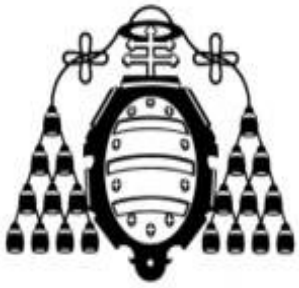

Modelos de predicción para partidas de élite en ajedrez

Pablo García Martínez

supervisado por

Enrique Miranda Menéndez e Ignacio Montes Gutiérrez

UNIVERSIDAD DE OVIEDO
Facultad de Ciencias
Máster en Análisis de Datos para la Inteligencia de Negocios
Trabajo Fin de Máster

Junio de 2022

Índice general

Resumen

El objetivo de este trabajo es la elaboración y comparación de varios modelos de predicción del resultado de las partidas de ajedrez entre jugadores de élite: victoria del jugador de blancas, tablas o victoria del jugador de negras.

Para ello, se realizó en primer lugar una revisión de la literatura, tal y como se detalla en el Capítulo 1. Nuestros modelos difieren en varios aspectos de los allí mencionados: por un lado, utilizaremos herramientas matemáticas diferentes de las consideradas en la literatura, como por ejemplo la regresión logística; por otro lado, nuestras predicciones van enfocadas a jugadores de élite (en el Top 100) de la clasificación; y por último, muchas de las variables predictoras consideradas son originales de este trabajo. En este capítulo introductorio se presenta también la base teórica de cada uno de los modelos tenidos en cuenta.

En el Capítulo 2 se hará una introducción al ajedrez acompañada de una descripción de las variables de interés en nuestros modelos.

A continuación, el Capítulo 3 detalla la obtención y depuración de la base de datos, resultante en un conjunto de más de 6000 partidas; los códigos de programación utilizados han sido incluidos en un Apéndice al final de esta memoria. Asimismo, se realiza un análisis estadístico de las distintas variables del conjunto de datos.

En el Capítulo 4 se expondrá cada uno de los modelos usados y se entrenarán para posteriormente comparar los resultados. En primer lugar se estudia si es posible obtener un modelo de regresión logística ordinal, siendo la variable dependiente el resultado de la partida. Posteriormente, se generarán diversos modelos de regresión logística multinomial. El primero de todos será un modelo con todas las variables, el cual, posteriormente, será simplificado. A continuación, se elaborarán estos modelos atendiendo al tipo de partida de ajedrez de la que se trate.

Seguidamente, se desarrollan tres modelos más, basados en árboles de decisión, K-vecinos más cercanos y máquinas de vector soporte. Todos estos modelos se entrenarán con el conjunto de datos obtenido en el capítulo anterior.

En el Capítulo 5 se realiza la validación de cada uno de los modelos, basado en un nuevo conjunto de datos. Además de todos los modelos desarrollados en el Capítulo 4, se considera también aquí un modelo más simple basado en el rating de cada uno de los jugadores.

Finalmente, en el Capítulo 6 se extraerán las conclusiones más relevantes y se presentan algunas posibles mejoras del modelo y otras líneas de trabajo futuro.

Capítulo 1

Introducción y conceptos preliminares

1.1. Revisión de la literatura

En primer lugar, hemos realizado una revisión de los modelos de predicción en la literatura. A continuación damos un breve resumen de los trabajos más importantes.

Chess Game Result Prediction System [1]

En este proyecto se realiza un modelo de entrenamiento para los sistemas de calificación de la Federación (FIDE) [2] haciendo uso de un conjunto de datos de un reciente período de once años con 2000 jugadores de ajedrez. Posteriormente se utiliza el modelo para predecir el resultado de las partidas. Este es muy diferente al que se va a realizar en este trabajo, ya que está basado en técnicas de procesos de Markov ocultos. Este es un modelo estadístico en el que se asume que el sistema a modelar es un proceso de Markov con parámetros desconocidos. En este trabajo, la fuerza del jugador es el estado oculto del proceso y también se considera la calificación del jugador en un mes determinado.

Entre las conclusiones del trabajo, se destaca que la variable que representa la fuerza del jugador sigue la distribución de una serie temporal ARIMA, mientras que la del resultado de la partida sigue una Bernoulli, cuyo parámetro es la probabilidad de victoria. En la validación realizada, se obtiene un porcentaje de acierto del $55'64\,\%$.

Predicting the Outcome of Chess Games based on Historical Data [3]

Este trabajo describe un enfoque usado en la competición "Puntuaciones de ajedrez - Elo versus Restos del Mundo", que tuvo lugar entre agosto y noviembre del año 2010. En él, se hace uso de las partidas llevadas a cabo por los jugadores participantes en esa competición y de un modelo conocido como Bradley-Terry [4], el cual es una herramienta relativamente sencilla y eficaz para obtener una clasificación de jugadores basada en las comparaciones por pares, es decir, el resultado de las partidas anteriores. Entre las hipótesis en las que se basa el modelo, destaca el postulado de que la fuerza de cada jugador puede venir aproximada por el puntaje esperado contra un desconocido jugador del que no se tiene calculada su fuerza. Este enfoque proporciona un procedimiento para calcular la fuerza de cada jugador basado en la fuerza de sus oponentes y en el puntaje promedio contra los

mismos. A su vez, este concepto de "fuerza" será de utilidad en nuestro trabajo, y es una de las inspiraciones para alguna de las variables con las que trabajaremos posteriormente.

En la validación realizada se obtiene un acierto cercano al 70 %.

Predicting Moves in Chess using Convolutional Neural Networks [5]

Este trabajo está basado en una red neuronal convolucional (CNN) de tres capas para hacer predicciones de jugadas en el ajedrez. El proyecto se divide en dos partes: por un lado, una CNN selectora de piezas es entrenada con el objetivo de anotar qué piezas blancas han de moverse, y las CNN selectoras de movimiento para cada pieza indican hacia dónde se pueden mover. A su vez, las redes fueron entrenadas utilizando 20.000 partidas consistentes de 245.000 movimientos realizados por jugadores con calificación ELO superior a 2000 del Free Internet Chess Server. Por ello, la población objetivo es diferente a la de nuestro trabajo, ya que nosotros hemos considerado a jugadores de élite (Top 100).

Además, este trabajo no pretende predecir el resultado, sino que tiene como objetivo detectar cuál va a ser el próximo movimiento.

Statistical Analysis on Result Prediction in Chess [6]

En este trabajo, los autores han propuesto una técnica que usa una base de datos existente de partidas de ajedrez y algoritmos de aprendizaje automático para predecir el resultado del juego. La base de datos cuenta con 10.000 partidas de ajedrez reales, importadas, elegidas de manera aleatoria sin importar quiénes eran los jugadores (a diferencia de nuestro trabajo, en el que nos centraremos en los jugadores de élite) y procesadas utilizando la base de datos de información de ajedrez de Shane (SCID); asimismo, se anota con puntuación de evaluación para cada medio movimiento utilizando el motor de ajedrez Stockfish.

Por su parte, en este trabajo se usan diferentes variables: número de movimientos, detalle de los movimientos, clasificación ECO del jugador, clasificación FIDE del jugador y, finalmente, el resultado. Aquí, hay grandes diferencias con nuestro trabajo, ya que aquí solo se considerará un sistema de rating.

Por su parte, en cuanto a los modelos usados en el estudio, cabe destacar que usan modelos como el Multi-Variante Lineal y Naive-Bayes para clasificar los diferentes atributos, obteniendo porcentajes de acierto cercanos al 70 %.

Predicting the Outcome of a Chess Game by Statistical and Machine Learning techniques [7]

Este trabajo está basado en la predicción de resultados de partidas de ajedrez tras veinte movimientos completados. Para ello, se hizo uso de una base de datos, a la que denominaron Base Gorgo, y que consta de tres millones de coincidencias y con jugadores de todos los rating, no solo de élite como en este trabajo. Tras limpiar la base de datos se le agregaron variables interesantes: nombre del torneo, lugar del torneo, fecha, jugador de blancas, jugador de negras, resultado. Notar que todas estas variables también son consideradas en nuestro trabajo.

Posteriormente, se elaboraron los modelos: un SVM con kernel RBF y lo compararon con un modelo de bosque aleatorio, obteniendo que funciona mejor el SVM.

1.2. Herramientas estadísticas

Pasamos a resumir brevemente las herramientas que utilizaremos en los modelos predictivos que desarrollaremos en esta memoria. Comenzamos pasando revista a las herramientas estadísticas, y más concretamente a la regresión logística.

1.2.1. Regresión logística binaria

En el ámbito estadístico, la regresión logística [8] es un tipo de análisis de regresión que se usa para predecir el resultado de una variable categórica a partir de una serie de variables, llamadas predictoras. Estas últimas, pueden ser tanto cuantitativas como cualitativas. Asimismo, atendiendo a si la variable respuesta tiene dos o más posibles niveles, se distinguen la regresión logística binaria y la multinomial.

Se considera modelo de regresión logística binaria a todo modelo de regresión logística cuya variable respuesta tiene dos posibles niveles. De esta manera, dicha variable será de tipo binario: Y $\sim$ B(p), siendo p la probabilidad de éxito. A su vez, estos modelos también están conformados por una serie de variables predictoras: $x_1, ..., x_n$, las cuales pueden ser tanto cuantitativas como cualitativas.

Por otra parte, se llama *odds* de un suceso al cociente entre la probabilidad de que ocurra frente a que no ocurra, es decir, $\frac{p}{1-p}$. Al logaritmo neperiano del odds se le conoce como función logit, y la regresión logística se basa en modelar el logit de la probabilidad de éxito desconocida como una función lineal de las variables predictoras x_j:

$$logit(p) = ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n + \epsilon.$$

siendo β_j (con j=1,...,n) parámetros desconocidos los cuales pueden ser estimados por el método de máxima verosimilitud; y, ϵ, el error.

De manera equivalente, la probabilidad buscada puede ser estimada como sigue:

$$p = \frac{1}{1+e^{-(\hat{\beta_0}+\hat{\beta_1}x_1+...+\hat{\beta_n}x_n)}}.$$

No obstante, tal y como se citó anteriormente, este modelo es eficaz para estudiar la relación entre una serie de variables independientes y una variable dependiente dicotómica. Así, dicho modelo no se puede aplicar en este trabajo, ya que la variable predictora (resultado) tiene 3 niveles: victoria, empate y derrota. Por ello, a continuación se van a presentar dos extensiones que permiten modelizar variables no dicotómicas.

1.2.2. Regresión logística ordinal

El modelo de regresión logística ordinal de odds acumulativos o proporcionales fue propuesto por McCullagh [9] y está basado en las hipótesis de que la variable respuesta es ordinal y los logaritmos de los odds forman una progresión aritmética. Considerando que hay m variables predictoras, sus ecuaciones logit son de la siguiente forma:

$$logit(P(Y \leq c)) = ln\frac{P(Y \leq c)}{P(Y > c)} = \alpha_c + \sum_{i=1}^{m} \beta_i x_i + \epsilon.$$

Por su parte, estos modelos son aplicables cuando la variable dependiente tiene asociado un orden, En definitiva, en nuestro caso, al tener una variable predictora ordinal con tres niveles (derrota<empate<victoria), el proceso consiste en reducir el modelo de regresión logística multinomial en dos modelos de regresión logística binaria:

- **Derrota vs Empate y Victoria**

$$logit(P(Y \leq Derrota)) = ln\left(\frac{P(Y \leq Derrota)}{P(Y > Derrota)}\right) = \alpha_{derrota} + \sum_{i=1}^{m} \beta_i x_i + \epsilon.$$

- **Derrota y Empate vs Victoria**

$$logit(P(Y \leq Empate)) = ln\left(\frac{P(Y \leq Empate}{P(Y > empate}\right) = \alpha_{empate} + \sum_{i=1}^{m} \beta_i x_i + \epsilon.$$

Para ello, en primer lugar se plantea el modelo ordinal con todas las variables y, a continuación se hace el test de líneas paralelas o de Odds proporcionales [10]. En él, la hipótesis nula es que los coeficientes de regresión son los mismos para las categorías de respuesta:

$$H_0 : \beta_i = cte.$$

Así que en esta ocasión necesitamos obtener un p-valor no significativo para que el modelo sea válido ($p \geq 0'05$).

1.2.3. Regresión logística multinomial

La regresión logística multinomial [11] es un tipo de modelo de regresión logística en la que la variable respuesta Y puede tomar $n + 1$ valores diferentes (siendo n mayor o igual que 2) y no tiene por qué existir un orden entre ellos (pero sí lo podría haber, como en este caso).

De esta manera, se fija una categoría de referencia de la variable respuesta, B_0, y se plantean tantas ecuaciones *logit* como categorías menos 1, siendo estas últimas denotadas por B_i (i: 1,...,número de categorías-1). Es decir, n ecuaciones de la forma:

$$ln\left(\frac{p_i}{p_0}\right) = \alpha_i + \sum_{j=1}^{m} \beta_{ij} \cdot x_j.$$

con i=1,...,n; j=1,...,m, con m siendo el número de variables predictoras;y denotando como $p_0 = P(B_0)$ y $p_i = P(B_i)$.

A su vez, haciendo uso de estas ecuaciones también quedan determinados los *logit* entre cualquier par de categorías:

$$\begin{aligned} ln\left(\frac{p_i}{p_k}\right) &= ln\left(\frac{p_i/p_0}{p_k/p_0}\right) = ln\left(\frac{p_i}{p_0}\right) - ln\left(\frac{p_k}{p_0}\right) \\ &= (\alpha_i + \sum_{j=1}^{m} \beta_{ij} \cdot x_j) - (\alpha_k + \sum_{j=1}^{m} \beta_{kj} \cdot x_j) = (\alpha_i - \alpha_k) + \left(\sum_{j=1}^{m} (\beta_{ij} - \beta_{kj}) \cdot x_j\right). \end{aligned}$$

Por su parte, para comprobar si las variables son significativas se hace uso del test de Wald [12], que tiene como hipótesis nula:

$$H_0 : \beta_i = 0 \ \forall \text{ i=1,...,n.}$$

Básicamente, una variable será significativa si su coeficiente asignado β_i es significativamente distinto de 0, por lo que una variable será importante para el modelo cuando se rechace la anterior hipótesis nula (p<0'05).

En cuanto a la bondad del modelo, esta se puede estudiar de dos formas:

- Test de la razón de verosimilitudes (TRV) [13]: La hipótesis nula H_0 en este test, plantea que todos los coeficientes del modelo son nulos. Por tanto, el modelo será adecuado cuando se obtenga significación estadística ($p < 0'05$).

- Pseudo R^2 [14, 15]: Cox y Snell, McFadden y Nagelkerke: En este caso, el significado de estos términos es distinto al R^2 de una regresión lineal. Así, mientras que para este último un coeficiente R^2 de McFadden es adecuado si es superior a 0'4, para la regresión logística multinomial será aceptable si es cercano a 0'1.

1.2.4. Árboles de clasificación

Los árboles de clasificación [16] están basados en la generación de diagramas de construcciones lógicas, muy similares a los sistemas de predicción basados en reglas, que sirven para representar y categorizar una serie de condiciones que ocurren de forma sucesiva, para la solución de un problema.

Los árboles de decisión están compuestos por nodos, vectores de números, flechas y etiquetas. Los nodos son los puntos en los que se ha de tomar una decisión entre varias posibles. De esta manera, a mayor número de nodos mayor número de posibles finales a los que se pueden llegar. Estos, por su parte, se encuentran unidos por flechas y, entre cada nodo y cada flecha, se encuentran las etiquetas, que dan nombre a cada acción. A su vez, los vectores de números son la solución final a la que se llega en función de las distintas posibilidades de las que se disponen.

Por su parte, una vez explicadas las partes que componen un árbol de decisión, cabe destacar cómo funciona este método. Este, tal y como se citó con anterioridad, consiste en la segmentación del espacio predictor (conjunto de posibles valores de las variables predictoras) en regiones más simples. Así, se parte de un nodo inicial que representa a toda la muestra y, de él salen dos ramas que dividen la muestra en dos subconjuntos, cada uno de los cuales es representado por un nuevo nodo. Este proceso se repite un número finito de veces, dando lugar a que el espacio predictor se divida en regiones de forma rectangular en la que la predicción de la respuesta es constante.

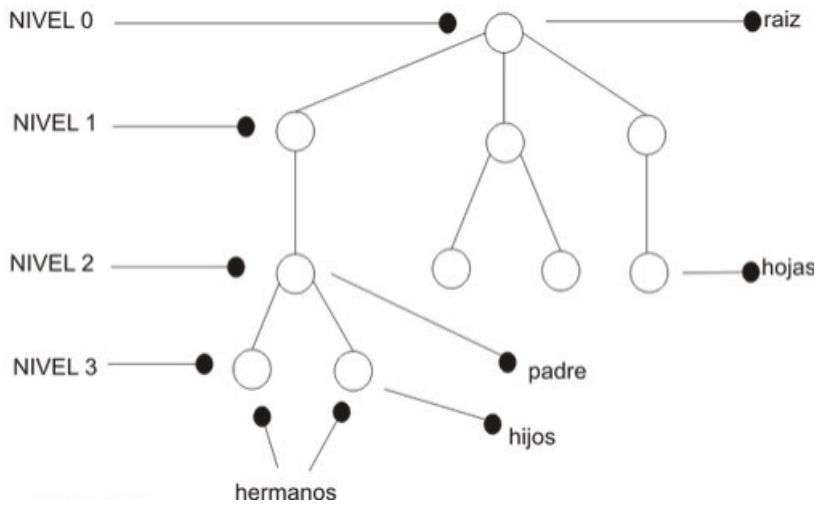

Figura 1.1: Ejemplo de árbol. Fuente: Google Sites

1.2.5. Máquinas de vector soporte

Las máquinas de vector soporte [17], también llamadas SVM, son un modelo lineal usado tanto para problemas de clasificación como para aquellos de regresión. La idea detrás de este algoritmo reside en hacer uso de un hiperplano para separar los datos en diferentes clases.

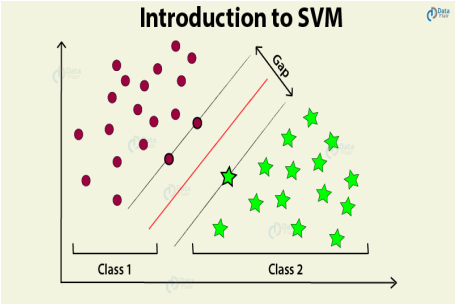

Figura 1.2: Ejemplo de máquinas de vector soporte. Fuente: vetyvet.com

Este tipo de algoritmos buscan el hiperplano con la mayor distancia con los puntos que estén más cerca de él mismo. De esta manera, en el caso en el que la variable a predecir solo tenga dos categorías, los puntos del vector que son etiquetados con una categoría se encontrarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado.

Por su parte, cuando se trata de predecir una variable con tres o más categorías, existen dos diferentes formas de realizar el SVM. La primera de ellas consiste en comparar una clase con otra (One-to-One) reiteradamente. La otra, por su parte, se basa en comparar una con el resto (One-to-Rest).

Por otra parte, uno de los hiperparámetros más importantes de este método es el "kernel". Esta es la función que usa el modelo para transformar los datos de entrada, y puede ser: "radial", "lineal", "polinómica", "sigmoide".

1.2.6. K-vecinos más cercanos

También se ha considerado el método de los k-vecinos más cercanos [18]. Éste es un modelo que sirve para clasificar valores buscando los puntos de datos "más similares" (por cercanía) aprendidos en la etapa de entrenamiento y haciendo conjeturas de nuevos puntos basado en esa clasificación. A su vez, este modelo tiene como una de sus complejidades la elección del número de k de vecinos usados para clasificar las nuevas observaciones. Una posible opción es considerar $\sqrt{n}$, donde n denota el número de entradas del conjunto de entrenamiento.

Por su parte, en cuanto a su funcionamiento, el algoritmo se basa en dos fases: una de entrenamiento y otra de clasificación. La fase de entrenamiento consiste en almacenar los vectores característicos y las etiquetas de las clases de los ejemplos de entrenamiento.

En la fase de clasificación, la evaluación del ejemplo (del que no se conoce su clase) es representada por un vector en el espacio característico. Se calcula la distancia (previamente elegida) entre los vectores almacenados y el nuevo vector, y se seleccionan los k ejemplos más cercanos. Finalmente, el nuevo ejemplo es clasificado con la clase que más se repite en los vectores seleccionados.

Un ejemplo ilustrativo de este algoritmo es el siguiente:

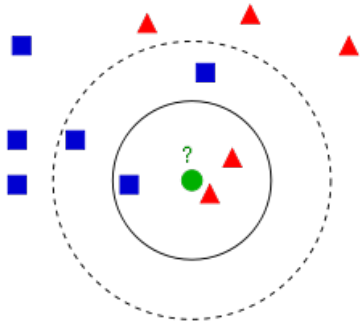

Figura 1.3: Ejemplo de K-vecinos más cercanos. Fuente: analyticsvidhya.com

En este ejemplo se puede apreciar como la clasificación del nuevo punto en estudio se ve influenciado por el número de vecinos considerados. Así, para k=3 se corresponde con una de las clases y, con k=5, se categoriza con "la contraria".

Capítulo 2

Variables de interés en un modelo predictivo en ajedrez

En este capítulo se realiza una breve introducción a los principales elementos del ajedrez y se presentan que las variables que hemos utilizado en nuestro modelo predictivo.

2.1. Ajedrez de alto nivel

2.1.1. Breve introducción al ajedrez

El ajedrez [19] es un juego que se desarrolla sobre un tablero y en el que participan dos personas. Cada uno de los jugadores cuenta con dieciséis piezas del mismo color (negras, o en contraposición, blancas) que puede desplazar sobre el tablero que está dividido en sesenta y cuatro casillas.

Figura 2.1: Tablero de ajedrez. Fuente: torre64.com

Las piezas disponibles y las reglas que siguen a la hora de ser movidas son las siguientes:

- **Peón**: Cada jugador cuenta con 8 de estas piezas. Cabe destacar que no existe una pieza más débil y numerosa que el peón y su movimiento más frecuente es hacia el frente, de a una casilla. No obstante, al inicar la partida, esta pieza puede dar dos pasos a la vez. Una de sus particularidades es que para capturar otras piezas debe hacerlo en diagonal.

Figura 2.2: Peón de ajedrez. Fuente: Wikipedia

- **Alfil**: Cada jugador dispone de dos alfiles al comienzo de la partida. Esta se desplaza en diagonal y, siempre y cuando encuentre espacios vacíos, puede seguir avanzando.

Figura 2.3: Alfil de ajedrez. Fuente: Wikipedia

- **Torre**: Cada jugador dispone de dos de estas figuras al comienzo de la partida y se podría decir que la torre es como el alfil pero en línea recta horizontal o vertical. No obstante, a diferencia del alfil, esta pieza puede colocarse en cualquier lugar del tablero.

Figura 2.4: Torre de ajedrez. Fuente: Wikipedia

- **Caballo**: Al igual que las dos piezas anteriores, cada jugador también dispone de dos de estas figuras al inicio de la partida. Esta es la única pieza capaz de saltar por encima de otras y su movimiento consiste en desplazarse en forma de L (dos casillas en línea recta horizontal o vertical, y luego avanzar una más formando un ángulo recto).

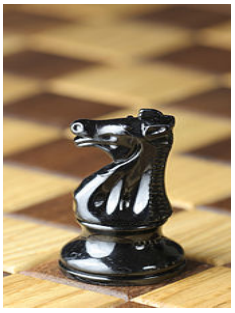

Figura 2.5: Caballo de ajedrez. Fuente: Wikipedia

- **Dama**: Cada jugador dispone de una y, es la pieza más poderosa, debido a que combina los movimientos del alfil y la torre.

Figura 2.6: Dama de ajedrez. Fuente: Wikipedia

- **Rey**: Cada jugador dispone de uno solo y es la pieza más importante. No obstante, sus movimientos se limitan a cualquiera de sus casillas adyacentes, por lo cual no puede avanzar más de un paso a la vez.

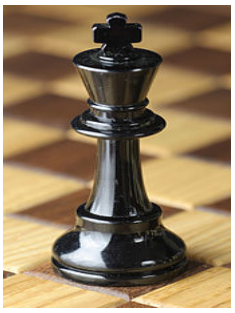

Figura 2.7: Rey de ajedrez. Fuente: Wikipedia

Una vez conocidas las piezas que conforman el ajedrez, cabe explicar el funcionamiento de una partida en sí. Así, una característica importante del ajedrez es que si un jugador logra que una de sus piezas llegue a la casilla en la que hay una pieza de su oponente, puede capturarla. De este modo, el rival pierde una de sus piezas. A su vez, el juego termina cuando un jugador consigue que el rey rival no pueda evitar una maniobra de captura: dicha jugada se conoce como **jaque mate**.

2.1.2. Federación Internacional de Ajedrez (FIDE) y ratings

La Federación Internacional de Ajedrez, FIDE [2], es una organización internacional que une las diversas federaciones nacionales de ajedrez. Fue fundada en París, Francia el 20 de julio de 1924 y su actual presidente desde 2018 es Arkady Dvorkovich.

(a) Logo FIDE. Fuente: FIDE.com

(b) Presidente actual FIDE. Fuente: Twitter

En la actualidad, la FIDE cuenta con 195 federaciones nacionales inscritas, siendo 176 de ellas pertenecientes a países de la ONU. Entre sus funciones se encuentra organizar el campeonato del mundo de ajedrez, redactar las reglas del ajedrez, publicar libros, nombrar a maestros internacionales, grandes maestros y árbitros y, calcular los **ratings** de los jugadores para cada uno de los tipos de partidas. Para esto último, la FIDE usa el sistema Elo, el cual fue inventado por Arpad Elo.

La puntuación Elo de un jugador se determina a partir de sus resultados frente a otros jugadores. De esta manera, y haciendo uso de la diferencia de la puntuación Elo entre dos jugadores, se determina lo de denominado en el argot como "puntuación esperada". Esta, para cada uno de los jugadores es la probabilidad de ganar más la mitad de la probabilidad de empate. Como ejemplo considérese a un jugador A, cuya puntuación Elo es R_A, y a un jugador B, cuya puntuación Elo es R_B. Usando una función logística, se deduce que la fórmula exacta de la puntuación esperada del jugador A es

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}.$$

De igual manera, la puntuación esperada para el jugador B es

$$E_B = \frac{1}{1 + 10^{(R_A - R_B)/400}}.$$

Como ejemplo consideremos a Magnus Carlsen y a Wesley So. A día 9 de mayo de 2022 el primero tiene un rating de 2864, por 2776 del segundo. Por ello, la puntuación esperada para Carlsen va a ser:

$$E_{Carlsen} = \frac{1}{1+10^{(2776-2864)/400}} = 0'624.$$

A su vez, la de So será:

$$E_{So} = \frac{1}{1+10^{(2864-2776)/400}} = 0'376.$$

Por ello, es favorito Carlsen.

Por otra parte, atendiendo a si un jugador determinado obtuvo una puntuación superior o inferior a la esperada, el sistema Elo incrementará o disminuirá su puntuación Elo. Así, y atendiendo al máximo ajuste posible por juego, llamado "factor K" (desde 10, para

quien haya superado en alguna ocasión la barrera de los 2400 puntos, 40 para los novatos y 20 para el resto), se tiene que si la puntuación esperada del jugador A es E_A y su puntuación al final fue S_A, la nueva puntuación Elo vendrá dada por la siguiente fórmula: $R'_A = R_A + K(S_A - E_A)$.

2.1.3. Auge del ajedrez

Cabe destacar que aunque el ajedrez es un deporte que ha sido jugado desde hace siglos, la llegada del Internet ha provocado un "boom" en los últimos años. Así, webs como "chess24.com", "chess.com" o "chessbase.com", han motivado un mayor seguimiento de este deporte.

Asimismo, a finales del 2020 se estrenó una serie en Netflix que provocó un incremento en el número de aficionados al ajedrez. Esta es "Gambito de dama" [20], cuyo título hace honor a una conocida apertura de partida en ajedrez. Dicha miniserie está protagonizada por Anya Taylor-Joy y recibió numerosos galardones, tales como dos Globos de Oro y, multitud de alabanzas por parte de la comunidad internacional de ajedrez.

Figura 2.9: Gambito de dama: Serie de Netflix. Fuente: Netflix

Por otra parte, en lo respectivo a España, entre junio y julio de este año se jugará en Madrid el próximo torneo de candidatos que decidirá al próximo retador de Carlsen, lo cual probablemente sirva para promocionar el ajedrez en nuestro país e incrementar la afición por este deporte.

2.2. Variables de interés

A la hora de elaborar un modelo predictivo de la probabilidad de los distintos resultados de una partida, en primer lugar, se tiene que decidir qué variables serán aquéllas en las que basemos nuestra predicción. A continuación se enumeran aquellas que han sido consideradas de interés en este trabajo.

- **Rating1** y **Rating2**: Estas variables denotan el rating del jugador de blancas y de negras, respectivamente, en el momento de la partida. Parece bastante claro que estas variables tendrán importancia a la hora de elaborar una predicción y que, cuanto mayor sea el rating de un jugador, mejor se desempeñará en el desarrollo de la partida. Además, para la recolección de estas variables se crearon otras que indicaba el nombre del jugador de blancas y de negras. No obstante, estas variables como tales no fueron usadas como variables independiente en el modelo.

  Como ejemplo, a lo largo de esta sección se va a considerar la última partida de la que se recogieron datos. En ella el jugador que jugaba con blancas era Alireza

Firouzja y su rating en ese momento era 2770. En esa partida se enfrentó a Shakhriyar Mamedyarov, cuyo rating en ese momento era de 2765 puntos.

Jugador blancas	Jugador negras	Rating1	Rating2
Firouzja, Alireza	Mamedyarov, Shakhriyar	2770	2765

- **Tipo partida**: En el ajedrez existen tres tipos de partidas: Blitz o relámpago, Rápida y Normal. Estas vienen determinadas por el tiempo disponible para afrontar la partida. Así, las partidas tipo Blitz o relámpago son las más rápidas de todas, ya que se juegan a un ritmo inferior a 10 minutos por jugador. Por su parte, las partidas rápidas son aquellas en las que todas las jugadas han de efectuarse en un tiempo fijo de entre 10 y 60 minutos para cada jugador. Por último, las partidas a ritmo normal son aquellas en las que cada jugador tiene al menos 60 minutos para la partida.

  Parece claro que se requieren distintas destrezas para destacar en una partida o en otra. Así, en una partida rápida o relámpago, la agilidad mental para realizar movimientos jugará un papel más fundamental que en una partida normal en la que no afecta tanto el tiempo. Por ello, parece intuitivo que esta variable jugará un papel fundamental a la hora de hacer la predicción.

  Como ejemplo de que un jugador puede ser mejor en unos tipos de partidas que en otros está Nakamura. Este actualmente es el número 1 del mundo a Blitz y Relámpago pero a ritmo normal solo es el número 10. Asimismo, el actual campeón del mundo de rápidas es Nodirbek Abdusattorov, que se encuentra en el puesto número 87 del ranking.

  En el ejemplo que estamos tratando en esta sección, la partida es del tipo "Normal":

Tipo partida
Normal

- **Tipo torneo**: Otra variable tenida en cuenta fue el tipo de torneo. Esta toma tres posibles valores: Individual, Equipos y Selecciones. Esta fue considerada debido a que en un torneo por equipos o selecciones el resultado individual del jugador puede afectar al resultado global del equipo. Por esta razón, a veces un jugador puede tender a ser conservador a pesar de tener ventaja al jugar con un rival de nivel inferior, o viceversa: a veces, a pesar de tener un rival fuerte, el jugador tiene que arriesgar porque su equipo lo necesita.

  A su vez, para recoger esta variable se tuvo que crear otra sobre el nombre del torneo. No obstante, esta no fue tenida en cuenta en el análisis.

  Por su parte, en nuestro ejemplo, el torneo es del tipo "Selecciones", en concreto, se trata del torneo "Europeo de selecciones":

Tipo torneo
Selecciones

- **Importancia torneo**: Otra variable que en principio parece determinante en el modelo es la importancia del torneo. Esto es debido a que parece sensato pensar que no es lo mismo para un jugador disputar un Mundial que un amistoso: distinta

intensidad, presión, etc. Por ello, se recogió esta variable que calificaba la importancia del torneo en distintos tipos:

- Tipo 1: campeonato del mundo y torneo de candidatos.
- Tipo 2: clasificatorios al torneo de candidatos (Chess World Cup, Grand Prix, FIDE Swiss).
- Tipo 3: cerrados entre jugadores TOP, Olimpiadas (por equipos), europeo (individual o por equipos) y mundial por equipos.
- Resto: resto de torneos.

En nuestro caso particular, la partida entre Alireza Firouzja y Shakhriyar Mamedyarov tuvo lugar en el torneo europeo por equipos de Terme-Olimia, catalogado como Tipo 3.

- **WhiteMaxRating** y **BlackMaxRating**: Estas variables consisten en los máximos ratings del jugador de blancas y el de negras hasta el momento de la partida. Dichas variables pueden ser importantes, ya que indican el "peak" máximo al que llegó el jugador y de lo que fue capaz en su momento.

  En el ejemplo, el jugador de blancas era Alireza Firouzja y su rating máximo en ese momento era 2770, el cual coincidía con su rating actual (**Rating1**). Por su parte, el jugador de negras era Shakhriyar Mamedyarov y su rating máximo en ese momento era 2820, que es superior a **Rating2**, lo que indica que el rating actual de este jugador es inferior en la actualidad que su máximo rating.

Jugador blancas	**Jugador negras**	**WhiteMaxRating**	**BlackMaxRating**
Firouzja, Alireza	Mamedyarov, Shakhriyar	2770	2820

- **WhiteRating2018**, **WhiteRating2019**, **WhiteRating2020**, **BlackRating2018**, **BlackRating2019** y **BlackRating2020** : Estas variables denotan el rating con el que el jugador que disputa la partida con blancas (y el de negras, respectivamente) acabó el año (2018, 2019 y 2020). Así, estas variables pueden resultar importantes en el estudio, ya que permiten vislumbrar si el jugador viene en una dinámica creciente o no.

  En el ejemplo anterior, tenemos los siguientes ratings para esos años:

Jugador blancas	**WhiteRating2018**	**WhiteRating2019**	**WhiteRating2020**
Firouzja, Alireza	2549	2618	2723
Jugador negras	**BlackRating2018**	**BlackRating2019**	**BlackRating2020**
Mamedyarov, Shakhriyar	2804	2817	2770

  Así, se puede apreciar que Firouzja va incrementando su rating con los años, lo que puede venir explicado por la juventud de este jugador. Por su parte, Mamedyarov incrementó su rating de 2018 a 2019, pero posteriormente se vio reducido en 2020.

- **WhiteRatingChange** y **BlackRatingChange**: Estas variables miden el cambio de rating del jugador de blancas y el de negras que se produjo entre el 31 de diciembre del año anterior de la partida y la fecha de la partida. Para ello, se necesitan varias variables: la fecha de la partida, el rating de cada jugador en el momento de la partida y el rating de cada jugador a 31 de diciembre del año anterior.

Para la primera, se recogió la fecha y se anotó en una variable, la cual no fue posteriormente usada en el modelo. A su vez, los ratings actuales vienen dados en las variables **Rating1** y **Rating2**. Por su parte, para conocer los ratings a 31 de diciembre del año anterior se hizo uso de una de las variables para cada jugador del ítem anterior.

En lo respectivo a las partidas para las que no han sido recogidas los ratings de un año antes, se hizo uso del primer valor registrado.

Como ejemplo, la partida que estamos tratando tuvo lugar el 21-11-2021. Por ello, teniendo en cuenta que el rating de Firouzja en el momento de la partida es 2770, y que su rating a 31 de diciembre de 2020 fue 2723, se tiene que WhiteRatingChange es igual a 47. A su vez, el rating de Mamedyarov en el momento de la partida es 2765, y su rating a 31 de diciembre de 2020 fue 2770, por lo que se tiene que BlackRatingChange es igual a -5:

Jugador blancas	**Fecha**	**Rating1**	**WhiteRating2020**	**WhiteRatingChange**
Firouzja, Alireza	21-11-2021	2770	2723	47
Jugador negras	**Fecha**	**Rating2**	**BlackRating2020**	**BlackRatingChange**
Mamedyarov, Shakhriyar	21-11-2021	2765	2770	-5

Por ello, se puede deducir que el jugador de blancas viene en dinámica positiva y el de negras en negativa, lo que puede ser un factor decisivo en la partida.

- **EdadWP** y **EdadBP**: Estas variables denotan la edad del jugador de blancas y el de negras en el momento de la partida. Estas parecen ser variables importantes en el modelo, ya que, a priori, aunque los jugadores más jóvenes tengan menos experiencia que uno veterano, también llegan con una dinámica buena y mejoran muy rápido, en cambio la gente de más edad suele haber llegado a su máximo y su tendencia ya no será tan positiva. Todo esto puede jugar un papel fundamental en el desarrollo de la partida.

  Por su parte, para su cálculo se usó la variable fecha de la partida y se crearon otras con la fecha de nacimiento del jugador de blancas y el de negras. No obstante, ninguna de estas fueron usadas en el modelo, sino que solo se crearon para construir otras a partir de ellas.

  En nuestro ejemplo, la partida tuvo lugar el 21-11-2021 y, Alireza Firouzja nació el 18-06-2003. Por ello, en el momento de la partida contaba con 18 años. A su vez, el jugador de negras, Shakhriyar Mamedyarov, nació el 12-04-1985, por lo que en el momento de la partida contaba con 36 años.

Jugador blancas	**Fecha**	**Fecha nacimiento**	**EdadWP**
Firouzja, Alireza	21-11-2021	18-06-2003	18
Jugador negras	**Fecha**	**Fecha nacimiento**	**EdadBP**
Mamedyarov, Shakhriyar	21-11-2021	12-04-1985	36

- **H2H**: Esta variable denota el *Head to head*, es decir, mide los resultados de los enfrentamientos individuales de los dos jugadores hasta el momento de la partida. En nuestro caso particular se va a calcular el H2H relativo al jugador de blancas (el de negras sería el complementario). Para ello, se multiplica por 1 el número de victorias de blancas y, a esto se le suma el número de empates multiplicado por 0.5. Una vez obtenida ess suma, se divide el resultado entre el número total de partidas.

Así, en nuestro ejemplo, Alireza Firouzja se enfrentó a Shakhriyar Mamedyarov en un total de 14 ocasiones con 3 victorias a favor de Firouzja, 5 a favor de Mamedyarov y 6 empates. Por ello, **H2H**= $\frac{3+6\cdot 0'5}{14} \approx 0'42$.

Otras variables que fueron calculadas fueron aquellas relacionadas con el estado de forma de los jugadores involucrados en la partida. Para ello, se consideraron dos formas de calcularlo:

- **EF1W** y **EF1B**: Estas variables denotan el estado de forma 1 del jugador de blancas y el de negras, respectivamente. Esto es la *performance* de las últimas 10 partidas.

  La *performance* mide el rendimiento del jugador en términos de sus resultados y el nivel de los jugadores a los que se ha enfrentado. Así, cuánto mayor es la *performance*, mayor es el rendimiento del jugador. Por su parte, se calcula utilizando esta fórmula:

  $$Performance = \frac{\text{(suma rating de los oponentes)}+400\cdot\text{(victorias-derrotas)}}{\text{número de partidas}}.$$

  Así, en nuestro ejemplo tenemos que, atendiendo a la fórmula anterior, el estado de forma 1 del jugador de blancas es igual a 2818'5 y, el de negras es 2790. Por ello, Firouzja llega mejor a la partida.

Jugador blancas	**Jugador negras**	**EF1W**	**EF1B**
Firouzja, Alireza	Mamedyarov, Shakhriyar	2818'5	2790

- **EF2W** y **EF2B**: Estas variables denotan otra forma de medir el estado de forma del jugador de blancas y el de negras, sin tener en cuenta el rating de sus últimos oponentes. Para su cálculo, se recopilan las 10 últimas partidas del jugador de blancas (y de negras, respectivamente) y se le suma al número de victorias del jugador de blancas (y de negras) el número de empates multiplicado por 0'5.

  De esta manera, en nuestro ejemplo, las últimas 10 partidas de Firouzja fueron 4 victorias, 5 empates y 1 derrota. Por ello, EF2W=$4 + 5 \cdot 0'5$=6'5.

  A su vez, las últimas 10 partidas de Mamedyarov fueron 2 victorias, 8 empates y 0 derrotas. Por ello, EF2B=$2 + 8 \cdot 0'5$=6.

  Por ello, y al igual que lo obtenido con el otro estado de forma considerado, Firouzja llega mejor a la partida:

Jugador blancas	**Jugador negras**	**EF2W**	**EF2B**
Firouzja, Alireza	Mamedyarov, Shakhriyar	6'5	6

Por otra parte, también se consideraron diferentes variables que tenían en cuenta el estilo del jugador. Estas fueron las siguientes:

- **LuchaWP** y **LuchaBP**: Estas variables denotan el nivel de lucha del jugador de blancas y el de negras, respectivamente. Esto sería el número medio de jugadas en las partidas de cada jugador hasta el momento de la partida.

  Por ejemplo, en la partida que estamos mostrando, Alireza Firouzja tiene una lucha de 47'39. Es decir, promedia aproximadamente 47 jugadas por partida hasta el momento. A su vez, el jugador de negras, Shakhriyar Mamedyarov tiene una lucha de 43'06. Es decir, promedia aproximadamente 43 jugadas por partida. Por tanto, esto

significa que Firouzja lucha más sus partidas que Mamedyarov, ya que tiene mayor número medio de jugadas.

Jugador blancas	**Jugador negras**	**LuchaWP**	**LuchaBP**
Firouzja, Alireza	Mamedyarov, Shakhriyar	47'39	43'06

- **AgresividadWP** y **AgresividadBP**: Otra variable considerada como importante para el modelo era la agresividad del jugador. Esta es complicada de medir, pero sí que parece creíble que un jugador que sea agresivo en las partidas tenga, a priori, más probabilidad de ganar o de perder que de empatar. Por ello, para su cálculo, se definió está variable como el porcentaje de partidas del jugador de blancas (y de negras, respectivamente) decididas, es decir, el porcentaje de partidas que no acabaron en tablas.

  En nuestro caso, el porcentaje de partidas decididas del jugador de blancas, Alireza Firouzja, es igual a 74'46 % y, para Shakhriyar Mamedyarov (jugador de negras), su porcentaje de partidas decididas es 54'46 %, por lo que es más conservador que su rival.

Jugador blancas	**Jugador negras**	**AgresividadWP**	**AgresividadBP**
Firouzja, Alireza	Mamedyarov, Shakhriyar	74'46 %	54'46 %

Así, de acuerdo a las 4 variables anteriores, Firoujza es más agresivo y más luchador que Mamedyarov.

Presentamos por último la variable a predecir:

- **Resultado**: Esta variable toma tres posibles valores: victoria blancas (1-0), empate (1/2-1/2) y derrota blancas (0-1).

  En el ejemplo expuesto, ganó Alireza Firouzja, por lo que ganó el jugador de blancas.

Capítulo 3

Base de datos

Se presenta en este capítulo la base de datos con la que trabajaremos y y se realiza un análisis descriptivo de la misma.

3.1. Extracción de los datos y preprocesamiento

3.1.1. La web 2700chess

Para el desarrollo del trabajo se hizo uso de una página muy conocida entre los amantes del ajedrez. Ésta es *2700chess* [21] y en ella se puede acceder a datos de diferentes partidas, jugadores, etc. Por otra parte, en esta página se hacen actualizaciones en vivo diarias de las calificaciones de ajedrez de los mejores jugadores. Por lo general, estas clasificaciones se actualizan 1 minuto después de que finaliza un juego, si se juega en un torneo superior. Las clasificaciones en vivo se basan en clasificaciones oficiales (clasificaciones FIDE) que se actualizan una vez al mes.

Si nos adentramos en la página, a fecha 10 de abril de 2022, podemos observar lo siguiente:

Figura 3.1: Pantalla principal 2700chess. Fecha: 10-4-2022. Fuente: 2700chess.com

En primera plana la página muestra los ratings de los 35 mejores jugadores en la actualidad en partidas clásicas. Para acceder a estos pero en diferentes tipos de partidas,

basta con hacer un click en “Rapid” y en “Blitz”, mientras que, por su parte, para acceder al Top 100 de la actualidad, basta con pulsar en “Top100”.

En cuanto a la información de cada jugador que se recoge en esta pantalla dentro de la página, se puede observar como cada jugador tiene a la derecha de su nombre sus ratings actuales en partidas clásicas (y su variación), Blitz y Rápidas, además de su edad. Así, por ejemplo, a día 11 de abril de 2022, Magnus Carlsen ocupa la primera posición en el top, teniendo un rating de 2864 en partidas clásicas (sin variación), 2847 en partidas rápidas, 2832 en partidas Blitz y, con una edad de 31 años.

Figura 3.2: Magnus Carlsen: actual número 1 del mundo. Fuente: eleconomista.es

A su vez, a la derecha de la edad de cada jugador aparece un tablero de ajedrez. Si pinchamos en esa figura, nos encontramos con un listado de las partidas de ese jugador en cuestión. Así, para Magnus Carlsen obtenemos lo siguiente:

#	White Player	Rating	Black Player	Rating	Result	Moves	ECO	Site	Year
1	Carlsen, Magnus	2855	Ostmoe, Geir Sune Tallaksen	2469	1/2-1/2	84	B10	Oslo	2022
2	Caruana, Fabiano	2792	Carlsen, Magnus	2865	0-1	49	B31	Wijk aan Zee	2022
3	Carlsen, Magnus	2865	Vidit, Santosh Gujrathi	2727	1/2-1/2	51	C24	Wijk aan Zee	2022
4	Karjakin, Sergey	2743	Carlsen, Magnus	2865	1/2-1/2	16	C67	Wijk aan Zee	2022
5	Carlsen, Magnus	2865	Mamedyarov, Shakhriyar	2767	1-0	27	E04	Wijk aan Zee	2022
6	Shankland, Samuel L	2708	Carlsen, Magnus	2865	1/2-1/2	35	D32	Wijk aan Zee	2022
7	Praggnanandhaa R	2612	Carlsen, Magnus	2865	0-1	34	D30	Wijk aan Zee	2022
8	Carlsen, Magnus	2865	Rapport, Richard	2763	1-0	31	E04	Wijk aan Zee	2022
9	Grandelius, Nils	2672	Carlsen, Magnus	2865	1/2-1/2	54	B33	Wijk aan Zee	2022
10	Carlsen, Magnus	2865	Van Foreest, Jorden	2702	1/2-1/2	52	A07	Wijk aan Zee	2022
11	Duda, Jan Krzysztof	2760	Carlsen, Magnus	2865	1/2-1/2	38	C88	Wijk aan Zee	2022
12	Carlsen, Magnus	2865	Giri, Anish	2772	1-0	36	E10	Wijk aan Zee	2022
13	Esipenko, Andrey	2714	Carlsen, Magnus	2865	1/2-1/2	21	D45	Wijk aan Zee	2022
14	Vachier-Lagrave, Maxime	2787	Carlsen, Magnus	2892	1-0	47	C88	Warsaw	2021
15	Carlsen, Magnus	2892	Nepomniachtchi, Ian	2792	1-0	43	A07	Warsaw	2021
16	Amin, Bassem	2617	Carlsen, Magnus	2892	0-1	58	B30	Warsaw	2021
17	Alekseenko, Kirill	2663	Carlsen, Magnus	2892	1/2-1/2	72	C50	Warsaw	2021
18	Carlsen, Magnus	2892	Mastrovasilis, Dimitrios	2577	1-0	23	B46	Warsaw	2021
19	Grischuk, Alexander	2757	Carlsen, Magnus	2892	1-0	58	E04	Warsaw	2021
20	Carlsen, Magnus	2892	Giri, Anish	2778	0-1	43	C65	Warsaw	2021

Figura 3.3: Últimas partidas de Magnus Carlsen. Fuente: 2700chess.com

De esta manera, para cada partida se encuentra recogida información muy importante para nuestro trabajo: jugador de blancas y su rating en el momento de la partida, jugador de negras y su rating en el momento de la partida, resultado, número de movimientos, lugar y año. Además, si pulsamos en el tablero de ajedrez situado a la derecha del año, accedemos a más información relacionada con la partida. Así, si pulsamos en la partida de Magnus Carlsen frente a Geir Sune Tallaksen Ostmoe, obtenemos lo siguiente:

Figura 3.4: Carlsen vs Ostmoe. Fuente: 2700chess.com

Aquí se pueden observar nuevas variables, tales como la edad de los jugadores en el momento de la partida y la fecha exacta de ésta. Por otra parte, si pulsamos en "Head to Head", se nos redirigirá a una página en la que se nos indica el resultado de las partidas entre ambos jugadores:

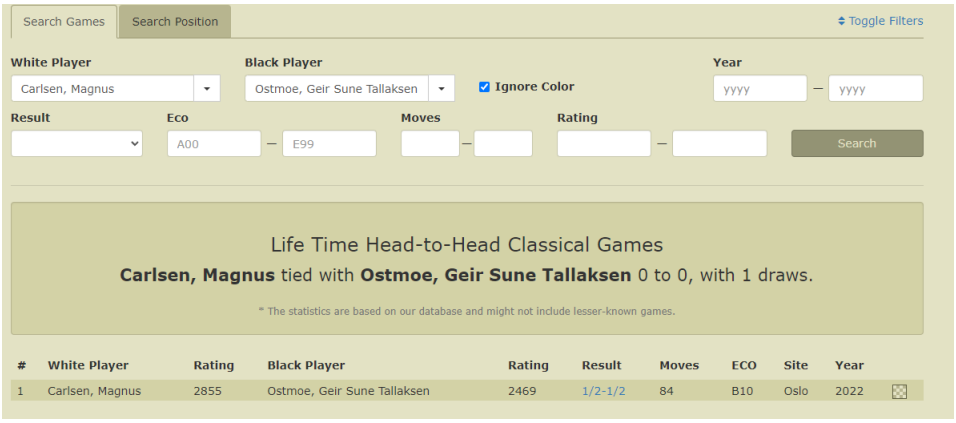

Figura 3.5: Carlsen vs Ostmoe: Head to Head. Fuente: 2700chess.com

En este caso, solo se disputó una partida entre estos jugadores, siendo un empate el resultado.

En cuanto a la información de cada jugador, basta con buscar el nombre del ajedrecista del que se desea obtener información. Por ejemplo, si buscamos a Magnus Carlsen se obtiene lo siguiente:

Figura 3.6: Información sobre Magnus Carlsen. Fuente: 2700chess.com

Así, de aquí se pueden obtener distintas variables, tales como el máximo rating del jugador (2882 para Carlsen) o su fecha de nacimiento (30 de noviembre de 1990 en el caso de Carlsen). Por último, en esta misma página se encuentra recogido en un diagrama el rating del jugador a lo largo del tiempo.

Figura 3.7: Evolución de Magnus Carlsen. Fuente: 2700chess.com

De esta manera, se puede obtener el rating del jugador a finales del año 2018, 2019 y 2020, siendo todas ellas variables para nuestro estudio.

3.1.2. Lectura de datos con R

Una vez introducida la página de la que se deciden extraer los datos, el siguiente paso consiste en su lectura a través de *R* [22]. Para ello, se hizo uso de una técnica conocida como *web scraping* [23], la cual sirve para extraer información de páginas web de modo automatizado.

Entre los beneficios de esta técnica se encuentra su sencillez para manejar amplias cantidades de datos de forma muy rápida. Así, para este trabajo se utilizó la librería *rvest* [24] de *R*, la cual permite realizar esta función.

3.1.3. Creación de variables

El primer paso para la creación de las variables fue detectar los nombres de los 100 primeros jugadores en el ranking. Una vez hecho esto, se creó una función que tenía como

valor de entrada los nombres de los jugadores del top 100 y, que daba como salida una lista de dataframes. Cada uno de ellos recogía, para el jugador correspondiente, un listado de todas las partidas que disputó entre 2018 y el momento de la recogida de datos. Además, acompañado de esto también aparecía el tipo de partida, la fecha y el torneo, siendo todas estas variables importantes para el estudio posterior.

A continuación, se unió esa lista de dataframes en uno solo, considerando solo aquellas partidas disputadas por integrantes del Top 100, eliminando las partidas duplicadas y corrigiendo errores en la nomenclatura de algún jugador u otros tipos de características que no fueron bien recogidas mediante el *web scraping*.

Una vez conseguida esta base de datos, se procedió a añadir las variables restantes. Para ello, en primer lugar se añadieron las fechas de nacimiento y los máximos ratings de los jugadores tanto de blancas como de negras. Esto, se realizó mediante varios pasos:

1º) Se hizo uso de la variable que recogía todos los nombres de los 100 mejores jugadores.

2º) Para cada uno de ellos se leyó su respectiva página buscando en ella su fecha de nacimiento y su rating máximo. Este paso fue hecho mediante un bucle.

Posteriormente, se obtuvieron las variables de rating en 2018, 2019 y 2020. Estas fueron las más difíciles de conseguir, debido a que no se disponía del rating a fecha 1 de enero del año en cuestión para muchos jugadores. Por ello, se tomó como referencia el último rating del año anterior.

Por último, se crearon las variables estado de forma, agresividad, lucha y "Head to head", haciendo uso de sus respectivas fórmulas y de la base de datos creada hasta ese momento.

Todo el código de esta parte se encuentra recogido en el apéndice A.

3.1.4. Base de datos final

Una vez realizado todo el proceso de la lectura de los datos y la creación de las variables, se obtuvo la base de datos final. En total, se almacenaron 6457 partidas. Una muestra de las 6 primeras partidas de la base de datos es la siguiente:

```
      White Player Rating1          Black Player Rating2  Result Year      fecha          torneo tipo_partida WhiteMaxRating
1   Firouzja, Alireza    2770  Mamedyarov, Shakhriyar    2765     1-0 2021 2021-11-21 r9-terme-olimia       Normal           2770
2      Aronian, Levon    2767 Vidit, Santosh Gujrathi    2739 1/2-1/2 2021 2021-11-21     r17-kolkata        Blitz           2871
3 Maghsoodloo, Parham    2601         Aronian, Levon    2767 1/2-1/2 2021 2021-11-21     r13-kolkata        Blitz           2706
4      Aronian, Levon    2767     Shankland, Samuel L    2660     0-1 2021 2021-11-21     r12-kolkata        Blitz           2871
5      Aronian, Levon    2767          Le, Quang Liem    2774     1-0 2021 2021-11-21     r10-kolkata        Blitz           2871
6 Grischuk, Alexander    2773          Adams, Michael    2714 1/2-1/2 2021 2021-11-21 r9-terme-olimia       Normal           2810
  BlackMaxRating WhiteRating2018 BlackRating2018 WhiteRating2019 BlackRating2019 WhiteRating2020 BlackRating2020 BlackRatingChange
1           2820            2549            2804            2618            2817            2723            2770               -5
2           2727            2795            2718            2695            2695            2784            2721               18
3           2871            2570            2795            2679            2695            2674            2784              -17
4           2731            2795            2668            2695            2725            2784            2683              -23
5           2758            2795            2732            2695            2699            2784            2683               91
6           2761            2767            2709            2771            2701            2777            2694               20
  WhiteRatingChange      H2H  LuchaWP AgresividadWP  LuchaBP AgresividadBP Edad_WP Edad_BP Importancia_torneo Tipo_torneo   EF1W   EF1B EF2W
1               47 0.4285714 47.39703     0.7446458 43.05814     0.5445736      18      36             Tipo3 Selecciones 2818.5 2790.0  6.5
2              -17 0.5277778 46.21191     0.4986150 49.40411     0.5102740      30      27             Resto  Individual 2707.9 2476.1  5.5
3              -73 0.4166667 46.64747     0.7845084 46.21191     0.4986150      21      30             Resto  Individual 2882.8 2752.5  7.0
4              -17 0.6250000 46.21191     0.4986150 47.10788     0.5269710      30      30             Resto  Individual 2803.0 2711.0  6.5
5              -17 0.6250000 46.21191     0.4986150 50.62849     0.5558659      30      32             Resto  Individual 2722.1 2629.1  5.5
6               -4 0.5714286 45.18256     0.4929006 43.12143     0.5214286      38      49             Tipo3 Selecciones 2711.2 2666.8  5.0
  EF2B
1  6.0
2  2.5
3  6.0
4  5.0
5  4.5
6  5.5
```

Figura 3.8: Muestra de las primeras 6 filas de la base de datos

Pasamos a continuación a realizar un breve análisis estadístico del conjunto de datos. Para ello, la información más relevante es la siguiente:

- **Rating1** y **Rating2**: Estas dos variables denotan el rating al comienzo de la partida del jugador de blancas y de negras, respectivamente.

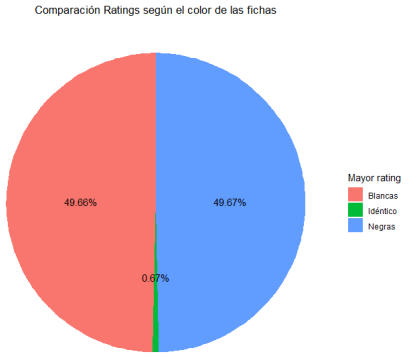

Figura 3.9: Comparación del rating del jugador según el color de las fichas

Por tanto, se puede observar que la proporción de veces en la que el jugador de blancas tiene mejor ranking que el de negras es similar al caso contario.

- **Edad_WP** y **Edad_BP**: Estas variables recogen la edad del jugador en el momento de la partida. Cabe destacar que los jugadores más jóvenes en aparecer en la base de datos son Vincent Keymer y Sarin Nihal, con 14 de años de edad. Por su parte, el jugador más longevo es Boris Gelfand, con 53 años.
- **Tipo_partida**, **Tipo_torneo** e **Importancia_Trorneo**: La primera variable recoge el tipo de partida, es decir, describe si la partida era "Normal", "Blitz" o "Rápida". Así, en este caso se aprecian diferencias en la cantidad de unas y de otras:

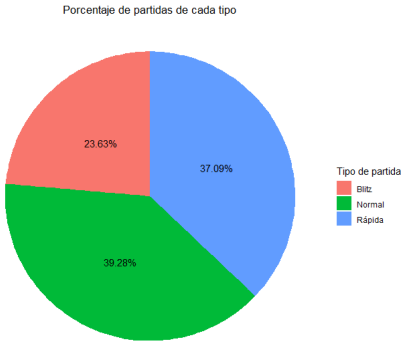

Figura 3.10: Proporción de partidas según el tipo

Por su parte, la segunda denota el tipo de torneo en el que se disputaba esa partida. En este caso hay muchas diferencias entre un tipo y otro, ya que la inmensa mayoría de las partidas son individuales:

Equipos	**Individual**	**Selecciones**
4'3 %	93'7 %	2 %

Por otra parte, la última variable denota la importancia del torneo: Tipo1, Tipo2, Tipo3 y Resto, habiendo la siguiente cantidad de partidas de cada una:

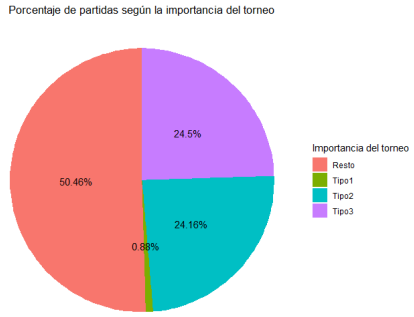

Figura 3.11: Proporción de partidas según la importancia del torneo

Tal y como se puede observar, del tipo 1 es de la que menos partidas se disponen, con apenas un 0'88 %. Por su parte, de la que más se tienen es de "Resto", con más de un 50 %.

- **Result**: Esta variable denota el resultado de la partida. Tiene 3 niveles: "1-0", "1/2-1/2", "0-1":

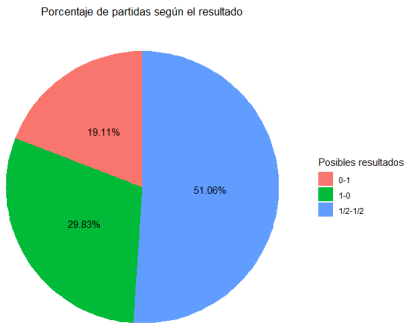

Figura 3.12: Proporción de partidas según el resultado

Tal y como se puede observar, la mayor parte de las partidas acabaron en empate. Además, estas categorías no se encuentran balanceadas, lo que puede ser una desventaja a la hora de hacer predicciones.

Una vez explicadas algunas de las variables más importantes por separado, se ha realizado un análisis bivariante con respecto a la variable a predecir, **Result**. Para ello, en primer lugar, se pretende analizar la variable **Result** en función de las variables cualitativas. Para ello, se realizaron diferentes tablas de contingencia del resultado en función

del tipo de partida, tipo de torneo e importancia de torneo. Además se realizó el test de independencia chi cuadrado de Pearson, el cual tiene como hipótesis nula la siguiente:

H_0= Las variables son independientes.

		Resultado				
		1-0	1/2-1/2	0-1	Total	p-valor
Tipo de partida	Normal	564 (25'2 %)	1354 (60'5 %)	319 (14'3 %)	2237	∼0
	Rápida	768 (30'2 %)	1281 (50'4 %)	492 (19'4 %)	2541	
	Blitz	617 (36'7 %)	610 (36'3 %)	452 (26'9 %)	1679	
Tipo de torneo	Equipos	29 (14'2 %)	148 (72'5 %)	27 (13'2 %)	2014	∼0
	Individual	1907 (30'9 %)	3028 (49'1 %)	1228 (19'9 %)	6163	
	Selecciones	13 (14'4 %)	69 (76'7 %)	8 (8'9 %)	90	
Importancia del torneo	Resto	1013 (31'1 %)	1554 (47'7 %)	691 (21'2 %)	3258	0'0027
	Tipo 1	17 (29'8 %)	32 (56'1 %)	8 (14 %)	57	
	Tipo 2	457 (29'3 %)	824 (52'8 %)	279 (17'9 %)	1560	
	Tipo 3	462 (29'2 %)	835 (52'8 %)	285 (18 %)	1582	

Tabla 3.1: Número (y % por filas) de partidas para cada variable según el resultado

Así, se puede observar que para cada uno de los distintos tipos de partidas, tipos de torneo e importancia de torneo, el resultado que más predomina en términos absolutos es el de tablas. A su vez, se obtuvo un p-valor muy cercano a 0 en los tres casos, por lo que se rechaza la hipótesis nula y al 5 % de significación se rechaza que las variable resultado sea independiente tanto del tipo de partida, como del tipo de torneo o de la importancia del torneo.

Por otra parte, el siguiente paso fue estudiar la relación entre la variable **Result** y las variables cuantitativas, obteniendo la siguiente tabla:

	Resultado			
	1-0	1/2-1/2	0-1	p-valor
Rating1	2757'04 (± 75'02)	2742 (± 62'20)	2726'23 (±86'23)	∼0
Rating2	2733'42 (± 85'60)	2746'69 (± 65'59)	2751'19 (±79'50)	∼0
WhiteMaxRating	2786'33 (± 54'74)	2774'52 (± 52'51)	2763'81 (±56'08)	∼0
BlackMaxRating	2768'70 (± 55'74)	148 (72'5 %)	27 (13'2 %)	∼0
WhiteRatingChange	7'76 (± 34'38)	3'01 (± 30'42)	6'52 (± 33'97)	∼0
BlackRatingChange	7'30 (± 34'89)	4'43 (± 31'07)	6'95 (± 34'16)	0'003
WhiteRating2018	2734'66 (± 77'88)	2727'47 (± 69'70))	2709'50 (± 82'96)	∼0
BlackRating2018	2717'18 (± 80'62)	2732'20 (± 68'78))	2781'85 (± 56'86)	∼0
WhiteRating2019	2738'63 (± 63'85)	2730'04 (± 58'43)	2718'26 (± 65'72)	∼0
BlackRating2019	2723'49 (± 64'69)	2734'04 (± 58'47)	2734'39 (± 65'07)	∼0
WhiteRating2020	2744'62 (± 62'15)	2732'05 (± 54'86)	2721'31 (± 58'68)	∼0
BlackRating2020	2726'58 (± 59'57)	2735'81 (± 57'02)	2740'74 (± 61'97)	∼0
EF1W	2755'00 (± 126'36)	2747'91 (± 117'68)	2727'60 (± 111'67)	∼0
EF1B	2738'62 (± 123'68)	2753'95 (± 117'38)	2754'07 (± 119'01)	∼0
EF2W	5'09 (± 1'28)	5'02 (± 1'18)	4'78 (± 1'27)	∼0
EF2B	4'90 (± 1'30)	5'08 (± 1'15)	5'16 (± 1'30)	∼0
AgresividadWP	0'55 (± 0'09)	0'53 (± 0'08)	0'56 (± 0'09)	∼0
AgresividadBP	0'55 (± 0'08)	0'53 (± 0'09)	0'55 (± 0'09)	∼0
LuchaWP	45'89 (± 2'15)	45'52 (± 2'37)	45'88 (± 2'37)	∼0
LuchaBP	45'77 (± 2'32)	45'52 (± 2'37)	45'98 (± 2'13)	∼0
H2H	0'57 (± 0'15)	0'49 (± 0'11)	0'42 (± 0'15)	∼0
Edad_WP	28'53 (± 6'32)	29'22 (± 6'73)	28'44 (± 7'15)	∼0

Edad_BP	28'38 (± 6'91)	29'31 (± 6'58)	28'15 (± 6'44)	∼0

Tabla 3.2: Media (y desviación típica) de partidas para cada variable según el resultado

Así, se puede observar que suponiendo normalidad (por el Teorema del Límite Central) y haciendo el Test de Levene para probar la igualdad de varianzas (H_0= Igualdad de varianzas entre los grupos), se obtiene que salvo para AgresividadBP, el resto variables no presentan igualdad de varianzas. Por ello, para estas se aplica el test Welch-ANOVA (H_0= Igualdad de medias en los grupos), rechazándose para todas ellas la hipótesis nula.

Por su parte, para AgresividadBP sí que se obtuvo igualdad de varianzas, por ello se aplica el test ANOVA (H_0= Igualdad de medias entre los grupos) y se obtiene que al 5 % no hay igualdad de medias para ninguna de las variables cuantitativas en estudio.

Por otra parte, posteriormente se realizó un estudio para saber qué variables estaban más relacionadas entre sí. Para ello, se hizo uso del coeficiente de correlación de Spearman, el cual es una medida que estudia si existe relación entre dos variables numéricas, con su correspondiente prueba, la cual tiene como hipótesis nula:

$$H_0 = \text{Las variables son incorreladass.}$$

De esta manera, se obtuvieron las siguientes conclusiones (marcadas con asterisco aquellos p-valores significativos al 5 %):

- Se observan correlaciones muy grandes y positivas entre las variables Rating1, WhiteMaxRating, WhiteRating2018, WhiteRating2019, WhiteRating2020. Así, cuando una de estas variables tome valores altos, las otras también los tomarán.

	Rating1	**WhiteRating2018**	**WhiteRating2019**	**WhiteRating2020**	**WhiteMaxRating**
Rating1	1	0'7378 *	0'6777 *	0'7577 *	0'7608 *
WhiteRating2018	0'7378 *	1	0'8415 *	0'8530 *	0'9096 *
WhiteRating2019	0'6777 *	0'8415 *	1	0'7956 *	0'7652 *
WhiteRating2020	0'7577 *	0'8530 *	0'7956 *	1	0'9078 *
WhiteMaxRating	0'7608 *	0'9096 *	0'7652 *	0'9078 *	1

- Correlaciones muy grandes y positivas entre Rating2, BlackMaxRating, BlackRating2018, BlackRating2019, BlackRating2020. De esta manera, cuando una de estas variables tome valores altos, las otras también los tomarán.

	Rating2	**BlackRating2018**	**BlackRating2019**	**BlackRating2020**	**BlackMaxRating**
Rating2	1	0'7337 *	0'6769 *	0'7560 *	0'7589 *
BlackRating2018	0'7337 *	1	0'8424 *	0'8507 *	0'9105 *
BlackRating2019	0'6769 *	0'8424 *	1	0'8001 *	0.7709 *
BlackRating2020	0'7560 *	0'8507 *	0'8001 *	1	0'9052 *
BlackMaxRating	0'7589 *	0'9105 *	0.7709 *	0'9052 *	1

- Correlación positiva entre EF1 y EF2 y, para cada una de ellas entre cada uno de los colores de las piezas. Así, cuando una de estas variables tome valores altos, las otras también los tomarán.

	EF1W	**EF1B**	**EF2W**	**EF2B**
EF1W	1	0'1021 *	0'8815 *	-0'0194
EF1B	0'1021 *	1	-0'0217	0'8849 *
EF2W	0'8815 *	-0'0217	1	-0'0586 *
EF2B	-0'0194	0'8849 *	-0'0586 *	1

- Correlación negativa ente agresividad y edad (para blanco y negro) y correlación positiva en la edad del jugador de blancas y el de negras. De esta manera, a medida que un jugador se hace mayor pierde agresividad.

	Edad_WP	**Edad_BP**	**AgresividadWP**	**AgresividadBP**
Edad_WP	1	0'0526 *	-0'4239 *	-0'0807
Edad_BP	0'0526 *	1	-0'0734 *	-0'4226 *
AgresividadWP	-0'4239 *	0'0734 *	1	0'0639 *
AgresividadBP	-0'0807 *	0'4226 *	0'0639 *	1

Por su parte, entre el resto de variables no se obtienen correlaciones significativas, siendo estas citadas las únicas reseñables.

Capítulo 4

Estimación del modelo de predicción

Tal y como se expuso en el capítulo 1, en función de si la variable categórica a predecir toma dos o más valores, se distinguen la regresión logística binaria y la multinomial. A su vez, esta última puede ser por dos tipos, dependiendo de si existe un orden lógico o no entre sus categorías. En este caso, la variable a predecir toma tres posibles valores (derrota, empate y victoria). Por ello, en este trabajo se van a considerar los modelos de regresión logística multinomial.

Posteriormente, se estudiarán también los otros modelos considerados en 1: árboles de de decisión, Máquinas de vector soporte y K-vecinos más cercanos.

4.1. Modelo de regresión logística ordinal

En primer lugar, se planteó un modelo de regresión logística ordinal de odds acumulativos o proporcionales. Para ello, inicialmente se plantea el modelo con todas las variables y, a continuación, se hace el test de líneas paralelas o de Odds proporcionales. En él, la hipótesis nula es

H_0 : los coeficientes de regresión son los mismos para las categorías de respuesta.

En esta ocasión necesitamos obtener un p-valor no significativo ($p \geq 0'05$). Para ello, se hará uso de la función *polr* [25] del paquete *ordinal* [26], para plantear el modelo; y de la función *poTest* [27] del paquete *car* [28], para realizar el test de Odds proporcionales. La salida obtenida es la siguiente:

Variable	**Coeficiente** (β_i)	**OR**	**p-valor**
Rating1	-0'0004	0'9995	0'0497
Rating2	0'0006	1'0006	**0'5786**
tipo_partida	—	—	—
Blitz	—	—	—
Normal	0'8860	2'4254	0'0000
Rápida	0'4604	1'5847	0'000087
WhiteMaxRating	-0'0004	0'9996	**0'9592**
BlackMaxRating	0'0011	1'0011	**0'4152**
WhiteRating2018	0'0012	1'0012	**0'1783**
BlackRating2018	0'0003	1'0003	0'0085

WhiteRating2019	-0'0019	0'9981	**0'2599**
BlackRating2019	-0'0018	0'9982	**0'1014**
WhiteRating2020	0'0002	1'0002	**0'7856**
BlackRating2020	0'0008	1'0008	**0'6857**
BlackRatingChange	-0'0008	0'9992	**0'5128**
WhiteRatingChange	-0'0023	0'9976	**0'5112**
H2H	-2'4731	0'0844	0'0000
LuchaWP	-0'0254	0'9749	**0'1048**
AgresividadWP	-1'9994	0'1354	0'0002
LuchaBP	-0'0227	0'9775	**0'1032**
AgresividadBP	-2'7957	0'0611	**0'0811**
Edad_WP	-0'0024	0'9976	0'0033
Edad_BP	-0'0047	0'9953	**0'1438**
Importancia_torneo	—	—	—
Resto	—	—	—
Tipo 1	-0'2621	0'7694	**0'9880**
Tipo 2	0'1343	1'1438	**0'0896**
Tipo 3	0'1539	1'1665	**0'3474**
Tipo_torneo	—	—	—
Equipos	—	—	—
Individual	-0'6102	0'5432	**0'1206**
Selecciones	0'0925	1'0969	**0'6549**
EF1W	-0'0005	0'9995	**0'9628**
EF1B	0'0004	1'0004	**0'9992**
EF2W	0'0975	1'1024	**0'6198**
EF2B	-0'0037	0'9963	**0'7169**

Como ya mencionamos, se necesita que en todas las variables el p-valor sea mayor que 0'05. Sin embargo, en este caso se obtiene un p-valor general mucho más pequeño ($\leq 2 \cdot 10^{-16}$) que ese umbral en alguno de los niveles, por lo que se rechaza la hipótesis nula. Por ello, al no cumplirse la condición teórica de los odds proporcionales, se plantea un modelo de regresión logística multinomial con respuesta nominal.

4.2. Modelo de regresión logística multinomial. Respuesta nominal

Al no ser óptimo el modelo logístico ordinal, el siguiente paso fue crear un modelo de regresión logística multinomial. Este, tal y como se explicó anteriormente, se trata de un modelo en el que la variable respuesta tiene varias categorías sin ningún orden entre ellas. Así, se fija una categoría de referencia, que en nuestro caso va a ser el nivel "1-0", i.e., gana blancas; y se plantean tantas ecuaciones logit como categorías menos una (en este caso, dos):

$$ln\frac{p_i}{p_0} = \alpha_i + \sum_{j=1}^{m=26} \beta_{ij} x_j + \epsilon.$$

con i=1,2 y refiriéndose con p_0 a la probabilidad de la categoría "1-0"; p_1 a la de la categoría "1-2/1-2"; y p_2 a la de la categoría "0-1".

Así, mediante el uso de la función *multinom* [29] del paquete *nnet* [30], se realizarán distintos modelos de regresión logística multinomial.

4.2.1. Primer modelo: todas las variables

En el primer modelo que se realizó se tuvieron en cuenta todas las variables predictoras, obteniéndose los siguientes resultados:

	0-1			**1/2-1/2**		
Variable	β_i	**OR**	**p-valor**	β_i	**OR**	**p-valor**
(Intercept)	11'7626	128371	**<0'001**	14'3063	1633684	**<0'001**
Rating1	-0'0017	0'998	**0'043**	-0'0009	0'999	0'2
Rating2	0'0010	1'001	0'2	0'0007	1'001	0'2
tipo_partida						
Blitz	—	—	—	—	—	—
Normal	-0'3777	0'69	**<0'001**	0'9285	2'53	**<0'001**
Rápida	-0'1170	0'89	**<0'001**	0'5524	1'74	**<0'001**
WhiteMaxRating	0'0000	1'001	>0'9	0'0000	0'999	0'5
BlackMaxRating	0'0000	1'001	>0'9	0'0000	1'001	>0'9
WhiteRating2018	-0'0008	0'999	0'6	0'0016	1'001	0'2
BlackRating2018	-0'0039	0'996	**0'021**	-0'0002	0'998	0'9
WhiteRating2019	0'0012	1'001	0'6	-0.0030	0'996	0'11
BlackRating2019	0'0032	1'003	0'2	-0'0017	0'998	0'3
WhiteRating2020	-0'0009	0'999	0'7	0'0011	1'001	0'5
BlackRating2020	-0'0014	0'998	0'4	0'0009	1'001	0'5
BlackRatingChange	-0'0011	0'998	0'4	-0'0010	0'998	0'3
WhiteRatingChange	-0'0016	0'998	0'3	-0'0030	0'996	**0'006**
H2H	-9'4824	0'00	**<0'001**	-4'9776	0'01	**<0'001**
LuchaWP	0'0108	1'01	0'074	-0'0184	0'98	**0'034**
AgresividadWP	0'7404	2'10	**<0'001**	-1'8156	0'16	**<0'001**
LuchaBP	0'0126	1'01	**0'021**	-0'0175	0'98	0'072
AgresividadBP	-0'6291	0'53	**<0'001**	-3'0463	0'05	**<0'001**
Edad_WP	0'0188	1'02	**0'012**	0'0050	1'01	0'4
Edad_BP	-0'0153	0'98	0'050	-0'0072	0'99	0'2
Importancia_torneo						
Resto	—	—	—	—	—	—
Tipo1	-0'0781	0'92	**<0'001**	-0'2411	0'79	**<0'001**
Tipo2	-0'1419	0'87	**<0'001**	0'1927	1'21	**<0'001**
Tipo3	-0'0518	0'95	**<0'001**	0'2291	1'26	**<0'001**
Tipo_torneo						
Equipos	—	—	—	—	—	—
Individual	-0'5417	0'58	**<0'001**	-0'8793	0'42	**<0'001**
Selecciones	-0'4054	0'67	**<0'001**	-0'0999	0'90	**<0'001**
EF1W	-0'0002	0'999	0.5	-0'0006	0'999	**0'017**
EF1B	0'0004	1'001	0'3	0'0005	1'001	0'071
EF2W	0'0477	1'05	**<0'001**	0'1208	1'13	**<0'001**
EF2B	-0'0375	0'96	**<0'001**	-0'0299	0'97	**<0'001**

Se observa que todas las variables parecen significativas (aquéllas con p-valor menor

o igual que 0'05) a excepción de WhiteMaxRating, WhiteRating2019, WhiteRating2019, WhiteRating2020, BlackMaxRating, BlackRatingChange, BlackRating2019, BlackRating-2020, Rating2, Edad_BP y EF1B. Esto tiene relación con lo explicado en 3.1.4, ya que muchas de estas variables se encuentran correlacionadas con otras.

Así, al haber varias variables no significativas, se va a proceder a simplificar el modelo. Para ello, se hace uso del criterio de información de Akaike [31], obteniendo como modelo óptimo el siguiente:

$$Result \sim Rating1 + tipo\_partida + BlackRating2018 + WhiteRatingChange + H2H + LuchaWP + AgresividadWP + AgresividadBP + Edad\_WP + Edad\_BP + Importancia\_torneo + Tipo\_torneo + EF2W$$

4.2.2. Modelo simplificado

Una vez simplificado el modelo, se ejecuta y se obtiene lo siguiente:

	0-1			**1/2-1/2**		
Variable	β_i	**OR**	**p-valor**	β_i	**OR**	**p-valor**
Intercept	13'4531	695987	**<0'001**	11'8159	135387	**<0'001**
Rating1	-0'0018	0'998	**<0'001**	-0'0013	0'998	**0'002**
tipo_partida						
Blitz	—	—	—	—	—	—
Normal	-0'3834	0'68	**<0'001**	0'9313	2'54	**<0'001**
Rápida	-0'1319	0'88	**<0'001**	0'5530	1'74	**<0'001**
BlackRating2018	-0'0015	0'998	**0'003**	-0'0005	0'998	0'2
WhiteRatingChange	-0'00139	0'998	0'3	-0'0031	0'998	**0'002**
H2H	-9'5687	0'00	**<0'001**	-5'1186	0'01	**<0'001**
LuchaWP	0'0109	1'01	0'5	-0'0229	0'98	0'10
AgresividadWP	0'7537	2'12	**<0'001**	-1'7229	0'18	**<0'001**
AgresividadBP	-0'6221	0'54	**<0'001**	-3'0738	0'05	**<0'001**
Edad_WP	0'0177	1'02	**0'006**	0'0035	1'003	0'5
Edad_BP	-0'0175	0'98	**0'010**	-0'0067	0'99	0'2
Importancia_torneo						
Resto	—	—	—	—	—	—
Tipo1	-0'0638	0'94	**<0'001**	-0'2439	0'78	**<0'001**
Tipo2	-0'1235	0'88	**<0'001**	0'1919	1'21	**<0'001**
Tipo3	-0'0396	0'96	**0'019**	0'2227	1'25	**<0'001**
Tipo_torneo						
Equipos	—	—	—	—	—	—
Individual	-0.5525	0'58	**<0'001**	-0'9082	0'40	**<0'001**
Selecciones	-0'4459	0'64	**<0'001**	-0'0843	0'92	**<0'001**
EF2W	0'0267	1'03	0'4	0'0691	1'07	**0'009**

La interpretación de estos resultados es la siguiente:

- Derrota frente a victoria de blancas:
  - Variables cuantitativas significativas con *Odds Ratio* superior a 1: AgresividadWP y Edad_WP. Esto significa que a mayor puntuación en estas variables, aumenta la probabilidad de derrota de blancas frente a victoria. En otras palabras, la probabilidad de derrota aumenta, con respecto a la victoria blanca, para jugadores más agresivos y más jóvenes.

- Variables cuantitativas significativas con *Odds Ratio* inferior a 1: Rating1, BlackRating2018, H2H, AgresividadBP y Edad_BP. Esto equivale a que a mayor puntuación en estas variables, disminuye la probabilidad de derrota de blancas frente a victoria. Complementario del ítem anterior.
- Variables cualitativas:
  - El Odds Ratio de partidas del tipo Normal y Rápida es inferior a 1 en cada una de ellas. Esto significa que la probabilidad de derrota del jugador blanco frente a victoria disminuye cuando la partida es de esos tipos con respecto al otro tipo de partidas.
  - El Odds Ratio de partidas con importancia tipo 1,2,3 es inferior a 1 en cada una de ellas. Esto significa que la probabilidad de derrota del jugador blanco frente a victoria disminuye cuando el torneo es de tipo 1,2 o 3 con respecto al resto de torneos.
  - El Odds Ratio de partidas de torneos del tipo Individual y Selecciones es inferior a 1 en cada una de ellas. Esto significa que la probabilidad de derrota del jugador blanco frente a victoria disminuye cuando el torneo es de tipo Individual o Selecciones con respecto a los de equipos.

- Empate frente a victoria de blancas:
  - Variables cuantitativas significativas con *Odds Ratio* superior a 1: EF2W. Esto significa que a mayor puntuación en esta variable, aumenta la probabilidad de empate frente a victoria de blancas.
  - Variables cuantitativas significativas con *Odds Ratio* inferior a 1: Rating1, WhiteRatingChange, H2H, AgresividadWP y AgresividadBP. Esto equivale a que a mayor puntuación en estas variables, disminuye la probabilidad de empate frente a victoria de blancas.
  - Variables cualitativas:
    - El Odds Ratio de partidas del tipo Normal y Rápida es superior a 1 en cada una de ellas. Esto significa que la probabilidad de empate del jugador blanco frente a victoria aumenta cuando la partida es de esos tipos con respecto a los otros tipos de partidas.
    - El Odds Ratio de partidas con importancia tipo 1 es inferior a 1. Esto significa que la probilidad de empate del jugador blanco frente a victoria disminuye cuando el torneo es de tipo 1 con respecto al resto de torneos. Por su parte, el Odds Ratio de partidas con importancia tipo 2 o 3 es superior a 1 en ambos casos. Por ello, la probabilidad de empate del jugador blanco frente a victoria disminuye cuando el torneo es de tipo 2 o 3 con respecto al resto de torneos
    - El Odds Ratio de partidas de torneos del tipo Individual y Selecciones es inferior a 1 en cada una de ellas. Esto significa que la probabilidad de empate del jugador blanco frente a victoria disminuye cuando el torneo es de tipo Individual o Selecciones con respecto a los de equipos.

Otro detalle importante que se puede observar de la tabla anterior, es que hay variables que no son significativas, tales como LuchaWP. A pesar de esto, estas variables no pueden ser eliminadas del modelo, ya que hacer eso provocaría que otras variables sí significativas dejasen de serlo.

Por su parte, una vez obtenido este modelo simplificado, se ejecuta, se comprueba que no se puede optimizar más, y se obtiene la matriz de confusión siguiente:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	739	421	177
1/2-1/2	1106	786	2673
0-1	104	151	300

Esta matriz de confusión es obtenida de esta base de datos pero, posteriormente, se obtendrá otra cuando se haga la validación. A su vez, de esta manera, y usando como medida el porcentaje de aciertos, se obtienen las siguientes conclusiones:

- El modelo tiene un 57'48 % de exactitud.
- El porcentaje de errores "graves"(predecir que va a ganar blancas y lo haga negras, o viceversa) es del 4'3 %.

4.2.3. Análisis de la bondad de ajuste

Antes de utilizar nuestros modelos para hacer predicciones, es necesario verificar que el modelo se ajuste bien a la base de datos.

Así, se realizó el test **Pseudo** R^2 [15, 14]. Los coeficientes de tipo R^2 son obtenidos de la misma manera que en otros modelos de regresión logística. Para ello, se usa la función *pr2* [32] del paquete *pscl* [33]. Así, haciendo uso de esa funcion y, teniendo en cuenta que, a mayor valor de los coeficientes más explicativo será el modelo, se obtuvieron los siguientes resultados:

McFadden	R^2ML	R^2CU
0'1205858	0'2192994	0'2515906

Por tanto, y aunque en un modelo de regresión lineal, un R^2 es alto a partir de 0'5 o 0'7 (según el campo de estudio), en un modelo de regresión logística se usan escalas diferentes. Por ello, los coeficientes son bastante aceptables, así que la capacidad explicativa de los modelos también lo será.

Por otra parte, también se realizó el **test de razón de verosimilitudes** [13]. Este, tiene como hipótesis nula que todos los coeficientes del modelo son nulos. Por ello, el modelo será adecuado cuando se obtenga significación estadística (p-valor $\leq$ 0'05). En este caso, y mediante el uso de la función *anova* de *R*, se obtiene un p-valor=0, que es más pequeño que 0'05. Por ello, el modelo parece adecuado.

4.3. Modelo de regresión logística multinomial en función del tipo de partida

Tal y como se observó en la sección anterior, el modelo obtenido tiene el tipo de partida como una de las variables significativas. Por ello, y atendiendo a la importancia de esta variable, se plantearon distintos modelos en función de cada uno de los posibles tipos de partida: "Blitz", "Rápidas", "Normal".

4.3.1. Partidas blitz

En primer lugar, se realizó el modelo multinomial para las partidas tipo Blitz. Para este caso, inicialmente se consideraron todas las variables, a excepción del tipo de partida (ya que en esta nueva base de datos solo tiene un nivel: Blitz) y, el tipo de torneo (la

muestra de torneos de selecciones era muy pequeña, apenas 3). Además, hay que tener en cuenta que para este conjunto de datos no hay tipo 1 en la Importancia del torneo.

Así, se planteó el modelo y se obtuvieron los siguientes resultados:

	0-1			**1/2-1/2**		
Variable	β_i	**OR**	**p-valor**	β_i	**OR**	**p-valor**
(Intercept)	14'7305	2496818	**<0'001**	17'9804	64383576	**<0'001**
Rating1	-0'0016	0'998	0'2	0'0006	1'001	0'6
Rating2	-0'0003	0'999	-0'0007	0'8	0'999	0'5
WhiteMaxRating	0'0059	1'01	0'063	0'0037	1'003	0'2
BlackMaxRating	0'0023	1'002	0'5	0'0036	1'003	0'2
WhiteRating2018	-0'0024	0'997	0'4	0'0014	1'001	0'6
BlackRating2018	-0'0027	0'997	0'4	0'0017	1'001	0'5
WhiteRating2019	-0'0011	0'998	0'8	-0'0071	0'99	0'067
BlackRating2019	-0'0019	0'998	0'6	-0'0094	0'99	**0'010**
WhiteRating2020	-0'0029	0'997	0'4	0'0015	1'001	0'6
BlackRating2020	0'0002	1'001	>0'9	0'0037	1'003	0'2
BlackRatingChange	-0'0004	0'999	0'9	0'0027	1'002	0'2
WhiteRatingChange	-0'0029	0'997	0'2	-0'0026	0'997	0'2
H2H	-8'4769	0'00	**<0'001**	-4'6803	0'01	**<0'001**
LuchaWP	-0'0275	0'97	0'2	-0'0555	0'95	**0'006**
AgresividadWP	-0'5354	0'59	**<0'001**	-2'5634	0'08	**<0'001**
LuchaBP	0'0343	1'03	0'2	-0'0428	0'96	0'082
AgresividadBP	-1'2734	0'28	**<0'001**	-3'9913	0'02	**<0'001**
Edad_WP	-0'0102	0'99	0'4	-0'0204	0'98	0'074
Edad_BP	0'0036	1'003	0'8	-0'0028	0'997	0'8
Importancia_torneo						
Resto	—	—	—	—	—	—
Tipo2	-0'0212	0'98	**<0'001**	0'2311	1'26	**<0'001**
Tipo3	0'0516	1'05	**<0'001**	0'0287	1'03	**<0'001**
EF1W	-0'0017	0'998	**0'013**	-0'0045	0'995	**<0'001**
EF1B	0'0025	1'002	**<0'001**	0'0021	1'002	**<0'001**
EF2W	0'1820	1'20	**<0'001**	0'3956	1'49	**<0'001**
EF2B	-0'1455	0'86	**<0'001**	-0'0906	0'91	**<0'001**

En este caso, se puede ver que las únicas variables no significativas (es decir, con p-valor superior a 0'05) son: Rating1, Rating2, WhiteMaxRating, WhiteRatingChange, WhiteRating2018, WhiteRating2019, WhiteRating2020, BlackRatingChange, BlackRating2018, BlackRating2020. Esto viene explicado por las correlaciones existentes entre algunas de las variables.

Así, se hizo uso del criterio de Akaike [31] para intentar reducir el modelo, llegando al siguiente:

$$Result \sim Rating1 + BlackMaxRating + BlackRating2019 + H2H + AgresividadWP + LuchaBP + AgresividadBP + EF1W + EF2W$$

En este caso, se reduce considerablemente el número de variables predictoras: 9. Además, se puede observar que todas las variables que no salieron significativas anteriormente, no aparecen en este modelo:

	0-1			1/2-1/2		
Variable	β_i	**OR**	**p-valor**	β_i	**OR**	**p-valor**
(Intercept)	14'2697	1574937	**<0'001**	12'7726	352416	**<0'001**
Rating1	-0'0017	0'998	0'057	0'0006	1'001	0'4
BlackMaxRating	0'0018	1'001	0'4	0'0056	1'01	**0'006**
BlackRating2019	-0'0039	0'996	0'069	-0'0058	0'99	**0'003**
H2H	-8'3356	0'00	**<0'001**	-4'5571	0'01	**<0'001**
AgresividadWP	-0'0688	0'93	**<0'001**	-1'9787	0'14	**<0'001**
LuchaBP	0'0393	1'04	0'2	-0'0426	0'96	0'089
AgresividadBP	-1'1938	0'30	**<0'001**	-3'5115	0'03	**<0'001**
EF1W	-0'0006	0'999	0'4	-0'0030	0'999	**<0'001**
EF2W	0'0853	1'09	**<0'001**	0'2727	1'31	**<0'001**

La interpretación de estos resultados es la siguiente:

- Derrota frente a victoria de blancas:
  - Variables cuantitativas significativas con *Odds Ratio* superior a 1: EF2W. Esto significa que a mayor puntuación en esta variable, aumenta la posibilidad de derrota de blancas frente a victoria.
  - Variables cuantitativas significativas con *Odds Ratio* inferior a 1: H2H, AgresividadBP y AgresividadWP. Esto equivale a que a mayor puntuación en estas variables, disminuye la posibilidad de derrota de blancas frente a victoria.
- Empate frente a victoria de blancas:
  - Variables cuantitativas significativas con *Odds Ratio* superior a 1: BlackMaxRating y EF2W. Esto significa que a mayor puntuación en estas variables, aumenta la posibilidad de empate frente a victoria de blancas.
  - Variables cuantitativas significativas con *Odds Ratio* inferior a 1: EF1W, H2H, BlackRating2019, AgresividadWP y AgresividadBP. Esto equivale a que a mayor puntuación en estas variables, disminuye la posibilidad de empate frente a victoria de blancas.

Otro detalle importante que se puede observar de la tabla anterior, es que hay variables que no son significativas, tales como LuchaBP. A pesar de esto, estas variables no pueden ser eliminadas del modelo, ya que hacer eso provocaría que otras variables sí significativas dejasen de serlo.

Así, una vez hecha esta simplificación del modelo, se ejecuta, se comprueba que no se puede simplificar más y se obtiene la matriz de confusión siguiente:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	352	217	116
1/2-1/2	199	306	161
0-1	66	87	175

A su vez, usando como medida el porcentaje de aciertos, se obtienen las siguientes conclusiones:

- El modelo tiene un 49'61 % de exactitud.

- El porcentaje de errores "graves"(predecir que va a ganar blancas y lo haga negras, o viceversa) es del 10'8 %.

4.3.2. Partidas rápidas

El siguiente modelo multinomial llevado a cabo fue seleccionando los datos pertenecientes a las partidas tipo rápidas. Inicialmente se tuvieron en cuenta todas las variables, a excepción del tipo de partida (ya que en esta nueva base de datos solo tiene un nivel: Rápidas) y el tipo de torneo (solo hay partidas individuales). Además, hay que tener en cuenta que para este conjunto de datos no hay tipo 1 en la Importancia del torneo.

Así, se planteó el modelo y se obtuvieron los siguientes resultados:

	0-1			**1/2-1/2**		
Variable	β_i	**OR**	**p-valor**	β_i	**OR**	**p-valor**
(Intercept)	4'7409	115	**<0.001**	13'7132	902731	**<0'001**
Rating1	-0'0032	0'996	**0'020**	-0'0025	0'997	**0'034**
Rating2	0'0019	1'001	0'2	0'0021	1'002	**0'046**
WhiteMaxRating	-0'0009	0'999	0'7	-0'0026	0'997	0'3
BlackMaxRating	0'0005	1'001	0'9	0'0026	1'001	0'2
WhiteRating2018	0'0007	1'001	0'8	0'0025	1'002	0'3
BlackRating2018	0'0057	0'99	0'062	0'00004	1'001	>0'9
WhiteRating2019	0'0000	1'001	>0'9	-0'0082	0'99	**0'019**
BlackRating2019	0'0082	1'01	**0'049**	-0'0029	0'997	0'4
WhiteRating2020	0'0007	1'001	0'8	0'0054	1'01	**0'038**
BlackRating2020	-0'0044	0'995	0'2	-0'0001	0'999	>0'9
BlackRatingChange	-0'0010	0'998	0'6	-0'0023	0'997	0'2
WhiteRatingChange	0'0007	1'001	0'7	-0'0022	0'997	0'2
H2H	-8'5983	0'00	**<0'001**	-4'1430	0'02	**<0'001**
LuchaWP	0'0375	1'04	**<0'001**	0'0071	1'01	0'6
AgresividadWP	1'0885	2'97	**<0'001**	-3'0701	0'05	**<0'001**
LuchaBP	0'0107	1'01	0'2	-0'0111	0'99	0'5
AgresividadBP	-1'1404	0'32	**<0'001**	2'8679	0'06	**<0'001**
Edad_WP	0'0271	1'03	**0'028**	0'0136	1'01	0'2
Edad_BP	-0'0118	0'99	0'4	-0'0060	0'99	0'5
Importancia_torneo						
Resto	—	—	—	—	—	—
Tipo2	-0'1862	0'83	**<0'001**	-0'0967	0'91	**<0'001**
Tipo3	-0'1123	0'89	**<0'001**	0'1299	1'14	**<0'001**
EF1W	-0'00001	0'999	>0'9	-0'0006	0'999	0'2
EF1B	0'0011	1'001	0'074	0'0011	1'001	**0'040**
EF2W	-0'0087	0'99	**<0'001**	0'1866	1'21	**<0'001**
EF2B	-0'1277	0'88	**<0'001**	-0'0886	0'92	**<0'001**

En este caso, se puede ver que las únicas variables no significativas (p-valor superior a 0'05) son: Edad_BP, Edad_WP, LuchaBP, Rating1, Rating2, WhiteMaxRating, WhiteRatingChange, WhiteRating2018, WhiteRating2019, WhiteRating2020, BlackMaxRating, BlacRatingChange, BlackRating2018 y BlackRating2020. Por ello, se hizo uso del criterio de Akaike [31] para intentar reducir el modelo, llegando al siguiente:

$$Result \sim Rating1 + Rating2 + WhiteRating2019 + H2H + AgresividadWP + AgresividadBP + Edad\_WP + EF2W$$

En este caso, se puede observar que todas las variables que no salieron significativas anteriormente, no aparecen en este modelo:

	0-1			**1/2-1/2**		
Variable	β_i	**OR**	**p-valor**	β_i	**OR**	**p-valor**
(Intercept)	5'8658	353	**<0'001**	11'6541	115169	**<0'001**
Rating1	-0'0025	0'997	**0'024**	-0'0023	0'997	**0'021**
Rating2	0'0007	1'001	0'3	-0'0023	1'001	**0'008**
WhiteRating2019	0'0007	1'001	0'5	-0'0019	0'998	0'053
H2H	-8'3110	0'00	**<0'001**	-4'1207	0'02	**<0'001**
AgresividadWP	1'3397	3'82	**<0'001**	-2'5730	0'08	**<0'001**
AgresividadBP	-0'8183	0'44	**<0'001**	-2'5953	0'07	**<0'001**
Edad_WP	0'0246	1'02	**0'017**	0'0103	1'01	0'2
EF2W	-0'0159	0'98	0'8	0'1489	1'16	**<0'001**

La interpretación de estos resultados es la siguiente:

- Derrota frente a victoria de blancas:
  - Variables cuantitativas significativas con *Odds Ratio* superior a 1: AgresividadWP y Edad_WP. Esto significa que cuanto mayor y más agresivo sea el jugador de blancas, aumenta la posibilidad de su derrota frente a victoria.
  - Variables cuantitativas significativas con *Odds Ratio* inferior a 1: Rating1, H2H y AgresividadBP. Esto equivale a que a mayor puntuación en estas variables, disminuye la posibilidad de derrota de blancas frente a victoria. Complementario del ítem anterior.
- Empate frente a victoria de blancas:
  - Variables cuantitativas significativas con *Odds Ratio* superior a 1: Rating2 y EF2W. Esto significa que a mayor puntuación en estas variables, aumenta la posibilidad de empate frente a victoria de blancas.
  - Variables cuantitativas significativas con *Odds Ratio* inferior a 1: Rating1, H2H AgresividadWP y AgresividadBP . Esto equivale a que a mayor puntuación en estas variables, disminuye la posibilidad de empate frente a victoria de blancas.

Otro detalle importante que se puede observar de la tabla anterior, es que hay variables que no son significativas, tales como WhiteRating2019. A pesar de esto, estas variables no pueden ser eliminadas del modelo, ya que hacer eso provocaría que otras variables sí significativas dejasen de serlo. Una vez hecha esta simplificación del modelo, se ejecuta, se comprueba que no se puede simplificar más y se obtiene la matriz de confusión siguiente:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	238	148	55
1/2-1/2	513	1099	388
0-1	17	34	49

A su vez, usando como medida el porcentaje de aciertos, se obtienen las siguientes conclusiones:

- El modelo tiene un 54'54 % de exactitud.
- El porcentaje de errores "graves"(predecir que va a ganar blancas y lo haga negras, o viceversa) es del 2'8 %.

4.3.3. Partidas a ritmo normal

El último modelo multinomial realizado fue seleccionando los datos pertenecientes a las partidas tipo normal. Para este caso, inicialmente se consideraron todas las variables, a excepción del tipo de partida (ya que en esta nueva base de datos solo tiene un nivel: Normal).

Así, se planteó el modelo y se obtuvieron los siguientes resultados:

	0-1			**1/2-1/2**		
Variable	β_i	**OR**	**p-valor**	β_i	**OR**	**p-valor**
(Intercept)	15'9818	8725466	**<0'001**	19'5876	321205002	**<0'001**
Rating1	0'0005	1'001	0'9	-0'0029	0'997	0'2
Rating2	0'0038	1'003	0'2	0'0017	1'001	0'4
WhiteMaxRating	-0'0059	0'99	0'10	-0'0023	0'997	0'4
BlackMaxRating	-0'0017	1'00	0'7	-0'0042	1'00	0'11
WhiteRating2018	-0'0019	0'998	0'5	0'0007	0'995	0'8
BlackRating2018	-0'0037	0'996	0'3	-0'0029	0'997	0'2
WhiteRating2019	0'0036	1'003	0'4	0'0032	1'003	0'3
BlackRating2019	0'0026	1'002	0'5	0'0036	1'003	0'2
WhiteRating2020	0'00003	1'001	>0'9	-0'0013	0'998	0'6
BlackRating2020	-0'0004	0'999	0'9	0'0012	1'001	0'6
BlackRatingChange	-0'0029	0'997	0'3	-0'0038	1'00	0'094
WhiteRatingChange	-0'0028	0'997	0'4	-0'0042	1'001	0'085
H2H	-11'9826	0'00	**<0'001**	-6'2879	0'00	**<0'001**
LuchaWP	0'0179	1'02	0'061	-0'0291	0'97	0'065
AgresividadWP	1'3746	3'95	**<0'001**	-0'3314	0'72	**<0'001**
LuchaBP	-0'0249	0'98	**0'003**	-0'0334	0'97	0'062
AgresividadBP	0'4845	1'62	**<0'001**	-2'7929	0'06	**<0'001**
Edad_WP	0'0546	1'06	**<0'001**	0'0221	1'02	**0'043**
Edad_BP	-0'0490	0'95	**0'003**	-0'0334	0'98	**0'024**
Importancia_torneo						
Resto	—	—	—	—	—	—
Tipo1	-0'0808	0'92	**<0'001**	-0'0334	1'18	**<0'001**
Tipo2	-0'1650	0'85	**<0'001**	0'6469	1'91	**<0'001**
Tipo3	-0'014	0'90	**<0'001**	0'7172	2'05	**<0'001**
EF1W	0'00002	1'001	>0'9	-0'0003	0'999	0'5
EF1B	-0'0012	0'998	0'12	-0'0002	0'999	0'7
EF2W	0'0818	1'09	**<0'001**	0'0501	1'05	**<0'001**
EF2B	0'0696	1'07	**<0'001**	-0'0029	0'997	**<0'001**
Tipo_torneo						
Equipos	—	—	—	—	—	

Individual	-0'5466	0'58	**<0'001**	-1'1368	0'32	**<0'001**
Selecciones	-0'4587	0'63	**<0'001**	-0'4465	0'64	**<0'001**

En este caso, las únicas variables no significativas (esto es, con un p-valor superior a 0'05) son: EF1B, EF1W, Rating1, Rating2, WhiteMaxRating, WhiteRatingChange, WhiteRating2018, WhiteRating2019, WhiteRating2020, BlackRatingChange, BlackRating2018, BlackRating2019, BlackRating2020. Esto viene explicado por la relación existente entre algunas variables vista en 3.1.4. Por ello, se hizo uso del criterio de Akaike [31] para intentar reducir el modelo, llegando al siguiente:

$$Result \sim Rating1 + WhiteRatingChange + H2H + AgresividadWP + AgresividadBP + Edad\_WP + Edad\_BP + Importancia\_torneo + Tipo\_torneo$$

En este caso, se puede observar que todas las variables que no salieron significativas anteriormente (a excepción de WhiteRatingChange), no aparecen en este modelo:

	0-1			**1/2-1/2**		
Variable	β_i	**OR**	**p-valor**	β_i	**OR**	**p-valor**
(Intercept)	12'3449	125286	**<0'001**	16'1439	16963158	**<0'001**
Rating1	-0'0026	0'997	**<0'001**	-0'0034	0'996	**<0'001**
WhiteRatingChange	-0'0033	0'996	0'2	-0'0059	0'99	**0'002**
H2H	-11'9477	0'00	**<0'001**	-6'1174	0'00	**<0'001**
AgresividadWP	1'4131	4'03	**<0'001**	-0'5067	0'70	**<0'001**
AgresividadBP	0'6076	1'90	**<0'001**	-3'1082	0'04	**<0'001**
Edad_WP	0'0413	1'04	**0'002**	0'0259	1'02	**0'016**
Edad_BP	-0'0548	0'95	**<0'001**	-0'0259	0'97	**<0'001**
Importancia_torneo						
Resto	—	—	—	—	—	—
Tipo1	-0'0031	0'996	0'2	0'1477	1'18	**<0'001**
Tipo2	-0'2106	0'81	**<0'001**	0'6264	1'89	**<0'001**
Tipo3	-0'1254	0'88	**0'001**	0'6957	2'02	**<0'001**
Tipo_torneo						
Equipos	—	—	—	—	—	—
Individual	-0'5628	0'57	**<0'001**	-1'2005	0'30	**<0'001**
Selecciones	-0'3654	0'71	**<0'001**	-0'3422	0'68	**<0'001**

La interpretación de estos resultados es la siguiente:

- Derrota frente a victoria de blancas:
  - Variables cuantitativas significativas con *Odds Ratio* superior a 1: AgresividadWP, AgresividadBP y Edad_WP. Esto significa que a mayor puntuación en estas variables, aumenta la posibilidad de derrota de blancas frente a victoria.
  - Variables cuantitativas significativas con *Odds Ratio* inferior a 1: Rating1, H2H y Edad_BP. Esto equivale a que a mayor puntuación en estas variables, disminuye la posibilidad de derrota de blancas frente a victoria. Complementario del item anterior.
  - Variables cualitativas:

- El *Odds Ratio* de partidas con importancia tipo 2 y 3 es inferior a 1 en ambos casos. Esto significa que la probabilidad de derrota del jugador blanco disminuye cuando el torneo es de tipo 2 o 3 con respecto al resto de torneos.
- El *Odds Ratio* de torneos Individuales y de Selecciones es inferior a 1. Esto significa que la probabilidad de derrota del jugador blanco frente a victoria se multiplica disminuye cuando el torneo es individual o de selecciones con respecto a un torneo de equipos.

- Empate frente a victoria de blancas:
  - Variables cuantitativas significativas con *Odds Ratio* superior a 1: Edad_WP. Esto significa que a mayor puntuación en esta variable, aumenta la posibilidad de empate frente a victoria de blancas.
  - Variables cuantitativas significativas con *Odds Ratio* inferior a 1: Rating1, H2H, WhiteRatingChange, AgresividadWP y AgresividadBP . Esto equivale a que a mayor puntuación en estas variables, disminuye la posibilidad de empate frente a victoria de blancas.
  - Variables cualitativas:
    - El *Odds Ratio* de partidas con importancia tipo 1, 2 y 3 es inferior a 1 en los tres casos. Esto significa que la probabilidad de empate del jugador blanco disminuye cuando el torneo es de tipo 1, 2 o 3 con respecto al resto de torneos
    - El *Odds Ratio* de torneos Individuales y de Selecciones es inferior a 1. Esto significa que la probabilidad de empate del jugador blanco frente a victoria se multiplica disminuye cuando el torneo es individual o de selecciones con respecto a un torneo de equipos.

Una vez realizada la simplificación del modelo, se ejecuta, se comprueba que no se puede reducir más y se obtiene la matriz de confusión siguiente:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	203	80	32
1/2-1/2	337	1235	200
0-1	24	39	87

A su vez, usando como medida el porcentaje de aciertos, se obtienen las siguientes conclusiones:

- El modelo tiene un 68'17 % de exactitud.
- El porcentaje de errores "graves"(predecir que va a ganar blancas y lo haga negras, o viceversa) es del 2'5 %.

4.3.4. Bondad de ajuste de los modelos

Al igual que se hizo para el modelo general, analizamos a continuación la bondad de ajuste para los modelos de regresión en función del tipo de partida. En primer lugar, se realizó el test **Pseudo** R^2, obteniendo los siguientes resultados:

Tipo partida	**McFadden**	R^2**ML**	R^2**CU**
Blitz	0'09532	0'1874	0'2114
Rápida	0'07448	0'14159	0'1625
Normal	0'1791233	0'2831001	0'3354176

Se puede ver que, salvo para el modelo de las partidas a ritmo normal, los coeficientes toman valores muy bajos.

A su vez, también se realizó el **test de razón de verosimilitudes**, el cual tiene como hipótesis nula que todos los coeficientes del modelo son nulos.

Tipo partida	Blitz	Rápida	Normal
p-valor	0	0	0

Todos los p-valores son menores que 0'05, por lo que se pueden dar por buenos los modelos.

Por otra parte, atendiendo a los porcentajes de acierto, se puede apreciar que los modelos por separado funcionan mejor que el modelo junto.

4.4. Árboles de clasificación

En primer lugar, se construye un árbol con profundidad máxima, y fijando el parámetro de complejidad en 0 y el número mínimo de observaciones en un nodo para intentar su división en 2. Para él se consideran todas las variables y se hace uso de la función *rpart* [34] del paquete con el mismo nombre. Luego, identificando el parámetro de complejidad óptimo, se poda el árbol usando la función *prune* [35] del paquete *rpart* [34] para obtener un modelo más simple. La matriz de confusión que se obtiene es la siguiente:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	855	1089	5
1/2-1/2	662	2545	38
0-1	142	993	128

A su vez, usando como medida el porcentaje de aciertos, se obtienen las siguientes conclusiones:

- El modelo tiene un 54'64 % de exactitud, ligeramente inferior al obtenido con el modelo multinomial simplificado (57'48 %).
- El porcentaje de errores "graves"(predecir que va a ganar blancas y lo haga negras, o viceversa) es del 2'27 %, siendo inferior al obtenido con el modelo multinomial simplificado (4'3 %).

4.5. Máquinas de vector soporte

En nuestro caso particular, tras comprobar con diferentes tipos de kernel, se decidió usar el "radial". Así, y haciendo uso de la función *svm* [36] del paquete de *R* llamado *e1071* [37], se obtiene la siguiente matriz de confusión:

A su vez, usando como medida el porcentaje de aciertos, se obtienen las siguientes conclusiones:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	690	1217	42
1/2-1/2	219	2981	45
0-1	133	906	224

- El modelo tiene un 60'32 % de exactitud, superior al obtenido con el modelo multinomial simplificado (57'48 %).
- El porcentaje de errores "graves"(predecir que va a ganar blancas y lo haga negras, o viceversa) es del 2'71 %, siendo inferior al obtenido con el modelo multinomial simplificado (4'3 %).

4.6. K-vecinos más cercanos

Tal y como se explicó en la Sección 1.2.6, este modelo tiene como una de sus complejidades la elección del número de k de vecinos usados para clasificar las nuevas observaciones. Una posible opción es considerar $\sqrt{n}$, donde n denota el número de entradas del conjunto de entrenamiento. No obstante, se decide replicar el modelo para diversos valores de k y quedarse con aquel valor que genere una mayor tasa de acierto. Esto se consigue mediante el uso de la función *train.knn* [38] del paquete *kknn* [39]. Dicha función tiene un hiperparámetro, ks, el cual es un vector de valores específicos de k y, la función ejecuta el modelo para cada uno de esos valores de k, quedándose con el mejor. En este caso, la función *train.knn* decidió que la k más apropiada era 99:

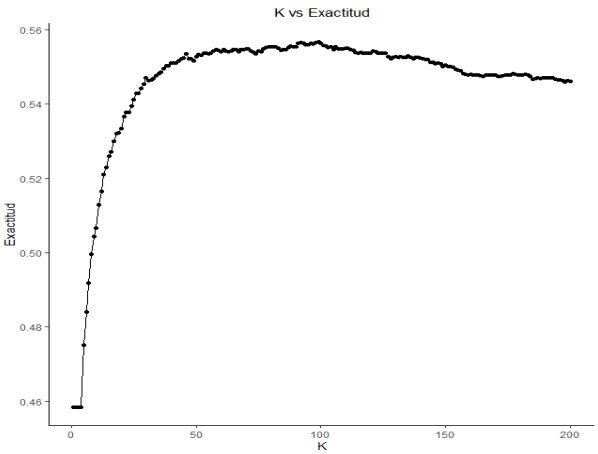

Figura 4.1: K vs Exactitud

Por su parte, se obtuvo la siguiente matriz de confusión:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	727	1160	62
1/2-1/2	171	3021	53
0-1	147	864	252

A su vez, usando como medida el porcentaje de aciertos, se obtienen las siguientes conclusiones:

- El modelo tiene un 61'95 % de exactitud, superior al obtenido con el modelo multinomial simplificado (57'48 %).
- El porcentaje de errores "graves"(predecir que va a ganar blancas y lo haga negras, o viceversa) es del 3'2 %, siendo inferior al obtenido con el modelo multinomial simplificado (4'3 %).

4.7. Comparación

Una vez entrenados los modelos, se obtiene lo siguiente:

Modelo\Acierto	**Exactitud**	**Errores "graves"**
Modelo multinomial general	57'48 %	4'3 %
Modelo según tipo de partida	-	-
Blitz	49'61 %	10'8 %
Rápidas	54'54 %	2'8 %
Normal	68'17 %	2'5 %
Árbol de clasificación	54'64 %	2'27 %
SVM	60'32 %	2'71 %
K-NN	61'95 %	3'2 %

De esta manera, se puede observar que los modelos según el tipo de partida funcionan mejor que el modelo general y que el resto de modelos considerados, siendo el de las partidas normales el que más exactitud alcanza con un 68'17 %. Posteriormente, los mejores modelos serían las máquinas de vector soporte (60'32 %) y K-vecinos más cercanos (61'95 %). Por su parte, en cuanto a los errores graves, el árbol es el que menos valor toma y, en general, los modelos según el tipo de partida funcionan mejor en este aspecto (en comparación con el multinomial general), a excepción del de las partidas tipo "Blitz".

Capítulo 5

Validación del modelo

Una vez creados los distintos modelos de predicción, pasamos a validar los mismos con un nuevo conjunto de datos.

5.1. Los torneos

Para realizar la validación, se han considerado las partidas de varios torneos de distinto tipo:

- Para las partidas tipo Blitz, se recogieron las partidas entre los jugadores pertenecientes al top 100 que tuvieron lugar en el Campeonato del Mundo Blitz/Relámpago 2021. En total hay 145 partidas.
- Se hizo lo propio con el Campeonato del Mundo Rápido 2021 para las partidas rápidas. En total hay 101 partidas.
- Para las partidas normales, se consideraron los juegos entre jugadores del top 100 durante el torneo Tata Steel 2022. En total, se recogieron 87 juegos.
- Para la validación sin distinguir el tipo de partida, se usa de conjunto de datos el compendio de los tres mencionados anteriormente. Así, en total, se tienen 333 partidas.

De esta forma, la proporción de datos de entrenamiento y de validación sería la siguiente:

Conjunto de datos	**Datos entrenamiento (% del total)**	**Datos de validación (% del total)**
Blitz	1685 (92.07 %)	145 (7.93 %)
Rápidas	2645 (96.32 %)	101 (3.68 %)
Normal	2801 (96.99 %)	88 (3.01 %)
Conjunto de datos completo	7131 (95.54 %)	333 (4.46 %)

5.2. Resultados del modelo multinomial

En primer lugar, se planteó la validación para el modelo completo (con todos los tipos de partidas). Para ello, se hizo uso del modelo simplificado resultante obtenido en la Sección 4.2.

Así, se realizaron predicciones con este modelo para las partidas de los distintos torneos mencionados anteriormente. La comparación entre los resultados esperados y observados viene resumida en la siguiente matriz de confusión:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	33	51	20
1/2-1/2	31	81	18
0-1	22	27	17

Como medida de porcentaje de aciertos, se obtienen las siguientes conclusiones:

- El modelo tiene un 43.67 % de exactitud.
- El porcentaje de errores "graves" (predecir que va a ganar blancas y lo haga negras, o viceversa) es del 14 %.

5.3. Resultados del modelo multinomial por tipo de partida

En este apartado se realizó la validación del modelo multinomial atendiendo al tipo de partida. Por ello, solo se tuvieron en cuenta como datos de validación a aquellas partidas del tipo que se estaban estudiando.

5.3.1. Partidas Blitz

En este caso, se realizó el modelo multinomial para las partidas tipo Blitz. Para ello, se consideró el modelo obtenido en el apartado 4.3.1:

La siguiente matriz de confusión resume la relación entre los resultados observados y esperados:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	24	20	15
1/2-1/2	14	13	11
0-1	15	11	13

Se obtienen las siguientes conclusiones:

- El modelo tiene un 44'52 % de exactitud.
- El porcentaje de errores "graves" (predecir que va a ganar blancas y lo haga negras, o viceversa) es del 19'35 %.

5.3.2. Partidas rápidas

Se considera a continuación la validación del modelo de regresión para las partidas rápidas, obtenido en el apartado 4.3.2:

La siguiente matriz de confusión compara los resultados esperados y los observados: Se obtienen las siguientes conclusiones:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	11	14	2
1/2-1/2	7	6	3
0-1	15	30	4

- El modelo tiene un 47'82 % de exactitud.

- El porcentaje de errores "graves" (predecir que va a ganar blancas y lo haga negras, o viceversa) es del 9'78 %.

Se observa así que este modelo es ligeramente superior a los anteriores.

5.3.3. Partidas normales

Consideramis finalmente el modelo multinomial para las partidas tipo normal. Para ello, se consideró el modelo obtenido en el apartado 4.3.3:

La validación de este modelo en el torneo mencionado anteriormente viene resumida en la siguiente matriz de confusión:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	4	13	1
1/2-1/2	8	35	0
0-1	1	9	1

Como medida de porcentaje de aciertos, se obtienen las siguientes conclusiones:

- El modelo tiene un 55'56 % de exactitud.
- El porcentaje de errores "graves" (predecir que va a ganar blancas y lo haga negras, o viceversa) es del 2'7 %.

Este modelo es por tanto muy superior a los anteriores.

5.4. Árboles de clasificación

Pasamos a continuación a validar el árbol de clasificación entrenado en la Sección 4.4, utilizando las partidas de todos los torneos de validación. Se obtiene lo siguiente:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	36	71	11
1/2-1/2	45	92	4
0-1	16	46	12

Como medida de porcentaje de aciertos, se obtienen las siguientes conclusiones:

- El modelo tiene un 42'04 % de exactitud.
- El porcentaje de errores "graves" (predecir que va a ganar blancas y lo haga negras, o viceversa) es del 8'1 %.

5.5. Máquinas de vector soporte

En lo que se refiere al modelo de máquinas de vector soporte entrenado en la Sección 4.5 y usando los datos de validación se obtuvo lo siguiente:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	32	61	11
1/2-1/2	32	91	7
0-1	21	31	14

A su vez, como medida de porcentaje de aciertos, se obtienen las siguientes conclusiones:

- El modelo tiene un 41'52 % de exactitud.
- El porcentaje de errores "graves"(predecir que va a ganar blancas y lo haga negras, o viceversa) es del 10'67 %.

5.6. K-vecinos más cercanos

Consideramos por último el modelo de K-vecinos más cercanos entrenado en la Sección 4.6. Utilizando las partidas de todos los torneos de validación se obtuvo lo siguiente:

Matriz de confusión	1-0	1/2-1/2	0-1
1-0	25	72	7
1/2-1/2	18	108	4
0-1	10	46	10

De aquí se extraen las siguientes conclusiones:

- El modelo tiene un 47'67 % de exactitud.
- El porcentaje de errores "graves"(predecir que va a ganar blancas y lo haga negras, o viceversa) es del 5'67 %.

5.7. Predicción haciendo uso del Rating

Tal y como se vio en la Sección 2.1.2, la puntuación Elo entre dos jugadores permite dar una estimación de la probabilidad estimada de puntuación entre ellos, la cual se define para cada jugador como su probabilidad de ganar más la mitad de la probabilidad de empate. De esta manera, dados dos jugadores A (jugador de blancas) y B (jugador de negras) con ratings R_A y R_B en el momento de la partida, respectivamente y, con probabilidad de empate igual a p, se tiene lo siguiente:

- P(Gana jugador A)=$E_A - \frac{1}{2} \cdot p = \frac{1}{1+10^{\frac{(R_B - R_A)}{400}}} - \frac{1}{2} \cdot p.$
- P(tablas)=p
- P(Gana jugador B)=$E_B - \frac{1}{2} \cdot p = \frac{1}{1+10^{\frac{(R_A - R_B)}{400}}} - \frac{1}{2} \cdot p.$

De esta manera, y para todas las partidas en el conjunto de datos de validación, es posible estimar la probabilidad de los distintos resultados de la partida, una vez fijada la probabilidad p de tablas. A partir de estas estimaciones, se puede dar como predicción del resultado aquél que tenga una mayor probabilidad.

Una vez obtenida la predicción se compara con el resultado real y se obtiene el porcentaje de aciertos final y el porcentaje de errores graves (predice que gana blancas y gana negras, y viceversa).

En este trabajo, se ha realizado esta validación para diferentes probabilidades de tablas. En concreto esta probabilidad va a variar de 0 a 1 con paso de 0'01. De esta manera, e iterando todo lo anterior, se obtiene la siguiente gráfica:

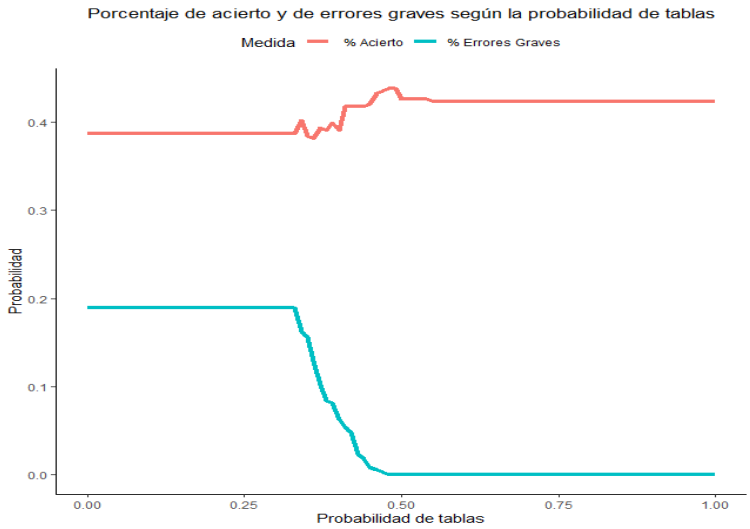

Figura 5.1: Acierto y errores graves según la probabilidad de tablas

De esta manera, se puede observar que el porcentaje de aciertos se va incrementando a medida que aumenta la probabilidad de tablas y que, por el contrario, la probabilidad de errores graves va disminuyendo. Por su parte, el mayor acierto se consigue con p=0'48, ya que con este valor se obtiene un porcentaje de acierto del 43'84 % y un porcentaje de errores graves del 0 %. La razón de esto último viene dada por el hecho de que para un probabilidad de tablas igual a 0'48 es muy probable predecir tablas. Por ello, es poco posible cometer un error grave, ya que no va a predecir victoria de negras ni de negras.

Por otra parte, si estimamos la probabilidad de tablas, p, a partir de la base de datos, se obtiene que toma un valor igual a 0'503. Así, para este valor se obtiene una probabilidad de acierto de 0'429 (inferior al de la p óptima) y un porcentaje de errores graves del 0 %.

A su vez, este modelo funciona peor que los anteriores pero, no obstante, las diferencias no son tan grandes como podría ser de esperar en un principio.

5.8. Comparativo

La siguiente tabla resume los resultados de la validación con los distintos modelos:

Modelo\Acierto	**Exactitud**	**Errores "graves"**
Modelo multinomial general	43'67 %	14 %
Modelo según tipo de partida	-	-
Blitz	44'52 %	19'35 %
Rápidas	47.82 %	9'78 %
Normal	55'56 %	2'7 %
Árbol de clasificación	42'04 %	8'1 %
SVM	41'52 %	10'67 %
K-NN	47'67 %	5'67 %
Modelo Rating p óptima	43'84 %	0 %
Modelo Rating con modelo estimado	42'94 %	0 %

Se puede observar que los modelos según el tipo de partida tienen mayor acierto que el modelo multinomial general y que el resto de modelos considerados, siendo el de las partidas normales el que más exactitud alcanza con un 55'56 %. Por su parte, en cuanto a los errores graves, el modelo realizado para las partidas del tipo "Normal" es el que menos valor toma y, en general, los modelos según el tipo de partida funcionan mejor en este aspecto, a excepción del de las partidas tipo "Blitz".

Además, cabe reseñar que los modelos que peor funcionan son el multinomial general y el que atendía al rating, ya que con elos apenas se llega al 44 % de acierto.

Por último, estos resultados en la validación entran en concordancia con lo visto en el apartado 4.7, ya que en él los modelos que más acertaban en el entrenamiento (modelo multinomial según el tipo de partida, K-NN y SVM) son los que más aciertan aquí.

Capítulo 6

Conclusiones

Tal y como se habíamos propuesto al inicio del trabajo, se han construido diversos modelos capaces de predecir con gran acierto el resultado de una partida de ajedrez. En primer lugar y, para ello, se realizó un análisis descriptivo de la base de datos creada. Ahí, se apreciaron correlaciones muy grandes entre las variables Rating1, WhiteMaxRating, WhiteRating2018, WhiteRating2019 y WhiteRating2020. Por su parte, lo mismo ocurría para las variables homólogas del jugador de negras. Además también se observó una correlación positiva entre los dos estados de forma considerados y, una correlación negativa entre la edad del jugador y su agresividad.

El primero, haciendo uso de un modelo de regresión logística multinomial, utilizaba una serie de variables predictoras conocidas, que estaban incorporadas en la base de datos previamente creada, y como variable respuesta el resultado de la partida. Posteriormente, se simplificó el modelo para que solo incorporase aquellas variables significativas, observándose que las relaciones vistas en el análisis descriptivo jugaban un papel muy importante, ya que muchas de las variables no aparecían al estar relacionadas con otras.

A continuación, se realizaron 3 modelos de regresión logística multinomial más. Uno para cada tipo de partida. Esto fue realizado ya que con el modelo obtenido anteriormente se tenía al tipo de partida como una de las variables significativas. Una vez hecho esto, se realizaron otros modelos: árbol de decisión, K-vecinos más cercanos y máquinas de vector soporte.

Cabe destacar que en cuanto al entrenamiento los modelos según el tipo de partida funcionan mejor que el modelo general y que el resto de modelos considerados, siendo el de las partidas normales el que más exactitud alcanza con un 68'17 %.

Una vez entrenado el conjunto de datos, se realizó la validación de cada uno de los modelos. Para ello se construyó una nueva base de datos que incoporaba resultados de diversos torneos. Además, a todos los modelos previamente entrenados se le incoporó uno nuevo. Este consistía en predecir el resultado de la partida haciendo uso del rating de cada uno de los jugadores que la disputaban. Una vez hecha la validación se puede observar que los modelos según el tipo de partida tienen mayor acierto que el modelo multinomial general y que el resto de modelos considerados. Por lo tanto, esto coincide con lo visto en el entrenamiento.

En cuanto a las limitaciones del estudio cabe destacar que a la hora de crear la base de datos hubo diversos problemas para conseguir las variables de algunos jugadores. Eso venía dado a que la página de la que se obtuvo la información no contaba con suficientes datos de esos jugadores. Así, una forma de mejorar estos modelos sería haber aumentado el tamaño de la base de datos ya que, aunque esta cuenta con 6457 partidas, el hecho de poder haber incorporado más partidas podría haber hecho más exactos los modelos. Por

otra parte, otra manera de mejorar los resultados habría sido considerar más variables. Por ejemplo, en alguno de los artículos expuestos en la revisión de la literatura se consideraban diversos tipos de rating, lo cual posiblemente da mejores resultados que considerar uno solo.

A su vez, otra opción para incrementar el acierto de estas predicciones es la de haber considerado otros modelos, tales como Redes Neuronales Convolucionales (como hacían en [5], Random Forest (realizado en [7]) o Naive-Bayes (en [6]). De esta manera, se tendrían diversos modelos estudiados.

Apéndice A

Códigos

A.1. Lectura de datos y creación de variables

A.1.1. Carga de datos para los primeros 100 jugadores

```
library(rvest)
#Copiamos el nombre de los 100 primeros jugadores
url<-read_html("https://2700chess.com/?per-page=100")%>%
                              #En primer lugar leemos la página
html_nodes("tbody")%>%
html_nodes("td")%>%
html_nodes("a") #Detectamos dónde se encuentra la información que buscamos.
#Seleccionamos solo el nombre y apellido del jugador.
url_nombres<-url[grepl("games?search",url,fixed=TRUE)]
nombres <-sapply(1:length(url_nombres),function(i) html_attr(url_nombres[i],"href"))
#Ahora se tiene una lista con todos los nombres de los 100 primeros jugadores

#Función para la carga de datos de los 100 primeros jugadores
pag<-function(nombres){
  #Almacenamos las URLS
  urls<-c()
  for (i in 1:30){
    urls[i]<-paste0('https://2700chess.com',nombres,'&page=',i)
  }
  #Leemos las páginas
  pages<-sapply(1:length(urls),function(i) read_html(urls[i])%>%
                  html_nodes("table") %>%
                  html_table()
  )
  #Detectamos cuántas páginas hay para el jugador en cuestión
  vector<-c()
  for(i in 1:30){
    if(isTRUE(nrow(pages[i][[1]])<=1)){
      vector<-c(vector,i)
    }
  }
  #Creamos un vector llamado valores con el número de páginas
  #existentes para el jugador
  todos<-1:30
  valores<-setdiff(todos,vector)
  #Repetimos el proceso de lectura de las páginas, ahora con la cantidad
  #de páginas exactas
  urls<-c()
  for (i in valores){
    urls[i]<-paste0('https://2700chess.com',nombres,'&page=',i)
  }

  pages<-sapply(1:length(urls),function(i) read_html(urls[i])%>%
                  html_nodes("table") %>%
                  html_table()
  )
  #Saco las urls de esa url con las fechas
```

```
 urls_2<-sapply(1:length(urls),function(i)read_html(urls[i])%>%
                  html_nodes("table")%>%
                  html_nodes("td")%>%
                  html_nodes("a")%>%
                  html_attrs()
 )
 #Creamos un vector con el tipo de partida
 posiciones<-lapply(1:(length(urls)), function(i) read_html(urls[i])%>%
                      html_nodes("table")%>%
                      html_nodes("tr"))
 for (i in 1:length(urls)){
   posiciones[[i]]<-posiciones[[i]][-1]
   #Eliminamos el primer valor (irrelevante)
 }
 pos_blitz<-list() #Tipo Blitz
 pos_rapido<-list() #Tipo rápido
 for (i in 1:length(urls)){
   #Encontramos las de Blitz
   pos_blitz[[i]]<-which(grepl("text-blitz date",posiciones[[i]])==TRUE)
   #Encontramos las rápidas
   pos_rapido[[i]]<-which(grepl("text-rapid date",posiciones[[i]])==TRUE)
 }
 #Una vez detectadas las partidas de cada tipo, creamos un vector que
 denote en qué partida están
 posicion_blitz<-c()

 for (i in 1:length(pos_blitz)){
   if(length(pos_blitz[[i]]!=0)){
     b<-(i-1)*length(posiciones[[1]])+pos_blitz[[i]]
     posicion_blitz<-c(posicion_blitz,b)
   }}

 posicion_rapido<-c()
 for (i in 1:length(pos_rapido)){
   if(length(pos_rapido[[i]]!=0)){
     b<-(i-1)*length(posiciones[[1]])+pos_rapido[[i]]
     posicion_rapido<-c(posicion_rapido,b)
   }}
 c<-1:nrow(do.call(rbind,pages))
 posicion_normal<-c[-union(posicion_blitz,posicion_rapido)]
 #El resto de partidas son normales

 #Creamos un vector con las fechas y el torneo
 #Seleccionamos las partes que nos interesan de las urls
 #Para algunos jugadores la información se leía en modo matricial
 if(is.matrix(urls_2)==TRUE){ #
  urls_2<-c(urls_2)
  for(i in 1:length(urls_2)){
    urls_2[[i]]<-urls_2[[i]][1]
  }
   urls_3<-do.call(rbind,urls_2)
   nodelete<-seq(0,3000,2)
   nodelete<-nodelete[-1]
   urls_3<-urls_3[-nodelete]


   #El torneo forma parte del nombre de la url. Lo extraemos
   torneo<-c()
   for (i in 1:length(urls_3)){
     torneo[i]<-substr(urls_3[i],regexpr("r[0-9]",
     urls_3[i]),regexpr("20[0-2]",urls_3[i])-2)
   }
   urls_4<-vector(mode="list",length=length(urls_3))
   #Las fechas se encuentran también en la url. Las extraemos
for(i in 1:length(urls_3)){
  urls_4[i]<-substr(urls_3[i], nchar(urls_3[i])-10+1, nchar(urls_3[i]))
}

 urls_4<-do.call(rbind,urls_4)
 paginas<-do.call(rbind,pages)
```

```
  paginas<-paginas[,-c(1,ncol(paginas))]
  paginas$fecha<-urls_4 #Añadimos las fechas
  paginas$torneo<-torneo
  #Creamos un vector para el tipo de partida
  paginas$tipo_partida<-rep(0,length(urls)*length(posiciones[[1]]))
  #Señalamos que partidas son Blitz
  paginas$tipo_partida[posicion_blitz]<-"Blitz"
  #Señalamos que partidas son Normales
  paginas$tipo_partida[posicion_normal]<-"Normal"
  #Señalamos que partidas son Rápidas
  paginas$tipo_partida[posicion_rapido]<-"Rápida"
  paginas<-subset(paginas,Year>=2019)} else if
  #Para aquellos jugadores cuya información se almacenaba en listas
  (is.list(urls_2)==TRUE){
    fechas<-vector(mode="list",length=length(urls_2))
  for (i in 1:length(urls_2)){
    for(j in 1:length(urls_2[[i]])){
      fechas[[i]][[j]]=substr(urls_2[[i]][[j]][1],
                              nchar(urls_2[[i]][[j]][1])-10+1,
                                    nchar(urls_2[[i]][[j]][1]))
    }
  } #Mismo proceso que en lo anterior pero pasando la lista a vector
    urls_3<-vector(mode="list",length=length(urls_2))
    for (i in 1:length(urls_2)){
      for(j in 1:length(urls_2[[i]])){
        urls_3[[i]][[j]]=urls_2[[i]][[j]][1]

      }
    }
    pares<-seq(2,length(unlist(urls_3)),2)
    urls_3<-unlist(urls_3)[-pares]
    torneo<-c()
    for (i in 1:length(urls_3)){
      torneo[i]<-substr(urls_3[i],regexpr("r[0-9]",
      urls_3[i]),regexpr("20[0-2]",urls_3[i])-2)
    }
    pares<-seq(2,length(unlist(fechas)),2)
   fechas<-unlist(fechas)[-pares]
   paginas<-do.call(rbind,pages)
   paginas<-paginas[,-c(1,ncol(paginas))]
   paginas$fecha<-fechas #Añadimos las fechas
   paginas$torneo<-torneo
   #Creamos vector para el tipo de partida
   paginas$tipo_partida<-rep(0,nrow(do.call(rbind,pages)))
   paginas$tipo_partida[posicion_blitz]<-"Blitz"
   paginas$tipo_partida[posicion_normal]<-"Normal"
   paginas$tipo_partida[posicion_rapido]<-"Rápida"
   paginas<-subset(paginas,Year>=2019)
  }


  return(paginas) #Devolvemos la información
}
```

A.1.2. Creación de la base de datos para cada jugador

```
#Creamos la base de datos para cada jugador
nombres<-sapply(1:length(url_nombres),function(i) html_attr(url_nombres[i],"href"))
#Se crea una lista con los datos de cada uno de los jugadores del Top100
datos_indiv<-lapply(1:length(nombres),function(i) try(pag(nombres[i])))

save(datos_indiv,file="top100.RData") #Los guardamos
load("C:/Users/Usuario/Desktop/uniovi/Máster/Segundo/TFM/top100.RData")

datos_indiv
#Creamos la base de datos general

#Extraemos los apellidos
apellidos_top100<-substr(nombres, 15, regexpr("%",nombres)-1)
#Corregimos errores
```

```
apellidos_top100[15]="Dominguez Perez"
apellidos_top100[35]="Vallejo Pons"
apellidos_top100[36]="Van Foreest"
apellidos_top100[45]="Anton Guijarro"
apellidos_top100[78]="Nihal"
apellidos_top100[84]="Narayanan"
```

A.1.3. Creación de un único dataframe

```
#Volcamos la lista en un único dataframe
datos<-do.call(rbind,datos_indiv)
library(tidyverse)
library(dplyr)
#Renombramos las variables del rating
names(datos)[2]<-"Rating1"
names(datos)[4]<-"Rating2"
#Eliminamos las partidas repetidas
data_1<-datos %>%distinct()
#Extraemos los nombres de los jugadores
nombre_top100<-c()
for (i in 1:100){
nombre_top100[i]<-substr(nombres[i],
                         regexpr("\\+",nombres[i])+1,
                         nchar(nombres[i])
                         )}
                         #Corregimos errores
nombre_top100<-gsub("\\+", " ", nombre_top100)
nombre_top100[85]="Yifan"
nombre_top100<-gsub("%2C","",nombre_top100)

#Juntamos los nombres y apellidos, separados por coma
nombres_top100<-paste0(apellidos_top100,",", " ",nombre_top100)
#Corregimos errores
nombres_top100[15]<-"Dominguez Perez, Leinier"
nombres_top100[35]<-"Vallejo Pons, Francisco"
nombres_top100[84]<-"Narayanan, SL."
nombres_top100[45]<-"Anton Guijarro, David"
nombres_top100[35]<-"Van Foreest, Jorden"
#Mediante dos vectores detectamos que nombres juegan de blancas
y cuáles de negras
a<-vector(mode="list",length=100)
for(i in 1:length(nombres_top100)){
  a[[i]]<-which(grepl(nombres_top100[i], data_1$`White Player`)==TRUE)
}
data_1<-data_1[sort(unlist(a)),]

b<-vector(mode="list",length=100)
for(i in 1:length(nombres_top100)){
  b[[i]]<-which(grepl(nombres_top100[i], data_1$`Black Player`)==TRUE)
}
#Ordenamos
data_1<-data_1[sort(unlist(b)),]
#Corregimos errores
for (i in 1:nrow(data_1)){
  if(data_1$`White Player`[i]=="Jones, Gawain C B"){
    data_1$`White Player`[i]="Jones, Gawain C"
  }else if
  (data_1$`Black Player`[i]=="Jones, Gawain C B"){
    data_1$`Black Player`[i]="Jones, Gawain C"
  }else if
  (data_1$`White Player`[i]=="Fedoseev, Vladimir3"){
    data_1$`White Player`[i]="Fedoseev, Vladimir"
  }else if
  (data_1$`Black Player`[i]=="Fedoseev, Vladimir3"){
    data_1$`Black Player`[i]="Fedoseev, Vladimir"
  }
}
#Comprobamos que estén todos los nombres
setdiff(nombres_top100, unique(data_1$`White Player`)) #Están todos
setdiff(unique(data_1$`White Player`),nombres_top100)
#save(data_1,file="data_1.RData") #Guardamos los datos
```

A.1.4. Adición de nuevas variables

```
#Vamos a añadir las fechas de nacimiento

#Seleccionamos los jugadores atendiendo al color de las piezas
white<-unique(data_1$`White Player`)
black<-unique(data_1$`Black Player`)
#Los juntamos evitando repetidos
players<-unique(c(white,black))
#Me voy a quedar solo con los apellidos
players2<-gsub(" ","+", players)
players3<-substr(players2,1,regexpr(",",players2)-1)
which(nchar(players3)==0)
#Corregimos errores o anomalías
players4<-replace(players3,which(nchar(players3)==0),
players2[which(nchar(players3)==0)])
players4[44]<-"Wang+Yue"
players4[93]<-"Wang+Hao"
players4[30]<-"narayanan.s.l"
unique(players4)
#Leemos las páginas de cada uno de los jugadores
urls3<-sapply(1:length(players4),function(i)
paste0('https://2700chess.com/players/',tolower(players4[i])))
#Problema con notkevich,sarrau,skrondal,utegaliyev,triggestad,
llobell cortell, cuenca jimenez y marinovic: No tiene pagina

#De los que se puede extraemos los títulos y la información
titulos<-sapply(1:length(urls3), function(i) try(read_html(urls3[i])%>%
                                                   html_nodes("body")%>%
                                                   html_nodes("dl")%>%
                                                   html_nodes("dt")%>%
                                                   html_text2()) )
informacion<-sapply(1:(length(urls3)), function(i) read_html(urls3[i])%>%
                      html_nodes("body")%>%
                      html_nodes("dl")%>%
                      html_nodes("dd")%>%
                      html_text2() )
#Creamos un vector con la fecha de nacimiento
nacimiento<-c()

for (i in 1:length(urls3)){
  #La fecha de nacimiento se encuentra en la lista información
  nacimiento[i]<-informacion[[i]][which(titulos[[i]]=="Born")]


}

#Creamos un vector con el PEAK máximo. Se encuentra en información
maximos<-c()
for (i in 1:length(urls3)){
  if("FIDE Peak"%in%titulos[[i]])
    maximos[i]<-informacion[[i]][which(titulos[[i]]=="FIDE Peak")]
  else{
    maximos[i]=NA
  }

}
#Creamos un dataframe con el nombre de los jugadores, las fechas de
nacimiento y su peak
fechasnac_max<-data.frame("Jugadores"=gsub("\\+", " " ,players4),
"Nacimiento"=substr(nacimiento,1,11),"Peak"=substr(maximos,1,4))

#Volvemos a data_1 y añadimos la fecha de nacimiento de las blancas y negras

#Para el jugador de blancas
data_1$WhiteBirth=rep(NA,nrow(data_1))
data_1$WhiteMaxRating=rep(NA,nrow(data_1))
for(i in 1:nrow(fechasnac_max)){
  for(j in 1:nrow(data_1)){
    if(grepl(fechasnac_max$Jugadores[i],data_1$`White Player`[j],fixed=TRUE)==TRUE){
      data_1$WhiteBirth[j]=fechasnac_max$Nacimiento[i]
```

```
      data_1$WhiteMaxRating[j]=fechasnac_max$Peak[i]
    }
  }
}

#Para el jugador de negras
data_1$BlackBirth=rep(NA,nrow(data_1))
data_1$BlackMaxRating=rep(NA,nrow(data_1))
for(i in 1:nrow(fechasnac_max)){
  for(j in 1:nrow(data_1)){
    if(grepl(fechasnac_max$Jugadores[i],data_1$`Black Player`[j],fixed=TRUE)==TRUE){
      data_1$BlackBirth[j]=fechasnac_max$Nacimiento[i]
      data_1$BlackMaxRating[j]=fechasnac_max$Peak[i]
    }
  }
}

################
#Variación ranking año a año

#Corregimos anomalías
jugadores<-gsub("\\+", " " ,players4)

#Año 2018
 #Leemos la página

ranking_2018<-function(player){
  a<-read_html(paste0("https://www.2700chess.com/players/",tolower(player)))%>%
    html_nodes("script")%>%
    html_text2()

  library(tidyverse)
  #Leemos la tabla con las partidas de la página
  tabla<-tibble(text = a[[10]]) %>%
    mutate(page = 1:length(text),
           text = str_split(text, pattern = "\\[")) %>%
    unnest() %>%
    group_by(page) %>%
    mutate(line = 1:length(text)) %>%
    ungroup(page) %>%
    select(page, line, text)

  #Vemos si hay alguna partida a fecha 1 de enero
  #para extraer el rating del jugador
  ranking2018=c()
  if(any(grepl("2018-01-01",tabla$text,fixed=TRUE)==TRUE)){
    for (j in 1:nrow(tabla)){
      if(grepl("2018-01-01",tabla$text[j],fixed=TRUE)==TRUE){
        ranking2018=substr(tabla$text[j],14,17)
      }
    }
    return(ranking2018)
  }
  else {ranking2018=NA} #Si no, ponemos NA y lo corregimos después
}
#Creamos un dataframe con el ranking2018
rankings_2018<-data.frame(matrix(ncol=2,nrow=0))
for (i in 1:length(jugadores)){
  rankings_2018[i,1]<-jugadores[i]
  rankings_2018[i,2]<-ranking_2018(jugadores[i])
}

#Repetimos el proceso para el año 2019
ranking_2019<-function(player){
  a<-read_html(paste0("https://www.2700chess.com/players/",tolower(player)))%>%
    html_nodes("script")%>%
    html_text2()


  library(tidyverse)
```

```
  tabla<-tibble(text = a[[10]]) %>%
    mutate(page = 1:length(text),
           text = str_split(text, pattern = "\\[")) %>%
    unnest() %>%
    group_by(page) %>%
    mutate(line = 1:length(text)) %>%
    ungroup(page) %>%
    select(page, line, text)


  ranking2019=c()
  if(any(grepl("2019-01-01",tabla$text,fixed=TRUE)==TRUE)){
    for (j in 1:nrow(tabla)){
      if(grepl("2019-01-01",tabla$text[j],fixed=TRUE)==TRUE){
        ranking2019=substr(tabla$text[j],14,17)
      }}

    return(ranking2019)
  }
  else{ranking2019=NA}
}

#Creo un data.frame para todos
rankings_2019<-data.frame(matrix(ncol=2,nrow=0))
for (i in 1:length(jugadores)){
  rankings_2019[i,1]<-jugadores[i]
  rankings_2019[i,2]<-ranking_2019(jugadores[i])
}

#Repetimos el proceso para el año 2020
ranking_2020<-function(player){
  a<-read_html(paste0("https://www.2700chess.com/players/",tolower(player)))%>%
    html_nodes("script")%>%
    html_text2()


  library(tidyverse)

  tabla<-tibble(text = a[[10]]) %>%
    mutate(page = 1:length(text),
           text = str_split(text, pattern = "\\[")) %>%
    unnest() %>%
    group_by(page) %>%
    mutate(line = 1:length(text)) %>%
    ungroup(page) %>%
    select(page, line, text)


  ranking2020=c()
  if(any(grepl("2020-01-01",tabla$text,fixed=TRUE)==TRUE)){
    for (j in 1:nrow(tabla)){
      if(grepl("2020-01-01",tabla$text[j],fixed=TRUE)==TRUE){
        ranking2020=substr(tabla$text[j],14,17)
      }
    }
    return(ranking2020)
  }
  else{ranking2020=NA}
}

#Creo un data.frame para todos

rankings_2020<-data.frame(matrix(ncol=2,nrow=0))
for (i in 1:length(jugadores)){
  rankings_2020[i,1]<-jugadores[i]
  rankings_2020[i,2]<-ranking_2020(jugadores[i])

}
```

```
#Creamos un dataframe con los ratings de cada año
data2<-data.frame("Jugador"=jugadores,"2018"=rankings_2018[,2],
"2019"=rankings_2019[,2],"2020"=rankings_2020[,2])
data2$X2018[data2$X2018=="null"]<-NA
data2$X2019[data2$X2019=="null"]<-NA
data2$X2020[data2$X2020=="null"]<-NA

#Añadimos las variaciones del ranking al cjto de datos
data_1$WhiteRating2018<-rep(NA,nrow(data_1))
for (i in 1:length(data_1$`White Player`)){
  for(j in 1:length(data2$Jugador))
    if(grepl(data2$Jugador[j],data_1$`White Player`[i])){
      data_1$WhiteRating2018[i]<-data2$X2018[j]
    }
}
data_1$BlackRating2018<-rep(NA,nrow(data_1))
for (i in 1:length(data_1$`Black Player`)){
  for(j in 1:length(data2$Jugador))
    if(grepl(data2$Jugador[j],data_1$`Black Player`[i])){
      data_1$BlackRating2018[i]<-data2$X2018[j]
    }
}

data_1$WhiteRating2019<-rep(NA,nrow(data_1))
for (i in 1:length(data_1$`White Player`)){
  for(j in 1:length(data2$Jugador))
    if(grepl(data2$Jugador[j],data_1$`White Player`[i])){
      data_1$WhiteRating2019[i]<-data2$X2019[j]
    }
}

data_1$BlackRating2019<-rep(NA,nrow(data_1))
for (i in 1:length(data_1$`Black Player`)){
  for(j in 1:length(data2$Jugador))
    if(grepl(data2$Jugador[j],data_1$`Black Player`[i])){
      data_1$BlackRating2019[i]<-data2$X2019[j]
    }
}

data_1$WhiteRating2020<-rep(NA,nrow(data_1))
for (i in 1:length(data_1$`White Player`)){
  for(j in 1:length(data2$Jugador))
    if(grepl(data2$Jugador[j],data_1$`White Player`[i])){
      data_1$WhiteRating2020[i]<-data2$X2020[j]
    }
}
data_1$BlackRating2020<-rep(NA,nrow(data_1))
for (i in 1:length(data_1$`Black Player`)){
  for(j in 1:length(data2$Jugador))
    if(grepl(data2$Jugador[j],data_1$`Black Player`[i])){
      data_1$BlackRating2020[i]<-data2$X2020[j]
    }
}

#Creamos un vector de variación del rating para cada color de piezas
data_1$BlackRatingChange<-rep(NA,nrow(data_1))
data_1$WhiteRatingChange<-rep(NA,nrow(data_1))
#Convertimos a numérico los ratings
data_1$Rating1<-as.numeric(data_1$Rating1)
data_1$Rating2<-as.numeric(data_1$Rating2)
data_1$BlackRating2020<-as.numeric(data_1$BlackRating2020)
data_1$BlackRating2019<-as.numeric(data_1$BlackRating2019)
data_1$BlackRating2018<-as.numeric(data_1$BlackRating2018)
data_1$WhiteRating2020<-as.numeric(data_1$WhiteRating2020)
data_1$WhiteRating2019<-as.numeric(data_1$WhiteRating2019)
data_1$WhiteRating2018<-as.numeric(data_1$WhiteRating2018)
#Variación
for(i in 1:nrow(data_1)){
  if((data_1$Year[i]==2021)==TRUE){
    data_1$BlackRatingChange[i]=data_1$Rating2[i]-data_1$BlackRating2020[i]
```

```
    data_1$WhiteRatingChange[i]=data_1$Rating1[i]-data_1$WhiteRating2020[i]
  }
  else if ((data_1$Year[i]==2020)==TRUE){
    data_1$BlackRatingChange[i]=data_1$BlackRating2020[i]-data_1$BlackRating2019[i]
    data_1$WhiteRatingChange[i]=data_1$WhiteRating2020[i]-data_1$WhiteRating2019[i]
  }
  else {
    data_1$BlackRatingChange[i]=data_1$BlackRating2019[i]-data_1$BlackRating2018[i]
    data_1$WhiteRatingChange[i]=data_1$WhiteRating2019[i]-data_1$WhiteRating2018[i]
  }
}



#Calculamos el Head to head

#Corregimos anomalías en los nombres
#para poder trabajar con ellos
apellido_blanco<-substr(data_1$`White Player`,1,regexpr(",",data_1$`White Player`)-1)

nombre_blanco<-substr(data_1$`White Player`,regexpr(",",data_1$`White Player`)+2,
nchar(data_1$`White Player`))
apellido_negro<-substr(data_1$`Black Player`,1,regexpr(",",data_1$`Black Player`)-1)

nombre_negro<-substr(data_1$`Black Player`,regexpr(",",data_1$`Black Player`)+2,
nchar(data_1$`Black Player`))

#Creamos un dataframe con los nombres y los apellidos
d<-data.frame("AB"=apellido_blanco,"NB"=nombre_blanco,
"AN"=apellido_negro,"NN"=nombre_negro)
d<-unique(d)


#Con todos los nombres

a<-vector("list",length=nrow(d))
#Creamos un vector resultado y leemos las páginas
resultado<-vector("list",length=nrow(d))
for (i in 1:nrow(d)){
  a[[i]]<-read_html(paste("https://2700chess.com/games?search=",d$AB[i],"%2C+",
  d$NB[i],"+vs+",d$AN[i],"%2C+",d$NN[i],sep=""))%>%
    html_nodes("body")%>%
    html_nodes("div")%>%
    html_nodes("p")

}

#Extraemos los resultados que hubo en las partidas.
for(i in 1:nrow(d)){
  if(any(grepl("Total Score", a[[i]],fixed=TRUE)==TRUE)){
    resultado[[i]]<-a[[i]][which(grepl("Total Score", a[[i]],fixed=TRUE)==TRUE)]%>%html_text2()
  } else {resultado[[i]]<-a[[i]][which(grepl("lead", a[[i]],fixed=TRUE)==TRUE)]%>%html_text2()
  }
}

resultado<-do.call(rbind,resultado)
library(stringr)

#Extraemos los números solo
Numextract <- function(string){
  unlist(regmatches(string,gregexpr("[[:digit:]]+\\.*[[:digit:]]*",string)))
}
#Para extraer los resultados obtenemos el resultado total,
#el número de victorias del jugador y el de empates
resultados_partidos<-vector("list",length=nrow(d))
# process string
for(i in 1:nrow(d)){
resultados_partidos[[i]]<-as.numeric(Numextract(resultado[i]))
}
ganador<-c()
```

```
for(i in 1:nrow(d)){
  if(grepl("Total Score", resultado[i],fixed=TRUE)==TRUE){
  if(grepl("beat", resultado[i],fixed=TRUE)==TRUE){
    ganador[i]<-substr(resultado[i],regexpr("\\)",resultado[i])+2,
    regexpr("beat",resultado[i])-2)
  }else
  {ganador[i]<-substr(resultado[i],regexpr("\\)",resultado[i])+2,
  regexpr("tied",resultado[i])-2)}
  }else
  {if(grepl("beat", resultado[i],fixed=TRUE)==TRUE){
    ganador[i]<-substr(resultado[i],regexpr("\\)",resultado[i])+2,
    regexpr("beat",resultado[i])-2)
  }else if
    (grepl("tied", resultado[i],fixed=TRUE)==TRUE)
  {ganador[i]<-substr(resultado[i],regexpr("\\)",resultado[i])+2,
  regexpr("tied",resultado[i])-2)}}
}

grepl(ganador[1],paste0(d$AB,", ",d$NB)[1])
head_to_head<-c()
for (i in 1:nrow(d)){
  if(grepl(ganador[i],paste0(d$AB,", ",d$NB)[i])==TRUE)
  {head_to_head[i]=(resultados_partidos[[i]][1]+
  resultados_partidos[[i]][3]/2)/(resultados_partidos[[i]][1]+resultados_partidos[[i]][2]+
  resultados_partidos[[i]][3])} else
  {head_to_head[i]=(resultados_partidos[[i]][2]+
  resultados_partidos[[i]][3]/2)/(resultados_partidos[[i]][1]+resultados_partidos[[i]][2]+
  resultados_partidos[[i]][3])}
}
#Metemos la variable head to head
d$head_to_head<-head_to_head

nombre_blanco<-paste0(d$AB,", ",d$NB)
nombre_negro<-paste0(d$AN,", ",d$NN)
#Mramos si el jugador juega de blancas o de negras, para meter el head to head
x<-data.frame("Nombre_blanco"=nombre_blanco,"Nombre_negro"=nombre_negro,"H2H"=head_to_head)
y<-data.frame("WP"=data_1$`White Player`,"BP"=data_1$`Black Player`)

y$"H2H"<-rep(0,nrow(y))
for(i in 1:nrow(y)){
  for(j in 1:nrow(x)){
    if(y$"WP"[i]==x$"Nombre_blanco"[j] && y$"BP"[i]==x$"Nombre_negro"[j] )
    y$"H2H"[i]<-x$H2H[j]
  }}
#La añadimos al dataframe
data_1$"H2H"<-y$H2H

#save(data_1,file="data_1.RData")

#install.packages("DescTools")
library("DescTools")
nombres<-c()
for (i in 1:length(datos_indiv)){
  nombres[i]<-Mode(datos_indiv[[i]]$`White Player`)
}
names(datos_indiv)<- nombres
#Damos el nombre del jugador a cada ítem de la lista
#Guardamos
save(datos_indiv,file="top100.RData")


#Calculamos el nivel de agresividad y lucha de cada jugador
lucha<-c()
agresividad<-c()
for(i in 1:length(datos_indiv)){
  agresividad[i]<-length(datos_indiv[[i]]$Result[datos_indiv[[i]]$Result!="1/2-1/2"])/
    length(datos_indiv[[i]]$Result)
  lucha[i]<-mean(datos_indiv[[i]]$Moves)
}
agr_luch<-data.frame("Nombres"=nombres_top100,"Lucha"=lucha,"Agresividad"=agresividad)
```

```
#Lo metemos en el data_1
data_1$LuchaWP<-rep(0,nrow(data_1))
data_1$AgresividadWP<-rep(0,nrow(data_1))
for(i in 1:nrow(data_1)){
  for(j in 1:nrow(agr_luch)){
    if(data_1$`White Player`[i]==agr_luch$"Nombres"[j]){
      data_1$LuchaWP[i]<-agr_luch$Lucha[j]
      data_1$AgresividadWP[i]<-agr_luch$Agresividad[j]}
  }}

data_1$LuchaBP<-rep(0,nrow(data_1))
data_1$AgresividadBP<-rep(0,nrow(data_1))
for(i in 1:nrow(data_1)){
  for(j in 1:nrow(agr_luch)){
    if(data_1$`Black Player`[i]==agr_luch$"Nombres"[j]){
      data_1$LuchaBP[i]<-agr_luch$Lucha[j]
    data_1$AgresividadBP[i]<-agr_luch$Agresividad[j]}
  }}

#save(data_1,file="data_1.RData")

#Edad a la hora de la partida

#Corregimos errores a mano
for(i in 1:nrow(data_1)){
  if(isTRUE(data_1$`White Player`[i]=="Vidit, Santosh Gujrathi" )){
    data_1$WhiteBirth[i]<-"24 Oct 1994"
  } else if (isTRUE(data_1$`Black Player`[i]=="Vidit, Santosh Gujrathi" )){
    data_1$BlackBirth[i]<-"24 Oct 1994"
  } else if (isTRUE(data_1$`White Player`[i]=="Narayanan, SL." )){
    data_1$WhiteBirth[i]<-"10 Jan 1998"
  } else if (isTRUE(data_1$`Black Player`[i]=="Narayanan, SL." )){
    data_1$BlackBirth[i]<-"10 Jan 1998"
  } else if (isTRUE(data_1$`White Player`[i]=="Sevian, Samuel" )){
    data_1$WhiteBirth[i]<-"26 Dec 2000"
  } else if (isTRUE(data_1$`Black Player`[i]=="Sevian, Samuel" )){
    data_1$BlackBirth[i]<-"26 Dec 2000"
  } else if (isTRUE(data_1$`White Player`[i]=="Melkumyan, Hrant" )){
    data_1$WhiteBirth[i]<-"30 Apr 1989"
  } else if (isTRUE(data_1$`Black Player`[i]=="Melkumyan, Hrant" )){
    data_1$BlackBirth[i]<-"30 Apr 1989"
  } else if (isTRUE(data_1$`White Player`[i]=="Cheparinov, Ivan" )){
    data_1$WhiteBirth[i]<-"26 Nov 1986"
  } else if (isTRUE(data_1$`Black Player`[i]=="Cheparinov, Ivan" )){
    data_1$BlackBirth[i]<-"26 Nov 1986"
  } else if (isTRUE(data_1$`White Player`[i]=="Ni, Hua" )){
    data_1$WhiteBirth[i]<-"31 May 1983"
  } else if (isTRUE(data_1$`Black Player`[i]=="Ni, Hua" )){
    data_1$BlackBirth[i]<-"31 May 1983"
  }
}


for(i in 1:nrow(data_1)){
  if(isTRUE(data_1$`White Player`[i]=="Grandelius, Nils" )){
    data_1$WhiteMaxRating[i]<-"2694"
  } else if (isTRUE(data_1$`Black Player`[i]=="Grandelius, Nils" )){
    data_1$BlackMaxRating[i]<-"2694"
  } else if (isTRUE(data_1$`White Player`[i]=="Salem, A.R. Saleh" )){
    data_1$WhiteMaxRating[i]<-"2690"
  } else if (isTRUE(data_1$`Black Player`[i]=="Salem, A.R. Saleh" )){
    data_1$BlackMaxRating[i]<-"2690"
  } else if (isTRUE(data_1$`White Player`[i]=="Narayanan, SL." )){
    data_1$WhiteMaxRating[i]<-"2658"
  } else if (isTRUE(data_1$`Black Player`[i]=="Narayanan, SL." )){
    data_1$BlackMaxRating[i]<-"2658"
  } else if (isTRUE(data_1$`White Player`[i]=="Predke, Alexandr" )){
    data_1$WhiteMaxRating[i]<-"2696"
  } else if (isTRUE(data_1$`Black Player`[i]=="Predke, Alexandr" )){
```

```
  data_1$BlackMaxRating[i]<-"2696"
} else if (isTRUE(data_1$`White Player`[i]=="Sarana, Alexey" )){
  data_1$WhiteMaxRating[i]<-"2664"
} else if (isTRUE(data_1$`Black Player`[i]=="Sarana, Alexey" )){
  data_1$BlackMaxRating[i]<-"2664"
} else if (isTRUE(data_1$`White Player`[i]=="Keymer, Vincent" )){
  data_1$WhiteMaxRating[i]<-"2639"
} else if (isTRUE(data_1$`Black Player`[i]=="Keymer, Vincent" )){
  data_1$BlackMaxRating[i]<-"2639"
} else if (isTRUE(data_1$`White Player`[i]=="Swiercz, Dariusz" )){
  data_1$WhiteMaxRating[i]<-"2670"
} else if (isTRUE(data_1$`Black Player`[i]=="Swiercz, Dariusz" )){
  data_1$BlackMaxRating[i]<-"2670"
}else if (isTRUE(data_1$`White Player`[i]=="Oparin, Grigoriy" )){
  data_1$WhiteMaxRating[i]<-"2660"
} else if (isTRUE(data_1$`Black Player`[i]=="Oparin, Grigoriy" )){
  data_1$BlackMaxRating[i]<-"2660"
} else if (isTRUE(data_1$`White Player`[i]=="Deac, Bogdan-Daniel" )){
  data_1$WhiteMaxRating[i]<-"2651"
} else if (isTRUE(data_1$`Black Player`[i]=="Deac, Bogdan-Daniel" )){
  data_1$BlackMaxRating[i]<-"2651"
} else if (isTRUE(data_1$`White Player`[i]=="Nihal, Sarin" )){
  data_1$WhiteMaxRating[i]<-"2655"
} else if (isTRUE(data_1$`Black Player`[i]=="Nihal, Sarin" )){
  data_1$BlackMaxRating[i]<-"2655"
} else if (isTRUE(data_1$`White Player`[i]=="Sevian, Samuel" )){
  data_1$WhiteMaxRating[i]<-"2677"
} else if (isTRUE(data_1$`Black Player`[i]=="Sevian, Samuel" )){
  data_1$BlackMaxRating[i]<-"2677"
} else if (isTRUE(data_1$`White Player`[i]=="Cori, Jorge" )){
  data_1$WhiteMaxRating[i]<-"2689"
} else if (isTRUE(data_1$`Black Player`[i]=="Cori, Jorge" )){
  data_1$BlackMaxRating[i]<-"2689"
} else if (isTRUE(data_1$`White Player`[i]=="Shevchenko, Kirill" )){
  data_1$WhiteMaxRating[i]<-"2647"
} else if (isTRUE(data_1$`Black Player`[i]=="Shevchenko, Kirill" )){
  data_1$BlackMaxRating[i]<-"2647"
} else if (isTRUE(data_1$`White Player`[i]=="Sjugirov, Sanan" )){
  data_1$WhiteMaxRating[i]<-"2678"
} else if (isTRUE(data_1$`Black Player`[i]=="Sjugirov, Sanan" )){
  data_1$BlackMaxRating[i]<-"2678"
} else if (isTRUE(data_1$`White Player`[i]=="Guseinov, Gadir" )){
  data_1$WhiteMaxRating[i]<-"2667"
} else if (isTRUE(data_1$`Black Player`[i]=="Guseinov, Gadir" )){
  data_1$BlackMaxRating[i]<-"2667"
} else if (isTRUE(data_1$`White Player`[i]=="Ponkratov, Pavel" )){
  data_1$WhiteMaxRating[i]<-"2662"
} else if (isTRUE(data_1$`Black Player`[i]=="Ponkratov, Pavel" )){
  data_1$BlackMaxRating[i]<-"2662"
} else if (isTRUE(data_1$`White Player`[i]=="Hou, Yifan" )){
  data_1$WhiteMaxRating[i]<-"2686"
} else if (isTRUE(data_1$`Black Player`[i]=="Hou, Yifan" )){
  data_1$BlackMaxRating[i]<-"2686"
} else if (isTRUE(data_1$`White Player`[i]=="Demchenko, Anton" )){
  data_1$WhiteMaxRating[i]<-"2679"
} else if (isTRUE(data_1$`Black Player`[i]=="Demchenko, Anton" )){
  data_1$BlackMaxRating[i]<-"2679"
} else if (isTRUE(data_1$`White Player`[i]=="Robson, Ray" )){
  data_1$WhiteMaxRating[i]<-"2682"
} else if (isTRUE(data_1$`Black Player`[i]=="Robson, Ray" )){
  data_1$BlackMaxRating[i]<-"2682"
} else if (isTRUE(data_1$`White Player`[i]=="Kuzubov, Yuriy" )){
  data_1$WhiteMaxRating[i]<-"2699"
} else if (isTRUE(data_1$`Black Player`[i]=="Kuzubov, Yuriy" )){
  data_1$BlackMaxRating[i]<-"2699"
} else if (isTRUE(data_1$`White Player`[i]=="Gledura, Benjamin" )){
  data_1$WhiteMaxRating[i]<-"2654"
} else if (isTRUE(data_1$`Black Player`[i]=="Gledura, Benjamin" )){
  data_1$BlackMaxRating[i]<-"2654"
} else if (isTRUE(data_1$`White Player`[i]=="Nyzhnyk, Illya" )){
```

```
    data_1$WhiteMaxRating[i]<-"2676"
  } else if (isTRUE(data_1$`Black Player`[i]=="Nyzhnyk, Illya" )){
    data_1$BlackMaxRating[i]<-"2676"
  }
}
sort(unique(data_1$`White Player`))
tail(data_1$torneo)
#Nos quedamos solo con el nombre del torneo
data_1$Nombretorneo<-sapply(1:length(data_1$torneo),
function(i) substr(data_1$torneo[i],regexpr("-",data_1$torneo[i])+1,nchar(data_1$torneo[i])))
#Creamos otra variable con la ronda
data_1$ronda<-substr(data_1$torneo,1,regexpr("-",data_1$torneo)-1)
#Calculamos la edad restándole a la fecha de la partida la del nacimiento del jugador
Sys.setlocale("LC_TIME", "C")
age_blanco<-c()
for(i in 1:nrow(data_1)){
age_blanco[i]<-as.numeric(try(difftime(data_1$fecha[,1][i],
as.Date(data_1$WhiteBirth[i],format="%d %b %Y"),unit="weeks")/52.25))
}
data_1$Edad_WP<-floor(age_blanco)
age_negro<-c()
for(i in 1:nrow(data_1)){
  age_negro[i]<-as.numeric(try(difftime(data_1$fecha[,1][i],
  as.Date(data_1$BlackBirth[i],format="%d %b %Y"),unit="weeks")/52.25))
}
data_1$Edad_BP<-floor(age_negro)

#save(data_1,file="data_1.RData")




#Correción rating carlsen y So
valores<-which(is.na(data_1$Rating1)==TRUE)
data_1$Rating1[valores]<-c(2855,2778,2885,2774,2699,2774,2699,2765,2782,2765
,2763,2763,2763,2699)
data_1$Rating2[valores]<-c(2778,2885,2778,2699,2774,2699,2774,2782,2765,2782,
2763,2763,2763,2736)
which(is.na(data_1$Rating1)==TRUE)
#Variación
for(i in valores){
  if((data_1$Year[i]==2021)==TRUE){
    data_1$BlackRatingChange[i]=data_1$Rating2[i]-data_1$BlackRating2020[i]
    data_1$WhiteRatingChange[i]=data_1$Rating1[i]-data_1$WhiteRating2020[i]
  }
  else if ((data_1$Year[i]==2020)==TRUE){
    data_1$BlackRatingChange[i]=data_1$BlackRating2020[i]-data_1$BlackRating2019[i]
    data_1$WhiteRatingChange[i]=data_1$WhiteRating2020[i]-data_1$WhiteRating2019[i]
  }
  else {
    data_1$BlackRatingChange[i]=data_1$BlackRating2019[i]-data_1$BlackRating2018[i]
    data_1$WhiteRatingChange[i]=data_1$WhiteRating2019[i]-data_1$WhiteRating2018[i]
  }
}

#Añadimos los best ratigs de los que faltan:

nombres_faltan<-c("Deac, Bogdan-Daniel","Demchenko, Anton" ,   "Gledura, Benjamin" ,
"Hou, Yifan" ,"Keymer, Vincent"   ,  "Kuzubov, Yuriy"   ,  "Melkumyan, Hrant"   ,
"Ni, Hua"   , "Nihal, Sarin"     ,   "Nyzhnyk, Illya"   ,   "Oparin, Grigoriy" ,
"Predke, Alexandr"  , "Robson, Ray"       ,  "Sarana, Alexey"    ,  "Sevian, Samuel",
"Shevchenko, Kirill", "Sjugirov, Sanan", "Tari, Aryan", "Swiercz, Dariusz")

max_nombres_faltan<-c(2651, 2679, 2656, 2686, 2663, 2695, 2678, 2724,
                      2661,2676, 2680,2684,
                      2680,2666,2684,2655,  2680,2652,2670)
for(i in 1:length(nombres_faltan)){
    posiciones_blanco<- which(data_1$`White Player`==nombres_faltan[i])
    data_1$WhiteMaxRating[posiciones_blanco]=max_nombres_faltan[i]
```

```
    posiciones_negro<-which(data_1$`Black Player`==nombres_faltan[i])
    data_1$BlackMaxRating[posiciones_negro]=max_nombres_faltan[i]
}



  for(i in 1:length(datos_indiv)){
    names(datos_indiv[[i]])[2]<-"Rating_WP"
    names(datos_indiv[[i]])[4]<-"Rating_BP"
  }

datos_indiv[[1]]

#Corregimos errores
nombres_falta_R2018<-unique(data_1$`White Player`[ which(is.na(data_1$WhiteRating2018))])
rating_R2018<-c(2593,2549,2629,2571,2570,2573,2634,2652,
                2595,2590,2562,2408,
                2607,2649,2549,2614,
                2658,2538,2643,2652,
                2646,2640,2645,2635,
                2623,2605,2645,2621)
nombres_falta_N_R2018<-unique(data_1$`Black Player`[ which(is.na(data_1$BlackRating2018))])
rating_R_N_2018<-rating_R2018[c(2,1,4,22,5,6,7,8,9,3,17,11,16,10,18,
14,23,15,20,25,24,27,12,21,13,19,28,26)]

nombres_falta_R2019<-unique(data_1$`White Player`[ which(is.na(data_1$WhiteRating2019))])
rating_R2019<-c(2625,2618,2612,2584,2642,2593,2611,2637,
                2630,2500,2613,2649,
                2603,2647,2686,2539,
                2635,2624,2615)
nombres_falta_N_R2019<-unique(data_1$`Black Player`[ which(is.na(data_1$BlackRating2019))])
rating_R_N_2019<-rating_R2019[c(2,1,4,5,6,7,3,15,9,14,8,16,12,13,17,10,11,19,18)]

nombres_falta_R2020<-unique(data_1$`White Player`[ which(is.na(data_1$WhiteRating2020))])
rating_R_2020<-c(2629,2644,2637,2527,
                 2626,2652,2553,2654,
                 2641,2656,2654)
nombres_falta_N_R2020<-unique(data_1$`Black Player`[ which(is.na(data_1$BlackRating2020))])
rating_R_N_2020<-rating_R2019[c(1,3,10,2,6,8,7,5,9,4,11)]

for(i in 1:length(nombres_falta_R2018)){
  posiciones_blanco<- which(data_1$`White Player`==nombres_falta_R2018[i])
  data_1$WhiteRating2018[posiciones_blanco]=rating_R2018[i]
  posiciones_negro<- which(data_1$`Black Player`==nombres_falta_N_R2018[i])
  data_1$BlackRating2018[posiciones_negro]=rating_R_N_2018[i]
}
for(i in 1:length(nombres_falta_R2019)){
  posiciones_blanco<- which(data_1$`White Player`==nombres_falta_R2019[i])
  data_1$WhiteRating2019[posiciones_blanco]=rating_R2019[i]
  posiciones_negro<- which(data_1$`Black Player`==nombres_falta_N_R2019[i])
  data_1$BlackRating2019[posiciones_negro]=rating_R_N_2019[i]
}

for(i in 1:length(nombres_falta_R2020)){
  posiciones_blanco<- which(data_1$`White Player`==nombres_falta_R2020[i])
  data_1$WhiteRating2020[posiciones_blanco]=rating_R_2020[i]
  posiciones_negro<- which(data_1$`Black Player`==nombres_falta_N_R2020[i])
  data_1$BlackRating2020[posiciones_negro]=rating_R_N_2020[i]
}


valores1<-which(is.na(data_1$WhiteRatingChange)==TRUE)
for(i in valores1){
  if((data_1$Year[i]==2021)==TRUE){

    data_1$WhiteRatingChange[i]=data_1$Rating1[i]-data_1$WhiteRating2020[i]
  }
  else if ((data_1$Year[i]==2020)==TRUE){
```

```
    data_1$WhiteRatingChange[i]=data_1$WhiteRating2020[i]-data_1$WhiteRating2019[i]
  }
  else {

    data_1$WhiteRatingChange[i]=data_1$WhiteRating2019[i]-data_1$WhiteRating2018[i]
  }
}
valores2<-which(is.na(data_1$BlackRatingChange)==TRUE)
for(i in valores2){
  if((data_1$Year[i]==2021)==TRUE){

    data_1$BlackRatingChange[i]=data_1$Rating2[i]-data_1$BlackRating2020[i]
  }
  else if ((data_1$Year[i]==2020)==TRUE){

    data_1$BlackRatingChange[i]=data_1$BlackRating2020[i]-data_1$BlackRating2019[i]
  }
  else {

    data_1$BlackRatingChange[i]=data_1$BlackRating2019[i]-data_1$BlackRating2018[i]
  }
}


#Añadimos el estado de forma
data_1$Num_partida<-1:nrow(data_1)
for(i in 1:nrow(data_1)){
  data_1$`White Player`[i]
  data_1$`Black Player`[i]
  jugador_blanco<-data_1[which(data_1$`White Player`==data_1$`White Player`[i] |
                                 data_1$`Black Player`==data_1$`White Player`[i]),]

  j<-which(jugador_blanco$Num_partida==i)
  #Hacemos uso de las dos fórmula expuestas en el trabajo para cada uno de los EF
  if(isTRUE(nrow(jugador_blanco)-j)>=10){
    rating_rival<-c()
    ganador<-c()
    empates<-c()
    for(z in 1:10){
      if(isTRUE(jugador_blanco$`White Player`[z+j-1]!=jugador_blanco[i,1])){
        rating_rival[z]<-jugador_blanco$Rating1[z+j-1]
      } else
      {rating_rival[z]<-jugador_blanco$Rating2[z+j-1]}
    }
    for(z  in 1:10){
      if(isTRUE(jugador_blanco$`White Player`[z+j-1]==jugador_blanco[i,1] &&
      jugador_blanco$Result[z+j-1]=="1-0")){ganador[z]<-"Sí"} else if
      (isTRUE(jugador_blanco$`Black Player`[z+j-1]==jugador_blanco[i,1] &&
      jugador_blanco$Result[z+j-1]=="0-1"))
      {ganador[z]<-"Sí"} else
      {ganador[z]<-"No"}
    }
    for(z  in 1:10){
      if(isTRUE(jugador_blanco$`White Player`[z+j-1]==jugador_blanco[i,1] &&
      jugador_blanco$Result[z+j-1]=="1/2-1/2")){empates[z]<-"Sí"} else if
      (isTRUE(jugador_blanco$`Black Player`[z+j-1]==jugador_blanco[i,1] &&
      jugador_blanco$Result[z+j-1]=="1/2-1/2"))
      {empates[z]<-"Sí"} else
      {empates[z]<-"No"}
    }

    data_1$EstadoFormaWP[i]<-(sum(rating_rival)+400*(sum(ganador=="Sí")-
    (1-sum(ganador=="Sí")-sum(empates=="Sí"))))/10
  } else{
    rating_rival<-c()
    ganador<-c()
    empates<-c()
    for(z in 1:(nrow(jugador_blanco)-j+1)){
      if(isTRUE(jugador_blanco$`White Player`[z+j-1]!=jugador_blanco[i,1])){
        rating_rival[z]<-jugador_blanco$Rating1[z+j-1]
```

```
      } else
      {rating_rival[z]<-jugador_blanco$Rating2[z+j-1]}
    }
    for(z  in 1:(nrow(jugador_blanco)-j+1)){
      if(isTRUE(jugador_blanco$`White Player`[z+j-1]==jugador_blanco[i,1] &&
      jugador_blanco$Result[z+j-1]=="1-0")){ganador[z]<-"Sí"} else if
      (isTRUE(jugador_blanco$`Black Player`[z+j-1]==jugador_blanco[i,1] &&
      jugador_blanco$Result[z+j-1]=="0-1"))
      {ganador[z]<-"Sí"} else
      {ganador[z]<-"No"}
    }
    for(z  in 1:(nrow(jugador_blanco)-j+1)){
      if(isTRUE(jugador_blanco$`White Player`[z+j-1]==jugador_blanco[i,1] &&
      jugador_blanco$Result[z+j-1]=="1/2-1/2")){empates[z]<-"Sí"} else if
      (isTRUE(jugador_blanco$`Black Player`[z+j-1]==jugador_blanco[i,1] &&
      jugador_blanco$Result[z+j-1]=="1/2-1/2"))
      {empates[z]<-"Sí"} else
      {empates[z]<-"No"}
    }
    data_1$EstadoFormaWP[i]<-(sum(rating_rival)+400*(sum(ganador=="Sí")-
    (1-sum(ganador=="Sí")-sum(empates=="Sí"))))/(nrow(jugador_blanco)-j+1)
  }

}


for(i in 1:nrow(data_1)){
  data_1$`White Player`[i]
  data_1$`Black Player`[i]
  jugador_negro<-data_1[which(data_1$`White Player`==data_1$`Black Player`[i] |
                                data_1$`Black Player`==data_1$`Black Player`[i]),]

  j<-which(jugador_negro$Num_partida==i)

  if(isTRUE(nrow(jugador_negro)-j)>=10){
    rating_rival<-c()
    ganador<-c()
    empates<-c()
     for(z in 1:10){

      if(isTRUE(jugador_negro$`Black Player`[z+j-1]!=jugador_negro[i,3])){
        rating_rival[z]<-jugador_negro$Rating2[z+j-1]
      } else
      {rating_rival[z]<-jugador_negro$Rating1[z+j-1]}
    }
    for(z  in 1:10){
      if(isTRUE(jugador_negro$`White Player`[z+j-1]==jugador_negro[i,3] &&
      jugador_negro$Result[z+j-1]=="1-0")){ganador[z]<-"Sí"} else if
      (isTRUE(jugador_negro$`Black Player`[z+j-1]==jugador_negro[i,3] &&
      jugador_negro$Result[z+j-1]=="0-1"))
      {ganador[z]<-"Sí"} else
      {ganador[z]<-"No"}
    }
    for(z  in 1:10){
      if(isTRUE(jugador_negro$`White Player`[z+j-1]==jugador_negro[i,3] &&
      jugador_negro$Result[z+j-1]=="1/2-1/2")){empates[z]<-"Sí"} else if
      (isTRUE(jugador_negro$`Black Player`[z+j-1]==jugador_negro[i,3] &&
      jugador_negro$Result[z+j-1]=="1/2-1/2"))
      {empates[z]<-"Sí"} else
      {empates[z]<-"No"}
    }
    data_1$EstadoFormaBP[i]<-(sum(rating_rival)+400*(sum(ganador=="Sí")-
    (1-sum(ganador=="Sí")-sum(empates=="Sí"))))/10
  } else{
    rating_rival<-c()
    ganador<-c()
    empates<-c()
    for(z in 1:(nrow(jugador_negro)-j+1)){
      if(isTRUE(jugador_negro$`BlackPlayer`[z+j-1]!=jugador_negro[i,3])){
        rating_rival[z]<-jugador_negro$Rating2[z+j-1]
```

```
      } else
      {rating_rival[z]<-jugador_negro$Rating1[z+j-1]}
    }
    for(z  in 1:(nrow(jugador_negro)-j+1)){
      if(isTRUE(jugador_negro$`White Player`[z+j-1]==jugador_negro[i,3] &&
      jugador_negro$Result[z+j-1]=="1-0")){ganador[z]<-"Sí"} else if
      (isTRUE(jugador_negro$`Black Player`[z+j-1]==jugador_negro[i,3] &&
      jugador_negro$Result[z+j-1]=="0-1"))
      {ganador[z]<-"Sí"} else
      {ganador[z]<-"No"}
    }
    for(z  in 1:(nrow(jugador_negro)-j+1)){
      if(isTRUE(jugador_negro$`White Player`[z+j-1]==jugador_negro[i,3] &&
      jugador_negro$Result[z+j-1]=="1/2-1/2")){empates[z]<-"Sí"} else if
      (isTRUE(jugador_negro$`Black Player`[z+j-1]==jugador_negro[i,3] &&
      jugador_negro$Result[z+j-1]=="1/2-1/2"))
      {empates[z]<-"Sí"} else
      {empates[z]<-"No"}
    }
    data_1$EstadoFormaBP[i]<-(sum(rating_rival)+400*(sum(ganador=="Sí")-
    (1-sum(ganador=="Sí")-sum(empates=="Sí"))))/(nrow(jugador_negro)-j+1)
  }

}

save(data_1,file="data_1.RData")
```

A.2. Análisis descriptivo

A.2.1. Análisis descriptivo variables cualitativas vs Resultado

```
library(tidyverse)
library(janitor)
library(jpeg)
library(ggplot2)
library(grid)
data_1ord$WhiteMaxRating<-as.numeric(data_1ord$WhiteMaxRating)
data_1ord$BlackMaxRating<-as.numeric(data_1ord$BlackMaxRating)
data_1ord<-data_1ord %>%
  mutate_if(sapply(data_1ord, is.character), as.factor)
str(data_1ord)
summary(data_1ord)
data_1ord$`White Player`=factor(data_1ord$`White Player`)
summary(data_1ord$`White Player`)


a=c()
for (i in 1:nrow(data_1ord)){
  if (data_1ord$Rating1[i]> data_1ord$Rating2[i]){a[i]="Blancas"}
  if (data_1ord$Rating1[i]< data_1ord$Rating2[i]){a[i]="Negras"}
  if (data_1ord$Rating1[i]== data_1ord$Rating2[i]){a[i]="Idéntico"}}
a=factor(a)

df2<-data.frame(
  pos=c("Blancas", "Idéntico", "Negras"),
  proporciones=c(sum(a=="Blancas")/nrow(data_1ord),sum(a=="Idéntico")/
  nrow(data_1ord),sum(a=="Negras")/nrow(data_1ord))
)
df2$porcentajes<-paste(round(100*df2$proporciones, 2), "%", sep="")
ggplot(df2, aes(x = "", y = proporciones, fill = pos)) +
  geom_col() +
  theme_void()+
  geom_text(aes(label = porcentajes),
            position = position_stack(vjust = 0.5))+
  ggtitle("Comparación Ratings según el color de las fichas")+
  theme(plot.title = element_text(hjust = 0.5))+
  coord_polar(theta = "y")+
  labs(fill='Mayor rating')
```

```
df3<-data.frame(
  pos=c("Blitz", "Normal" ,"Rápida"),
  proporciones=c(sum(data_1ord$tipo_partida=="Blitz")/nrow(data_1ord),
  sum(data_1ord$tipo_partida=="Normal")/nrow(data_1ord),
  sum(data_1ord$tipo_partida=="Rápida")/nrow(data_1ord))
)
df3$porcentajes<-paste(round(100*df3$proporciones, 2), "%", sep="")
ggplot(df3, aes(x = "", y = proporciones, fill = pos)) +
  geom_col() +
  theme_void()+
  geom_text(aes(label = porcentajes),
            position = position_stack(vjust = 0.5))+
  ggtitle("Porcentaje de partidas de cada tipo")+
  theme(plot.title = element_text(hjust = 0.5))+
  coord_polar(theta = "y")+
  labs(fill='Tipo de partida')

d=na.omit(data_1ord)
df4<-data.frame(
  pos=c("Resto", "Tipo1", "Tipo2", "Tipo3"),
  proporciones=c(sum(d$Importancia_torneo=="Resto")/nrow(d),
  sum(d$Importancia_torneo=="Tipo1")/nrow(d),sum(d$Importancia_torneo=="Tipo2")/nrow(d),
                 sum(d$Importancia_torneo=="Tipo3")/nrow(d))
)
df4$porcentajes<-paste(round(100*df4$proporciones, 2), "%", sep="")
ggplot(df4, aes(x = "", y = proporciones, fill = pos)) +
  geom_col() +
  theme_void()+
  geom_text(aes(label = porcentajes),
            position = position_stack(vjust = 0.5))+
  ggtitle("Porcentaje de partidas según la importancia del torneo")+
  theme(plot.title = element_text(hjust = 0.5))+
  coord_polar(theta = "y")+
  labs(fill='Importancia del torneo')

df5<-data.frame(
  pos=c("1-0", "1/2-1/2", "0-1"),
  proporciones=c(sum(data_1ord$Result=="1-0")/nrow(data_1ord),
  sum(data_1ord$Result=="1/2-1/2")/nrow(data_1ord),
                 sum(data_1ord$Result=="0-1")/nrow(data_1ord))
)
df5$porcentajes<-paste(round(100*df5$proporciones, 2), "%", sep="")
ggplot(df5, aes(x = "", y = proporciones, fill = pos)) +
  geom_col() +
  theme_void()+
  geom_text(aes(label = porcentajes),
            position = position_stack(vjust = 0.5))+
  ggtitle("Porcentaje de partidas según el resultado")+
  theme(plot.title = element_text(hjust = 0.5))+
  coord_polar(theta = "y")+
  labs(fill='Posibles resultados')
```

A.2.2. Análisis descriptivo variables cuantitativas vs Resultado

```
library(tidyverse)
data_1ord$WhiteMaxRating<-as.numeric(data_1ord$WhiteMaxRating)
data_1ord$BlackMaxRating<-as.numeric(data_1ord$BlackMaxRating)
data_1ord<-data_1ord %>%
  mutate_if(sapply(data_1ord, is.character), as.factor)
str(data_1ord)
datos<-na.omit(data_1ord) # Importante quitar los NAs para que luego funcione bien
datos$Result<-factor(datos$Result,levels=levels(datos$Result)[c(2,1,3)])
load("C:/Users/usuario/Desktop/uniovi/Máster/Segundo/TFM/DatosTFM.RData")
names(data_1ord) <- make.names(names(data_1ord))
summary(datos)
library(abind, pos=26)
library(e1071, pos=27)
numSummary(datos[,c("AgresividadBP", "AgresividadWP", "BlackMaxRating",
"BlackRating2018", "BlackRating2019", "BlackRating2020", "BlackRatingChange", "Edad_BP",
  "Edad_WP", "EF1B", "EF1W", "EF2B", "EF2W", "H2H", "LuchaBP", "LuchaWP",
```

```
  "Rating1", "Rating2", "WhiteMaxRating", "WhiteRating2018", "WhiteRating2019",
  "WhiteRating2020", "WhiteRatingChange"), drop=FALSE], groups=datos$Result,
  statistics=c("mean", "sd", "IQR", "quantiles"), quantiles=c(0,.25,.5,.75,1))

#Rating1
Tapply(Rating1 ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(Rating1 ~ Result, data=datos, center="median") #Varianzas distintas
AnovaModel.3 <- aov(Rating1 ~ Result, data=datos)
summary(AnovaModel.3)
with(datos, numSummary(Rating1, groups=Result, statistics=c("mean", "sd")))
oneway.test(Rating1 ~ Result, data=datos) # Welch test
#Medias distintas

#Rating2
Tapply(Rating2 ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(Rating2 ~ Result, data=datos, center="median") #Varianzas distintas

AnovaModel.4 <- aov(Rating2 ~ Result, data=datos)
summary(AnovaModel.4)
with(datos, numSummary(Rating2, groups=Result, statistics=c("mean", "sd")))
oneway.test(Rating2 ~ Result, data=datos) # Welch test
#Medias distintas


#WhiteMaxRating

Tapply(WhiteMaxRating ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(WhiteMaxRating ~ Result, data=datos, center="median") #Varianzas distintas

AnovaModel.5 <- aov(WhiteMaxRating ~ Result, data=datos)
summary(AnovaModel.5)
with(datos, numSummary(WhiteMaxRating, groups=Result, statistics=c("mean", "sd")))
oneway.test(WhiteMaxRating ~ Result, data=datos) # Welch test
#Distintas medias


#BlackMaxRating

Tapply(BlackMaxRating ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(BlackMaxRating ~ Result, data=datos, center="median") #Varianzas distintas

AnovaModel.6 <- aov(BlackMaxRating ~ Result, data=datos)
summary(AnovaModel.6)
with(datos, numSummary(BlackMaxRating, groups=Result, statistics=c("mean", "sd")))
oneway.test(BlackMaxRating ~ Result, data=datos) # Welch test
#Distintas medias

#WhiteRatingChange

Tapply(WhiteRatingChange ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(WhiteRatingChange ~ Result, data=datos, center="median") #Varianzas distintas


AnovaModel.7 <- aov(WhiteRatingChange ~ Result, data=datos)
summary(AnovaModel.7)
with(datos, numSummary(WhiteRatingChange, groups=Result, statistics=c("mean", "sd")))
oneway.test(WhiteRatingChange ~ Result, data=datos) # Welch test
 #Distintas medias

#BlackRatingChange
Tapply(BlackRatingChange ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(BlackRatingChange ~ Result, data=datos, center="median") #Varianzas distintas



AnovaModel.8 <- aov(BlackRatingChange ~ Result, data=datos)
summary(AnovaModel.8)
with(datos, numSummary(BlackRatingChange, groups=Result, statistics=c("mean", "sd")))
oneway.test(BlackRatingChange ~ Result, data=datos) # Welch test
#Medias distintas
```

```
#WhiteRating2018
Tapply(WhiteRating2018 ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(WhiteRating2018 ~ Result, data=datos, center="median") #Distintas varianzas

AnovaModel.9 <- aov(WhiteRating2018 ~ Result, data=datos)
summary(AnovaModel.9)
with(datos, numSummary(WhiteRating2018, groups=Result, statistics=c("mean", "sd")))
oneway.test(WhiteRating2018 ~ Result, data=datos) # Welch test
#Distintas medias

#BlackRating2018

Tapply(BlackRating2018 ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(BlackRating2018 ~ Result, data=datos, center="median") #Distintas varianzas


AnovaModel.10 <- aov(BlackRating2018 ~ Result, data=datos)
summary(AnovaModel.10)
with(datos, numSummary(BlackRating2018, groups=Result, statistics=c("mean", "sd")))
oneway.test(BlackRating2018 ~ Result, data=datos) # Welch test
 #Distintas medias

#BlackRating2019
Tapply(BlackRating2019 ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(BlackRating2019 ~ Result, data=datos, center="median") #Distintas varianzas


AnovaModel.11 <- aov(BlackRating2019 ~ Result, data=datos)
summary(AnovaModel.11)
with(datos, numSummary(BlackRating2019, groups=Result, statistics=c("mean", "sd")))
oneway.test(BlackRating2019 ~ Result, data=datos) # Welch test
#Distintas medias


#WhiteRating2019
Tapply(WhiteRating2019 ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(WhiteRating2019 ~ Result, data=datos, center="median") #Distintas varianzas


AnovaModel.12 <- aov(WhiteRating2019 ~ Result, data=datos)
summary(AnovaModel.12)
with(datos, numSummary(WhiteRating2019, groups=Result, statistics=c("mean", "sd")))
oneway.test(WhiteRating2019 ~ Result, data=datos) # Welch test
#Distintas medias


#BlackRating2020
Tapply(BlackRating2020 ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(BlackRating2020 ~ Result, data=datos, center="median") #Distintas medias


AnovaModel.14 <- aov(BlackRating2020 ~ Result, data=datos)
summary(AnovaModel.14)
with(datos, numSummary(BlackRating2020, groups=Result, statistics=c("mean", "sd")))
oneway.test(BlackRating2020 ~ Result, data=datos) # Welch test
#Distintas medias


#WhiteRating2020
Tapply(WhiteRating2020 ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(WhiteRating2020 ~ Result, data=datos, center="median") #Distintas varianzas


AnovaModel.13 <- aov(WhiteRating2020 ~ Result, data=datos)
summary(AnovaModel.13)
with(datos, numSummary(WhiteRating2020, groups=Result, statistics=c("mean", "sd")))
oneway.test(WhiteRating2020 ~ Result, data=datos) # Welch test
#Distintas medias
```

```
#EF1W
Tapply(EF1W ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(EF1W ~ Result, data=datos, center="median") #Distintas varianzas


AnovaModel.15 <- aov(EF1W ~ Result, data=datos)
summary(AnovaModel.15)
with(datos, numSummary(EF1W, groups=Result, statistics=c("mean", "sd")))
oneway.test(EF1W ~ Result, data=datos) # Welch test
#Distintas medias

#EF1B
Tapply(EF1B ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(EF1B ~ Result, data=datos, center="median") #Distintas varianzas


AnovaModel.16 <- aov(EF1B ~ Result, data=datos)
summary(AnovaModel.16)
with(datos, numSummary(EF1B, groups=Result, statistics=c("mean", "sd")))
oneway.test(EF1B ~ Result, data=datos) # Welch test
#Distintas medias

#EF2W
Tapply(EF1W ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(EF2W ~ Result, data=datos, center="median") #Distintas varianzas


AnovaModel.17 <- aov(EF2W ~ Result, data=datos)
summary(AnovaModel.17)
with(datos, numSummary(EF2W, groups=Result, statistics=c("mean", "sd")))
oneway.test(EF2W ~ Result, data=datos) # Welch test
#Distintas medias

#EF2B
Tapply(EF2B ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(EF2B ~ Result, data=datos, center="median") #Distintas varianzas


AnovaModel.18 <- aov(EF2B ~ Result, data=datos)
summary(AnovaModel.18)
with(datos, numSummary(EF2B, groups=Result, statistics=c("mean", "sd")))
oneway.test(EF2B ~ Result, data=datos) # Welch test
#Distintas medias


#AgresividadWP
Tapply(AgresividadWP ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(AgresividadWP ~ Result, data=datos, center="median") #Distintas varianzas

AnovaModel.20 <- aov(AgresividadWP ~ Result, data=datos)
summary(AnovaModel.20)
with(datos, numSummary(AgresividadWP, groups=Result, statistics=c("mean", "sd")))
oneway.test(AgresividadWP ~ Result, data=datos) # Welch test
#Distintas medias

#AgresividadBP
Tapply(AgresividadBP ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(AgresividadBP ~ Result, data=datos, center="median") #Igualdad de varianzas

library(mvtnorm, pos=26)
library(survival, pos=26)
library(MASS, pos=26)
library(TH.data, pos=26)
library(multcomp, pos=26)
library(abind, pos=31)
AnovaModel.1 <- aov(AgresividadBP ~ Result, data=datos)
summary(AnovaModel.1)
with(datos, numSummary(AgresividadBP, groups=Result, statistics=c("mean", "sd")))
#No hay igualdad de medias
```

```
#LuchaWP
Tapply(LuchaWP ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(LuchaWP ~ Result, data=datos, center="median") #No hay igualdad de varianzas


AnovaModel.22 <- aov(LuchaWP ~ Result, data=datos)
summary(AnovaModel.22)
with(datos, numSummary(LuchaWP, groups=Result, statistics=c("mean", "sd")))
oneway.test(LuchaWP ~ Result, data=datos) # Welch test
#Distintas medias



#LuchaBP
Tapply(LuchaBP ~ Result, var, na.action=na.omit, data=datos) # variances by group
leveneTest(LuchaBP ~ Result, data=datos, center="median") #No hay igualdad de varianzas


AnovaModel.21 <- aov(LuchaBP ~ Result, data=datos)
summary(AnovaModel.21)
with(datos, numSummary(LuchaBP, groups=Result, statistics=c("mean", "sd")))
oneway.test(LuchaBP ~ Result, data=datos) # Welch test
#Distintas medias
```

A.3. Modelo de predicción

```
#Cargamos los paquetes necesarios
library(dplyr)
library(nnet) # Para hacer multinom
library("pscl") # Para calcular el R^2
library(generalhoslem) # #Generalised Hoslem
library(ordinal) # Regresión ordinal
library(tidyverse)
library(RcmdrMisc)
library(tidyverse)
library(janitor)
library(ggplot2)
library(ggrepel)
library(GGally)
library(RColorBrewer)
library(VIM)

library(MASS)
library(rpart)
library(rpart.plot)
library(nnet)
library(kknn)

library(pROC)
library(ipred)
library(fastAdaboost)
library(randomForest)
library(e1071)
## Preparar la base de datos:
load(data_1ord)
names(data_1ord) <- make.names(names(data_1ord))
summary(data_1ord) #Resumen de los datos
#Convertimos variables a numérico
data_1ord$WhiteMaxRating<-as.numeric(data_1ord$WhiteMaxRating)
data_1ord$BlackMaxRating<-as.numeric(data_1ord$BlackMaxRating)
#Convertimos variables string en factores
data_1ord<-data_1ord %>%
  mutate_if(sapply(data_1ord, is.character), as.factor)
str(data_1ord)
#Eliminamos las filas con NA's
datos<-na.omit(data_1ord)
```

```
#Convertimos la variable Result a factor con
los niveles que queremos
datos$Result<-factor(datos$Result,
levels=levels(datos$Result)[c(2,1,3)])
```

A.3.1. Modelo ordinal

```
RegOrd2 <- polr(Result ~ Rating1+Rating2+tipo_partida+WhiteMaxRating+BlackMaxRating+
WhiteRating2018+BlackRating2018+WhiteRating2019+BlackRating2019+
WhiteRating2020+BlackRating2020+BlackRatingChange+WhiteRatingChange+
H2H+LuchaWP+AgresividadWP+LuchaBP+AgresividadBP+Edad_WP+Edad_BP+
Importancia_torneo+Tipo_torneo+EF1W+EF1B+EF2W+EF2B, data=datos, Hess=TRUE)
summary(RegOrd2)

# Tests for Proportional Odds
p=car::poTest(RegOrd2) # No salen Odds proporcionales.
p
#No se puede usar el modelo ordinal
```

A.3.2. Modelos Multinomial

```
### Modelo multinomial 1: completo, TODAS las variables como predictoras

#Hacemos el modelo
RM1<-multinom(Result ~ Rating1+Rating2+tipo_partida+WhiteMaxRating+BlackMaxRating+
WhiteRating2018+BlackRating2018+WhiteRating2019+BlackRating2019+
WhiteRating2020+BlackRating2020+BlackRatingChange+WhiteRatingChange+
H2H+LuchaWP+AgresividadWP+LuchaBP+AgresividadBP+Edad_WP+Edad_BP+
Importancia_torneo+Tipo_torneo+EF1W+EF1B+EF2W+EF2B, data=datos)
summary(RM1)
exp(coef(RM1)) #Odds Ratio
z <- summary(RM1)$coefficients/summary(RM1)$standard.errors
p <- (1 - pnorm(abs(z), 0, 1)) * 2
p
# Este vector p contiene los p-valores
que comprueban la significatividad de las variables

exp(coef(step(RM1))) # Modelo paso a paso.
Por defecto, usa la opción "both" y el criterio AIC
# Tras hacer el modelo paso a paso, da como resultado este modelo:


#Modelo simplificado

RM1simpli<-multinom(Result ~ Rating1 + tipo_partida + BlackRating2018 +
WhiteRatingChange + H2H + LuchaWP +
AgresividadWP + AgresividadBP + Edad_WP +
Edad_BP + Importancia_torneo +
Tipo_torneo + EF2W, data=datos)
summary(RM1simpli)
z1bis <- summary(RM1simpli)$coefficients/summary(RM1simpli)$standard.errors
p1bis <- (1 - pnorm(abs(z1bis), 0, 1)) * 2
p1bis # Este vector contiene los p-valores
que comprueban la significatividad de las variables
step(RM1simpli) # Este ya no se puede simplificar más.

# Cálculo de la matriz de confusión.
PredProb <- fitted(RM1simpli)
Pmax<-apply(PredProb,1,max)
aa<-(PredProb==Pmax)%*%(0:2)
aa<-as.factor(aa)
levels(aa)<-c("1-0","0-1","1/2-1/2")
tab<-table(aa,datos$Result)
sum(diag(tab))/sum(tab) # % de individuos clasificados correctamente
sum(tab[1,2]+tab[2,1])/sum(tab)
# Cálculo del R^2
library("pscl")
pR2(RM1simpli)  # McFadden: 0.119
```

```
#TRV
m0=multinom(Result~1, data=datos)
anova(m0,RM1simpli)
#Generalised Hoslem (se necesita p-valor>0.05)
logitgof(datos$Result,fitted(RM1simpli)) # pv=0.06954




##########################################
### Hago el modelo para cada tipo de partida
Blitz<-datos[datos$tipo_partida=="Blitz" &
datos$Importancia_torneo!="Tipo1",]
Blitz <- droplevels(Blitz)
Rapid<-datos[datos$tipo_partida=="Rápida",]
Rapid<-droplevels(Rapid)
Normal<-datos[datos$tipo_partida=="Normal",]
Normal<-droplevels(Normal)
## BLITZ
### Correlaciones
nums <- unlist(lapply(datos, is.numeric))
summary(Blitz)
cor(Blitz[,nums])
# Quito el tipo de torneo porque no hay de selecciones

tbl <-
  nnet::multinom(Result ~Rating1+Rating2+WhiteMaxRating+BlackMaxRating+
WhiteRating2018+BlackRating2018+WhiteRating2019+BlackRating2019+
WhiteRating2020+BlackRating2020+BlackRatingChange+WhiteRatingChange+
H2H+LuchaWP+AgresividadWP+LuchaBP+AgresividadBP+Edad_WP+Edad_BP+
Importancia_torneo+EF1W+EF1B+EF2W+EF2B, data=Blitz) %>%

RM1b<-multinom(Result ~Rating1+Rating2+WhiteMaxRating+BlackMaxRating+
WhiteRating2018+BlackRating2018+WhiteRating2019+BlackRating2019+
WhiteRating2020+BlackRating2020+BlackRatingChange+WhiteRatingChange+
H2H+LuchaWP+AgresividadWP+LuchaBP+AgresividadBP+Edad_WP+Edad_BP+
Importancia_torneo+EF1W+EF1B+EF2W+EF2B, data=Blitz)
summary(RM1b)
exp(coef(RM1b)) #Odds ratio
zb <- summary(RM1b)$coefficients/summary(RM1b)$standard.errors
pb <- (1 - pnorm(abs(zb), 0, 1)) * 2
pb # Este vector contiene los p-valores
que comprueban la significatividad de las variables
RM1bSimpli<-step(RM1b) # Modelo paso a paso.
Por defecto, usa la opción "both" y el
criterio AIC
exp(coef(RM1bSimpli)) #Odds ratio

# Cálculo de la matriz de confusión.
PredProb <- fitted(RM1bSimpli)
Pmax<-apply(PredProb,1,max)
aa<-(PredProb==Pmax)%*%(0:2)
aa<-as.factor(aa)
levels(aa)<-c("1-0","0-1","1/2-1/2")
tabb<-table(aa,Blitz$Result)
sum(diag(tabb))/sum(tabb)# % de individuos clasificados correctamente
sum(tabb[1,2]+tabb[2,1])/sum(tabb)
# Cálculo del R^2
library("pscl")
pR2(RM1bSimpli)


#TRV
m0=multinom(Result~1, data=Blitz)
anova(m0,RM1bSimpli)
#Generalised Hoslem (se necesita p-valor>0.05)
logitgof(Blitz$Result,fitted(RM1bSimpli)) # pv=0.3358

## RAPID
```

```
summary(Rapid)

RM1r<-multinom(Result ~ Rating1+Rating2+WhiteMaxRating+BlackMaxRating+
WhiteRating2018+BlackRating2018+WhiteRating2019+BlackRating2019+
WhiteRating2020+BlackRating2020+BlackRatingChange+WhiteRatingChange+
H2H+LuchaWP+AgresividadWP+LuchaBP+AgresividadBP+Edad_WP+Edad_BP+
Importancia_torneo+EF1W+EF1B+EF2W+EF2B, data=Rapid)
summary(RM1r)
exp(coef(RM1r)) #Odds ratio
zr <- summary(RM1r)$coefficients/summary(RM1r)$standard.errors
pr <- (1 - pnorm(abs(zr), 0, 1)) * 2
pr # Este vector contiene los p-valores
que comprueban la significatividad de las variables
RM1rSimpli<-step(RM1r) # Modelo paso a paso.
Por defecto, usa la opción "both" y el criterio AIC

exp(coef(RM1rSimpli)) #Odds ratio

# Cálculo de la matriz de confusión.
PredProb <- fitted(RM1rSimpli)
Pmax<-apply(PredProb,1,max)
aa<-(PredProb==Pmax)%*%(0:2)
aa<-as.factor(aa)
levels(aa)<-c("1-0","0-1","1/2-1/2")
tabr<-table(aa,Rapid$Result)
sum(diag(tabr))/sum(tabr)# % de individuos
clasificados correctamente
sum(tabr[1,2]+tabr[2,1])/sum(tabr)
# Cálculo del R^2
library("pscl")
pR2(RM1rSimpli)  # McFadded: 0.078

#TRV
m0=multinom(Result~1, data=Rapid)
anova(m0,RM1rSimpli)
#Generalised Hoslem (se necesita p-valor>0.05)
logitgof(Rapid$Result,fitted(RM1rSimpli)) # pv=0.5769


## NORMAL
summary(Normal)

RM1n<-multinom(Result ~ Rating1+Rating2+WhiteMaxRating+BlackMaxRating+
WhiteRating2018+BlackRating2018+WhiteRating2019+BlackRating2019+
WhiteRating2020+BlackRating2020+BlackRatingChange+WhiteRatingChange+
H2H+LuchaWP+AgresividadWP+LuchaBP+AgresividadBP+Edad_WP+Edad_BP+
Importancia_torneo+EF1W+EF1B+EF2W+EF2B+Tipo_torneo, data=Normal)
summary(RM1n) # Modelo paso a paso.
Por defecto, usa la opción "both" y el criterio AIC
exp(coef(RM1n)) #Odds ratio
zn <- summary(RM1n)$coefficients/summary(RM1n)$standard.errors
pn <- (1 - pnorm(abs(zn), 0, 1)) * 2
pn # Este vector contiene los p-valores
que comprueban la significatividad de las variables
RM1nSimpli<-step(RM1n)

exp(coef(RM1nSimpli)) #Odds ratio

# Cálculo de la matriz de confusión.
PredProb <- fitted(RM1nSimpli)
Pmax<-apply(PredProb,1,max)
aa<-(PredProb==Pmax)%*%(0:2)
aa<-as.factor(aa)
levels(aa)<-c("1-0","0-1","1/2-1/2")
tabn<-table(aa,Normal$Result)
sum(diag(tabn))/sum(tabn)# % de individuos clasificados correctamente
sum(tabn[1,2]+tabn[2,1])/sum(tabn)
# Cálculo del R^2
library("pscl")
pR2(RM1nSimpli)  # McFadded: 0.178
```

```
#TRV
m0=multinom(Result~1, data=Normal)
anova(m0,RM1nSimpli)
#Generalised Hoslem (se necesita p-valor>0.05)
logitgof(Normal$Result,fitted(RM1nSimpli)) # pv=0.5068
```

A.3.3. Árbol de decisión

```
arbol <- rpart(Result ~ AgresividadBP + AgresividadWP + BlackMaxRating +
BlackRatingChange + Edad_BP + Edad_WP + EF1B +
EF1W + EF2B + EF2W + H2H +
Importancia_torneo + LuchaBP + LuchaWP +
Rating1 + Rating2 + tipo_partida +
Tipo_torneo + WhiteMaxRating + WhiteRatingChange,
data=datos, control = rpart.control(xval = 100,
minsplit = 2))
#Podamos el árbol
arbol_mejor <- prune(arbol, cp = arbol$cptable[as.numeric(
which(arbol$cptable[,"xerror"] <
sum(arbol$cptable[which.min(
arbol$cptable[,"xerror"]),
c("xerror","xstd")]))[1]), "CP"])

#Hacemos la predicción
pred_arbol <- predict(arbol_mejor,datos, type = "class")
#Calculamos el acierto
acierto_arbol <- mean(datos$Result == pred_arbol)
tabla <- table(datos$Result, pred_arbol)
```

A.3.4. K-vecinos más cercanos

```
knn <- train.kknn(Result ~ AgresividadBP +
AgresividadWP + BlackMaxRating + BlackRatingChange +
Edad_BP + Edad_WP + EF1B + EF1W + EF2B + EF2W + H2H
+ Importancia_torneo + LuchaBP + LuchaWP + Rating1 +
Rating2 + tipo_partida +
Tipo_torneo + WhiteMaxRating +
WhiteRatingChange, data=datos, ks = 1:100)
#k va de 1 a 100

## Hacemos la predicción
pred_knn <- predict(knn, datos)
#Calculamos el acierto
acierto_knn <- mean(datos$Result == pred_knn)
```

A.3.5. Máquinas de vector soporte

```
mvs <- svm(Result~AgresividadBP + AgresividadWP +
BlackMaxRating + BlackRatingChange + Edad_BP +
Edad_WP + EF1B + EF1W + EF2B + EF2W + H2H +
Importancia_torneo + LuchaBP + LuchaWP + Rating1 +
Rating2 + tipo_partida +
Tipo_torneo + WhiteMaxRating + WhiteRatingChange,
data = datos, probability = TRUE)
#Hacemos la predicción
pred_svm=predict(mvs, datos[,-5])
#Calculamos el acierto
acierto_svm <- mean(datos$Result == pred_svm)

#Matriz de confusión
tabla <- table(datos$Result, pred_svm)
(tabla[1,3]+tabla[3,1])/sum(tabla)
```

A.3.6. Validación Modelos

Validación modelos multinomial

```
#BLITZ
entrena_Blitz=data_1ord[data_1ord$tipo_partida=="Blitz",]
entrena_Blitz$fecha<-NULL
entrena_Blitz$tipo_partida<-NULL
valida_Blitz=Validacion[Validacion$tipo_partida=="Blitz",-c(1,2)]
valida_Blitz$fecha<-NULL
valida_Blitz$tipo_partida<-NULL

#preparamos el conjunto de datos
library(RcmdrMisc)
entrena_Blitz<-na.omit(entrena_Blitz)
summary(entrena_Blitz)
entrena_Blitz$WhiteMaxRating<-as.numeric(entrena_Blitz$WhiteMaxRating)
entrena_Blitz$BlackMaxRating<-as.numeric(entrena_Blitz$BlackMaxRating)
entrena_Blitz <- as.data.frame(unclass(entrena_Blitz),
# Convert all columns to factor
                       stringsAsFactors = TRUE)


summary(valida_Blitz)
valida_Blitz$WhiteMaxRating<-as.numeric(valida_Blitz$WhiteMaxRating)
valida_Blitz$BlackMaxRating<-as.numeric(valida_Blitz$BlackMaxRating)
valida_Blitz <- as.data.frame(unclass(valida_Blitz),
# Convert all columns to factor
                              stringsAsFactors = TRUE)
entrena_Blitz$Result=factor(entrena_Blitz$Result,levels=
levels(entrena_Blitz$Result)[c(2,1,3)])
valida_Blitz$Result=factor(valida_Blitz$Result,levels=
levels(valida_Blitz$Result)[c(2,1,3)])
valida_Blitz<-na.omit(valida_Blitz)
####
#Regresión multinormial
library(nnet)
library(mlogit)
library(gtsummary)
library(magrittr)

multinom_pivot_wider <- function(x) {
  # check inputs match expectatations
  if (!inherits(x, "tbl_regression") || !inherits(x$model_obj, "multinom")) {
    stop("`x=` must be class 'tbl_regression' summary of a `nnet::multinom()` model.")
  }

  # create tibble of results
  df <- tibble::tibble(outcome_level = unique(x$table_body$groupname_col))
  df$tbl <-
    purrr::map(
      df$outcome_level,
      function(lvl) {
        gtsummary::modify_table_body(
          x,
          ~dplyr::filter(.x, .data$groupname_col %in% lvl) %>%
            dplyr::ungroup() %>%
            dplyr::select(-.data$groupname_col)
        )
      }
    )

  tbl_merge(df$tbl, tab_spanner = paste0("**", df$outcome_level, "**"))
}

# dummy data

# multinom model tabulated with gtsummary
tbl <-
  nnet::multinom(Result ~ Rating1+Rating2+WhiteMaxRating+BlackMaxRating+
              WhiteRating2018+BlackRating2018+WhiteRating2019+BlackRating2019+
```

```
            WhiteRating2020+BlackRating2020+BlackRatingChange+WhiteRatingChange+
            H2H+LuchaWP+AgresividadWP+LuchaBP+AgresividadBP+Edad_WP+Edad_BP+
            Importancia_torneo+EF1W+EF1B+EF2W+EF2B, data=entrena_Blitz) %>%
  tbl_regression(exponentiate = TRUE,intercept=TRUE) %>%
  bold_p(t = 0.05)%>%
  multinom_pivot_wider()%>%
  italicize_levels()%>%
  modify_header(label = "**Variable**",ci_1="**95% IC1**",ci_2="**95% IC2**",
  p.value_1 = "**p-valor**",p.value_2 = "**p-valor**") %>%
  modify_footnote(ci_1 = "IC1 = Intervalo de confianza 1",ci_2 =
  "IC2 = Intervalo de confianza 2", abbreviation = TRUE)



RM1b<-multinom(Result ~ Rating1+Rating2+WhiteMaxRating+BlackMaxRating+
         WhiteRating2018+BlackRating2018+WhiteRating2019+BlackRating2019+
         WhiteRating2020+BlackRating2020+BlackRatingChange+WhiteRatingChange+
         H2H+LuchaWP+AgresividadWP+LuchaBP+AgresividadBP+Edad_WP+Edad_BP+
         Importancia_torneo+EF1W+EF1B+EF2W+EF2B, data=entrena_Blitz)
#SIMPLIFICAMOS CON AIC
RM1bSimpli<-step(RM1b)
pred_reg <- predict(RM1bSimpli, valida_Blitz, type = "class")
acierto_reg <- mean(valida_Blitz$Result == pred_reg)
tabla <- table(valida_Blitz$Result, pred_reg) #Matriz de confusión
levels(pred_reg)
levels(valida_Blitz$Result)
#37% de acierto
sum(tabla[1,2]+tabla[2,1])/sum(tabla)

#Rápida
entrena_rapida=data_1ord[data_1ord$tipo_partida=="Rápida",]
entrena_rapida$fecha<-NULL
entrena_rapida$tipo_partida<-NULL
valida_rapida=Validacion[Validacion$tipo_partida=="Rápida",-c(1,2)]
valida_rapida$fecha<-NULL
valida_rapida$tipo_partida<-NULL
#preparamos el conjunto de datos
library(RcmdrMisc)
entrena_rapida<-na.omit(entrena_rapida)
summary(entrena_rapida)
entrena_rapida$WhiteMaxRating<-as.numeric(entrena_rapida$WhiteMaxRating)
entrena_rapida$BlackMaxRating<-as.numeric(entrena_rapida$BlackMaxRating)
entrena_rapida <- as.data.frame(unclass(entrena_rapida),# Convert all columns to factor
                                stringsAsFactors = TRUE)

valida_rapida<-na.omit(valida_rapida)
summary(valida_rapida)
valida_rapida$WhiteMaxRating<-as.numeric(valida_rapida$WhiteMaxRating)
valida_rapida$BlackMaxRating<-as.numeric(valida_rapida$BlackMaxRating)
valida_rapida <- as.data.frame(unclass(valida_rapida),# Convert all columns to factor
                               stringsAsFactors = TRUE)
entrena_rapida$Result=factor(entrena_rapida$Result,levels=
levels(entrena_rapida$Result)[c(2,1,3)])
valida_rapida$Result=factor(valida_rapida$Result,levels=
levels(valida_rapida$Result)[c(2,1,3)])

####
#Regresión multinormial
library(nnet)
library(mlogit)
RM1r<-multinom(Result ~ Rating1+Rating2+WhiteMaxRating+BlackMaxRating+
         WhiteRating2018+BlackRating2018+WhiteRating2019+BlackRating2019+
         WhiteRating2020+BlackRating2020+BlackRatingChange+WhiteRatingChange+
         H2H+LuchaWP+AgresividadWP+LuchaBP+AgresividadBP+Edad_WP+Edad_BP+
         Importancia_torneo+EF1W+EF1B+EF2W+EF2B, data=entrena_rapida)
#SIMPLIFICAMOS CON AIC
RM1rSimpli<-step(RM1r)
pred_reg <- predict(RM1rSimpli, valida_rapida, type = "class")
acierto_reg <- mean(valida_rapida$Result == pred_reg)
tabla <- table(valida_rapida$Result, pred_reg) #Matriz de confusión
```

```
levels(pred_reg)
levels(valida_Blitz$Result)
#48% de acierto
sum(tabla[1,2]+tabla[2,1])/sum(tabla)

#NORMAL
entrena_normal=data_1ord[data_1ord$tipo_partida=="Normal",]
entrena_normal$fecha<-NULL
entrena_normal$tipo_partida<-NULL
library(readxl)
valida_normal=readXL("C:/Users/usuario/Desktop/uniovi/Máster/Segundo/TFM/validacionTata.xlsx")
valida_normal=valida_normal[,-c(1,2)]
valida_normal$fecha<-NULL
valida_normal$tipo_partida<-NULL
save(valida_normal,file="valida_normal.RData")
#preparamos el conjunto de datos
library(RcmdrMisc)
entrena_normal<-na.omit(entrena_normal)
summary(entrena_normal)
entrena_normal$WhiteMaxRating<-as.numeric(entrena_normal$WhiteMaxRating)
entrena_normal$BlackMaxRating<-as.numeric(entrena_normal$BlackMaxRating)
entrena_normal <- as.data.frame(unclass(entrena_normal),# Convert all columns to factor
                                stringsAsFactors = TRUE)

valida_normal<-na.omit(valida_normal)
summary(valida_normal)
valida_normal$WhiteMaxRating<-as.numeric(valida_normal$WhiteMaxRating)
valida_normal$BlackMaxRating<-as.numeric(valida_normal$BlackMaxRating)
valida_normal<- as.data.frame(unclass(valida_normal),# Convert all columns to factor
                              stringsAsFactors = TRUE)
entrena_normal$Result=factor(entrena_normal$Result,levels=
levels(entrena_normal$Result)[c(2,1,3)])
valida_normal$Result=factor(valida_normal$Result,levels=
levels(valida_normal$Result)[c(2,1,3)])
valida_normal$Importancia_torneo=factor(gsub(" ", "",valida_normal$Importancia_torneo))
####
#Regresión multinormial
library(nnet)
library(mlogit)
RM1n<-multinom(Result ~ Rating1+Rating2+WhiteMaxRating+BlackMaxRating+
            WhiteRating2018+BlackRating2018+WhiteRating2019+BlackRating2019+
            WhiteRating2020+BlackRating2020+BlackRatingChange+WhiteRatingChange+
            H2H+LuchaWP+AgresividadWP+LuchaBP+AgresividadBP+Edad_WP+Edad_BP+
            Importancia_torneo+EF1W+EF1B+EF2W+EF2B+Tipo_torneo, data=entrena_normal)
#SIMPLIFICAMOS CON AIC
RM1nSimpli<-step(RM1n)
pred_reg <- predict(RM1nSimpli, valida_normal, type = "class")
acierto_reg <- mean(valida_normal$Result == pred_reg)
tabla <- table(valida_normal$Result, pred_reg) #Matriz de confusión
levels(pred_reg)
levels(valida_normal$Result)
#52%
sum(tabla[1,2]+tabla[2,1])/sum(tabla)

###
#Todos juntos
Validacion=Validacion[,-c(1,2)]
Validacion$fecha<-NULL
valida_normal$tipo_partida<-rep("Normal",nrow(valida_normal))

valida_todos<-rbind(Validacion,valida_normal)

#preparamos el conjunto de datos

data_1ord<-na.omit(data_1ord)
summary(data_1ord)
data_1ord$WhiteMaxRating<-as.numeric(data_1ord$WhiteMaxRating)
data_1ord$BlackMaxRating<-as.numeric(data_1ord$BlackMaxRating)
data_1ord <- as.data.frame(unclass(data_1ord),# Convert all columns to factor
                                stringsAsFactors = TRUE)
```

```
valida_todos<-na.omit(valida_todos)
summary(valida_todos)
valida_todos$WhiteMaxRating<-as.numeric(valida_todos$WhiteMaxRating)
valida_todos$BlackMaxRating<-as.numeric(valida_todos$BlackMaxRating)
valida_todos <- as.data.frame(unclass(valida_todos),# Convert all columns to factor
                              stringsAsFactors = TRUE)
data_1ord$Result=factor(data_1ord$Result,levels=levels(data_1ord$Result)[c(2,1,3)])
valida_todos$Result=factor(valida_todos$Result,levels=levels(valida_todos$Result)[c(2,1,3)])

#Regresión multinormial
library(nnet)
library(mlogit)
RM1t<-multinom(Result ~ Rating1+Rating2+WhiteMaxRating+BlackMaxRating+
         WhiteRating2018+BlackRating2018+WhiteRating2019+BlackRating2019+
         WhiteRating2020+BlackRating2020+BlackRatingChange+WhiteRatingChange+
         H2H+LuchaWP+AgresividadWP+LuchaBP+AgresividadBP+Edad_WP+Edad_BP+
         EF1W+EF1B+EF2W+EF2B+Tipo_torneo+tipo_partida+
         Importancia_torneo, data=data_1ord)
#SIMPLIFICAMOS CON AIC
RM1tSimpli<-step(RM1t)
pred_reg <- predict(RM1tSimpli, valida_todos, type = "class")
acierto_reg <- mean(valida_todos$Result == pred_reg)
tabla <- table(valida_todos$Result, pred_reg) #Matriz de confusión
levels(pred_reg)
levels(valida_todos$Result)

#44%

Validacion=Validacion[,-c(1,2)]
Validacion$fecha<-NULL
valida_normal$tipo_partida<-rep("Normal",nrow(valida_normal))
valida_todos<-rbind(Validacion,valida_normal)
data_1ord$WhiteMaxRating<-as.numeric(data_1ord$WhiteMaxRating)
data_1ord$BlackMaxRating<-as.numeric(data_1ord$BlackMaxRating)
data_1ord<-data_1ord %>%
  mutate_if(sapply(data_1ord, is.character), as.factor)
str(data_1ord)
datos<-na.omit(data_1ord) # Importante quitar los NAs para que luego funcione bien
datos$Result<-factor(datos$Result,levels=levels(datos$Result)[c(2,3,1)])
valida_todos$Result<-as.factor(valida_todos$Result)
valida_todos$Result<-factor(valida_todos$Result,levels=levels(valida_todos$Result)[c(2,3,1)])
library(RcmdrMisc)
library(tidyverse)
library(janitor)
library(ggplot2)
library(ggrepel)
library(GGally)
library(RColorBrewer)
library(VIM)

library(MASS)
library(rpart)
library(rpart.plot)
library(nnet)
library(kknn)

library(pROC)
library(ipred)
library(fastAdaboost)
library(randomForest)
library(e1071)
valida_todos$Importancia_torneo[valida_todos$Importancia_torneo=="Tipo 3"]="Tipo3"
```

Árbol de decisión

```
arbol <- rpart(Result ~ AgresividadBP + AgresividadWP + BlackMaxRating +
BlackRatingChange + Edad_BP + Edad_WP + EF1B + EF1W + EF2B + EF2W + H2H +
Importancia_torneo + LuchaBP + LuchaWP + Rating1 + Rating2 + tipo_partida +
Tipo_torneo + WhiteMaxRating + WhiteRatingChange, data=datos, control =
```

```
rpart.control(xval = 100, minsplit = 2))
arbol_mejor <- prune(arbol, cp = arbol$cptable[as.numeric(which(
arbol$cptable[,"xerror"] < sum(arbol$cptable[which.min(arbol$cptable[,"xerror"]),
c("xerror","xstd")]))[1]), "CP"])

## Búsqueda del punto de corte óptimo para la clasificación
pred_arbol <- predict(arbol_mejor,valida_todos, type = "class")
acierto_arbol <- mean(valida_todos$Result == pred_arbol)
tabla <- table(valida_todos$Result, pred_arbol)
(tabla[1,3]+tabla[3,1])/sum(tabla)
```

K-vecinos más cercanos

```
knn <- train.kknn(Result ~ AgresividadBP + AgresividadWP + BlackMaxRating +
BlackRatingChange + Edad_BP + Edad_WP +
EF1B + EF1W + EF2B + EF2W + H2H +  LuchaBP + LuchaWP + Rating1 + Rating2 +
tipo_partida + WhiteMaxRating + WhiteRatingChange, data=datos, ks = 1:200)
a=knn$MISCLASS
d=data.frame("k"=seq(1,200),"Exactitud"=1-a)
ggplot(d, aes(x=k,y=optimal))+
  geom_line()+
  geom_point()+
  labs(y= "Exactitud", x = "K")+
  ggtitle("K vs Exactitud")+
  theme_classic()+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(legend.position = "top")
## Búsqueda del punto de corte óptimo para la clasificación
pred_knn <- predict(knn, val[,-c(26,27)])
tabla <- table(val$Result, pred_knn)

acierto_knn <- mean(val$Result == pred_knn)
```

Máquinas de vector soporte

```
mvs <- svm(Result ~ AgresividadBP + AgresividadWP + BlackMaxRating +
BlackRatingChange + Edad_BP + Edad_WP +
EF1B + EF1W + EF2B + EF2W + H2H + LuchaBP + LuchaWP + Rating1 + Rating2 +
tipo_partida + WhiteMaxRating + WhiteRatingChange, data = datos, probability = TRUE)
pred_svm=predict(mvs, val)
acierto_svm <- mean(datos$Result == pred_svm)
tabla <- table(val$Result, pred_svm)
```

Modelo usando el rating

```
Rating1=c(Validacion$Rating1,ValidacionTata$Rating1 )

Rating2=c(Validacion$Rating2,ValidacionTata$Rating2)

Result=c(Validacion$Result,ValidacionTata$Result)
Result=factor(Result,levels=c("1-0", "1/2-1/2", "0-1"))

p=0.1
val=function(p){
A=c()
B=c()
Pred=c()
for (i in 1:length(Result)){
  A[i]=1/(1+10^((Rating2[i]-Rating1[i])/400))-0.5*p
  B[i]=1/(1+10^((Rating1[i]-Rating2[i])/400))-0.5*p
  if (max(A[i],B[i],p)==A[i]){Pred[i]="1-0"}
  if (max(A[i],B[i],p)==p){Pred[i]="1/2-1/2"}
  if(max(A[i],B[i],p)==B[i]){Pred[i]="0-1"}
}
Pred=factor(Pred,levels=c("1-0", "1/2-1/2", "0-1"))

tabla <- table(Result, Pred)
```

```
Acierto<-mean(Result == Pred)
Grave= (tabla[1,3]+tabla[3,1])/sum(tabla)
ac=data.frame("p"=p,"Acierto"=Acierto, "Grave"=Grave)
return(ac)}


s=c()
for (p in seq(0,1,0.01)){
  s=rbind(s,val(p))
}

ggplot(s, aes(x=p)) +
  geom_line(aes(y = Acierto,colour=" % Acierto"),size=1.5) +
  geom_line(aes(y = Grave, colour="% Errores Graves"),size=1.5) +
  labs(y= "Probabilidad", x = "Probabilidad de tablas")+
  ggtitle("Porcentaje de acierto y de errores graves según la probabilidad de tablas")+
  theme_classic()+
  scale_color_discrete(name='Medida')+
  theme(plot.title = element_text(hjust = 0.5))+
  theme(legend.position = "top")

max(s$Acierto) #0.4384384 y error grave 0, con p=0.48 y 0.49
max(s$Grave) # 0.1891892

#p a partir de la base de datos
summary(datos$Result)
#Probabilidad de tablas:

p=summary(datos$Result)["1/2-1/2"]/sum(summary(datos$Result))
val(p)
```

Bibliografía

[1] Zheyuan Fan, Yuming Kuang, and Xiaolin Lin. Chess game result prediction system. CS 229 Machine Learning Project Report, 2013.

[2] International chess federation. `https://www.fide.com/`.

[3] Diogo R. Ferreira. Predicting the outcome of chess games based on historical data. Technical report, IST–Technical University of Lisbon, 2010.

[4] John Handley. Comparative analysis of Bradley-Terry and Thurstone-Mosteller paired comparison models for image quality assessment. *Proceedings of the IS&T PICS Conference*, pages 108–112, 2004.

[5] Barak Oshri. Predicting moves in chess using convolutional neural networks. 2015.

[6] Paras Lehana, Department of Computer Science and Engineering, Jaypee Institute of Information Technology, Noida, 201304, India, Sudhanshu Kulshrestha, Nitin Thakur, and Pradeep Asthana. Statistical analysis on result prediction in chess. *Int. j. inf. eng. electron. bus.*, 10(4):25–32, 2018.

[7] H.A.R. Pulido. *Predicting the Outcome of a Chess Game by Statistical and Machine Learning Techniques*. Tesis Doctoral. Universitat Politècnica de Catalunya. Facultat d'Informàtica de Barcelona, 2016.

[8] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958.

[9] Peter McCullagh. Regression models for ordinal data. *J. R. Stat. Soc.*, 42(2):109–127, 1980.

[10] Keith McNulty. *Handbook of regression modeling in people analytics: With examples in R and python*. Chapman and Hall/CRC, Boca Raton, 2021.

[11] Abdalla M El-Habil. An application on multinomial logistic regression model. *Pak. J. Stat. Oper. Res.*, 8(2):271, 2012.

[12] Cti Reviews. *Regression, models, methods and applications: Statistics, regression analysis*. Cram101, La Vergne, TN, Estados Unidos de América, 2016.

[13] P. McCullagh and J.A. Nelder. *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman & Hall, 1989.

[14] J Scott Long and Jeremy Freese. *Regression models for categorical dependent variables using Stata, third edition*. Stata Press, Philadelphia, PA, 3 edition, 2014.

[15] J Scott Long. *Regression models for categorical and limited dependent variables*. SAGE Publications, Thousand Oaks, CA, 1997.

[16] Wray Buntine. Learning classification trees. *Stat. Comput.*, 2(2):63–73, 1992.

[17] Elena Campo León. Introducción a las máquinas de vector soporte (SVM) en aprendizaje supervisado. 2016.

[18] Antonio Mucherino, Petraq J. Papajorgji, and Panos M. Pardalos. *k-Nearest Neighbor Classification*, pages 83–106. Springer New York, New York, NY, 2009.

[19] `https://feda.org/feda2k16/wp-content/uploads/Leyes-2017.pdf`.

[20] Walter Tevis. *The Queen's Gambit*. Blackstone Publishing, 2018.

[21] `https://2700chess.com/`. Fecha de acceso: 2022-5-9.

[22] Brian D Ripley. The R project in statistical computing. *MSOR connect.*, 1(1):23–25, 2001.

[23] Jaime Lopez. Web scraping. 2018.

[24] M. H. Wickham, H. Wickham. Package 'rvest'. `https://cran.r-project.org/web/packages/rvest/rvest.pdf`, 2016.

[25] polr function - RDocumentation. `https://www.rdocumentation.org/packages/MASS/versions/7.3-57/topics/polr`. Fecha de acceso: 2022-5-9.

[26] R. H. B. Christensen. ordinal—regression models for ordinal data, 2019. R package version 2019.12-10. https://CRAN.R-project.org/package=ordinal.

[27] PoTest function - RDocumentation. `https://www.rdocumentation.org/packages/car/versions/3.0-12/topics/poTest`. Fecha de acceso: 2022-5-9.

[28] car package - RDocumentation. `https://www.rdocumentation.org/packages/car/versions/3.0-12`. Fecha de acceso: 2022-5-9.

[29] multinom function - RDocumentation. `https://www.rdocumentation.org/packages/nnet/versions/7.3-17/topics/multinom`. Fecha de acceso: 2022-5-9.

[30] nnet package - RDocumentation. `https://www.rdocumentation.org/packages/nnet/versions/7.3-17`. Fecha de acceso: 2022-5-9.

[31] Joseph E Cavanaugh and Andrew A Neath. The akaike information criterion: Background, derivation, properties, application, interpretation, and refinements. *Wiley Interdiscip. Rev. Comput. Stat.*, 11(3):e1460, 2019.

[32] Pr2 function - RDocumentation. `https://www.rdocumentation.org/packages/epr/versions/3.0/topics/pr2`. Fecha de acceso: 2022-5-9.

[33] Alex Tahk, Achim Zeileis, Christina Maimone, Jim Fearon, Zoe Meers Maintainer, and Simon Jackman. Package 'pscl'. `https://cran.r-project.org/web/packages/pscl/pscl.pdf`. Fecha de acceso: 2022-5-9.

[34] Beth Maintainer. Package 'rpart'. `https://cran.r-project.org/web/packages/rpart/rpart.pdf`, 2022. Fecha de acceso: 2022-5-9.

[35] prune.rpart function - RDocumentation. https://www.rdocumentation.org/packages/rpart/versions/4.1.16/topics/prune.rpart. Fecha de acceso: 2022-5-9.

[36] svm function - RDocumentation. https://www.rdocumentation.org/packages/e1071/versions/1.7-9/topics/svm. Fecha de acceso: 2022-5-9.

[37] E1071 package - RDocumentation. https://www.rdocumentation.org/packages/e1071/versions/1.7-9. Fecha de acceso: 2022-5-9.

[38] kknn function - RDocumentation. https://www.rdocumentation.org/packages/kknn/versions/1.3.1/topics/kknn. Fecha de acceso: 2022-5-9.

[39] Kknn. https://www.rdocumentation.org/packages/kknn/versions/1.3.1. Fecha de acceso: 2022-5-9.