos_atp %>%
  mutate(
    avg_age1 = ifelse(player1_id == winner_id, winner_age, loser
        _age),
    avg_rank1 = ifelse(player1_id == winner_id, winner_rank,
        loser_rank),
    avg_height1 = ifelse(player1_id == winner_id, winner_ht,
        loser_ht),
    avg_ace1 = ifelse(player1_id == winner_id, w_ace, l_ace),
    avg_df1 = ifelse(player1_id == winner_id, w_df, l_df),
    avg_svpt1 = ifelse(player1_id == winner_id, w_svpt, l_svpt),
    avg_1stIn1 = ifelse(player1_id == winner_id, w_1stIn, l_1
        stIn),
    avg_1stWon1 = ifelse(player1_id == winner_id, w_1stWon, l_1
        stWon),
    avg_2ndWon1 = ifelse(player1_id == winner_id, w_2ndWon, l_2
        ndWon),
    avg_SvGms1 = ifelse(player1_id == winner_id, w_SvGms, l_
        SvGms),
    avg_bpSaved1 = ifelse(player1_id == winner_id, w_bpSaved, l_
        bpSaved),
    avg_bpFaced1 = ifelse(player1_id == winner_id, w_bpFaced, l_
        bpFaced),
    avg_age2 = ifelse(player2_id == winner_id, winner_age, loser
        _age),
    avg_rank2 = ifelse(player2_id == winner_id, winner_rank,
        loser_rank),
    avg_height2 = ifelse(player2_id == winner_id, winner_ht,
        loser_ht),
    avg_ace2 = ifelse(player2_id == loser_id, l_ace, w_ace),
    avg_df2 = ifelse(player2_id == loser_id, l_df, w_df),
    avg_svpt2 = ifelse(player2_id == loser_id, l_svpt, w_svpt),
    avg_1stIn2 = ifelse(player2_id == loser_id, l_1stIn, w_1stIn
        ),
    avg_1stWon2 = ifelse(player2_id == loser_id, l_1stWon, w_1
        stWon),
    avg_2ndWon2 = ifelse(player2_id == loser_id, l_2ndWon, w_2
        ndWon),
    avg_SvGms2 = ifelse(player2_id == loser_id, l_SvGms, w_SvGms
        ),
    avg_bpSaved2 = ifelse(player2_id == loser_id, l_bpSaved, w_
        bpSaved),
    avg_bpFaced2 = ifelse(player2_id == loser_id, l_bpFaced, w_
        bpFaced),
    winner_rank_frac = 1 / winner_rank,
```

```
    loser_rank_frac = 1 / loser_rank,
    winner_rank_log = log(winner_rank),
    loser_rank_log = log(loser_rank),
    winner_rank_sqrt = sqrt(winner_rank),
    loser_rank_sqrt = sqrt(loser_rank),
    player1_win = ifelse(player1_id == winner_id, 1, 0)
  )

#Correlacion entre variables
cor(x=datos_partidos_atp$avg_rank1,y=datos_partidos_atp$avg_
    rank2) #0.08663417
cor(x=datos_partidos_atp$avg_rank1,y=(datos_partidos_atp$avg_
    rank1-datos_partidos_atp$avg_rank2)) #0.6205655
cor(x=datos_partidos_atp$avg_rank1,y=(datos_partidos_atp$avg_
    rank1*(datos_partidos_atp$avg_rank1-datos_partidos_atp$avg_
    rank2))) #0.7307631

#Correlation plot
#Variables winner
#pairs(datos_partidos_atp_compl_numeric[,c
    (1,2,5,6,7,8,9,10,11,12,13,14,24)])
#pairs(datos_partidos_atp_compl_numeric[best_of == 3, c
    (5,8,9,10,11,12)])
#pairs(datos_partidos_atp_compl_numeric[best_of == 5, c
    (5,8,9,10,11,12)])
#Variables loser
#pairs(datos_partidos_atp_compl_numeric[,c
    (3,4,5,15,16,17,18,19,20,21,22,23,26)])
#Variables winner-loser enfrentadas
datos_partidos_atp_compl_numeric_wl <- subset(datos_partidos_atp
    , winner_rank <= 1000 & loser_rank <= 1000)
plot(datos_partidos_atp_compl_numeric_wl$winner_rank,datos_
    partidos_atp_compl_numeric_wl$loser_rank, xlab = "winner_rank
    ", ylab = "loser_rank")
plot(datos_partidos_atp_compl_numeric_wl$winner_rank_log,datos_
    partidos_atp_compl_numeric_wl$loser_rank_log, xlab = "winner_
    rank_log", ylab = "loser_rank_log")
plot(datos_partidos_atp_compl_numeric_wl$winner_rank_frac,datos_
    partidos_atp_compl_numeric_wl$loser_rank_frac, xlab = "winner
    _rank_frac", ylab = "loser_rank_frac")
plot(datos_partidos_atp_compl_numeric_wl$winner_rank_sqrt,datos_
    partidos_atp_compl_numeric_wl$loser_rank_sqrt, xlab = "winner
    _rank_sqrt", ylab = "loser_rank_sqrt")
plot(datos_partidos_atp_compl_numeric$w_ace,datos_partidos_atp_
    compl_numeric$l_ace, xlab = "w_ace", ylab = "l_ace")
plot(datos_partidos_atp_compl_numeric$w_1stIn,datos_partidos_atp
    _compl_numeric$l_1stIn, xlab = "w_1stIn", ylab = "l_1stIn")

#Matriz de correlacion
round(cor(datos_partidos_atp_compl_numeric[c(5,8,9,10,11,12)])
    ,2)
```

```
#Analisis descriptivo con los datos_partidos_atp
#Analisis descriptivo edad de los ganadores y perdedores en 2022
datos_partidos_atp_edad <- datos_partidos_atp[!is.na(datos_
    partidos_atp$winner_age),]
datos_partidos_atp_edad <- datos_partidos_atp_edad[!is.na(datos_
    partidos_atp_edad$loser_age),]
datos_partidos_atp_edad
var(datos_partidos_atp_edad$winner_age)
var(datos_partidos_atp_edad$loser_age)
hist(datos_partidos_atp_edad$winner_age, col = "green", main = "
    Edad Ganadores", xlab="Edad ganadores", ylab = "Frecuecia")
hist(datos_partidos_atp_edad$loser_age, col = "green", main = "
    Edad Perdedores", xlab="Edad perdedores", ylab = "Frecuecia")
boxplot(datos_partidos_atp_edad$winner_age, col = "green", main
    = "Edad Ganadores")
boxplot(datos_partidos_atp_edad$loser_age, col = "green", main =
     "Edad Perdedores")

#Analisis descriptivo superficie de juego
str(surface)
barplot(table(surface)[3:5], col=c("red","green","blue"), ylim=c
    (0,30000), xlab="Superficie", ylab = "Numero de partidos")
#Tabla de frecuencias
table(surface)[2:5]
prop.table(table(surface))[2:5]

#Numero total de torneos ATP en estas temporadas
num_torneos<-length(unique(tourney_name))
num_torneos

#Torneos de cada categoria (tourney_level) por ano
torneos_por_categoria <- datos_partidos_atp %>% group_by(year,
    tourney_level) %>% summarise(num_torneos = n_distinct(tourney
    _name)) %>% pivot_wider(names_from = year, values_from = num_
    torneos)
torneos_por_categoria

#Grafico torneos por ano
datos_torneos <- datos_partidos_atp %>% group_by(year) %>%
    summarize(num_torneos = n_distinct(tourney_id))
ggplot(datos_torneos, aes(x = factor(year), y = num_torneos)) +
  geom_bar(stat = "identity", fill = "purple") +
  labs(x = "Ano",
       y = "Numero de torneos") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ylim(0, 80)

#Analisis de la nacionalidad
#Partidos de jugadores de USA
USA_plays<-datos_partidos_atp %>% filter(winner_ioc == "USA" |
    loser_ioc == "USA")
```

```
USA_plays

#Veces que una nacionalidad participa en un partido
sort(table(c(winner_ioc,loser_ioc)),decreasing=T)

#Comparativa estatura vs numero de aces
altura_aces <- ggplot(datos_partidos_atp, aes(x = winner_ht, y =
    w_ace/w_SvGms)) +
  geom_point(pch = 1, col = "black") +
  labs(x = "Altura en cm", y = "Aces por juego") +
  theme_bw()
altura_aces + geom_smooth(method = "loess")


#Analisis de la variable score
#Crear variable numero de sets jugados
# Crear la variable sets_played contando el numero de guiones en
    la variable score
datos_partidos_atp$sets_played <- str_count(datos_partidos_atp$
    score, "-")

#Contar el numero de partidos en los que hay al menos un tie
    break
num_tiebreak <- sum(str_detect(datos_partidos_atp$score, "\\("))
num_tiebreak / nrow(datos_partidos_atp)

#Porcentaje de veces que vence un jugador con ranking inferior
win_no_favorito <- datos_partidos_atp %>%
  filter(winner_rank > loser_rank) %>%
  count()

win_no_favorito/nrow(datos_partidos_atp)

datos_2023 <- datos_partidos_atp %>% filter(year == 2023)
#Jugadores que disputan mas partidos en 2023
torneos_por_jugador <- datos_2023 %>%
  mutate(jugador = ifelse(winner_name == loser_name, winner_name
      , c(winner_name, loser_name))) %>%
  group_by(jugador) %>%
  summarise(torneos_disputados = n_distinct(tourney_name)) %>%
  arrange(desc(torneos_disputados))

#Dataframe con porcentaje de victorias de cada jugador y otros
    datos de interes
todos_jugadores <- unique(c(datos_2023$winner_name, datos_2023$
    loser_name))

#Crear un DataFrame vacio para almacenar la informacion de los
    jugadores
df_jugadores <- data.frame(nombre = todos_jugadores,
    stringsAsFactors = FALSE)
```

```r
#Funcion para calcular el porcentaje de victorias de un jugador
    en 2023
calcular_porcentaje_victorias <- function(nombre_jugador) {
  total_partidos <- sum(datos_2023$winner_name == nombre_jugador
      | datos_2023$loser_name == nombre_jugador)
  victorias <- sum(datos_2023$winner_name == nombre_jugador)
  porcentaje_victorias <- (victorias / total_partidos) * 100
  return(porcentaje_victorias)
}

#Funcion para calcular las estadisticas por partido
calcular_estadisticas_por_partido <- function(nombre_jugador,
    variable) {
  total_partidos <- sum(datos_2023$winner_name == nombre_jugador
      | datos_2023$loser_name == nombre_jugador)
  total_variable <- sum(datos_2023[[paste0("w_", variable)]][
      datos_2023$winner_name == nombre_jugador],
                        datos_2023[[paste0("l_", variable)]][
                            datos_2023$loser_name == nombre_
                            jugador])
  media_variable <- total_variable / total_partidos
  return(media_variable)
}

#Funcion para calcular el tiempo en pista
calcular_tiempo_en_pista <- function(nombre_jugador) {
  datos_jugador <- filter(datos_2023, winner_name == nombre_
      jugador | loser_name == nombre_jugador)
  total_minutos <- sum(datos_jugador$minutes)
  total_horas <- total_minutos / 60
  return(total_horas)
}

df_jugadores$porcentaje_victoria <- sapply(df_jugadores$nombre,
    calcular_porcentaje_victorias)
variables_por_partido <- c("ace", "df", "svpt", "1stIn", "SvGms"
    , "bpSaved", "bpFaced")

for (variable in variables_por_partido) {
  df_jugadores[[variable]] <- sapply(df_jugadores$nombre,
      function(nombre_jugador) {
    calcular_estadisticas_por_partido(nombre_jugador, variable)
  })
}
df_jugadores$jug <- sapply(df_jugadores$nombre, function(nombre_
    jugador) {
  total_partidos <- sum(datos_2023$winner_name == nombre_jugador
      | datos_2023$loser_name == nombre_jugador)
  return(total_partidos)
})
df_jugadores$minutes <- sapply(df_jugadores$nombre, calcular_
    tiempo_en_pista)
```

```
df_jugadores <- df_jugadores %>% filter(jug >= 10) %>% rename(
    time = minutes)
df_jugadores <- df_jugadores %>% arrange(desc(porcentaje_
    victoria))
df_jugadores_round<-round(df_jugadores[,c(2:11)], 2)
df_jugadores<-cbind(df_jugadores[,1],df_jugadores_round)

#Mejores jugadores %victoria
#Jugadores que acabaron en el top 10 la temporada 2023
nombre_jugador <- c("Novak Djokovic", "Carlos Alcaraz", "Daniil
    Medvedev", "Jannik Sinner", "Andrey Rublev",
                    "Stefanos Tsitsipas","Alexander Zverev", "
                        Holger Rune", "Hubert Hurkacz", "Taylor
                        Fritz")

#Data frame con los jugadores y sus porcentajes
#Referencia R-blogguers
tabla_porcentaje_victorias_top10 <- data.frame()
for (jugador in nombre_jugador) {
  seleccion_jugadores <- datos_partidos_atp[(datos_partidos_atp$
      winner_name == jugador) | (datos_partidos_atp$loser_name ==
       jugador), ]
  seleccion_jugadores$nombre <- jugador
  tabla_seleccion_jugadores <- seleccion_jugadores %>% group_by(
      nombre, year) %>%
    summarise('Porcentaje_de_victorias' = sum(winner_name ==
        nombre)/n(), 'Partidos jugados' = n())

  tabla_porcentaje_victorias_top10 <- bind_rows(tabla_porcentaje
      _victorias_top10, tabla_seleccion_jugadores)
}

#Plot
tabla_porcentaje_victorias_top10_2020_2023 <- filter(tabla_
    porcentaje_victorias_top10, year >= 2020 & year <= 2023)
ggplot(tabla_porcentaje_victorias_top10_2020_2023, aes(x = as.
    numeric(year), y = Porcentaje_de_victorias, color = nombre))
    +
  geom_line() +
  labs(x = "Ano", y = "Porcentaje de victorias") +
  theme_minimal()

#nombre_jugador eran los jugadores top10 al finalizar 2023
#Extraer estadisticas y comparaciones de ellos en ese ano
id_jugador_top10 <- c("104925", "207989", "106421", "206173", "
    126094",
                      "126774","100644", "208029", "128034", "
                          126203")

partidos_top10 <- subset(datos_partidos_atp_rep_perjug, name %in
    % nombre_jugador)
partidos_top10_2023 <- subset(partidos_top10, year == 2023)
```

```
ggplot(data = partidos_top10_2023, aes(x = aces, y = df)) +
  geom_point(aes(color = nombre_jugador)) +
  labs(title = "Comparacion de Aces vs. Dobles Faltas",
       x = "Aces",
       y = "Dobles Faltas",
       color = "Jugador") +
  theme_minimal()


#Grafico torneos disputados cada ano
partidos_2023_2023 <- datos_partidos_atp %>%
  filter(winner_name %in% nombre_jugador | loser_name %in%
      nombre_jugador) %>%
  mutate(year = as.integer(format(date_new, "%Y"))) %>%
  filter(year >= 2020 & year <= 2023)

partidos_por_ano <- partidos_2023_2023 %>%
  group_by(jugador = ifelse(winner_name %in% nombre_jugador,
      winner_name, loser_name), year) %>%
  summarise(torneos_jugados = n_distinct(tourney_name))

partidos_por_ano <- as.data.frame(partidos_por_ano)

ggplot(partidos_por_ano, aes(x = year, y = torneos_jugados,
    color = jugador)) +
  geom_line() +
  labs(
       x = "Ano",
       y = "Numero de torneos jugados",
       color = "Jugador") +
  theme_minimal()


detach(datos_partidos_atp)

#Analisis de los torneos Grand Slam
#Al mejor de 5 sets
#Duracion de partido en los grand slam
grand_slam<- subset(datos_partidos_atp, tourney_level == "G")
grand_slam
boxplot(grand_slam$minutes ~ grand_slam$tourney_name, xlab="
    Grand Slam", ylab="Tiempo de juego", col=c("yellow","red","
    blue","green"), las=1, names=c("AO","RG","US","W"))


#PCA y Clustering por jugadores
#Lectura
ranking_actual <- read.csv("atp_rankings_current.csv")
datos_players <- read.csv("atp_players.csv")
#Fusionar datasets
```

```
datos_fusion <- merge(datos_players, ranking_actual, by = "
    player")
#Ordenar segun el ranking
datos_fusion <- datos_fusion[order(datos_fusion$rank, decreasing
    = FALSE), ]
datos_fusion$player_name <- paste(datos_fusion$name_first, datos
    _fusion$name_last, sep = " ")

nombre_df <- data.frame(player_name = datos_fusion[1:300,]$
    player_name)
nombre <- datos_fusion[1:300,]$player_name

#Porcentaje de victorias en cada superficie para cada jugador
superficies <- unique(datos_partidos_atp$surface)

for (superficie in superficies) {
  nombre_df[paste0("prop_victoria_", superficie)] <- rep(NA,
      nrow(nombre_df))
}

for (i in 1:nrow(nombre_df)) {
  jugador <- nombre[i]
  for (superficie in superficies) {
    partidos_jugador_superficie <- subset(datos_partidos_atp, (
        winner_name == jugador | loser_name == jugador) & surface
         == superficie)
    total_partidos <- nrow(partidos_jugador_superficie)
    if (total_partidos > 0) {
      victorias_jugador <- nrow(subset(partidos_jugador_
          superficie, winner_name == jugador))
      porcentaje_victorias <- victorias_jugador / total_partidos
           * 100
      nombre_df[i, paste0("prop_victoria_", superficie)] <-
          porcentaje_victorias
    } else {
      nombre_df[i, paste0("prop_victoria_", superficie)] <- 0
    }
  }
}


#Renombrar
nombre_df <- nombre_df %>%
  rename(clay_porc = prop_victoria_1,
         grass_porc = prop_victoria_2,
         hard_porc = prop_victoria_3)

#Anadir las demas variables de datos fusionados
nombre_df <- data.frame(nombre_df,
                        rank = datos_fusion[1:300,]$rank,
                        height = datos_fusion[1:300,]$height)
```

```
#Edad, viene fecha de nacimiento
fecha_now <- as.Date("2024-01-01")
datos_fusion$dob <- as.Date(as.character(datos_fusion$dob), "%Y%
    m%d")
datos_fusion$edad <- floor(as.numeric(difftime(fecha_now, datos_
    fusion$dob, units = "days") / 365))
nombre_df <- data.frame(nombre_df,
                        edad = datos_fusion[1:300,]$edad)

#Variable numero de victorias
nombre_df$win_count <- rep(0, nrow(nombre_df))
for (i in 1:nrow(nombre_df)) {
  jugador <- nombre[i]
  partidos_jugador <- subset(datos_partidos_atp, winner_name ==
      jugador | loser_name == jugador)
  victorias_jugador <- nrow(subset(partidos_jugador, winner_name
       == jugador))
  nombre_df[i, "win_count"] <- victorias_jugador
}

# Eliminar NAs
nombre_df <- nombre_df[complete.cases(nombre_df), ]

#PCA
pca_resultados <- prcomp(nombre_df[,c(2:8)], scale = TRUE)
#Resumen de los componentes principales
summary(pca_resultados)
#Graficar la proporcion de varianza explicada por cada
    componente
plot(pca_resultados, type = "l")
#Graficar el biplot (Componentes principales 1 y 2)
biplot(pca_resultados, scale = 0)

pca_primeras <- pca_resultados$x[, 1:2]
nombre_df <- cbind(nombre_df, pca_primeras)

#Clusters en funcion de los porcentajes de victoria por
    superficie
#k-means, 3 clusters
set.seed(123)
num_clusters <- 3
clustering <- kmeans(nombre_df[, -1], centers = num_clusters)

nombre_df$cluster <- clustering$cluster
print(nombre_df)
print(clustering$centers)

ggplot(nombre_df, aes(x = PC1, y = PC2, color = factor(cluster))
    ) +
  geom_point(size = 3) +
  labs(x = "PC1",
       y = "PC2",
```

```
        color = "Cluster") +
  theme_minimal()
```

A.3. Implementacion de los metodos

En esta sección se adjunta el código R empleado en este trabajo para realizar los métodos predictivos.

```
#Train / test
train <- round(2/3 * nrow(datos_partidos_normalizados))
test <- nrow(datos_partidos_normalizados) - train
datos_entrenamiento <- datos_partidos_normalizados[1:train, ]
datos_prueba <- datos_partidos_normalizados[(train+1):nrow(datos
    _partidos_normalizados), ]

#Primer partido 2023
primero_2023 <- which(year(datos_partidos_atp$date_new) == 2023)
    [1]
#Dividir los datos en conjuntos de entrenamiento y prueba
datos_entrenamiento <- datos_partidos_normalizados[1:(primero_
    2023 - 1), ]
datos_prueba <- datos_partidos_normalizados[primero_2023:nrow(
    datos_partidos_normalizados), ]

#Ajustar modelo de regresion logistica con datos de
    entrenamiento
modelo_logistico <- glm(as.factor(player1_win) ~ ., data = datos
    _entrenamiento, family = "binomial")
summary(modelo_logistico)

#Obtener probabilidades de victoria para datos de prueba
predicciones <- predict(modelo_logistico, datos_prueba, type = "
    response")
predicciones_log <- ifelse(predicciones > 0.5, 1, 0)
precision_rlog <- mean(predicciones_log == datos_prueba$player1_
    win)
precision_rlog

#arboles y Random Forests para clasificacion
#Factor para que haga clasificacion y no regresion
m_RF <- randomForest(as.factor(player1_win) ~ ., data = datos_
    entrenamiento)
plot(m_RF)
predicciones_rf <- predict(m_RF, datos_prueba)
precision_rf <- mean(predicciones_rf == datos_prueba$player1_win
    )
precision_rf

#Importancia de las variables
m_RF$importance
```

```
variables <- c("surface", "round", "diff_avg_height", "minutes",
     "diff_avg_age", "diff_prop_last3month","diff_win_count", "
    diff_prop_last10matches", "diff_rank_log", "diff_prop")
mean_decrease_gini <- c(4032.6038, 2112.4363, 1599.0638,
    1575.4125, 1398.4833, 1388.4449, 1008.0464, 951.0070,
    607.2130, 280.9903 )
barplot(sort(mean_decrease_gini),
        names.arg = variables,
        xlab = "Gini",
        col = "orange",
        xlim = c(0, max(mean_decrease_gini) * 1.1),
        las = 1,
        cex.names = 0.5,
        horiz = TRUE)

#SVM
modelo_svm <- svm(as.factor(player1_win) ~ ., data = datos_
    entrenamiento, kernel = "poly")
summary(modelo_svm)
predicciones_svm <- predict(modelo_svm, newdata = datos_prueba)
precision_svm <- sum(predicciones_svm == datos_prueba$player1_
    win) / length(predicciones_svm)
precision_svm

#Boosting
#Convertir datos a matriz
datos_entrenamiento_matriz <- as.matrix(datos_entrenamiento[, c
    (1:11)])
datos_prueba_matriz <- as.matrix(datos_prueba[, c(1:11)])

modelo_boosting <- xgboost(data = datos_entrenamiento_matriz,
    label = datos_entrenamiento$player1_win, nrounds = 100,
    objective = "binary:logistic")
predicciones <- predict(modelo_boosting, datos_prueba_matriz)
predicciones_boost <- ifelse(predicciones > 0.5, 1, 0)
precision_boost <- mean(predicciones_boost == datos_prueba$
    player1_win)
precision_boost

#Resumen de las precisiones
precisiones<-c(precision_rlog,precision_rf,precision_svm,
    precision_boost)
precisiones

#Tasas de falsos positivos para cada modelo
fp_rlog <- nrow(fallos_rlog) / sum(predicciones_rlog == 1)
fp_rf <- nrow(fallos_rf) / sum(predicciones_rf == 1)
fp_svm <- nrow(fallos_svm) / sum(predicciones_svm == 1)
fp_boost <- nrow(fallos_boost) / sum(predicciones_boost == 1)

#Mostrar las tasas de falsos positivos para cada modelo
fp <- c(fp_rlog, fp_rf, fp_svm, fp_boost)
```

```
names(fp) <- c("Regresion Logistica", "Random Forest", "SVM", "
    Gradient Boosting")
fp

#Conteo de victorias que realmente fueron derrotas
fallos_rlog_fn <- subset(datos_prueba_completos, player1_win !=
    predicciones_rlog & predicciones_rlog == 0)
fallos_rf_fn <- subset(datos_prueba_completos, player1_win !=
    predicciones_rf & predicciones_rf == 0)
fallos_svm_fn <- subset(datos_prueba_completos, player1_win !=
    predicciones_svm & predicciones_svm == 0)
fallos_boost_fn <- subset(datos_prueba_completos, player1_win !=
     predicciones_boost & predicciones_boost == 0)

#Tasas de falsos positivos para cada modelo
fn_rlog <- nrow(fallos_rlog_fn) / sum(predicciones_rlog == 0)
fn_rf <- nrow(fallos_rf_fn) / sum(predicciones_rf == 0)
fn_svm <- nrow(fallos_svm_fn) / sum(predicciones_svm == 0)
fn_boost <- nrow(fallos_boost_fn) / sum(predicciones_boost == 0)

#Mostrar las tasas de falsos positivos para cada modelo
fn <- c(fn_rlog, fn_rf, fn_svm, fn_boost)
names(fn) <- c("Regresion Logistica", "Random Forest", "SVM", "
    Gradient Boosting")
fn

#Curvas ROC
pred_prob_rlog <- predict(modelo_logistico, datos_prueba, type =
     "response")
pred_prob_rf <- predict(m_RF, datos_prueba, type = "prob")[,2]
pred_prob_svm <- predict(modelo_svm, datos_prueba, decision.
    values = TRUE)
pred_prob_svm <- attr(pred_prob_svm, "decision.values")
pred_prob_boost <- predict(modelo_boosting, datos_prueba_matriz)

perf_rlog <- performance(prediction(pred_prob_rlog, datos_prueba
    $player1_win), "tpr", "fpr")
perf_rf <- performance(prediction(pred_prob_rf, datos_prueba$
    player1_win), "tpr", "fpr")
perf_svm <- performance(prediction(pred_prob_svm, datos_prueba$
    player1_win), "tpr", "fpr")
perf_boost <- performance(prediction(pred_prob_boost, datos_
    prueba$player1_win), "tpr", "fpr")

plot(perf_rlog, col = "blue", lwd = 2)
plot(perf_rf, col = "red", add = TRUE, lwd = 2)
plot(perf_svm, col = "green", add = TRUE, lwd = 2)
plot(perf_boost, col = "orange", add = TRUE, lwd = 2)

legend("bottomright", legend = c("Regresion Logistica", "Random
    Forest", "SVM", "Gradient Boosting"), col = c("red", "blue",
    "green", "orange"), lwd = 2, cex=0.7)
```

```
#AUC
auc_rlog <- performance(pred_rlog, "auc")@y.values[[1]]
auc_rf <- performance(pred_rf, "auc")@y.values[[1]]
auc_svm <- performance(pred_svm, "auc")@y.values[[1]]
auc_gb <- performance(pred_boost, "auc")@y.values[[1]]
```

A.3.1. Implementacion Bradley Terry

```
library(BradleyTerry2)
library(brglm)

#Obtener los jugadores comunes
jugadores_comunes <- intersect(levels(player1_id), levels(
    player2_id))
jugadores_comunes <- as.factor(jugadores_comunes)

datos_Bradley_Terry <- subset(datos_partidos_atp, player1_id %in
    % jugadores_comunes & player2_id %in% jugadores_comunes)
datos_Bradley_Terry$player1_id <- factor(datos_Bradley_Terry$
    player1_id, levels = jugadores_comunes)
datos_Bradley_Terry$player2_id <- factor(datos_Bradley_Terry$
    player2_id, levels = jugadores_comunes)

datos_Bradley_Terry$player1_id
datos_Bradley_Terry$player2_id

#Train / test
indices_entrenamiento <- sample(1:nrow(datos_Bradley_Terry), 0.8
     * nrow(datos_Bradley_Terry))
datos_entrenamiento <- datos_Bradley_Terry[indices_entrenamiento
    , ]
datos_prueba <- datos_Bradley_Terry[-indices_entrenamiento, ]

#Modelos
modelo_bt <- BTm(outcome=player1_win, player1=player1_id,
    player2=player2_id, data = datos_entrenamiento)
summary(modelo_bt)
predicciones1 <- predict(modelo_bt, newdata = datos_prueba, type
     = "response")
predicciones_clases1 <- ifelse(predicciones1 > 0.5, 1, 0)
precision1 <- mean(predicciones_clases1 == datos_prueba$player1_
    win)
precision1 #0.6610899

modelo_bt4 <- BTm(player1_win, player1_id, player2_id,
                  ~ winner_rank[..] + loser_rank [..] +
                    minutes[..] + winner_ht [..] + loser_ht [..]
                        + (1|..),
                  family = binomial(link = "probit"), data =
                      datos_entrenamiento)
```

```
summary(modelo_bt4)
predicciones4 <- predict(modelo_bt4, newdata = datos_prueba,
    type = "response")
predicciones_clases4 <- ifelse(predicciones4 > 0.5, 1, 0)
precision4 <- mean(predicciones_clases4 == datos_prueba$player1_
    win)
precision4 #0.6590185
```